

UNIVERSIDAD DE CASTILLA-LA MANCHA  
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS



UN ENFOQUE ESTRUCTURADO PARA EL  
DESARROLLO DE INTERFACES DE USUARIO 3D

TESIS DOCTORAL

D. JOSÉ PASCUAL MOLINA MASSÓ

DIRIGIDA POR

DR. D. PASCUAL GONZÁLEZ LÓPEZ

ALBACETE, FEBRERO 2008



Dedicado a Ana, mi familia y amigos.



*“(...) Los Feds siguen trabajando en Planilandia. Nada de equipos tridimensionales, nada de visores ni sonido estéreo (...)”*  
“Snow Crash”, Neal Stephenson, 1992



# Tabla de contenidos

<b>Tabla de contenidos</b> .....	<b>i</b>
<b>Índice de figuras</b> .....	<b>ix</b>
<b>Índice de tablas</b> .....	<b>xi</b>
<b>Agradecimientos</b> .....	<b>xiii</b>
<b>Resumen</b> .....	<b>xv</b>
<b>Abstract</b> .....	<b>xvii</b>
<b>1 Introducción</b> .....	<b>1</b>
1.1 La interacción persona-ordenador .....	1
1.2 Las interfaces de usuario 3D .....	2
1.3 El desarrollo estructurado como objetivo.....	3
1.4 Organización de esta memoria .....	5
<b>2 La interfaz de usuario 3D y su diseño</b> .....	<b>7</b>
2.1 Las interfaces de usuario WIMP y post-WIMP.....	7
2.2 Gráficos 3D, interfaces de usuario 3D y Realidad Virtual .....	11
2.3 Interfaces de usuario 3D y sus aplicaciones .....	13
2.3.1 Interfaces para Realidad Virtual .....	15
2.3.2 Realismo para acercar la informática a más usuarios .....	16
2.3.3 3D para superar los límites físicos de la pantalla.....	17
2.3.4 3D para mejorar la organización de datos y tareas .....	18
2.3.5 Visualización de información .....	19
2.3.6 Creación y manipulación en 3D.....	20
2.3.7 Entretenimiento y diversión.....	20
2.3.8 Interfaces para Realidad Aumentada .....	21
2.4 Un nuevo espacio de diseño .....	22
2.4.1 Interacción 2D en entornos 3D .....	22
2.4.2 Extendiendo el continuo de interfaces de usuario.....	24
2.5 Elementos que integran las interfaces de usuario 3D.....	29
2.5.1 Espacio 3D .....	29
2.5.2 Objetos físicos y virtuales.....	30
2.5.3 Comportamiento .....	32
2.5.4 Interacción.....	33
2.6 Elementos básicos de la interacción .....	34
2.6.1 Diálogos interactivos .....	34

---

2.6.2 Tareas de interacción .....	35
2.6.3 Técnicas de interacción, objetos de interacción y widgets .....	38
2.6.4 Dispositivos físicos .....	40
2.7 El diseño de la interfaz de usuario 3D .....	42
2.7.1 El problema de los dispositivos físicos.....	44
2.7.2 Diseño de técnicas de interacción .....	45
2.7.3 Diseño de controles.....	46
2.7.4 Diseño de objetos 3D.....	47
2.7.5 Librerías de programación .....	48
2.7.6 Prototipos y especificación de la interfaz .....	49
2.7.7 Notaciones para la especificación de la interfaz.....	49
2.7.8 Herramientas software para el diseño de interfaces.....	53
2.7.9 Lenguajes de marcas para describir interfaces.....	53
2.7.10 Guías de diseño, usabilidad y evaluación .....	55
2.8 Resumen .....	56
<b>3 Metodologías para el desarrollo de interfaces de usuario 3D .....</b>	<b>59</b>
3.1 Introducción.....	59
3.2 Producción de cine de animación 3D .....	60
3.2.1 Cadena de producción de los estudios de animación Pixar .....	61
3.2.1.1 Descripción.....	61
3.2.1.2 Análisis.....	62
3.3 Creación de mundos virtuales para PC.....	64
3.3.1 Creación en dos o tres pasos .....	64
3.3.1.1 Descripción.....	64
3.3.1.2 Análisis.....	65
3.3.2 Metodología de diseño de EV's observada por Kaur .....	66
3.3.2.1 Descripción.....	66
3.3.2.2 Análisis.....	67
3.3.3 Redefinición del proceso de desarrollo por Kaur .....	67
3.3.3.1 Descripción.....	67
3.3.3.2 Análisis.....	68
3.3.4 Creación de mundos virtuales con VRML/X3D.....	69
3.3.4.1 Descripción.....	69
3.3.4.2 Análisis.....	71
3.3.5 Análisis del grupo .....	72
3.4 Desarrollo participativo de mundos virtuales.....	74
3.4.1 Desarrollo centrado en el usuario en el grupo VIRART.....	75
3.4.1.1 Descripción.....	75
3.4.1.2 Análisis.....	76

---

3.4.2 El ciclo de diseño de interacción 3D según Celentano y Pittarello .....	77
3.4.2.1 Descripción.....	77
3.4.2.2 Análisis.....	78
3.4.3 Análisis del grupo .....	79
3.5 Uso del análisis de tareas en la creación de mundos virtuales .....	79
3.5.1 Evaluación secuencial según Gabbard.....	80
3.5.1.1 Descripción.....	80
3.5.1.2 Análisis.....	81
3.5.2 Metodología para la creación de ambientes virtuales 3D .....	81
3.5.2.1 Descripción.....	81
3.5.2.2 Análisis.....	82
3.5.3 VEDS: Virtual Environment Development Structure.....	83
3.5.3.1 Descripción.....	83
3.5.3.2 Análisis.....	85
3.5.4 Ingeniería de entornos virtuales con X3D según Polys .....	87
3.5.4.1 Descripción.....	87
3.5.4.2 Análisis.....	88
3.5.5 Análisis del grupo .....	89
3.6 Desarrollo basado en ingeniería del software.....	90
3.6.1 Metodología de diseño para EV's según Fencott .....	91
3.6.1.1 Descripción.....	91
3.6.1.2 Análisis.....	93
3.6.2 CLEVR: Concurrent and LEvel by level development of VR systems.....	95
3.6.2.1 Descripción.....	95
3.6.2.2 Análisis.....	96
3.6.3 SENDA: Metodología de desarrollo de mundos virtuales habitados .....	97
3.6.3.1 Descripción.....	97
3.6.3.2 Análisis.....	100
3.6.4 Análisis del grupo .....	102
3.7 Creación de interfaces de usuario más allá del PC.....	104
3.7.1 El proyecto INQUISITIVE .....	105
3.7.1.1 Descripción.....	105
3.7.1.2 Análisis.....	106
3.7.2 Estrategias dirigidas por el contenido o por la interacción ....	107
3.7.2.1 Descripción.....	107
3.7.2.2 Análisis.....	109
3.7.3 VRID: Virtual Reality Interface Design .....	110

---

3.7.3.1	Descripción.....	110
3.7.3.2	Análisis.....	111
3.7.4	MPIUA: Modelo de Proceso de la Ingeniería de la Usabilidad y de la Accesibilidad.....	113
3.7.4.1	Descripción.....	113
3.7.4.2	Análisis.....	115
3.7.5	Diseño de entornos virtuales según Sutcliffe.....	116
3.7.5.1	Descripción.....	116
3.7.5.2	Análisis.....	119
3.7.6	Análisis del grupo .....	121
3.8	Métodos para el desarrollo de interfaces convencionales .....	123
3.8.1	Metodología de diseño según Foley <i>et al.</i> .....	123
3.8.1.1	Descripción.....	123
3.8.1.2	Análisis.....	125
3.8.2	LUCID: Logical User-Centred Interaction Design.....	126
3.8.2.1	Descripción.....	126
3.8.2.2	Análisis.....	127
3.8.3	OVID: Object View Interaction Design .....	128
3.8.3.1	Descripción.....	128
3.8.3.2	Análisis.....	130
3.8.4	IDEAS: Interface Development Environment within OASIS.....	131
3.8.4.1	Descripción.....	131
3.8.4.2	Análisis.....	133
3.8.5	Análisis del grupo .....	133
3.9	La metodología IDEAS-3D .....	135
3.10	Resumen .....	138
<b>4</b>	<b>Hacia la metodología TRES-D .....</b>	<b>145</b>
4.1	Introducción.....	145
4.2	Propósito y alcance.....	146
4.3	A partir de aquí.....	147
<b>5</b>	<b>Meta-modelos para una nueva metodología .....</b>	<b>149</b>
5.1	Introducción.....	149
5.2	Meta-modelo de objetos .....	149
5.2.1	Clasificación de objetos según comportamiento e interacción .....	150
5.2.2	Elementos de un objeto.....	152
5.3	Meta-modelo de elementos de la interacción .....	155
5.3.1	El diálogo.....	156

5.3.2 Las tareas y las operaciones.....	156
5.3.3 Técnicas de interacción, acciones y controles .....	158
5.3.4 Dispositivos físicos .....	162
5.4 Meta-modelo del espacio.....	163
5.4.1 El espacio 2D digital.....	164
5.4.2 El espacio 3D virtual.....	166
5.4.3 El espacio 3D real .....	169
5.4.4 Transiciones intra-, inter- y trans-espaciales .....	172
5.5 Resumen .....	174
<b>6 El modelo de proceso de la metodología TRES-D.....</b>	<b>177</b>
6.1 Introducción.....	177
6.2 Fases y etapas del modelo de proceso .....	178
6.3 Estudio previo.....	181
6.3.1 Presentación del problema .....	183
6.3.2 Análisis del problema .....	184
6.3.2.1 El tipo de aplicación.....	185
6.3.2.2 El perfil del usuario .....	185
6.3.2.3 Las tareas y los objetos.....	186
6.3.3 Propuesta de la solución .....	188
6.3.3.1 Generación de ideas .....	189
6.3.3.2 Validación de ideas .....	190
6.3.3.3 Presentación de la propuesta .....	191
6.4 Estudio detallado .....	191
6.4.1 Diseño .....	192
6.4.1.1 Diseño I.....	194
6.4.1.1.1 Las tareas.....	194
6.4.1.1.2 Los objetos .....	195
6.4.1.1.3 El espacio .....	196
6.4.1.1.4 Validación .....	197
6.4.1.2 Diseño II.....	197
6.4.1.2.1 Las tareas.....	198
6.4.1.2.2 Los objetos .....	200
6.4.1.2.3 El espacio .....	202
6.4.1.2.4 Validación .....	203
6.4.1.3 Herramientas ejemplares para guiar al diseñador .....	203
6.4.2 Implementación.....	207
6.4.2.1 Las tareas.....	208
6.4.2.2 Los objetos .....	209
6.4.2.3 Validación .....	209

---

6.4.2.4	La librería VUIToolkit como ejemplo .....	210
6.4.3	Despliegue y mantenimiento.....	213
6.5	Resumen .....	213
<b>7</b>	<b>Casos de estudio.....</b>	<b>217</b>
7.1	Introducción.....	217
7.2	Desarrollo de interfaces basadas en guantes.....	218
7.2.1	TTristéreo.....	219
7.2.1.1	Presentación del problema .....	219
7.2.1.2	Análisis del problema.....	220
7.2.1.3	Propuesta de solución.....	221
7.2.1.4	Diseño .....	222
7.2.1.5	Implementación.....	223
7.2.2	Virtual Reality Prismaker .....	224
7.2.2.1	Presentación del problema .....	224
7.2.2.2	Análisis del problema.....	224
7.2.2.3	Propuesta de solución.....	225
7.2.2.4	Diseño .....	226
7.2.2.5	Implementación.....	227
7.2.3	Una interfaz de película: “Minority Report” .....	228
7.2.3.1	Presentación del problema .....	228
7.2.3.2	Análisis del problema.....	229
7.2.3.3	Propuesta de solución.....	229
7.2.3.4	Diseño .....	230
7.2.3.5	Implementación.....	231
7.3	Prototipado de interfaces de usuario distribuidas.....	232
7.3.1	Aproximación al problema .....	232
7.3.2	El entorno de usuario y las plataformas.....	234
7.3.3	Interfaces de usuario .....	235
7.3.4	Ejemplo de migración.....	236
7.4	Resumen .....	236
<b>8</b>	<b>Conclusiones .....</b>	<b>239</b>
8.1	Echando la vista atrás... ..	239
8.2	Contribuciones.....	240
8.3	Publicaciones.....	245
8.4	Proyectos .....	247
8.5	Estancias en el extranjero .....	247
8.6	Trabajos futuros.....	248

---

<b>9 Conclusions .....</b>	<b>251</b>
9.1 Summary.....	251
<b>Anexo A La librería VUIToolkit.....</b>	<b>255</b>
A.1 Introducción.....	255
A.2 El marco de referencia Cameleon y el lenguaje UsiXML.....	257
A.3 La interfaz de usuario concreta en UsiXML.....	259
A.4 Correspondencia entre UsiXML y VRML97/X3D .....	261
A.5 Diseño e implementación de la librería VUIToolkit .....	264
A.5.1 Diseño de la arquitectura.....	265
A.5.2 Implementación de los widgets 3D.....	266
A.5.2.1 VUIWindow .....	272
A.5.2.2 VUIButton, VUIToggleButton y VUICheckBox.....	273
A.5.2.3 VUITextComponent y VUIImageComponent.....	274
A.5.2.4 VUISlider y VUICursor .....	275
A.5.2.5 VUIComboBox .....	276
<b>Referencias.....</b>	<b>277</b>



# Índice de figuras

Figura 2.1 El continuo realidad-virtualidad de Milgram y Kishino.....	22
Figura 2.2 Continuo digital-virtual-real de interfaces de usuario .....	25
Figura 2.3 Categorías del modelo de Fencott e Isdale .....	31
Figura 2.4 De las tareas a las técnicas de interacción según Sastry <i>et al.</i> .....	40
Figura 2.5 Diferentes modelos de guantes ofrecen distintas tecnologías y número de sensores. ....	44
Figura 2.6 Técnicas de selección ray casting y go-go.....	45
Figura 3.1 Bocetos, figuras, modelos 3D e instantáneas pertenecientes a la producción del corto de animación “El juego de Geri” .....	62
Figura 3.2 Creación de mundos virtuales con VRML según B. Hay.....	70
Figura 3.3 Desarrollo de mundos virtuales centrado en el usuario según Neale y Nichols.....	75
Figura 3.4 Diseño de entornos virtuales según Fencott .....	92
Figura 3.5 Desarrollo de mundos virtuales habitados según SENDA .....	100
Figura 3.6 Diseño de sistemas interactivos según MPIUA.....	114
Figura 3.7 Los cuatro niveles de la metodología IDEAS-3D .....	136
Figura 5.1 Clasificación propuesta de objetos .....	151
Figura 5.2 Modelo propuesto para los objetos.....	155
Figura 5.3 Descomposición de las tareas de interacción.....	157
Figura 5.4 Técnicas de interacción: acciones virtuales y controles, y su relación con las tareas.....	158
Figura 5.5 El avatar como una marioneta del usuario.....	159
Figura 5.6 Relación entre componente y control .....	160
Figura 5.7 Widgets 3D de la librería VUIToolkit.....	161
Figura 5.8 Obgets en un juguete virtual.....	161
Figura 5.9 Tipos de controles.....	162
Figura 5.10 Los espacios digital, virtual y real en el meta-modelo .....	164
Figura 5.11 El concepto de lugar es común a todos los espacios .....	167
Figura 5.12 Una interfaz de usuario 2D en un entorno virtual 3D y viceversa. ....	168
Figura 5.13 El puerto de vista relaciona el espacio virtual con el digital .....	169
Figura 5.14 Superficie y volumen de interacción en una misma pantalla de ordenador .....	171
Figura 5.15 Las superficies y volúmenes de interacción relacionan el espacio real con los espacios digital y virtual.....	172
Figura 6.2 Modelo de proceso de la metodología TRES-D.....	180
Figura 6.3 Herramienta para guiar en la elección de una técnica de selección y manipulación.....	205

---

Figura 6.4 Herramienta para guiar en la elección de una técnica de introducción de texto.....	206
Figura 6.5 Ordenador portátil virtual con widgets 3D de la librería VUIToolkit en su pantalla .....	211
Figura 7.1 Primer boceto de TTristéreo e interfaz de usuario final .....	222
Figura 7.2 Gestos de coger, rotar y solar una pieza en TTristéreo.....	223
Figura 7.3 Gestos “pinch” para coger y soltar en VRPrismaker.....	226
Figura 7.4 El entorno virtual de VRPrismaker y usuario.....	227
Figura 7.5 Diagrama de estados y gestos asociados a cada operación.....	231
Figura 7.6 Vista general del entorno mostrando las cinco plataformas .....	234
Figura 7.7 Ventanas virtualizadas de las dos aplicaciones de este caso de estudio .....	235
Figura 7.8 Secuencia de imágenes que muestra la migración desde un portátil al Pocket PC.....	236
Figura A.1 El marco de trabajo Cameleon.....	258
Figura A.2 Descomposición de un modelo de interfaz concreta en CIO’s .....	260
Figura A.3 Definición de <i>toggleButton</i> en VRML97 .....	267
Figura A.4 Definición de <i>toggleButton</i> en X3D .....	269
Figura A.5 Widgets 3D en una interfaz de usuario final vista a través de Cortona .....	271
Figura A.6 Vista de frente de los widgets 3D, esta vez a través del visor Flux	271

# Índice de tablas

Tabla 2.1 Estado de las interfaces actuales, según [Furness, 2001].....	8
Tabla 2.2 Características de las interfaces actuales frente a las non-WIMP, según [Jacob, 1999].....	10
Tabla 2.3 Características de las interfaces convencionales frente a las de Realidad Virtual según [Tanriverdi, 2001] .....	10
Tabla 2.4 Usos de la tecnología 3D según diferentes autores.....	14
Tabla 2.5 Lista propuesta de usos de las interfaces de usuario 3D .....	15
Tabla 2.6 Clases de objetos según Eastgate.....	31
Tabla 2.7 Clasificaciones de objetos en base a su comportamiento .....	32
Tabla 2.8 Analogía de Foley <i>et al.</i> entre elementos de la interacción y del lenguaje natural.....	35
Tabla 2.9 Tareas de interacción en aplicaciones gráficas 3D .....	37
Tabla 2.10 Tareas de interacción en aplicaciones de Realidad Virtual.....	38
Tabla 2.11 Tareas apropiadas para diferentes dispositivos.....	41
Tabla 3.1 Diagramas en la separación de vistas de Kim <i>et al.</i> frente al modelo SBF de McIntosh .....	97
Tabla 3.2 Uso de diagramas en las propuestas de desarrollo basadas en ingeniería del software.....	103
Tabla 3.3 Secuencia de etapas según el tipo de estrategia .....	108
Tabla 3.4 Las dos fases de diseño de VRID .....	110
Tabla 3.5 Similitudes entre los modelos que proponen Tanriverdi <i>et al.</i> y otros autores.....	112
Tabla 3.6 Distinción entre flujo y control en varias propuestas.....	122
Tabla 3.7 Cobertura de las actividades según rol y propuesta .....	139
Tabla 3.8 Herramientas más representativas citadas en cada propuesta .....	140
Tabla 5.1 Diferentes estilos de interacción según [Foley, 1996] .....	157
Tabla 7.1 Propiedades de pantalla de las diferentes plataformas.....	235
Tabla A.1 Correspondencias posibles entre el nivel CUI de UsiXML y VRML97/X3D .....	263



# Agradecimientos

Ya está. Muchas horas de trabajo quedan atrás. Horas de lectura, con los lápices de colores subrayando las palabras de otros. Horas de análisis y reflexión, frente a una pizarra blanca o una pared con Post-it's. Horas de discusión, las más productivas. Horas de programación, las más entretenidas. Horas de redacción, las más largas. Sin duda, esas fueron las más largas. Horas de carreras y prisas, siempre las hubo. Pero ahora escribo mis últimas palabras en esta Tesis Doctoral para dedicarlas a vosotros. Para mí son estas las más importantes, pues si yo he dedicado horas a este trabajo, vosotros habéis dedicado las vuestras a mí.

En primer lugar, le estoy muy agradecido a mi director, Pascual, quien ya lo fuera también de mi proyecto final de carrera, y a quien le debo la oportunidad y el privilegio de investigar e impartir docencia en Gráficos por Ordenador y Realidad Virtual. No fueron fáciles los comienzos, al dejar atrás los buenos resultados que obteníamos en la aceleración del trazado de rayos para volcarnos en un nuevo campo, el de la Interacción Persona-Ordenador, que por aquel entonces era un campo emergente. En todo este tiempo no he sido el mejor doctorando pero, frente a mis defectos, él ha sido paciente y ha mantenido su confianza en mí hasta llegar, por fin, a este momento.

Aunque no fueran fáciles aquellos comienzos, tampoco entonces estaba solo, sino muy bien acompañado por la gente que entonces se reunió para formar el grupo de investigación LoUISE. En ellos siempre he encontrado el apoyo necesario, y me han dado una y mil facilidades para dedicar el tiempo necesario al trabajo que aquí se presenta. En particular, quiero agradecer a María su compromiso, al hacer posible la estancia de investigación que realicé en la Université catholique de Louvain, y que resultó fundamental en el desarrollo de este trabajo.

Durante los meses que duró aquella estancia tuve la suerte de estar bajo la tutela del profesor Jean Vanderdonckt. A él debo agradecer el tiempo y la atención que entonces me prestó. Pero no sólo entonces, pues aquello fue el comienzo de una colaboración que se mantiene hasta ahora y que ha dado muchos frutos, en forma de múltiples publicaciones hasta llegar a la presente Tesis. Sin duda, esta se ha beneficiado del enorme conocimiento del campo que atesora este reconocido investigador, su facilidad para dar con nuevas ideas, su capacidad para plasmarlas en el papel y, sobre todo, su inagotable ánimo y empeño.

Tampoco estuve solo aquellos siete meses que pasé en esa pequeña localidad belga llamada Louvain-la-Neuve, pues en aquella experiencia me acompañaban Víctor y Paco, entonces también candidatos al título de Doctor. Hoy sin duda son dos grandes doctores, pero sobre todo siguen siendo dos grandes amigos que siempre me han ayudado cuando lo he necesitado. Como también me han ayudado mis compañeros de promoción Julia y Gregorio, así como otros más veteranos, como Blanca, Elena, Paco, Pedro, Quique, y tantos otros del Departamento de Sistemas Informáticos que me han demostrado su apoyo con sus constantes ánimos, y todos ellos han hecho que me sienta parte de esta Escuela en la que tanto tiempo pasamos.

Mi agradecimiento también es para todos aquellos alumnos que confiaron en mí la dirección de su proyecto final de carrera, y que con su trabajo han colaborado también en la realización de esta Tesis Doctoral. Todos habéis sido un ejemplo para mí, enseñándome a sacar fuerzas en aquellos momentos en los que flaquean. Arturo, Antonio, Diego, Francisco, Víctor, Gabriel, Antíoco, Fernando, Julián, y Jonatan son hoy Ingenieros en Informática. Arturo y Diego siguen hoy en día su carrera investigadora en este campo, y son dos compañeros infatigables a los que les agradezco aún más si cabe su confianza en este proyecto. El laboratorio de Realidad Virtual es lo que es gracias a todos ellos. Este trabajo es también, por tanto, suyo.

Por supuesto, a mis padres, a ellos les debo todo lo que soy. Siempre han estado ahí, haciendo mi vida lo más fácil posible para que sólo tuviera que preocuparme por terminar mis estudios y mi trabajo, porque ellos ya se preocupaban por mí. Pero si hay alguien que sabe el esfuerzo que me ha costado llegar hasta aquí es Ana, a ella nunca podré agradecerle lo suficiente todo lo que ha hecho por mí, tantos días en los que parecía no salir de la Escuela por terminar siempre tal o cual artículo, y más de uno en los que me traía la cena al despacho porque se hacía muy tarde y seguía sin salir de allí. Gracias de todo corazón por esperarme. A ellos, a mi familia y amigos, les dedico este trabajo.

No quiero terminar estas líneas sin acordarme de mi padrino, Federico. Aún recuerdo cuando él y mi padre trajeron a casa aquel MSX comprado en Galerías Preciados. También recuerdo a Jesús Lorés y su inagotable sonrisa. Sus consejos aún resuenan en mi cabeza. Descansen en paz los dos.

# Resumen

Las Interfaces de Usuario 3D, también conocidas por sus siglas UI3D o mejor por las propias en inglés, 3DUI, hacen referencia a aquellas interfaces en las que la interacción tiene lugar en el espacio tridimensional. Este área de la Interacción Persona-Ordenador puede parecer nueva, pero lo cierto es que estas interfaces llevan muchos años siendo estudiadas, si bien suelen presentarse bajo otros nombres, como gráficos 3D interactivos, Realidad Virtual o Realidad Aumentada. De hecho, uno de los iconos de estas interfaces, el visiocasco, comparte orígenes en el tiempo con otro icono, el ratón, esta vez de las interfaces de escritorio. Sin embargo, estas últimas disfrutan hoy en día un estado de madurez que todavía no han alcanzado las tridimensionales. La diferencia es que, tras todos estos años, las interfaces de escritorio cuentan con un conjunto de estilos de interacción y widgets bien conocidos, y su desarrollo se basa en métodos ya consolidados. En cambio, las interfaces de usuario 3D carecen de un conjunto estándar de técnicas de interacción y controles, y su creación está fuertemente dirigida por la experiencia e intuición del propio desarrollador.

No obstante, en estos años se ha hecho un gran esfuerzo por identificar cuáles son las tareas de interacción “universales” que puedan encontrarse en prácticamente cualquier interfaz de usuario 3D, y unido a ese esfuerzo se han propuesto numerosas técnicas de interacción que resolvían la realización de esas tareas en diferentes entornos, algunas muy conocidas como ray casting o go-go. Sin embargo, la experiencia del desarrollador sigue marcando su creación, los programadores aplican sus habilidades de ingeniería del software, los diseñadores de interfaces sus conocimientos en interacción persona-ordenador, y los artistas su saber hacer en la creación de contenidos. El problema es que muchas propuestas metodológicas ofrecen sólo la visión de uno de estos roles, que es sólo una visión parcial del desarrollo de la interfaz 3D, pues no sólo involucra tareas e interacción sino también contenidos, y para ello es necesario combinar la aproximación ingenieril de unos con la artística de otros.

Ese es, precisamente, unos de los objetivos que persigue el marco metodológico TRES-D que se presenta en este trabajo doctoral, y que no es otro que el de ofrecer un enfoque estructurado orientado tanto a tareas e interacción como a objetos y contenido, dos líneas de trabajo paralelas que se entrecruzan en el desarrollo. De este modo, se persigue involucrar tanto a diseñadores de interfaces de usuario y programadores como a artistas y creadores de contenido digital. Pero también al cliente, al experto en el dominio y, por supuesto, al

usuario. Todo ello en un proceso caracterizado por dos grandes fases y en donde el fin de la primera –estudio previo- es elaborar una propuesta de solución que, sólo si es aceptada, es entonces desarrollada por completo en la segunda – estudio detallado-. Esta división, más propia de tiempos pretéritos en donde el riesgo en el desarrollo del software era mayor, se trae aquí de nuevo precisamente por el riesgo que involucra el desarrollo en un campo aún inmaduro como es el de las interfaces tridimensionales.

Pero la propuesta TRES-D no se limita únicamente a un modelo de proceso. Aquí se presenta también un nuevo continuo digital-virtual-real con el que se pretende describir el espacio de diseño de estas interfaces. Tres meta-modelos, uno para los objetos, otro para la interacción y otro para el espacio, proporcionan además a los desarrolladores un lenguaje común con el que comprender mejor estas interfaces, plasmar sus ideas y comunicarlas al resto del equipo. Para guiar al diseñador, se proponen también dos herramientas para la selección de técnicas de interacción en la forma de un árbol de decisión, uno para la tarea de selección/manipulación de objetos y otra para la introducción de texto, ambas en entornos inmersivos. Y para el programador, se incluye una librería, VUIToolkit, que se caracteriza por transformar las interfaces planas en una representación tridimensional que facilite la transición de la metáfora de escritorio a los entornos virtuales. Todo ello se completa con cuatro casos de estudio, con los que se ha querido también, por qué no, contribuir a la búsqueda de esa aplicación definitiva a la que tantos aluden.

# Abstract

Three-Dimensional User Interfaces, or 3DUIs, refer to those interfaces where interaction takes place in three-dimensional space. This area of Human-Computer Interaction is not a novel one, as these interfaces have been studied for several years now, though they have often been presented under other names, such as interactive 3D graphics, Virtual Reality or Augmented Reality. In fact, one of the icons of these interfaces, the head-mounted display, was invented short after another popular icon, the mouse, this being of desktop interfaces. However, the latter have now a state of maturity that three-dimensional interfaces have not reached yet. Desktop interfaces rely on a set of well-known interaction styles and widgets, and their development is based on well-funded methods. Quite the opposite, 3D user interfaces lack of a standard set of interaction techniques and controls, and their development relies on the experience and intuition of the developer.

Despite that fact, progress has been done in identifying the “universal” tasks that can be found in almost every 3D user interface, and together with it, many interaction techniques have been proposed to perform such tasks in different environments, some of them becoming quite popular such as ray casting or go-go. However, experience remains the key to successful development, and in this sense programmers apply their knowledge in software engineering practices, user interface designers rely on their skills in human-computer interaction, and artists use their know-how in content creation. The problem is that many proposed methods approach this development from the view of just one of these roles, then offering a partial solution to that development, which is not only about code, interaction or content but all these things together. Thus, it is necessary to combine both an engineering approach with a creative one.

That is one of the objectives addressed by the TRES-D framework, which is presented in this doctoral dissertation, offering a structured approach that is oriented to task and interaction as well as to objects and content. This way, the proposal promotes the participation of user interface designers and programmers as well as artists and digital content creators. In addition to them, it also involves the client, the domain expert and, of course, the user. The whole process is characterized by two main phases. The aim of the first one –the previous study- is to propose a solution to a given problem and, only if it is accepted, then the second phase follows –the detailed study- to complete the proposed design, and to implement and deploy it. This division in two phases reminds of former approaches to software development that were meant to reduce risks, and that is

exactly its purpose here, to reduce the risks that the development of experimental interfaces involves.

All in all, TRES-D is not simply a process model. A new digital-virtual-real continuum is also presented, which is meant to describe the design space of three-dimensional interfaces. Three meta-models, one for objects, one for interaction and a last one for space, offer developers a common language to better understand these interfaces, bring their ideas to paper, and communicate them to other developers. In order to help the designer when selecting the right interaction technique, two decision trees are given as guidance tools, one for object selection/manipulation tasks and another one for text input, both for immersive environments. To help the programmer when translating existing interfaces to three-dimensional environments, a new library is introduced, named VUIToolkit, which transforms plain widgets into 3D ones by adding true depth to them. Finally, the TRES-D proposal is demonstrated with four case studies, each showing a different application of 3D user interfaces.

# 1

## Introducción

### 1.1 La interacción persona-ordenador

Con los primeros computadores aparecieron los primeros programas de ordenador y con ellos el principio de un arte como es el de la programación, que a lo largo de todos estos años ha ido evolucionando para convertirse en toda una ciencia como es la ingeniería del software. Durante las primeras décadas, el esfuerzo de los ingenieros del software se dirigía hacia el desarrollo de herramientas y aplicaciones que cumplieran todos los requisitos funcionales que se les exigían. La interfaz de usuario no representaba un aspecto muy importante en el desarrollo, en parte porque las limitaciones técnicas de aquellos momentos imponían austeras interfaces basadas en comandos, formularios o menús, y que en cualquier caso se presuponía que el usuario aprendería a utilizar el sistema con el debido tiempo y esfuerzo.

Sin embargo, esta situación cambió a partir de los años 80, cuando la tecnología hizo posible la comercialización de ordenadores para la oficina y el hogar con interfaces gráficas, y las aplicaciones basadas en la metáfora del escritorio y el ratón comenzaron a ser utilizadas por millones de personas. Se reconoció que, por encima de unos requisitos mínimos de funcionalidad, el factor más importante del éxito de una herramienta informática era la facilidad de uso de su interfaz.

La Interfaz de Usuario o IU –en inglés, *User Interface* o UI- pasó así a ocupar un papel protagonista dentro del desarrollo de cualquier aplicación, y como consecuencia de ello creció enormemente el interés por la Interacción Persona-Ordenador o IPO. Conocida también bajo las siglas HCI –en inglés, *Human-Computer Interaction*- o CHI –*Computer-Human Interaction*-, se trata de un campo de estudio multidisciplinar [Preece, 1994] [Shneiderman, 1998] que conecta conocimientos tan usuales dentro de los estudios de informática, como la programación, la ingeniería del software, la inteligencia artificial o la telemática, con otros conocimientos que pueden resultar ajenos a ellos, como la lingüística, la psicología, el diseño, la sociología o la ergonomía.

Hoy en día, la importancia adquirida por este campo de estudio se refleja en el gran número de talleres, conferencias, congresos y otros simposios que se organizan cada año con esta temática. Por ejemplo, AIPO (Asociación

Interacción Persona-Ordenador) [AIPO, URL] viene organizando todos los años desde el 2000 las jornadas Interacción como un foro en el que docentes, investigadores y profesionales de España y Latinoamérica exponen y comparten sus ideas. Toda esta actividad se traduce en una extensa bibliografía en la que se recogen los avances que se han ido produciendo en el campo y que le han llevado a su actual estado de madurez. Esto es así, al menos, en cuanto a las interfaces de escritorio se refiere.

## 1.2 Las interfaces de usuario 3D

Esa madurez a la que se hacía referencia en el apartado anterior puede confundirse, sin embargo, con una falta de ideas en el diseño de las interfaces de usuario. Y es que las aplicaciones de hoy en día presentan un aspecto no muy diferente al de aquellas que corrían en los primeros entornos basados en la metáfora del escritorio, salvando el relativo “realismo” ganado con la adición de sombras y suavizados a los botones y otros componentes de la interfaz. Aparentemente, este tipo de interfaz de usuario es suficientemente bueno para las actuales tareas de escritorio.

La realidad es que son muchos los esfuerzos que se están realizando en este campo para avanzar hacia interfaces de usuario mucho más ricas y mejor adaptadas a las habilidades y capacidades de cada persona –véase, p. ej., [Furness, 2001]-. Y las personas, como tales, somos seres vivos que se desenvuelven en un espacio tridimensional. Nuestros dos ojos nos proporcionan una visión binocular que, gracias al fenómeno de la estereopsis, nos ayuda a conocer la profundidad de los objetos que vemos. Nuestros dos oídos nos facilitan la localización de los sonidos en el espacio. La información que nos llega de nuestros sentidos, junto con la posición de cada parte de nuestro cuerpo, nos permite localizarnos a nosotros mismos en el espacio tridimensional. Cuando nos comunicamos con otras personas, acompañamos nuestras palabras con movimientos de las manos en el espacio. En definitiva, porque el mundo real es tridimensional, resulta inevitable preguntarse si las tres dimensiones del espacio son también la respuesta a esa búsqueda de una nueva interfaz con el ordenador.

Las Interfaces de Usuario 3D o IU3D –en inglés, *3D User Interface* o 3DUI- llevan, no obstante, muchos años siendo estudiadas. Lo que hace que pueda parecer un campo novedoso es que hasta hace poco no se presentaban bajo este nombre, sino bajo otros, como los gráficos 3D interactivos, la Realidad Virtual o la Realidad Aumentada, muchas veces tecnologías emergentes que no llegaban más allá de los laboratorios de investigación que las concebían, o de las selectas

industrias en las que se aplicaban, por no encontrar la “cabeza de playa” que les permitiera desembarcar en el masivo mercado del ordenador personal.

Así, la tecnología de gráficos 3D fue en su momento dominio exclusivo de los profesionales en informática gráfica que disponían de ordenadores de gama alta sobre los que ejecutaban software muy especializado. Las estaciones de trabajo necesitaban un considerable tiempo de computación e ingentes cantidades de memoria para representar en pantalla complejos modelos tridimensionales ricos en colores y texturas.

Actualmente, el hardware especializado para gráficos 3D, en la forma de tarjetas de expansión para PC, puede encontrarse en las tiendas de informática a precios asequibles pero con unas prestaciones que superan con creces las de aquellas estaciones, con cifras de procesamiento de polígonos que hablan de millones por segundo. Avances como el slot de expansión AGP y el PCI-Express o las librerías para gráficos OpenGL y Direct3D lo han hecho posible. Más aún, ese hardware especializado se distribuye con cada nuevo ordenador personal, gracias a la gran demanda por parte de los usuarios de PC's, no sólo para videojuegos, sino también para aplicaciones multimedia.

Precisamente, la presencia de ese hardware en los ordenadores personales ha animado a muchos a pensar que las interfaces de usuario tridimensionales no tardarían mucho tiempo en revolucionar la forma en que trabajamos con los ordenadores. Sin embargo, de los muchos proyectos que se han sucedido persiguiendo esa idea nos ha quedado una importante lección que aprender, que el éxito que tienen los gráficos 3D en los videojuegos no puede extrapolarse sin más al resto de aplicaciones de ordenador. La mera adaptación de los estilos tradicionales de interacción empleados en las interfaces de escritorio no proporciona una solución satisfactoria, sino que deben plantearse nuevas metáforas que enriquezcan nuestra interacción con el ordenador al añadir la tercera dimensión y que nos permitan aprovechar las singulares características de dispositivos tales como los sistemas de visión estereoscópica, de sonido 3D o de posicionamiento, entre otros.

### **1.3 El desarrollo estructurado como objetivo**

Las interfaces de usuario 3D no han alcanzado, con todo, el mismo estado de madurez que presentan las interfaces de escritorio. Ya en [Herndon, 1994] se apuntaba que las aplicaciones gráficas 3D eran significativamente más difíciles de diseñar, implementar y usar que sus homólogas 2D. Una de las razones que explican esta situación es que el diseño espacial en las interfaces de usuario 3D es también significativamente mayor que en 2D, y queda aún mucho por

explorar. Los diseñadores deben enfrentarse a una gran variedad de dispositivos de entrada y de salida y a una tecnología que puede experimentar cambios muy rápidamente, con nuevas pantallas y sensores que requieren nuevas y apropiadas técnicas de interacción y una consecuente reevaluación del conocimiento y experiencia ganado hasta el momento [Bowman, 2001].

No obstante, la comunidad investigadora se interesó mucho por identificar cuáles eran las tareas de interacción más importantes que pudieran encontrarse en prácticamente cualquier interfaz de usuario 3D, aquellas a las que se dio por llamar “universales”, y ese interés se acompañó de numerosas propuestas de técnicas de interacción que resolvían la realización de esas tareas en diferentes entornos. Las técnicas más conocidas hoy en día, como p. ej. ray casting o gogo, se introdujeron en aquel momento, y desde entonces el ritmo de aparición de nuevas técnicas ha bajado mucho, como se recoge en [Bowman, 2006].

Sin embargo, aunque son muchas las técnicas de interacción 3D que se recogen en la bibliografía actual y no faltan investigaciones acerca del factor humano y la interacción en entornos tridimensionales, se echa en falta una opinión unánime sobre la forma de combinar los resultados de todas esas investigaciones y construir con ellos un marco único que guíe al diseñador en el desarrollo de interfaces de usuario 3D, dado que ese proceso ha dependido tradicionalmente del propio desarrollador y su experiencia pasada.

Un ejemplo es el desarrollo de mundos virtuales, el cual no es extraño que sea asumido por uno o varios creadores de contenido 3D dada la importancia que ese contenido tiene en la interfaz, centrándose en primer lugar en el propio modelado de los objetos en tres dimensiones, para al final añadir las correspondientes animaciones e interactividad a esos objetos. Esta práctica no es, sin embargo, la más aconsejable, ya que en el diseño de los objetos debe tenerse presente, desde el principio, su interactividad. No hacerlo así puede llevar a problemas de usabilidad, como ya descubriera Kaur en su trabajo doctoral [Kaur, 1998]. Como remedio, la propia Kaur propuso una nueva metodología acompañada de un conjunto de guías de diseño. Y como Kaur, muchos otros también han presentado sus propias propuestas para el desarrollo de mundos virtuales –véase, p. ej. [Fencott, 1999] o [Eastgate, 2001]–.

Cabe preguntarse, no obstante, si esas y otras metodologías que se han propuesto para aplicaciones concretas de las interfaces de usuario 3D son igualmente útiles para el desarrollo, en general, de cualquier interfaz tridimensional, y si no lo son, cuál debería ser entonces el proceso a seguir. Este el reto que se asume en este trabajo doctoral y que, como se verá a lo largo de la presente memoria, se ha traducido en el estudio de un amplio número de metodologías ya existentes y la propuesta, tras dicho estudio, de un nuevo marco metodológico al que se ha llamado TRES-D.

## 1.4 Organización de esta memoria

Este documento se ha dividido en un total de nueve capítulos a lo largo de los cuales se desarrolla todo el trabajo realizado, desde una revisión inicial del estado actual en diseño de interfaces tridimensionales a la puesta en práctica del nuevo marco de trabajo propuesto. Además, se ha incluido un anexo en el que se dan los detalles de implementación de la librería VUIToolkit que acompaña a la metodología propuesta. A continuación se resume brevemente el contenido de cada uno de los ocho capítulos que restan y del citado anexo:

- En el **segundo capítulo** se tratará de dar una visión lo más completa posible de las interfaces de usuario 3D, cuál es su definición, cuáles son sus aplicaciones, cuáles son los elementos que integran y qué problemas entraña el diseño de estas interfaces y de cada elemento en particular. Como primeras aportaciones, se dará una definición propia para estas interfaces, y se describirá el espacio de diseño de las mismas como un continuo digital-virtual-real.
- El **capítulo tres** recoge la revisión de metodologías a la que antes se ha hecho referencia. En total se recogen veintiséis propuestas de diferentes autores, cada una con su particular punto de vista y aproximación al problema, pero también con puntos en común que han permitido distribuirlos en siete grupos diferentes. Grupo por grupo, metodología por metodología, cada una de ellas es resumida primero, y analizada después. Además, cada grupo termina con un nuevo análisis en el que se discuten las fortalezas y flaquezas del mismo. Este capítulo incluye, además, una primera propuesta del autor de esta Tesis, a la que se ha llamado IDEAS-3D por ser una revisión de otra metodología, llamada IDEAS, para el desarrollo de interfaces 3D.
- Con el **cuarto capítulo** comienza la exposición de la propuesta que se presenta con este trabajo doctoral, definiendo el objeto y el alcance de esa nueva propuesta, y por supuesto dándole un nombre, que no es otro que TRES-D. Esta propuesta incluye tres meta-modelos que se explican en el **capítulo quinto**, uno para los objetos, otro para los elementos de interacción y un último para el espacio. Esos meta-modelos proporcionan el lenguaje en el que se basa la metodología, si bien el modelo de proceso, como tal, no se explica hasta el **capítulo seis**. Junto a ese modelo de modelo se describen también un par de guías para la selección de técnicas de interacción, útiles en la etapa de diseño, y se

presenta también la librería VUIToolkit como recurso para los desarrolladores ya en la etapa de implementación.

- El **capítulo siete** recoge cuatro casos de estudio con los que se lleva a la práctica la propuesta realizada. Tres de esos cuatro casos de estudio se utilizan para ilustrar el modelo de proceso de la metodología TRES-D, a través de la creación de tres interfaces de usuario 3D que, en particular, comparten el uso de guantes de datos como interfaz de entrada, si bien cada guante es distinto como respuesta a las necesidades concretas de cada uno de esos proyectos. El cuarto caso de estudio se centra, en cambio, en la librería VUIToolkit, mostrando su aplicación en el diseño y prototipado de interfaces de usuario distribuidas.
- El **octavo capítulo**, y último, resume todo el trabajo realizado, listando las principales contribuciones y destacando las publicaciones en las que se recogen distintas partes de ese trabajo. También se recogen otras actividades realizadas durante el tiempo que ha llevado el mismo, como los proyectos en los que se ha colaborado y la realización de una estancia en la Universidad catholique de Louvain bajo la supervisión del profesor Dr. Jean Vanderdonckt. Como no podía faltar, también se señalan en este capítulo aquellos aspectos de la propuesta que precisan ser revisados o ampliados y que, por tanto, se proponen como trabajo futuro. La redacción en inglés de este capítulo se ha incluido como otro nuevo, el **capítulo nueve**.
- Finalmente, se incluye un **anexo** en el que se amplía la información dada hasta entonces sobre la librería VUIToolkit, explicando el marco de referencia Cameleon y el lenguaje UsiXML del que parte su desarrollo, reparando en especial en el nivel CUI de ese lenguaje UsiXML, para después explicar la arquitectura que comparten las dos versiones realizadas de esta librería, y finalmente detallar uno a uno los elementos que la componen.

## 2

# La interfaz de usuario 3D y su diseño

## 2.1 Las interfaces de usuario WIMP y post-WIMP

Diferentes estilos de interacción han caracterizado largos periodos de la historia de las interfaces de usuario. Primero fue la interfaz basada en una línea de comandos o instrucciones, luego la basada en menús y formularios, y después la basada en manipulación directa. Adecuado a la tecnología disponible en su tiempo, cada nuevo estilo representaba un nuevo avance en la interacción con el usuario con respecto a la anterior, ganando en número de usuarios.

El periodo actual –tercero, según [van Dam, 1997]- se caracteriza por la interfaz de usuario WIMP (siglas en inglés correspondientes a *Windows, Icons, Menus y Pointers*). Esta interfaz fue concebida en los años 70 en los laboratorios Xerox PARC, y materializada a principios de los 80 en el ordenador Xerox Star. Su interfaz gráfica de usuario (en inglés GUI, *Graphical User Interface*) mostraba, en dos dimensiones, múltiples espacios de trabajo en forma de ventanas rectangulares, e iconos que representaban elementos habituales de una oficina, como carpetas, documentos, etc. Un invento de Douglas Engelbart, el ratón (1965), permitía al usuario del Star manipular de forma directa los elementos de la interfaz, desplazando el puntero por la pantalla.

Después de más de veinte años desde su aparición, la popularización de las interfaces de usuario WIMP ha hecho posible que la mayoría de los usuarios no tengan que leer más manuales al convertirse en un estándar *de facto* entre las interfaces de aplicación. De hecho, las aplicaciones actuales no se diferencian en mucho de las primeras aplicaciones de escritorio, si bien las interfaces WIMP tal y como las conocemos hoy son el resultado de un refinamiento a lo largo de los años. Algunos detalles de la interacción han sido perfeccionados, la presentación ha mejorado gracias a los avances en resolución y paleta de colores de la tecnología de salida, y han aparecido variantes del ratón, como el ratón con rueda para explorar documentos o el ratón con retorno táctil.

Aparentemente, las interfaces de usuario WIMP son lo bastante buenas para las tareas de escritorio convencionales. Como se comenta en la Web del proyecto Infinite-3D [Fei, URL1], se ha evolucionado desde una dimensión (1D, refiriéndose al flujo de caracteres en una terminal de texto) con un esquema de intercambio de información esperar-y-ejecutar, a las dos dimensiones (2D,

---

refiriéndose a estas interfaces gráficas de escritorio) con un esquema dirigido por eventos. Se reconoce que el beneficio de la evolución es sustancial, pero también que aún existen limitaciones, algunas de las cuales se recogen en la Tabla 2.1 y otras se comentan a continuación.

- 
- La información aún está muy codificada
  - La presentación no es tridimensional (vista y oído)
  - El campo de vista de las pantallas es muy reducido (no sumergen al usuario, ni siquiera sacan partido de la visión periférica)
  - El usuario observa la información desde afuera (no se aprovecha la forma en la que el ser humano organiza lo que percibe a su alrededor)
  - Modalidades de entrada poco flexibles (no se utiliza el habla o la mirada)
  - La presentación no es transparente (no puede superponerse al mundo real)
  - Las interfaces exigen que el usuario se adapte al ordenador
  - Las interfaces no son intuitivas (requieren tiempo de aprendizaje)
  - Resulta complicado involucrar a múltiples participantes
- 

**Tabla 2.1** Estado de las interfaces actuales, según [Furness, 2001]

La interfaz de una aplicación WIMP no es transparente al usuario como lo sería la interfaz ideal, no resulta natural mover el ratón para desplazar el puntero por la pantalla [Furness, 2001] o realizar acciones como el doble-clic [Tebutt, URL], y mucho menos es adecuado para personas de cualquier edad o condición física [van Dam, 1997]. Aunque el aprendizaje por separado de cada elemento de la interfaz (widget) es sencillo, su profusión en las aplicaciones exige al usuario un esfuerzo tan grande que este se resiste a aprender más funciones de las que ya conoce. Esa profusión obliga al usuario a ejecutar largas secuencias de acciones “apuntar y pulsar” (en inglés, “point and click”) hasta alcanzar la función deseada, perdiendo tiempo en la manipulación de la interfaz [van Dam, 1997], o abarrotan la pantalla con elementos que nunca llega a utilizar, perdiendo espacio de presentación. Además, esos elementos 2D son apropiados para aplicaciones 2D como procesadores de texto u hojas de cálculo, pero su correspondencia con tareas 3D es mucho menos natural, remediada en las aplicaciones 3D con paneles de botones 2D que rodean el mundo 3D –lo que van Dam denomina el “modelo de TV”- para manipularlo de forma indirecta, ampliando la distancia cognitiva entre el usuario y la tarea.

Las aplicaciones WIMP parecen desaprovechar la capacidad de cálculo y de almacenamiento de información de los ordenadores, la cual ha aumentado de forma increíble desde los inicios de este tercer periodo, y parece que seguirá creciendo al menos al ritmo marcado por la Ley de Moore (multiplicándose por dos cada 18-24 meses). [Tebutt, URL] recuerda que un Apple Mac de 1984 contaba con 128 KB de memoria frente a los más de 128 MB de un ordenador actual, y sin embargo continuamos haciendo las mismas cosas que entonces sin

que la interfaz haya mejorado de forma dramática. Además, también desaprovechan la capacidad de percibir y de actuar del ser humano. El habla, el oído y el tacto no participan en lo que [Jacob, 1999] describe como diálogos inherentemente secuenciales, por turnos (al estilo ping-pong) y con un único flujo de entrada/salida. Bill Buxton (científico jefe de Alias/Wavefront/Sgi) describe la interfaz de usuario WIMP como la interfaz perfecta para criaturas con un sólo ojo, un sólo dedo y ningún otro órgano sensorial –citado en [van Dam, 1997]-.

Por ello, a pesar de las muchas ventajas que han traído las interfaces de usuario WIMP, se considera que el estado actual de las interfaces de ordenador es pobre [Furness, 2001] y no es suficiente [van Dam, 1997]. Para Furness, el gran reto a corto plazo de la era de la información es ser capaces de aprovechar la tremenda capacidad que se alcanzará en medios digitales, computación y comunicaciones, pero al no existir correspondencia con las capacidades del ser humano, se está limitando la forma en la que la información fluye hacia y desde la mente de la persona. Entre las voces críticas está también la del Consejo Nacional de Investigación de los Estados Unidos, que en uno de sus informes destaca la recomendación de dejar atrás la tecnología y paradigmas de los años 60, y desarrollar nuevos enfoques en la interacción [National, 1997]. Y es que el ratón, inventado hace ya cuarenta años, parece el último adelanto en las interfaces persona-ordenador, cuando existen otras tecnologías hoy en día sobre las que se debería pensar en pos de una nueva generación de interfaces de usuario –la cuarta, de nuevo según van Dam-.

La interfaz que caracterizará este nuevo periodo recibe el nombre de post-WIMP [van Dam, 1997], pero también el de non-WIMP [Jacob, 1996] o post-PC [Nooface, URL], nombres que transmiten la idea de una interfaz de usuario que no es la actual. Para van Dam, una interfaz post-WIMP es aquella que cuenta con al menos una técnica de interacción que no depende de los clásicos widgets 2D como son los menús y los iconos, y cree que las interfaces post-WIMP involucrarán finalmente a todos nuestros sentidos en un lenguaje de comunicación natural, en paralelo, con múltiples usuarios. El reconocimiento de trazos de lápiz en un dispositivo de mano como un PDA, o los entornos virtuales, son ejemplos de interfaces post-WIMP para este autor. Entre otros componentes post-WIMP, van Dam cita las interfaces 3D, el reconocimiento de voz y gestos, la realimentación táctil y de fuerza, y entornos virtuales de diferentes grados de inmersión. Para Jacob, las interfaces non-WIMP se caracterizan por una interacción continua entre el usuario y el ordenador a través de varios canales o dispositivos paralelos, asíncronos. Estas características también se recogen en [Herndon, 1994], pero Jacob las compara además con las interfaces actuales (véase Tabla 2.2, tomada de [Jacob, 1999]). Según este autor, esas características pueden verse claramente en las interfaces de Realidad Virtual, aunque también son compartidas con otras tecnologías emergentes,

como juegos y entretenimiento, agentes inteligentes, interfaces basados en lápiz o en el movimiento del ojo, y la computación ubicua. Por último, la Web Nooface.com fue creada para dar soporte al intercambio de ideas acerca de la siguiente generación de interfaces de usuario, y recoge noticias y comentarios que se clasifican según un conjunto de categorías, como interfaces de usuario 3D visuales, interfaces para juegos y entretenimiento, interfaces de audio y voz, interfaces para dispositivos móviles y ordenadores vestibles, entre otras.

WIMP	Non-WIMP
Una única serie de entradas y salidas	Diálogos paralelos, asíncronos pero interrelacionados
Unidades de información (tokens) de tipo discreto	Entradas y salidas discretas y continuas
Unidades de información claras	Entrada probabilística, más difícil de separar en unidades
La secuencia es significativa, no el tiempo	Requisitos de tiempo real, cálculos basados en límites de tiempo
Instrucciones de usuario explícitas	Observación pasiva (no basada en instrucciones) del usuario

**Tabla 2.2** Características de las interfaces actuales frente a las non-WIMP, según [Jacob, 1999]

Es de destacar que las interfaces de Realidad Virtual son representantes de ese conjunto de características que se buscan en una interfaz post-WIMP frente a las limitaciones de las interfaces actuales. La Tabla 2.3 compara las características de las interfaces convencionales y de Realidad Virtual, según [Tanriverdi, 2001]. Sobre esas interfaces, los entornos virtuales y las interfaces de usuario 3D discurrirá el siguiente apartado.

Características	Interfaces convencionales	Interfaces de Realidad Virtual
Gráficos de los objetos	En su mayoría 2D	Principalmente 3D
Tipos de objetos	En su mayoría objetos virtuales	Objetos virtuales y físicos
Comportamientos de los objetos	En su mayoría objetos pasivos	Objetos pasivos y activos
Patrones de comunicación	En su mayoría simples	Principalmente complejos
Interacciones persona-ordenador	En su mayoría explícitas	Explícitas e implícitas

**Tabla 2.3** Características de las interfaces convencionales frente a las de Realidad Virtual según [Tanriverdi, 2001]

## 2.2 Gráficos 3D, interfaces de usuario 3D y Realidad Virtual

En muchas ocasiones, investigadores y desarrolladores utilizan el término “3D” o “3-D” para referirse a las tres dimensiones del espacio, pero también a los gráficos 3D y a su tecnología asociada, a las interfaces de usuario 3D y también a la Realidad Virtual. Dado que es bastante común que aparezcan y se mezclen estos diferentes términos en la bibliografía, es necesaria una definición de cada uno de ellos.

En primer lugar, existe una diferencia entre gráficos 3D y lo que se conoce como gráficos 2.5D. En estos últimos se utiliza la tecnología y los algoritmos de gráficos 2D para dibujar elementos 2D en la pantalla y reproducir, mediante operaciones sobre mapas de bits (*bitmaps*) y de forma limitada, algunas de las claves que emplea nuestro cerebro para extraer información de profundidad de las imágenes, como la superposición (de ventanas en un escritorio, por ejemplo), el tamaño relativo de los objetos (se considera más alejado el más pequeño), o el paralaje de movimiento (el fondo se desplaza a menor velocidad). En cambio, la tecnología y los algoritmos de gráficos 3D permiten obtener la vista desde cualquier punto de un mundo tridimensional reproduciendo con fidelidad las anteriores claves y más, aunque a un coste computacional mayor.

Sin embargo, Toni Parisi –uno de los padres del lenguaje VRML [VRML Spec, URL]- escribía que la mayoría de los gráficos 3D son en esencia 2D pues finalmente son representados en una ventana bidimensional en la pantalla, si bien añadía que también era posible encontrar sistemas donde la representación es “totalmente 3D”, como por ejemplo el sistema CAVE ([WWW-VRML, URL], §3/7/2003). De nuevo, la diferencia está en las claves visuales que se proporcionan, que en la habitual pantalla de PC son sólo monoculares, mientras que en un sistema estereoscópico como el CAVE se añaden otras binoculares. En casos como el último se habla entonces de “gráficos 3D estéreo” para recalcar esta característica, aunque la disparidad binocular sea la única de clave visual que se añade.

Puede que lo que lleve a Parisi considerar un sistema estereoscópico como una representación “totalmente 3D” sea el hecho de que la estereoscopia esté sobreestimada, como respondía un lector de Nielsen a su argumento de que la representación en 2D es mejor que en 3D porque nuestros ojos están orientados hacia delante, no somos ranas [Nielsen, 1998]. Sin embargo, es cierto que una de las dificultades de las 3D es la percepción de la profundidad [Foley, 1996], y en este aspecto la disparidad binocular nos permite determinar mejor esa profundidad, aunque esa habilidad se limite a los objetos cercanos, como

recuerda Chad Wingrave ([3D UI, URL], §30/7/2004). Como ejemplo de ello, para “the TaskGallery” [Microsoft, URL] se decidió que las ventanas no flotaran en el aire por no poder transmitir claves binoculares en un PC al uso, por lo que se optó por incluir soportes que las hicieran reposar sobre el suelo, como explica Dantzich –uno de sus creadores- en ([WWW-VRML, URL], §8/10/1999).

Con todo, se hable de gráficos 2.5D, 3D o 3D estéreo, debe entenderse que todos ellos son gráficos 3D diferenciándose en el número de claves visuales y la forma empleada para simularlas. Cuantas más claves se proporcionen mayor será la ilusión de profundidad, pero los gráficos serán igualmente 3D. De hecho, no es opinión única del que escribe que las tres principales claves visuales son: la perspectiva, la disparidad binocular, y la paralaje de movimiento (esta última sobre todo cuando está relacionada con el movimiento de nuestra cabeza).

Sobre la Realidad Virtual, es sabido que muchos investigadores prefieren utilizar el término “entorno virtual” (EV) –en inglés “virtual environment” o VE- para huir de la idea –muchas veces exagerada- que el público en general tiene de esta tecnología. También se refieren a ella como “realidad artificial” o “inmersión espacial” [Marsh, 1998]. La relación entre los gráficos 3D y la Realidad Virtual es abordada por Eric Maranne en ([WWW-VRML, URL], §4/7/2003), explicando que los gráficos 3D son a la Realidad Virtual lo que el papel es a los libros, esto es, un constituyente básico, y al igual que el papel, los gráficos 3D se ajustan a una gran variedad de fines. La diferencia entonces entre Realidad Virtual y otros sistemas de gráficos 3D es la interactividad, como bien dice [Eastgate, 2001] y otros tantos autores.

La interactividad en Realidad Virtual es posible gracias a su interfaz de usuario. Para muchos autores, las interfaces de Realidad Virtual son una parte de los entornos virtuales, la mayoría de los cuales –como apunta [Larimer, 2003]- proporcionan una interfaz de usuario 3D. Para Maranne, sin embargo, la propia aplicación de Realidad Virtual es la interfaz, añadiendo además que la Realidad Virtual y las interfaces de usuario 3D son dos campos distintos con objetivos diferentes aunque compartan las mismas herramientas y presentación, pero el mercado de las interfaces 3D es el aprendizaje y el de la Realidad Virtual es el entrenamiento. Precisamente, sobre las interfaces 3D, Maranne comenta que el mayor interés está en reducir el nivel de abstracción, ya que añadir una dimensión en la representación de la información puede marcar una gran diferencia. Los comentarios de Maranne pueden relacionarse con los aportados por Fei en ([3D UI, URL], §10/8/2004), donde indica que la prioridad de las interfaces de usuario 3D es favorecer la comunicación entre el cerebro y el ordenador, mientras que la prioridad de la Realidad Virtual es el realismo. Aunque más que el realismo, el reto más importante en los entornos virtuales es mantener el sentimiento de presencia [Cuppens, 2004].

El que aquí escribe es de la idea mayoritaria de que las interfaces de usuario 3D son una parte –muy importante– de las aplicaciones de Realidad Virtual, pero también de otros entornos, compartiendo por ello técnicas, herramientas y también problemas, aunque los fines no sean los mismos. Como definición, una interfaz de usuario 3D es *aquella en la que el lenguaje que emplea el usuario para comunicar órdenes e información al ordenador, y/o el lenguaje que emplea el ordenador para transmitir información al usuario, está basado en el espacio físico y sus tres dimensiones*. Esto es, bien el usuario emplea dispositivos 3D para interactuar con el sistema, bien el sistema presenta imágenes 3D al usuario, o ambas cosas. Los dispositivos 3D permiten que movimientos del usuario en las tres dimensiones del espacio real sean tomados como entrada de la aplicación e interpretados en su contexto. El concepto de “imagen 3D” no se limita únicamente al sentido de la vista, esto es, a los gráficos 3D, sino que se extiende al resto de los sentidos, pudiéndose hablar también de imágenes sonoras o táctiles. Esta definición general persigue abarcar las diferentes formas que puede adoptar una interfaz de usuario 3D, y que podemos encontrar en diferentes tipos de aplicaciones. El siguiente apartado abordará esta diversidad de aplicaciones.

## 2.3 Interfaces de usuario 3D y sus aplicaciones

La tecnología de las interfaces de usuario 3D es utilizada en múltiples dominios, y no sólo como parte de las aplicaciones de Realidad Virtual. Así, [Herndon, 1994] subraya el éxito de aplicaciones de gráficos 3D en CAD, CAE, y visualización médica y científica. Ese éxito también lo tienen los videojuegos 3D, que en el mercado del PC corre en paralelo a la creciente popularidad de las tarjetas gráficas 3D. Pero, aparte de los videojuegos, según [Leavitt, 2001] no se ha identificado aún la que podríamos llamar “aplicación definitiva”, traducción de la expresión en inglés “killer application”. De hecho, ese es uno de los argumentos que tratan de justificar el hecho de que las aplicaciones WIMP sigan siendo las más populares. Igualmente, existe un gran interés en su identificación, pues las interfaces de Realidad Virtual, y en general las interfaces de usuario 3D, han sido presentadas como candidatas a convertirse en la siguiente generación de interfaces.

Las expectativas sobre esta tecnología crecieron con el interés que despertaron los medios sobre la Realidad Virtual. Las prisas por encontrar sin aparente éxito esa aplicación definitiva llevó a la Realidad Virtual a ser acusada –como apunta [Sutcliffe, 2003]– de ser una tecnología en busca de una aplicación real. En ([WWW-VRML, URL], §4/7/2003), David Arendash cree que los usos que se les ha dado a las 3D han sido en realidad intentos de crear un problema para el cual las 3D son la solución, mientras que Maranne advierte que la tecnología no debe ser quien lleve la dirección, es sólo el motor. Marc Bernatchez resume todo

ello explicando que por encima de lo atractiva que pueda ser una tecnología, esta debería establecerse en base a una necesidad real, para resolver limitaciones y problemas actuales ([3D UI, URL], §10/8/2004). Y es que, como contestaba un lector a [Nielsen, 1998], en ocasiones se fuerza la utilización de las 3D simplemente por la novedad.

Demasiado tiempo ha estado este campo dirigido por ese factor de la novedad, como escribía Wingrave en la lista de correo ([3D UI, URL], §19/2/2002). Como explicaba también Angelillo en un correo a esa lista ese mismo día, no es que el entusiasmo sobre ese concepto original de la Realidad Virtual que iba a cambiar nuestras vidas haya acabado en nada (“the failure of gloves and googles”), sino que esa idea ha evolucionado hacia otras menos complejas pero más útiles. Según [Herndon, 1994], la investigación actual involucra explorar el espacio de diseño de interfaces 3D para las actuales aplicaciones gráficas 3D y buscar nuevas aplicaciones que serán mejoradas con la adición de la tercera dimensión.

Es imperativo, por tanto, dejar a un lado la urgencia por encontrar esa aplicación definitiva, y en cambio identificar los usos para los cuales la tecnología 3D es apropiada. Aunque, como se apunta en [Leavitt, 2001], llevará tiempo encontrar todos los usos de las 3D, podemos empezar abstrayendo esos usos de los diferentes dominios en los que se aplican. Así, en [Stuart, 2001] y [Sutcliffe, 2003] encontramos clasificaciones acerca de los usos de la Realidad Virtual, los cuales se recogen en la Tabla 2.4, aunque no listan todos los usos de las 3D. [Shneiderman, 2002] añade algún uso más en este sentido, en una clasificación que también se ha recogido en la misma tabla, pero aún quedarían otros.

Aplicaciones de los entornos virtuales según [Stuart, 2001]	Dominios de aplicación de la Realidad Virtual según [Sutcliffe, 2003]	Visualizaciones 3D según [Shneiderman, 2002]
<ul style="list-style-type: none"> <li>• Operación “online”.</li> <li>• Entrenamiento y ensayo “offline”.</li> <li>• Comprensión “online”.</li> <li>• Aprendizaje y adquisición de conocimiento “offline”.</li> <li>• Diseño “online”.</li> <li>• Entretenimiento.</li> <li>• Comunicación.</li> <li>• Herramientas para el estudio de las capacidades humanas motoras y de percepción.</li> </ul>	<ul style="list-style-type: none"> <li>• Educación y entrenamiento.</li> <li>• Análisis de requisitos y prototipos virtuales (por ejemplo, paseos por edificios virtuales).</li> <li>• Entretenimiento y diversión.</li> <li>• Tele-operación.</li> <li>• Marketing.</li> </ul>	<ul style="list-style-type: none"> <li>• Entornos virtuales inmersivos.</li> <li>• “Desktop 3D” para mundos 3D.</li> <li>• “Desktop 3D” para mundos artificiales.</li> <li>• “Desktop 3D” para visualización de información.</li> <li>• Diagramas 3D.</li> </ul>

**Tabla 2.4** Usos de la tecnología 3D según diferentes autores

En base a esas clasificaciones se han identificado un conjunto de usos de las tecnologías 3D, al cual se han añadido otros usos no recogidos en las mismas, obteniendo como resultado el listado de usos que aquí se propone y que se recoge en la Tabla 2.5.

- 
- Interfaces para Realidad Virtual
  - Realismo para acercar la informática a más usuarios
  - 3D para superar los límites físicos de la pantalla
  - 3D para mejorar la organización de datos y tareas
  - Visualización de información
  - Creación y manipulación en tres dimensiones
  - Entretenimiento y diversión
  - Interfaces para Realidad Aumentada
- 

**Tabla 2.5** Lista propuesta de usos de las interfaces de usuario 3D

En los apartados que siguen se abordará la descripción de cada uno de los usos que se han identificado en la propuesta dada, y que no deben entenderse como usos excluyentes entre sí.

### 2.3.1 Interfaces para Realidad Virtual

Uno de los usos que se da a la tecnología 3D no es otro que el de simular espacios tridimensionales como los que podemos observar en el mundo real. El grado de correspondencia con la realidad dependerá de los objetivos de la aplicación. Así, podemos encontrar entornos virtuales con los que se pretende reproducir determinados ambientes reales, tanto pasados (arqueología), presentes (simuladores, ciudades virtuales, tele-presencia), como futuros (prototipos virtuales).

En el caso de los simuladores, se persigue que el usuario pueda desarrollar unas habilidades en el mundo virtual que luego podrá poner en práctica en ese mundo real, y por ello Stuart califica este uso como entrenamiento y ensayo “offline”. Resulta muy útil cuando el entrenamiento en el mundo real es mucho más costoso o comporta riesgos que se desean evitar, como en caso de los simuladores de vuelo o los ensayos de operaciones quirúrgicas. En este caso, una de las prioridades en el diseño de la interfaz es lograr un gran realismo, con el fin de que el conocimiento aprendido o las habilidades entrenadas puedan trasladarse con éxito al mundo real y no se vean perjudicadas por cualidades o vicios del sistema de Realidad Virtual. Otra de las prioridades es lograr que el

sentimiento de presencia del usuario sea elevado, con el fin de que experimente las acciones en el mundo virtual como si estuviera en el ambiente real.

En el caso de la tele-operación –u operación “online”, como la califica Stuart-, también se persigue que sienta la realidad tal como es, pero la razón ahora es que las acciones que se realizan en el mundo virtual tienen un efecto sobre el ambiente real que está siendo operado a distancia. Sin embargo, a diferencia de los simuladores, en este caso el operador puede ser asistido por el ordenador en su trabajo, dándole facilidades que en la realidad no tendría.

En cualquier caso, reproducir la realidad exactamente como es no suele ser práctico, bien por tiempo de desarrollo o coste en equipos, bien por las propias limitaciones de la tecnología. Por ello, usualmente encontramos entornos que guardan gran similitud con los reales, pero en los que sus creadores se han tomado ciertas libertades, como por ejemplo utilizar un algoritmo automático para generar bosques. El autor puede seguir este camino diluyendo cada vez más la correspondencia existente entre el entorno virtual y el ambiente real, utilizando por ejemplo modelos reales para crear *ambientes hipotéticos*. Siguiendo este camino, dejaríamos atrás cualquier correspondencia con ambientes reales, dando paso a los *mundos imaginarios*.

Además de su correspondencia con ambientes que existen en la realidad, podemos clasificar los mundos virtuales en base a su grado de naturalidad, esto es, la medida en que simulan las leyes físicas que rigen el mundo real y a las que el usuario está habituado. En este sentido, Sutcliffe distingue entre *entornos virtuales naturales*, *entornos naturales híbridos*, y *entornos artificiales*. Los primeros se asemejan tanto a la realidad que, idealmente, el usuario no se daría cuenta de que son sintéticos. Los entornos naturales híbridos, aunque basados en la realidad, permiten romper ciertas leyes básicas de la física, como por ejemplo la gravedad. Finalmente, los entornos artificiales guardan poca o ninguna relación con la realidad, teniendo el autor total libertad al crearlos.

### **2.3.2 Realismo para acercar la informática a más usuarios**

Esta categoría guarda un gran parecido con las interfaces de Realidad Virtual vistas en el apartado anterior, especialmente con su uso en simuladores, pues tanto en ese caso como en éste se simula la realidad, aunque ahora con un objetivo bien diferente. Si en un simulador se persigue que la experiencia del usuario pueda ser luego aplicada a la realidad, en esta ocasión se persigue la máxima transferencia del conocimiento del mundo real al virtual, de modo que el usuario pueda enfrentarse a la informática con éxito.

Si, como decía Jeff Sostein en ([VRML Email, URL], §31/3/2004), es humano entender en un primer momento un nuevo medio casi enteramente en términos del antiguo, en IBM debieron pensar que resultaría más fácil entender el nuevo medio si se percibe igual al antiguo. En este caso, el nuevo medio son los ordenadores y el antiguo la realidad. Como se afirma en las guías de diseño RealPlaces [IBM, URL], el equipo de desarrollo de IBM creía en el realismo para acercar la informática a muchos más usuarios. Al mismo tiempo, se pretendía mostrar el valor de una interfaz 3D que podría ser usada como sustituta del escritorio actual, un siguiente paso en la evolución de las interfaces gráficas de usuario.

Este uso de la interacción en tres dimensiones ha suscitado, sin embargo, cierta controversia. Así, Maxim Kononenko, de la compañía ParallelGraphics, se preguntaba por qué debía andar hasta una estantería para acceder a un texto cuando podía conseguirlo con un simple clic de ratón ([WWW-VRML, URL], §9/4/2003). [Nielsen, 1998] cree que al imitar la realidad se está haciendo un mal uso de la tecnología 3D, y en comparación explica que en el diseño Web el objetivo es ofrecer un medio que sea mejor que la propia realidad. Sin embargo, una de las contestaciones que recibió Nielsen indicaba que la principal ventaja de la tecnología 3D no es reflejar la realidad, sino precisamente poder superarla. Por esa razón, el equipo de IBM indicaba también en las guías de diseño RealPlaces que había que ser cuidadoso y no copiar las limitaciones y comportamientos no intuitivos del mundo real.

Con todo, imitar la realidad tratando al mismo tiempo de salvar sus dificultades puede no ser tan sencillo con la tecnología actual. Así, [Eastgate, 2001] advierte que algunas acciones relativamente simples en el mundo real pueden ser muy complejas en el entorno virtual con la interacción disponible. Peor aún, según ese autor muchos problemas pueden existir ya en el mundo real, pero ser agravados en el entorno virtual.

### **2.3.3 3D para superar los límites físicos de la pantalla**

Otro de los usos que se le da a la tecnología 3D es el de exprimir al máximo el espacio físico que proporciona la pantalla, y que limita la información que puede presentarse de forma simultánea en la misma. Como describe [Pittarello, 2001], las interfaces bidimensionales proporcionan barras de desplazamiento, lentes de zoom y otros controles para examinar la información que queda oculta por las limitaciones de espacio y resolución. Sin embargo, como apunta Leonard Daly ([WWW-VRML, URL], §9/4/2003), el enorme número de controles que deben proporcionar las interfaces actuales las está llevando precisamente a ir más allá de los gráficos 2D.

Gracias a los conocidos mecanismos de proyección de tres dimensiones a las dos que tiene el característico rectángulo de la pantalla, así como las técnicas de estereoscopia basadas en la multiplexación de imágenes en el espacio y en el tiempo, es posible transmitir claves monoscópicas y binoculares de profundidad que permitan aumentar el flujo de información entre el usuario y el ordenador. Ciertamente es que, con la proyección, los objetos lejanos no se muestran tan nítidos como si estuvieran en primer plano, como critica [Nielsen, 1998], pero la intuición del ser humano le ayuda a ver más de lo que realmente se muestra. Un ejemplo de ello es la presentación de información en perspectiva llamada “perspective wall” y descrita en [Robertson, 1993], y en donde lo importante es lo que parece ser, no lo que es [Assumpcao, 1991].

Otro ejemplo es el empleo de entornos 3D en el que sus elementos, como las paredes de la galería del proyecto “the TaskGallery” [Microsoft, URL], son utilizados como nuevos espacios para las tareas. En palabras de Dantzych en un mensaje enviado a la lista ([3D UI, URL], §7/10/1999), es como si contásemos con múltiples escritorios. Aunque hay también quien –como Jerry Isdale en esa discusión de correo- opina que sería mucho mejor multiplicar el número de pantallas que recurrir al uso de artefactos como los descritos en este apartado.

### **2.3.4 3D para mejorar la organización de datos y tareas**

En esta categoría se sitúan todos o casi todos los proyectos de escritorios 3D que buscan ofrecer una alternativa o una evolución sobre los actuales entornos de ventanas bidimensionales. Según Wingrave ([3D UI, URL], §30/7/2004), una de las ventajas de esos entornos 2D es que permiten más información en un mismo espacio. Sin embargo, esos mismos entornos están simulando la tercera dimensión por la forma en que distribuyen y solapan unas ventanas sobre otras, según apuntaba Nick Elmqvist –uno de los creadores del proyecto 3D Workspace Manager– en ([3D UI, URL], §4/10/1999). Para Wingrave, la tecnología 3D puede hacer eso mucho mejor. Según Fei ([3D UI, URL], §10/8/2004), contar con un espacio de trabajo 3D nos permite distribuir mejor las aplicaciones, reduciendo la saturación en la pantalla. Fei explica en su correo que la tercera dimensión puede incrementar el espacio para las aplicaciones en un factor de 3 o 4 con respecto a un escritorio 2D, y una vista de 360° alrededor del usuario lo haría en un factor de 5 o 6. Elmqvist afirma que el simple hecho de tener aplicaciones 2D en un espacio 3D permite al usuario distribuir las ventanas de una forma más intuitiva, opinión que comparte también Fei.

Según [Leftwich, 1993], una de las ventajas de una interfaz de usuario así es que posibilita al usuario desarrollar una relación espacial tridimensional con la

información con la que interactúa. Leftwich razona que, dado que vivimos en un mundo tridimensional, resulta algo simplemente natural. Una de las contestaciones a [Nielsen, 1998] apunta que cualquier información abstracta, para que tenga sentido para las personas, debe estar espacialmente referenciada. Y siguiendo este hilo argumental, Reed Hegdes explica en ([WWW-VRML, URL], §4/7/2003) que algo importante tiene lugar en nuestro cerebro cuando hablamos de entrar o salir, de estar dentro o estar fuera, que nuestra mente está muy relacionada con el espacio, con la navegación, percepción y comprensión del espacio. Para Leftwich, una interfaz de usuario tridimensional da pie a la utilización de la memoria espacial y otros medios psico-fisiológicos. Wingrave, por su parte, explica que es nuestra cognición espacial la que nos ayuda a acceder a esos datos y aplicaciones en tres dimensiones.

### 2.3.5 Visualización de información

Otro de los usos más frecuentes de la tecnología 3D es la representación de datos abstractos, usada en diagramas y gráficas. Parece lógico utilizar esta representación para conjuntos de datos que posean tres dimensiones, como por ejemplo en gráficas de tres variables, pero también con un número menor de dimensiones, añadiendo la tercera dimensión por motivos estéticos. El empleo de esta tecnología para conjuntos de datos de más dimensiones es discutible, y así [Nielsen, 1998] opina que utilizar las 3D para visualizar un espacio de 100 variables tan sólo deja 97 en lugar de 98 que restarían al utilizar una representación bidimensional.

Sin embargo, para Chris Thorne ([WWW-VRML, URL], §4/7/2003), las 3D ofrecen muchas más ventajas, pues algunas cosas son interpretadas más fácilmente en 3D. El ejemplo que expone es la representación de un modelo en tres dimensiones de un objeto real, pues su representación en vistas ortográficas 2D es más problemática para el usuario, quien tiene que hacer el esfuerzo mental para imaginar a partir de ellas dicho objeto. En este sentido, Parisi opinaba en ese intercambio de correos que la tecnología 3D es de lejos superior a las dos dimensiones para la representación de conocimiento. Pero la visualización científica –que trata con fenómenos 3D del mundo real como las tormentas o la anatomía humana- o el modelado 3D –de piezas mecánicas, obras arquitectónicas o aeronaves- es bien diferente de la visualización de información de datos abstractos multidimensionales y multivariantes [Shneiderman, 2001].

También Shneiderman indica que los abogados de cada estilo de presentación – 2D y 3D- claman la superioridad del estilo que defienden, pero que la evidencia empírica es aún insuficiente. Será por ello que el propio Nielsen reconoce que es correcto utilizar las tres dimensiones para visualizar datos abstractos, así como

objetos físicos que necesitan ser ilustrados en su forma sólida, si bien recomienda experimentar primero con vistas 2D, pues en algunas ocasiones el resultado es mejor. Uno de los factores a tener en cuenta, según Parisi, es el coste, ya que la creación de contenidos 3D es mucho más laboriosa que las ilustraciones 2D. Para Parisi, si el objetivo es mostrar una imagen estática, posiblemente la realización de un modelo 3D suponga demasiado esfuerzo para tan poco beneficio. Pero si se trata de proporcionar una experiencia 3D, la relación entre el coste y el beneficio se invertirá a favor de la tecnología 3D.

### **2.3.6 Creación y manipulación en 3D**

Además de visualizar la información en tres dimensiones desde diferentes perspectivas, un uso que suele acompañar al anterior es el de interactuar con la propia información, manipulándola en tres dimensiones. Un ejemplo es la creación y edición de modelos en tres dimensiones, un uso que las evidencias indican que no es apropiado para el estilo WIMP [Scali, 2003].

También se incluye en esta categoría la entrada de datos basada en gestos corporales, y un ejemplo ilustrativo es la interfaz de la aplicación que podía verse en la película “Minority Report” (2002), en la cual, si bien la información se presentaba en dos dimensiones, el usuario –en este caso un actor- interactúa con ella mediante gestos manuales expresados en las tres dimensiones del espacio.

### **2.3.7 Entretenimiento y diversión**

El entretenimiento es, junto con la educación, uno de los campos que el propio [Nielsen, 1998] reconoce adecuados para el empleo de las 3D. Y al hablar de esta categoría no debe pensarse sólo en los videojuegos en tres dimensiones, que podrían clasificarse dentro del uso de la tecnología 3D para interfaces de Realidad Virtual, sino también en otras aplicaciones que, entre otros objetivos, incluyen el de hacer las tareas más amenas para el usuario. Un ejemplo es el escritorio Win3D [ClockWise, URL], del cual no hace mucho se lanzó una versión orientada al público infantil. Otro ejemplo es el proyecto SUN Looking Glass, cuyos creadores afirman perseguir una experiencia más atractiva para el usuario. Y uno de los argumentos que Fei esgrime en defensa de sus proyectos Infinite-3D y Cube es, precisamente, que las 3D resultan más divertidas ([3D UI, URL], §10/8/2004).

De hecho, si bien Sandy Ressler se quejaba en ([VRML Email, URL], §23/3/2004) de que muchos de los elementos que se han probado para las

interfaces de usuario 3D –como estanterías, habitaciones o paisajes- no han funcionado, también reconocía una posible excepción, y es que los usuarios los encontraban más divertidos. Y eso se demuestra también en experimentos como el que referencia [Shneiderman, 2002], en el que se comparó el uso de la interfaz tridimensional “Data Mountain” de Microsoft frente a la tradicional interfaz bidimensional y se concluye que, aunque el tiempo que empleaban los usuarios en llevar a cabo sus acciones en 3D era un poco mayor que en 2D, a los usuarios parecía gustarles más la interfaz 3D.

Más aún, [Stuart, 2001] cuenta cómo ciertos estudios indican que el sistema Apple Lisa no era realmente una revolución en la facilidad de uso y aprendizaje, y aún así ese tipo de interfaces tuvo un gran éxito, por lo que se sugiere entonces que la gente encontraba su uso más entretenido. Según Stuart, existen razones para creer que los entornos virtuales bien diseñados pueden ser intrínsecamente motivadores y entretenidos.

### 2.3.8 Interfaces para Realidad Aumentada

La tecnología 3D también se utiliza para crear interfaces de usuario para Realidad Aumentada. A diferencia de la Realidad Virtual, en donde el usuario se sumerge en un mundo sintético que reemplaza completamente al real, el paradigma de la Realidad Aumentada no persigue reemplazar sino complementar al real, permitiendo al usuario ver el mundo real con objetos virtuales superpuestos o en composición con él. El usuario, por tanto, tiene una visión de la realidad "aumentada", al percibir los objetos virtuales coexistiendo en el mismo espacio tridimensional con los reales.

A decir verdad, la diferencia entre Realidad Virtual y Aumentada no es tan clara. Así, la dificultad de proporcionar retorno táctil y de fuerzas en un mundo virtual se puede solventar con la introducción en el sistema de objetos tangibles, lo que nos aproxima a la Realidad Aumentada [Sutcliffe, 2003]. En este sentido, el paso gradual desde los mundos virtuales hasta el mundo real fue definido ya por [Milgram, 1994] como un continuo *realidad-virtualidad*, entre cuyos extremos encontramos la Realidad Aumentada (AR, *Augmented Reality*), donde el mundo real predomina sobre el virtual, pero también la Virtualidad Aumentada (AV, *Augmented Virtuality*), donde el mundo virtual predomina sobre el real. A todo ese espacio intermedio, Milgram y Kishino lo llamaron Realidad Mixta (MR, *Mixed Reality*). La Figura 2.1 ilustra este continuo.



Figura 2.1 El continuo realidad-virtualidad de Milgram y Kishino

## 2.4 Un nuevo espacio de diseño

Aunque las interfaces de usuario 3D puedan ofrecer muchas de las características que se esperan en una interfaz de usuario post-WIMP, no parece que con ello la tecnología 3D vaya por sí sola a impulsar un nuevo periodo en la historia de las interfaces de usuario. Una previsión conservadora, como la que expresa Doug Bowman en la lista de correo ([3D UI, URL], §22/7/2004), es que el escritorio seguirá siendo bidimensional, con usos puntuales de las 3D. Más que eso, el futuro parece que estará caracterizado no por un único paradigma sino por una mezcla de diferentes tecnologías, a la que habría que añadir la Realidad Mixta, la computación ubicua o la visión por computador, un crisol o “melting pot” según palabras de Wingrave en ese mismo intercambio de correos en la lista 3D UI y en [Wingrave, 2005].

Siguiendo esa misma corriente de opinión, en esta Tesis se apuesta por un espacio de diseño de interfaces de usuario en el que conviven las dos y las tres dimensiones, y que se explica en los siguientes dos apartados.

### 2.4.1 Interacción 2D en entornos 3D

Uno de los lectores que respondía a las críticas hacia las 3D de [Nielsen, 1998] argumentaba que había que reconocer que uno no trabaja sólo en 2D. Pero habría que añadir que, igualmente, uno tampoco trabaja siempre en 3D y, como se apunta en [Herndon, 1994], sumar grados de libertad a tareas que no las necesitan puede ser la causa de una complejidad indeseada. De hecho, según Herbet Stocker ([WWW-VRML, URL], §10/11/2002), si se observan bien las interfaces de usuario de las máquinas que nos rodean todas ellas son 2D, disponen sus elementos lado a lado, excepto aquellas que han sido diseñadas para ajustarse a nuestro cuerpo.

En este sentido, los mundos virtuales no son muy diferentes de la realidad. Así, en [Bowman, 2001] se apunta que un error común en el diseño de interfaces de usuario 3D es creer que sólo debe utilizarse interacción 3D. [Larimer, 2003] explica cómo la mayoría de los entornos virtuales proporcionan una interfaz de usuario 3D que es apropiada, por ejemplo, para navegación y manipulación de objetos, pero existen otras tareas dentro del entorno virtual que requieren interacción en una o dos dimensiones. Por ello, no puede concebirse que la tecnología 3D vaya a excluir la interacción en dos dimensiones, sino que incluso –como comenta Miriam English en ([WWW-VRML, URL], §24/3/2003)- debe soportarla muy bien.

En [Molina, 2005a] se enumeran las diferentes razones por las que se debe facilitar la interacción 2D en entornos 3D, enfocadas al soporte de interfaces de usuario 2D en esos entornos, y de las cuales cabe destacar la manipulación indirecta y el legado de aplicaciones 2D. Así, si bien los entornos virtuales suelen ser considerados como una extensión del paradigma de la manipulación directa de las dos a las tres dimensiones, resulta difícil construir toda una interfaz basada únicamente en la manipulación directa [Foley, 1996]. El hecho de que la manipulación directa no siempre es obvia se comenta también en [Cuppens, 2004], y como consecuencia se apunta que a veces es necesario recurrir a la manipulación indirecta. Una forma de hacerlo consiste en utilizar menús y diálogos como los de las aplicaciones gráficas de usuario 2D, “aumentando” el entorno virtual [Larimer, 2003], y obteniendo así una interfaz de usuario híbrida 2D/3D [Cuppens, 2004].

La otra razón es que los entornos 3D puedan acoger el enorme legado de aplicaciones 2D, facilitando así la reutilización de esas aplicaciones en esos entornos. Este es un objetivo que está presente en el desarrollo de muchos entornos 3D, principalmente en escritorios 3D como –por ejemplo- 3D Window Manager [3D WM, URL] o the TaskGallery [Microsoft, URL], como sus propios creadores reconocían en discusiones públicas en la lista [3D UI, URL]. En el caso particular de RealPlaces –las guías de diseño desarrolladas por IBM para nuevos entornos 3D [IBM, URL]-, se introducía la idea de utilizar una versión virtual de una computadora en el nuevo espacio tridimensional con el fin de que se mostraran en ella esas aplicaciones 2D. Con ello también se persigue que el usuario pueda utilizar las aplicaciones a las que está acostumbrado mientras se hace al uso de nuevas interfaces de usuario 3D, favoreciendo la evolución hacia la tercera dimensión.

De nuevo, más que una revolución en las interfaces de usuario que relegue las interfaces de usuario 2D a un segundo plano en beneficio de las 3D, ambas deben coexistir. El fin debe ser que el usuario realice sus tareas en el espacio adecuado, facilitando la transición de uno a otro de forma ininterrumpida (“seamless”, según [Forsberg, 1997]).

## 2.4.2 Extendiendo el continuo de interfaces de usuario

Como se ha visto en el apartado 2.3.8, el continuo descrito por Milgram y Kishino (Figura 2.1) distribuye las interfaces de usuario a lo largo de un único eje desde el mundo real a los mundos virtuales, pasando por dos formas de Realidad Mixta: Realidad Aumentada y Virtualidad Aumentada. Basado en él, se propone aquí un nuevo continuo con el que se persigue ofrecer una visión más amplia de las interfaces de usuario, considerando no sólo interfaces de usuario 3D, sino también interfaces gráficas 2D e interfaces basadas en una línea de comandos o instrucciones, esto es, interfaces de usuario 1D.

En este nuevo continuo, las interfaces se distribuyen a lo largo de un eje en orden creciente de dimensiones, de una a tres, recordando en parte esa evolución de la que hablaba Fei y que se comentó en el apartado 2.1. De esta forma, el continuo deja de ser realidad-virtualidad para representar ahora uno más amplio, el *digital-virtual-real*, denotando con la palabra “digital” el mundo de dígitos binarios con los que operan los ordenadores y que es opuesto al mundo real en el que se desenvuelven los seres humanos. Además, de la misma forma que existe un espacio intermedio entre lo real y lo virtual llamado Realidad Mixta, también existe otro entre lo digital y lo virtual bautizado aquí como Virtualidad Mixta.

Más aún, este nuevo continuo digital-virtual-real añade un segundo eje que atiende al grado de inmersión que, comparado con la realidad, experimenta el usuario en el espacio mostrado por la interfaz, según la siguiente escala: *cero, bajo, medio, alto*. Así, con una interfaz clasificada entre los dos primeros valores, el usuario no tiene sensación de inmersión o esta es muy pobre. Entre bajo y medio, el usuario observa el espacio tridimensional a través de una ventana, posiblemente manipulando los objetos en ese espacio siguiendo el modelo de TV del que hablaba van Dam, y también comentado en el apartado 2.1. Entre medio y alto, el usuario se siente sumergido en el espacio tridimensional. El grado alto corresponde a la realidad misma.

Una primera aproximación a este nuevo continuo se describe en [Molina, 2005a]. Igualmente basado en el continuo descrito por Milgram y Kishino, se sustituía el extremo que representaba la Realidad Virtual por la siguiente sucesión de interfaces de usuario: interfaz gráfica 3D virtual, interfaz gráfica 3D digital, dibujo en 3D de una interfaz gráfica 2D, e interfaz gráfica 2D. Añadía también un segundo eje en el que se distinguía entre dos grados de inmersión: bajo y alto. Con el nuevo continuo se persigue entonces llevar más lejos que en esa aproximación inicial, aplicando leves correcciones sobre el mismo pero, sobre todo, ampliando aún más su alcance.

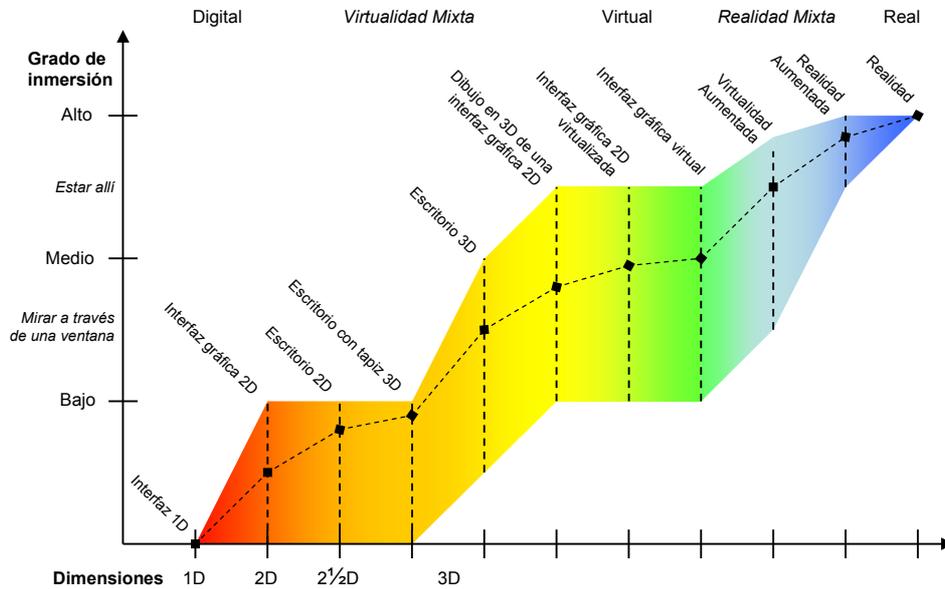


Figura 2.2 Continuo digital-virtual-real de interfaces de usuario

La Figura 2.2 ilustra este nuevo continuo, representando el eje horizontal el número de dimensiones, y el vertical el grado de inmersión. Cada segmento vertical discontinuo representa el rango de diferentes grados de inmersión que un determinado tipo de interfaz puede ofrecer, siendo el grado de inmersión típico marcado con un cuadrado. La unión de unos segmentos con otros dibuja un contorno que representa el espacio de diseño de las interfaces de usuario. La unión de los grados de inmersión típicos de las interfaces muestra la evolución típica de la experiencia del usuario a lo largo del continuo. La gráfica resultante es creciente y se caracteriza por tres tramos cóncavos y dos puntos de inflexión. A continuación se describirán una a una las interfaces indicadas en la figura, de menos a más dimensiones, de lo digital a lo real, y se justificará el anterior resultado:

- **Interfaz de usuario 1D:** Basada en una línea de comandos o instrucciones, normalmente un flujo de caracteres en una terminal de texto. Presenta una única dimensión, la que representa la sucesión de las instrucciones, sus parámetros y la respuesta del ordenador. El grado de inmersión es cero pues la interfaz no presenta imagen alguna de espacio en el que el usuario pueda verse sumergido por sus sentidos, si bien es cierto que, a partir de una descripción textual y usando su propia imaginación, el usuario puede hacerse una idea de un espacio, como

ocurre en los juegos MUD (*Multi User Dungeons*) o juegos online no gráficos.

- **Interfaz gráfica 2D:** Los elementos de esta interfaz se distribuyen a lo largo de una superficie bidimensional, típicamente una pantalla o monitor de ordenador. El grado de inmersión es bajo, ya que el usuario considera la superficie como una más de las que le rodean en el espacio real, más que un espacio en el que sumergirse. Sin embargo, estas interfaces suelen exhibir ciertas claves tridimensionales, en particular sombras, y aunque sus elementos no son más que bitmaps, el uso de estas claves supone un tímido pero claro acercamiento a las interfaces 3D, más acusado en el siguiente tipo de interfaz.
- **Interfaz de escritorio 2D:** Los entornos de ventanas exhiben diferentes tipos de elementos 2D, llamados widgets, sobre los que actúa el usuario superponiendo sobre ellos otro elemento, el cursor, que es desplazado por el usuario por la superficie de la pantalla al mover sobre la superficie de su mesa el ratón. Sin embargo, de nuevo el empleo de ciertas claves tridimensionales acercan estas interfaces a las 3D, más que por el empleo de sombras, por la continua simulación de superposiciones en el espacio, no sólo del cursor sobre los widgets, sino especialmente de unas ventanas sobre otras, como ya se comentó en los apartados 2.2 y 2.3.4. Por esta razón, se clasifica estas interfaces bajo el tipo 2.5D (dos dimensiones y media), si bien el grado de inmersión es aún bajo.
- **Escritorio con tapiz 3D:** El hecho de que el entorno de ventanas se adorne con un tapiz o fondo de escritorio representando un espacio tridimensional –por ejemplo, una fotografía de un paisaje– no cambia que los widgets, y la interacción con ellos, continúen pegados a las dos dimensiones de la pantalla. Sin embargo, es indudable que dicho tapiz 3D amplía la dimensión de profundidad del espacio de trabajo del usuario, por lo que, aunque el grado de inmersión siga siendo bajo, está más cerca del siguiente tipo de interfaz, la interfaz de escritorio 3D. De hecho, bien podría pasar como una instantánea de una interfaz de escritorio 3D, aunque cuando las ventanas ocultan el fondo sólo queda la interfaz gráfica 2D que es.
- **Interfaz de escritorio 3D:** Con este nombre se alude a interfaces como las que ofrecen los escritorios 3DNA [3DNA, URL] o Win3D [ClockWise, URL], las cuales se integran en entornos de ventanas existentes añadiendo a los mismos un espacio tridimensional por el que el usuario puede navegar, y a lo ancho y largo de ese espacio se distribuyen representaciones de los objetos con los que puede actuar,

como aplicaciones para ejecutar y documentos para abrir. Este comportamiento sí se corresponde con una interacción en tres dimensiones, por lo que ahora sí puede hablarse con propiedad de interfaz 3D. El grado de inmersión en este caso crece hasta un nivel medio, pues no deja de ser una interfaz de escritorio y el usuario, por tanto, experimenta el mundo 3D a través de la ventana que es la pantalla de su ordenador. De hecho, las ventanas y sus elementos son los propios de los entornos que extienden, tan sólo aparecen en el plano de la pantalla superpuestos al espacio que la interfaz dibuja tras ellos, y en el caso de llegar a ocultarlo del todo, de nuevo sólo quedaría la interfaz gráfica 2D. Por esta razón, se ha considerado que el grado de inmersión puede variar en este caso entre los valores cero-bajo y medio.

- **Dibujo en 3D de una interfaz gráfica 2D:** La diferencia entre el fondo tridimensional y el plano de la pantalla sobre el que se dibujan las ventanas se diluye en esta nueva interfaz, compartiendo los dos el mismo espacio, de forma tal que las ventanas pueden ser manipuladas en las tres dimensiones. Más allá del redimensionamiento y el desplazamiento con respecto a los márgenes de la pantalla, las ventanas pueden ahora alejarse y acercarse, incluso rotarse. Eso sí, las ventanas se representan como un bitmap dibujado sobre un polígono en el espacio tridimensional, una superficie plana en la que los widgets no tienen profundidad en este espacio aunque las sombras que les dibujen hagan creer al usuario lo contrario. Un ejemplo es el escritorio del proyecto Looking Glass [Sun, URL], si bien también se utiliza en entornos virtuales inmersivos, por lo que se ha considerado un grado de inmersión entre bajo y medio-alto.
- **Interfaz gráfica 2D virtualizada:** Si los elementos de las ventanas eran dibujados en las anteriores interfaces con primitivas 2D –lo que se conoce como gráficos *raster*–, este nuevo hito en el continuo destaca el uso de objetos 3D –típicamente polígonos adornados con texturas– para representar y dar profundidad a los widgets, transformándolos en lo que en aquí se convendrá en llamar widgets 3D. Una interfaz gráfica 2D así transformada es bautizada en [Molina, 2005a] con el nombre de interfaz de usuario virtualizada (VUI, *Virtualized User Interface*). El grado de inmersión varía igual que en la interfaz anterior, aunque típicamente esta nueva interfaz producirá en el usuario un grado sensiblemente mayor, al percibir este el relieve de los elementos con los que interacciona.
- **Interfaz gráfica virtual:** Es la interfaz presente en muchos entornos virtuales, en los cuales el usuario interacciona con objetos virtuales que, en especial en entornos naturales –siguiendo la clasificación de Sutcliffe

comentada en el apartado 2.3.1-, son imitación de objetos reales. Otros elementos artificiales también pueden estar presentes, típicamente para dar soporte a la interacción. También interfaces gráficas 2D, virtualizadas o no. El máximo grado de inmersión se mantiene aquí en medio-alto, pues aunque el usuario pueda experimentar un fuerte sentimiento de presencia, se entiende que las limitaciones actuales de la tecnología de Realidad Virtual impiden aún equipararse con la propia realidad.

- **Virtualidad Aumentada:** En este punto del continuo, este enlaza con el de Milgram y Kishino y su concepto de Realidad Mixta. Para superar las anteriores limitaciones de la tecnología, en especial la estimulación del sentido del tacto, en muchos otros entornos virtuales se complementa la interfaz gráfica virtual con objetos reales que el usuario puede tocar y así sentir su peso y/o tacto. Con ello, el grado de inmersión aumenta, aunque siguen existiendo limitaciones que mantienen la distancia con la realidad.
- **Realidad Aumentada:** La introducción de objetos tangibles nos encamina, como apuntaba Sutcliffe y así se comentó en el apartado 2.3.8, hacia la Realidad Aumentada. La diferencia entre Virtualidad y Realidad Aumentada se encuentra en la proporción de mundo virtual y real que se mezcla en la interfaz. En esta segunda forma de Realidad Mixta, los elementos virtuales “aumentan” el mundo real, el cual es el medio dominante y, por ello, el grado típico de inmersión se aproxima a la cota más alta.
- **Realidad:** La realidad misma es el referente en este continuo digital-virtual-real del máximo grado de inmersión, y así se muestra en la figura. Lógicamente, aquí no cabe hablar de barreras tecnológicas que limiten la inmersión del usuario, pero no se debe caer en el error de pensar por ello que la interacción en el mundo real es la ideal, pues también presenta problemas que pueden agravarse al ser imitadas en las interfaces persona-ordenador, como advertía Eastgate en el caso de los entornos virtuales (véase apartado 2.3.2).

Tomada entonces buena cuenta de las diferentes interfaces de usuario que pueden encontrarse a lo largo del continuo aquí propuesto, los siguientes apartados abordarán la problemática del diseño de las interfaces que son el objeto de esta Tesis, las interfaces de usuario 3D.

## 2.5 Elementos que integran las interfaces de usuario 3D

La interfaz de usuario tiene dos funciones: revelar el estado interno de la aplicación al usuario y cambiar ese estado interno siguiendo las instrucciones del usuario [Assumpcao, 1991]. Con ese doble propósito, diferentes elementos se suman en la interfaz. En el caso de los sistemas de Realidad Virtual, y según [Marsh, 1998], los componentes principales de todos ellos son tres: imaginaria, interacción y comportamiento. Para [Eastgate, 2001], son cuatro las piezas con las que se crean los entornos virtuales: topografía, objetos virtuales, comportamiento y puntos de vista. En los siguientes apartados se describirán, como elementos que forman las interfaces de usuario 3D, los cuatro siguientes: espacio 3D, objetos, comportamiento e interacción.

### 2.5.1 Espacio 3D

En el mundo virtual no sólo los objetos dan forma al espacio. En principio, ese espacio virtual es infinito, y el usuario puede moverse libremente por sus tres dimensiones, atravesando cualesquiera objetos que encuentre a su paso, como un fantasma. El tipo de aplicación sugerirá qué similitudes debe guardar el espacio virtual con el espacio real, y la tecnología determinará cuáles pueden conseguirse. En este sentido, [Eastgate, 2001] diferencia entre topografías *naturalistas*, en las que se reproducen las características del mundo real, y topografías *innaturales*, como portales o tele-transportes que permiten conectar partes distantes del entorno o estructuras tipo “tardis” (vocablo tomado de la serie de TV “Dr. Who” y que se utiliza para describir una construcción que es más grande en su interior de lo que aparenta desde el exterior). La Realidad Mixta, por ejemplo, puede predisponer el tipo de topografía a una naturalista.

El espacio puede estar delimitado por objetos que, a modo de obstáculo, impiden al usuario desplazarse más allá de ese punto. Estos lindes también pueden ser invisibles, y pueden servir para que el usuario no se aleje del espacio construido o trate de acercarse a imágenes situadas alrededor de ese espacio como decorados que simulan paisajes de fondo. Caminos, señales y objetos de referencia (*landmarks*) guían al usuario en su desplazamiento por el espacio. Diferentes estructuras ayudan a diferenciar zonas dentro del espacio, cada una de las cuales puede tener asociado un significado, creando espacios temáticos (como en el escritorio Win3D [ClockWise, URL]), asociando tareas a esos lugares (“3D rooms” en [Robertson, 1993], “places” en la guía RealPlaces [IBM, URL]), o contenido multimedia (“interaction locus” en [Pittarello, 2001]).

En espacios compartidos con múltiples participantes, esas zonas también pueden clasificarse en públicas y privadas [McIntosh, 2000]. Precisamente, en los entornos colaborativos también se tienen en cuenta el espacio personal de cada usuario, distinguiéndose tres espacios: aura –volumen en el que el usuario puede interactuar con otros objetos-, focus –volumen en el que un objeto puede percibir eventos-, y nimbus –volumen en el que los eventos del objeto pueden ser percibidos por otros- [Carlsson, 1993].

Referenciar los objetos al cuerpo del usuario puede servir para ubicar herramientas que pueda necesitar [Pierce, 1999], o situar alrededor del mismo objetos con los que interactúa, como por ejemplo ventanas de aplicación en una esfera centrada en el usuario (en InfoSpace [Leftwich, 1993], VEWL [Larimer, 2003] o en el escritorio 3D Sphere XP [SphereXP, URL]) o un cilindro (en el escritorio 3D Cube [Fei, URL2]).

### 2.5.2 Objetos físicos y virtuales

En el espacio 3D, los objetos parecen dispuestos jerárquicamente siguiendo una relación de continente-contenido o todo-parte, y de hecho los modelos de la escena suelen organizarse en forma de grafo, aunque ese grafo puede describir una jerarquía diferente a la que se muestra a la vista [Huda, URL1]. En el apartado anterior se han introducido ya algunos tipos de objetos que dan forma a ese espacio, como lindes, caminos, señales, referencias y otras estructuras, pero hay más tipos de objetos, aunque las clasificaciones varían de un autor a otro.

En primer lugar, puede distinguirse entre objetos reales y virtuales [Tanriverdi, 2001]. Los primeros no hay que modelarlos, pues forman parte del mundo real que se mezcla con el virtual. Los segundos pueden ser modelos de objetos reales, abstracciones de objetos reales u objetos imaginarios [Marsh, 1998].

Una segunda clasificación distingue entre objetos de la aplicación y objetos que permiten interactuar con los anteriores, llamados widgets [Conner, 1992] u objetos de interacción [Sastry, 2001]. Esta distinción se discutirá más adelante al hablar de tareas y técnicas de interacción.

En [Eastgate, 2001] se describen dos clasificaciones de objetos, recogidas en la Tabla 2.6. La primera de ellas es utilizada por el autor en la etapa de adquisición de recursos del mundo real para la creación del mundo virtual. En esta primera clasificación, Eastgate distingue entre cuatro tipos de objetos: fondo (*background*), contexto, características visuales primarias –sólo precisan ser visualmente correctos- y características funcionales primarias –deben ser correctos visual y funcionalmente-.

Clases de objetos para adquisición de recursos	Clases de objetos para reducción de polígonos
<ul style="list-style-type: none"> <li>• Fondo</li> <li>• Contexto</li> <li>• Características visuales primarias</li> <li>• Características funcionales primarias</li> </ul>	<ul style="list-style-type: none"> <li>• Referencias</li> <li>• Lindes</li> <li>• Dispositivos</li> <li>• Características que ayudan a reconocer</li> <li>• Claves</li> <li>• Informadores de tamaño</li> <li>• Sendas</li> <li>• Ayudas</li> </ul>

Tabla 2.6 Clases de objetos según Eastgate

Con la segunda clasificación, Eastgate persigue identificar qué objetos pueden reducirse en el número de polígonos que los forman, o pueden ser eliminados del mundo virtual, con el fin de obtener una mayor velocidad de generación de imágenes. Ocho categorías forman esta clasificación: referencias (*landmarks*), lindes, dispositivos –cualquier objeto que ofrezca interacción, por ejemplo una herramienta-, características que ayudan a reconocer, claves –en inglés “cues”, indican qué puede hacer el usuario y cómo-, informadores de tamaño, sendas y ayudas –en inglés “aids”, se introducen en el mundo virtual para proporcionar información no implícita en la geometría-. La diferencia entre las claves y las ayudas, según Eastgate, es que los segundos son añadidos, no tienen correspondencia en la realidad.

Otra clasificación de objetos es la que se expone en [Fencott, 2001] al explicar su Modelo de Oportunidades de Percepción, un conjunto de categorías que especifican el objetivo que se persigue con el objeto, en términos de simulación (denotación) –por ejemplo, una silla es una silla- y comunicación simbólica (connotación) –por seguir el ejemplo anterior, una silla también puede ser útil para alcanzar objetos en alto-. Estas categorías están organizadas en forma de árbol, y pueden verse en la Figura 2.3.

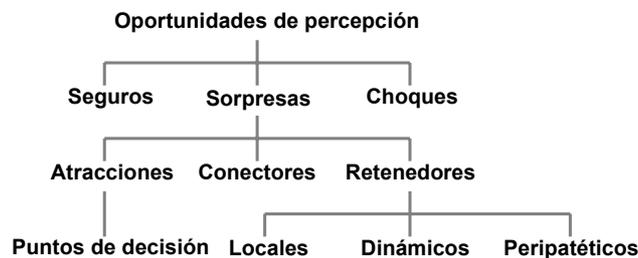


Figura 2.3 Categorías del modelo de Fencott e Isdale

En cuanto a las propiedades de los objetos, Eastgate cita como atributos básicos la forma, el tamaño y la apariencia. Comenta además que los objetos pueden estar formados por varios objetos hijo, dando forma a ese grafo de la escena del que antes se ha hablado. También describe otras propiedades que definen su comportamiento, pero serán abordadas en el siguiente apartado.

### 2.5.3 Comportamiento

Siguiendo con Eastgate, este autor explica que los objetos pueden también tener un determinado dinamismo y comportamiento, y que clasifica en: interactivo, autónomo, conectado y sin comportamiento. Estas distinciones también pueden servir para clasificar los objetos. En este sentido, la guía de diseño RealPlaces [IBM, URL] describe un modelo jerárquico de objetos que, en primer lugar, distingue entre los objetos que exhiben comportamiento y los que no. [Sutcliffe, 2003], por su parte, apunta cuatro clases de objetos al comentar el análisis del dominio de una aplicación concreta (un parque de negocios): estructuras principales, objetos pasivos, objetos activos –responden a eventos- y agentes – toman la iniciativa-. Ambas clasificaciones son recogidas en la Tabla 2.7.

Modelo de objetos de RealPlaces [IBM, URL]	Clases de objetos según [Sutcliffe, 2003]
<ul style="list-style-type: none"> <li>• Ambientales</li> <li>• Con comportamiento               <ul style="list-style-type: none"> <li>○ Decorativos</li> <li>○ Objetos que soportan las tareas                   <ul style="list-style-type: none"> <li>▪ Objetos de las tareas</li> <li>▪ Herramientas                       <ul style="list-style-type: none"> <li>• Basados en apuntador</li> <li>• Dispositivos</li> </ul> </li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Estructuras principales</li> <li>• Objetos pasivos</li> <li>• Objetos activos</li> <li>• Agentes</li> </ul>

**Tabla 2.7** Clasificaciones de objetos en base a su comportamiento

De nuevo, Eastgate explica que el dinamismo y el comportamiento de los objetos se manifiesta en cambios de: posición, orientación, tamaño, forma, color, sonido, función o propiedades. La exhibición de propiedades *físicas*, como masa y la velocidad, está supeditada a la simulación de las leyes físicas en el entorno virtual. En muchos casos no se hace, lo que lleva a que –por ejemplo- los objetos puedan ser atravesados por el usuario o que los objetos no caigan al soltarlos en caída libre, exhibiendo así propiedades *innaturales*. Como apunta Jeff Pierce ([3D UI, URL], §3/8/1999) no es sólo ausencia de gravedad lo que se suele observar en los mundos virtuales, sino también de inercia, los objetos no

continúan en movimiento al lanzarlos. Otra propiedad innatural que apunta Eastgate es la invisibilidad, una propiedad que también podría clasificarse como “mágica”.

Precisamente, [Tanriverdi, 2001] clasifica los comportamientos de los objetos según dos tipos: comportamientos físicos y comportamientos mágicos. El primer tipo hace referencia a aquellos cambios que son observables en el mundo real y, por esta razón, son más fáciles de comunicar a otros miembros del equipo sin describir todos los detalles. En cambio, el segundo tipo representa aquellos comportamientos raramente o nunca vistos en el mundo real. El concepto de “magia”, como se verá en el siguiente apartado, está muy presente en los mundos virtuales.

#### 2.5.4 Interacción

Por interacción debe entenderse no sólo el hecho de que el usuario pueda manipular los objetos 3D en el entorno, sino también que pueda navegar por ese entorno y observar los objetos desde diferentes puntos de vista.

De hecho, la navegación no es sino el control del usuario sobre su representación (“the presence” o “the self”, en palabras de [Sutcliffe, 2003]) en el mundo virtual, lo que en entornos multiusuario se conoce como “avatar”. No obstante, también podría considerarse control sobre el mundo si se considera que es este el que se mueve en relación al avatar. En cualquier caso, y siendo precisos, debe hablarse de viajar y no de navegar, pues según [Bowman, 2001] el concepto de navegación incluye un componente motor –viajar o “travel”- y otro cognitivo –decidir el camino o “wayfinding”-.

Más aún, lo que permiten los mundos virtuales es un control del punto de vista más poderoso que en la realidad. Así, el usuario puede observar el mundo virtual a través de los ojos de su avatar (vista *egocéntrica* según Eastgate, *endógena* según Sutcliffe), pero también desde fuera de ese cuerpo virtual (vista *exocéntrica* según Eastgate, *exógena* según Sutcliffe), pudiendo observarse a sí mismo –por ejemplo- a vista de pájaro, lo que también se conoce como vista del ojo de Dios. Igualmente, el mundo virtual hace posible que el usuario viaje como no lo puede hacer en la realidad, flotando o atravesando muros como fantasmas (lo que Eastgate llama “ghost mode”).

La interacción descrita en el párrafo anterior podría describirse también como magia o superpoderes. Precisamente, [Kaur, 1998] explica que la interacción en los entornos virtuales difiere del mundo real pues habrá *limitaciones*, y serán necesarias ciertas *sustituciones*, pero también por otorgar nuevos *poderes*. Pero

no siempre es apropiado utilizar “técnicas mágicas”. Como dice Pierce ([3D UI, URL], §23/11/2001), habrá situaciones en las que emular el mundo real es más conveniente. Un ejemplo son los simuladores, en donde las habilidades que se entrenan deben luego ponerse en práctica en el mundo real. En este sentido, habría que distinguir entre mundos virtuales realistas y mundos virtuales mágicos (Matt Conway, [3D UI, URL], §18/5/1999).

Lo aquí escrito no es más que una pequeña introducción a la interacción como uno de los cuatro elementos que forman las interfaces de usuario tridimensionales. Por su importancia, se le dedica a continuación un apartado aparte en el que se examinan, a su vez, los elementos de la propia interacción.

## 2.6 Elementos básicos de la interacción

Según [Foley, 1996], los elementos básicos de las interfaces de usuario son: diálogos interactivos, tareas de interacción, técnicas de interacción y acciones con dispositivos de entrada. Comparado con la lista de elementos dada en la anterior sección, se echan en falta otros elementos que también se ha visto que integran las interfaces de usuario 3D. Sin embargo, resulta muy apropiada para estudiar los detalles de la interacción, y por ello se utilizará como guía para los siguientes apartados.

### 2.6.1 Diálogos interactivos

Para explicar el significado de los elementos básicos, en [Foley, 1996] se recurre a una analogía con el lenguaje natural –véase Tabla 2.8-, haciendo corresponder los diálogos interactivos (*interactive dialogues*) con frases, y así se indica que, de la misma forma que las frases se construyen a partir de palabras, los diálogos se componen de secuencias de *tareas de interacción*.

Sin embargo, como se verá a continuación, las tareas de interacción son como el significado de las palabras, y cada una de las diferentes palabras que se pueden usar para expresar un mismo significado es una técnica de interacción. Por ello, sería más correcto decir que al igual que las frases se construyen a partir de palabras, cada una expresando un significado, los diálogos se componen de secuencias de *técnicas de interacción*, cada una implementando una tarea de interacción.

Elementos de la interacción	Elementos del lenguaje natural
Diálogos interactivos	Frases
Tareas de interacción	Significado de las palabras
Técnicas de interacción	Palabras (el léxico)
Acciones simples con dispositivos de entrada	Letras

**Tabla 2.8** Analogía de Foley *et al.* entre elementos de la interacción y del lenguaje natural

### 2.6.2 Tareas de interacción

De nuevo según [Foley, 1996], las tareas de interacción clasifican los tipos fundamentales de información que son introducidas con las técnicas de interacción. Como se ha adelantado en el apartado anterior, puede establecerse una correspondencia entre las tareas de interacción y el significado de las palabras.

Haciendo uso de esa analogía, podría decirse también que las tareas de interacción complejas se descomponen en tareas más básicas de igual forma que el sentido de una frase viene dado a partir de los significados individuales de las palabras que la forman. Foley *et al.* identifican tres Tareas de Interacción Compuestas (CITs, *Composite Interaction Tasks*): cajas de diálogo (para especificar múltiples entradas de información), construcción (para crear objetos que requieran dos o más posiciones) y manipulación (para cambiar la forma de un objeto, arrastrarlo, rotarlo o escalarlo).

Los átomos que se combinan para dar lugar a esas moléculas son las Tareas de Interacción Básicas (BITs, *Basic Interaction Tasks*), cada una de las cuales identifica una unidad de información que el usuario puede introducir y que tiene significado en el contexto de la aplicación. Para Foley *et al.*, estas tareas básicas son cuatro: situar (una posición), seleccionar (un objeto), texto (una cadena) y cuantificar (un valor numérico).

En relación a las tareas de interacción en el espacio 3D, Foley *et al.* apuntan que las tareas de situar y seleccionar se vuelven más complicadas, añaden una tarea de interacción adicional –rotación 3D–, y señalan que con frecuencia se hace necesaria la combinación de varias tareas de interacción 3D.

[Herndon, 1994] ya apuntaba que, aunque es reconocido que cada aplicación impone diferentes requisitos al diseño de su interfaz de usuario, existe un

número de tareas de interacción que pueden ser consideradas universales en las aplicaciones gráficas 3D. Precisamente, la viabilidad y utilidad de identificar esas tareas “universales” es un tema muy debatido en la lista de correo [3D UI, URL] (véase p. ej. agosto de 2004). En esa discusión, Pablo Figueroa opinaba que era importante encontrar un conjunto común de tareas que termine por convertirse en la base de los *toolkits* (literalmente “cajas de herramientas”, suelen ser librerías de programación). Sin embargo, la lista varía según el autor.

Por su parte, Herndon *et al.* identifican seis tareas universales: control del punto de vista, percepción, creación de objetos o definición de modelos, selección de objetos, colocación y edición de objetos, y definición de comportamientos o relaciones entre objetos.

En [Bowman, 1999] también se señala que muchas de las tareas de interacción en entornos virtuales pueden encuadrarse en una de las categorías que enumeran, y que describen como tareas universales. En este caso son tres: control del punto de vista o viajar, selección y manipulación. Además, en la taxonomía de selección/manipulación dada en esa publicación aparece otra tarea más: liberar objetos. Y en cuanto a la manipulación en particular, en [Bowman, 2001] se citan dos tareas básicas: colocación de objetos y rotación de objetos. En ambos trabajos se nombra también la tarea de control del sistema, la cual no suele incluirse entre las tareas universales pues se entiende que puede ser llevada a cabo mediante tareas de selección y/o manipulación. El propio Bowman comentaba en la lista de correo ([3D UI, URL], §16/8/2004) que otros autores podrían también incluir la tarea de entrada simbólica (introducir y editar letras, números y símbolos) y la tarea de modelado (crear y dar forma a objetos 3D).

Continuando con [Bowman, 1999], sus autores dividen a su vez las anteriores tareas en subtareas. Por ejemplo, la tarea selección agrupa las siguientes subtareas: indicación del objeto, indicación de selección y realimentación (*feedback*). Los propios autores señalan que es posible que algunas subtareas no estén directamente relacionadas con la tarea del usuario, como en este caso el *feedback* del sistema.

[Boyd, 1999], al hablar del desarrollo de un toolkit en el marco del proyecto [INQUISITIVE, URL], identifica cuatro tareas básicas de interacción: navegación, selección, manipulación y entrada de datos. También distinguen entre tareas de más alto nivel y tareas básicas, indicando que las primeras pueden interpretarse como una combinación de las segundas.

[Barrilleaux, 2001] enumera tres componentes que coinciden con algunas de las tareas de interacción expuestas hasta ahora: navegación, manipulación y acceso. La última tarea alude al acceso del usuario a la escena, dejándole que decida qué modelos deben incluirse en el mundo y cuáles extraer de él. Esos tres

componentes se basan a su vez en otros tres, que tratan los aspectos más fundamentales del diálogo entre el usuario y la aplicación: control (interpreta las entradas del usuario como acciones con significado en la aplicación), realimentación (permite al diseñador de la aplicación guiar al usuario) y visualización (permite al usuario observar la escena y los resultados de las acciones). Estas componentes básicas podrían asemejarse, por ejemplo, a las subtareas que describen Bowman y Hodges.

Por último, [Sutcliffe, 2003] afirma que los diálogos en Realidad Virtual no son diálogos como tales, sino la integración de los siguientes cuatro componentes: movimiento y navegación, manipulación de objetos e interacción con ellos, conversación con otros agentes, y características que no son de Realidad Virtual. Este último componente está relacionado con la introducción de datos y el estilo de interacción GUI. En relación a la manipulación de objetos y la interacción con los mismos, este autor también habla de seleccionar, indicando que puede integrarse con consultar, y cita también acciones como tocar y agarrar. La novedad en esta clasificación de Sutcliffe es la inclusión de la conversación con otros agentes como tarea, no contemplada en las anteriores. A este respecto, es interesante citar a I. Heldal, quien explicaba en ([3D UI, URL], §18/8/2004) que incluso puede distinguirse entre conversación (hablar a otros) comunicación (por escrito, dibujos, etc.) y consciencia (*awareness*) de los demás (signos y gestos).

En las siguientes dos tablas se resumen las principales seis clasificaciones de las tareas de interacción. En la Tabla 2.9 se han agrupado las taxonomías que describen las tareas en el contexto general de las aplicaciones gráficas 3D.

Herndon <i>et al.</i> , 1994	Foley <i>et al.</i> , 1996	Barrilleaux, 2001
<ul style="list-style-type: none"> <li>• Control del punto de vista</li> <li>• Percepción</li> <li>• Creación de objetos, definición de modelos</li> <li>• Selección de objetos</li> <li>• Colocación y edición de objetos                             <ul style="list-style-type: none"> <li>○ Transformaciones afines (p. ej. trasladar, rotar, escalar, alineación, etc.)</li> <li>○ Modificar otros parámetros (p. ej. color, sombreado, etc.)</li> </ul> </li> <li>• Programación                             <ul style="list-style-type: none"> <li>○ Definir el comportamiento de objetos</li> <li>○ Definir relaciones entre objetos</li> </ul> </li> </ul>	<p>Tareas de Interacción Compuestas (CITs):</p> <ul style="list-style-type: none"> <li>• Cajas de diálogo</li> <li>• Construcción</li> <li>• Manipulación: cambiar forma, arrastrar, rotar o escalar</li> </ul> <p>Tareas Básicas de Interacción (BITs):</p> <ul style="list-style-type: none"> <li>• Situar</li> <li>• Seleccionar</li> <li>• Texto</li> <li>• Cuantificar</li> </ul> <p>Otras tareas 3D:</p> <ul style="list-style-type: none"> <li>• Rotación 3D</li> </ul>	<p>Tres componentes:</p> <ul style="list-style-type: none"> <li>• Navegación</li> <li>• Manipulación</li> <li>• Acceso</li> </ul> <p>Basados en otros tres:</p> <ul style="list-style-type: none"> <li>• Control</li> <li>• Realimentación</li> <li>• Visualización</li> </ul>

**Tabla 2.9** Tareas de interacción en aplicaciones gráficas 3D

En la Tabla 2.10 se recogen aquellas que describen tareas en el contexto más particular de las aplicaciones de Realidad Virtual.

Bowman y Hodges, 1999	Boyd y Sastry, 1999	Sutcliffe, 2003
<ul style="list-style-type: none"> <li>• Control del punto de vista o viajar               <ul style="list-style-type: none"> <li>○ Selección de dirección / destino</li> <li>○ Selección de velocidad / aceleración</li> <li>○ Entrar condiciones</li> </ul> </li> <li>• Selección               <ul style="list-style-type: none"> <li>○ Indicación de objeto</li> <li>○ Indicación de selección</li> <li>○ Realimentación</li> </ul> </li> <li>• Manipulación               <ul style="list-style-type: none"> <li>○ Sujeción del objeto</li> <li>○ Posición del objeto</li> <li>○ Rotación del objeto</li> <li>○ Realimentación</li> </ul> </li> </ul> <p>Otras tareas:</p> <ul style="list-style-type: none"> <li>• Liberar               <ul style="list-style-type: none"> <li>○ Indicación de soltar</li> <li>○ Posición final del objeto</li> </ul> </li> <li>• Control del sistema</li> </ul>	<ul style="list-style-type: none"> <li>• Navegación</li> <li>• Selección</li> <li>• Manipulación</li> <li>• Entrada de datos</li> </ul>	<ul style="list-style-type: none"> <li>• Movimiento y navegación</li> <li>• Manipulación de objetos e interacción con ellos               <ul style="list-style-type: none"> <li>○ Seleccionar, consulta</li> <li>○ Tocar, agarrar</li> </ul> </li> <li>• Conversación con otros agentes</li> <li>• Características que no son de Realidad Virtual               <ul style="list-style-type: none"> <li>○ Entrada de datos</li> <li>○ Interacción estilo GUI</li> </ul> </li> </ul>

**Tabla 2.10** Tareas de interacción en aplicaciones de Realidad Virtual

### 2.6.3 Técnicas de interacción, objetos de interacción y widgets

Continuando con la explicación dada en [Foley, 1996], las técnicas de interacción se definen como formas de usar los dispositivos de entrada para introducir información. En su comparación con el lenguaje natural, las técnicas de interacción son para estos autores como las palabras (el léxico), y de la misma forma que podemos utilizar diferentes palabras para expresar un mismo significado, también podemos utilizar diferentes técnicas para realizar una misma tarea de interacción.

Foley *et al.* explican que el conjunto de técnicas de interacción determina en gran medida el “look and feel” de una interfaz, y que con el fin de conseguir una mayor consistencia entre diferentes interfaces, esos elementos suelen recopilarse en forma escrita en guías de estilo o implementadas en librerías (*toolkits*).

También explican que con el término widget se nombra así a las técnicas de interacción en el sistema de ventanas X Windows, como botones, menús, barras de desplazamiento, etc. Sin embargo, es discutible que un widget sea una técnica de interacción, y su uso de forma indistinta en la bibliografía –como se verá a continuación- resulta ciertamente confuso.

Así, en [Conner, 1992] se introduce el término “widget tridimensional” y es definido como un encapsulado de geometría y comportamiento, el cual se utiliza para controlar o mostrar información acerca de los objetos de la aplicación – interesante distinción entre widget y objeto de la aplicación-. Estos autores señalan que un widget no tiene porqué contar con una representación visual y exponen como ejemplo un widget de rotación 3D basado en el movimiento del ratón 2D sobre la superficie de una esfera virtual. El comportamiento de ese widget puede conseguirse sin ninguna representación visual, pero también puede aumentarse utilizando –por ejemplo- una esfera semitransparente que encierre el objeto. Sin embargo, este ejemplo de widget 3D sería más apropiado para ilustrar lo que otros autores llamarían técnica de interacción, en concreto una técnica de manipulación.

En [Herndon, 1994] se definen las técnicas de interacción como las interfaces que uno utiliza para completar una tarea particular. Una definición tal vez demasiado general, cualquier elemento de la interfaz se incluiría entonces en ese término. Es interesante, sin embargo, que ya por entonces se reconocía que las aplicaciones gráficas 3D eran en su mayoría implementaciones *ad hoc* para diseños que dependían de la tarea –lo que hace difícil su reutilización- y se esperaba que finalmente emergiese un conjunto de técnicas de interacción 3D para uso general.

[Bowman, 1999] coincide con Foley *et al.* al indicar que para cada una de las tareas de interacción universales, existen muchas técnicas de interacción propuestas. Como se comentó en el apartado anterior, Bowman y Hodges describen una taxonomía que identifica un conjunto de tareas universales, y divide cada una de ellas en subtareas. La descomposición se completa listando los diferentes posibles componentes de técnicas para cada una de esas subtareas. Para una tarea dada, es posible entonces componer una técnica de interacción eligiendo un componente particular de cada una de las subtareas. Cada subtarea representa, de este modo, una decisión que el diseñador debe tomar.

Por su parte, en [Boyd, 1999] se explica que una técnica de interacción hace referencia al estilo de interacción específico que confiere un proceso de interacción. También coinciden con Foley *et al.* y con Bowman y Hodges al decir que cada una de las tareas básicas de interacción puede ser llevada a cabo empleando un número de posibles técnicas de interacción. En [Sastry, 2001] ilustran cómo una tarea básica se concreta en una técnica de interacción, la cual

se implementa basada en objetos de interacción y objetos de aplicación (véase Figura 2.4). Esto último recuerda a la distinción entre widgets y objetos de aplicación de Conner *et al.* De hecho, como objetos de interacción también incluyen un conjunto de widgets. Sin embargo, es discutible que objetos como la mano virtual o el cursor sean incluidos entre los objetos de interacción y widgets, como proponen estos autores.

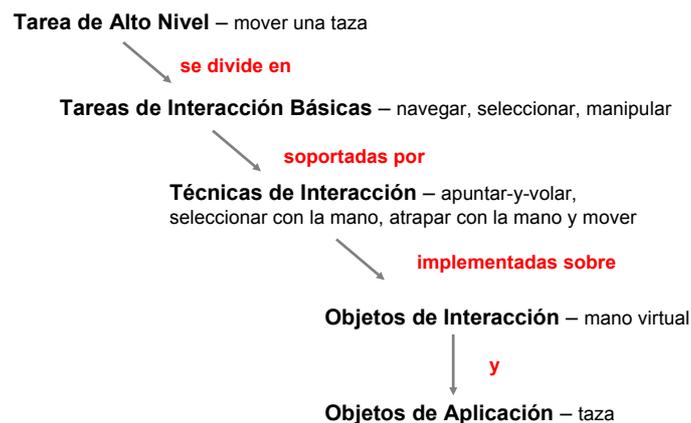


Figura 2.4 De las tareas a las técnicas de interacción según Sastry *et al.*

A las cuestiones que se ha dicho que son discutibles se tratará de dar respuesta más adelante en esta Tesis. Lo que no es discutible es, por ejemplo, la importancia de la metáfora en las técnicas de interacción. En este sentido, en [Poupyrev, 1998] se describe una taxonomía de técnicas de manipulación de objetos, y se explica que la mayoría de esas técnicas se basan en unas pocas metáforas básicas de interacción. Sin duda una afirmación que bien podría hacerse extensiva al resto de técnicas. Los autores prosiguen explicando que cada una de estas metáforas proporciona el modelo mental en el que se fundamenta la técnica de interacción: por una parte, expresa qué puede hacer el usuario y cómo (*affordances*) y, por otro lado, pone de manifiesto sus limitaciones (*constraints*). Las técnicas de interacción propuestas toman entonces esas metáforas y las extienden con el fin de superar sus limitaciones, aunque ello normalmente origina otras nuevas.

#### 2.6.4 Dispositivos físicos

La analogía de [Foley, 1996] compara en esta ocasión las acciones simples con dispositivos de entrada con las letras del alfabeto que forman las palabras, que como se ha visto en el apartado anterior serían las técnicas de interacción. [Bowman, 2001] explica que los dispositivos de entrada son las herramientas

físicas usadas para implementar diferentes técnicas de interacción, y que varias técnicas pueden implementarse con un mismo dispositivo. La cuestión – continúan Bowman *et al.* – es si el dispositivo es el más apropiado para la técnica dada.

Los dispositivos de entrada pueden ser catalogados según el tipo de evento que generen. Según [Sutcliffe, 2003], podemos hablar de dispositivos discretos y analógicos. A estos últimos, Bowman *et al.* los denominan continuos, y añaden otra categoría más, la de los dispositivos de entrada híbrida. Otra clasificación atendería a los grados de libertad del dispositivo (DOF, *Degrees Of Freedom*), siendo los dispositivos más populares los 2D –como el ratón–, según Foley *et al.*

Dispositivo	Grados de libertad	Tareas representativas
Control deslizante, dial	1	Control de volumen
Ratón	2	Selección, dibujo, posición 2D
Ratón 6D, tracker	6	Colocación, control de vista
Guante, máscara	16 o más	Animación de la mano, cara
Traje de cuerpo entero	100 o más	Animación de todo el cuerpo

**Tabla 2.11** Tareas apropiadas para diferentes dispositivos

También según Foley *et al.*, los dispositivos pueden ser evaluados a tres niveles diferentes: a nivel de dispositivo –características físicas–, de tarea –comparando dispositivos para la tarea– y de diálogo –evaluando secuencias de tareas–. A nivel de tarea, podemos establecer relaciones obvias entre dispositivos y tareas según los grados de libertad, como se lista en la Tabla 2.11, tomada de [Herndon, 1994]. Por ejemplo, los dispositivos 2D son apropiados para tareas 2D.

Pero como Herndon *et al.* explican, la diferencia de grados de libertad entre la propia tarea y el dispositivo físico puede ser un problema. Por una parte, si el espacio de la tarea tiene más grados de libertad que el dispositivo, la tarea toma la forma de un diálogo más complejo que la simple manipulación directa, como también explican Foley *et al.* Por ejemplo, un dispositivo 2D como el ratón sólo permite establecer en cada momento una correspondencia directa con dos de las dimensiones del espacio 3D. Al añadir más dimensiones se está forzando al ratón a dar mucho más de lo que se pretendía con él (Wingrave, [3D UI, URL], §30/7/2004). Suele resolverse definiendo diferentes modos entre los que el usuario cambia con los botones del ratón o los de una botonera que se presenta en pantalla (*dashboard*). Para [Nielsen, 1998], estos controles extra se interponen en la tarea principal del usuario.

Otro tanto ocurre cuando el espacio de la tarea tiene menos grados de libertad que el dispositivo, pues puede resultar confuso si el dispositivo no está físicamente limitado, e incluso muy fatigoso [Herndon, 1994]. Esto ocurre, por ejemplo, cuando el dispositivo es 3D pero el espacio de la tarea sólo tiene una o dos dimensiones. Herndon *et al.* explican igualmente cómo fuerzas tales como la gravedad o la fricción son empleadas por las personas para simplificar tareas sin ni siquiera pensar en ellas.

Aparte de los dispositivos de entrada, también hay que tener en cuenta los de salida (*displays*, según el uso más extendido en la bibliografía). En contraste con el lento progreso en dispositivos de entrada, el hardware de gráficos 3D ha experimentado en los últimos años un avance vertiginoso, tanto en tarjetas gráficas, proyectores y pantallas, incluyendo los nuevos sistemas estereoscópicos sin gafas. Ya en 1994 advertían Herndon *et al.* sobre la cada vez mayor disponibilidad de hardware y software 3D en todo tipo de ordenadores. Y aunque aún en 1996 la tecnología 3D del PC al uso era inmadura [Roberts, 1999], los avances en la tecnología han logrado elevar esas capacidades del PC, rebajando además los precios [Leavitt, 2001]. Estos avances han beneficiado sobre todo a los videojuegos 3D, pero también a los sistemas de visualización científica y de la información.

Pero el término “display” no sólo alude a la presentación visual, sino también acústica y táctil [Bowman, 2001], e incluso se habla de imágenes sonoras y táctiles [Furness, 2001]. Los dispositivos de realimentación de fuerzas (*force-feedback*, también *haptic displays*) y táctiles son actualmente un campo de intensa actividad investigadora, pero el uso de estos dispositivos es limitado, sustituyéndose en ocasiones por los propios objetos tangibles [Sutcliffe, 2003]. También puede transformarse la señal en sonido (*sonification* según Bowman *et al.*) o en imágenes de colores, por ejemplo. Es lo que Sutcliffe llama “sustitución modal”, pero que también se conoce como “trasposición de los sentidos” [Burdea, 1996].

Precisamente, el dispositivo a utilizar, ya sea de entrada o de salida, dependerá de la modalidad deseada. Según Sutcliffe, las modalidades para la comunicación persona-ordenador son: visión, audio (habla, sonido), tacto (realimentación de fuerzas, relieve), motor y olfato. Curiosamente no incluye el gusto, aunque ni este ni el olfato suelen tenerse en cuenta en el diseño de interfaces.

## 2.7 El diseño de la interfaz de usuario 3D

En los apartados anteriores se ha tratado de identificar los elementos que forman las interfaces de usuario 3D, un estudio que se ha estimado necesario previo al

abordaje del diseño de estas interfaces. Sin embargo, en el intento se ha dado con una gran variedad de términos y clasificaciones que hacen temer un diseño nada fácil.

No sorprende mucho que en [Herndon, 1994] se afirmara que las aplicaciones gráficas 3D son significativamente más difíciles de diseñar, implementar y usar que sus homólogas 2D, que el campo era aún muy inmaduro. Sin embargo, como se verá en este apartado, esa afirmación parece no haber perdido vigencia más de una década después. Así, muchos de los desarrollos son guiados por la intuición del diseñador, e implementados desde cero por programadores y creadores de modelos 3D. La reutilización es baja, y los problemas de uso con la interfaz son frecuentes. Por el contrario, la investigación en interfaces gráficas bidimensionales ha producido muchas soluciones caracterizadas por su estabilidad y madurez, y las reglas que rigen la interacción han sido formalizadas y son ampliamente aceptadas por los usuarios [Pittarello, 2001].

Según [Shneiderman, 1998], tres son los pilares sobre los que se apoya el diseño de la interfaz de usuario para lograr el éxito: documentos que describen las guías a seguir, herramientas software para interfaces de usuario, y la revisión por expertos y realización de tests de usabilidad. En el caso de las 2D, las guías, interfaces de programación (APIs, *Application Programming Interfaces*) y otras herramientas describen y proporcionan los elementos más básicos de las interfaces de escritorio –funcionalidad, apariencia, etc.- y la forma en que se utilizan en conjunto, liberando al diseñador de la necesidad de inventar e implementar esos elementos [Bowman, 2001]. Pero como estos últimos autores afirman, el diseño de las interfaces 3D no ha alcanzado aún ese estado de madurez.

Refiriéndose a los entornos virtuales, [Kaur, 1998] observa que la inmadurez de la tecnología complica innecesariamente el desarrollo, los diseñadores tienen que comprobar la viabilidad técnica de sus ideas antes de empezar la implementación. Pero el problema no es sólo la tecnología. Para [Fencott, 2001], la tecnología está disponible y continua siendo desarrollada y perfeccionada, pero con todo opina que la Realidad Virtual aún se encuentra en pañales (“incunabula stage” es la expresión utilizada por estos autores –del latín, *incunabŭla*, pañales-). Y es que toda innovación comienza con un periodo de experimentación (respuesta de un lector a [Nielsen, 1998]), y lleva tiempo llegar a comprender un nuevo medio y diseñar experimentos aceptables (Wingrave, [3D UI, URL], §19/2/2002). El problema, según el propio Wingrave, es que estamos impedidos por la dificultad de repetir los experimentos de los demás. Por contra, lo que sí parece repetirse son los errores pasados, que no son afrontados en los nuevos diseños (Wingrave, [3D UI, URL], §30/7/2004). Para este investigador, resulta imprescindible conocer qué se ha hecho o se está

haciendo en otros proyectos, no cometer los mismos errores y, por último, no descartar aquellas soluciones que, como las interfaces gráficas 2D, funcionan.

En los siguientes apartados se abordarán entonces, y uno a uno, los diferentes aspectos del diseño de una interfaz de usuario 3D, con especial atención a los problemas que plantean.

### 2.7.1 El problema de los dispositivos físicos

Una de las razones por las que las interfaces gráficas 2D son más fáciles de diseñar es porque sus conocidas técnicas de interacción están basadas en unos pocos dispositivos físicos, típicamente el ratón, el teclado y la pantalla. En cambio, como se explica en [Bowman, 2001], el espacio de diseño de las interfaces de usuario 3D es significativamente mayor, los diseñadores tienen que afrontar una gran variedad de dispositivos de entrada y salida, y la aparición de nuevos productos requiere inventar nuevas técnicas y reconsiderar lo aprendido hasta entonces. Un ejemplo de esa variedad lo encontramos en los guantes de datos, como se ilustra en la Figura 2.5.



**Figura 2.5** Diferentes modelos de guantes ofrecen distintas tecnologías y número de sensores.

Puede que el número de empresas que luchan por sobrevivir en este arriesgado mercado no sean muchas, pero podría decirse que al menos hay tantas como tipos diferentes de dispositivos y tecnologías. Cada una de esas diferentes tecnologías ofrece una ventaja sobre las demás, pero también tiene sus inconvenientes, con lo que no se puede decir que una tecnología sea mejor que las demás en todas las aplicaciones. Además, en algunos aspectos suelen quedar desfasadas frente a las tecnologías que aparecen en el mundo PC. Pero como los productos son caros, la renovación de los dispositivos antiguos y la adquisición de modelos innovadores no están al alcance de cualquiera. Ya en [Foley, 1996] se reconocía que no todo diseñador de interfaces de usuario tiene el lujo de seleccionar el dispositivo más apropiado, pues en ocasiones la elección está hecha de antemano. Y esa es una afirmación que en el caso de las 3D cobra mayor importancia si cabe.

## 2.7.2 Diseño de técnicas de interacción

De nuevo, el diseñador de interfaces gráficas 2D tiene la ventaja de que las técnicas de interacción son pocas, bien conocidas y están soportadas por el sistema operativo, como por ejemplo “apuntar y pulsar” o “arrastrar y soltar” con el ratón, o los atajos con el teclado. En el caso de las interfaces de usuario 3D, si bien es cierto que existen algunas técnicas ya viejas conocidas, como la técnica de emisión de rayos *–ray casting* en inglés- [Mine, 1995] o la técnica *go-go* [Poupyrev, 1996] *–para extender el brazo como el inspector Gadget, personaje de una popular serie de dibujos animados-* (véase Figura 2.6), no existe ningún conjunto estándar, y la falta de soporte software obliga a la implementación desde cero de las técnicas elegidas por el diseñador.



Figura 2.6 Técnicas de selección ray casting y go-go (autor: Arturo S. García)

Aunque como se ha visto anteriormente, existe un cierto conjunto de tareas que pueden considerarse “universales”, no está tan claro qué técnicas implementar para llevarlas a cabo. [Bowman, 2001] habla de dos enfoques a la hora de diseñar las interfaces de usuario 3D:

- El primero sigue una filosofía artística, orientado a diseños a corto plazo, en general soluciones *ad hoc*, pero proporciona unos cimientos sobre los que fundar una filosofía más sistemática. Según Bowman *et al.* el éxito en el diseño de la interfaz 3D se basaría en: apoyarse en la investigación sobre el factor humano en la interacción con ordenadores; reutilizar técnicas e ideas aportadas por otros diseñadores; recurrir a la creatividad de uno mismo para inventar nuevas técnicas e interfaces; finalmente, usar los modelos y estrategias existentes en el diseño de interfaces 3D. En cuanto a “inventar” nuevas técnicas, los autores distinguen entre ideas que se apoyan en el mundo real *–simulación de la interacción física-* y las que se basan en la “magia” *–tomadas de la literatura o el cine, pero también rompiendo con las ideas preconcebidas del usuario* (Pierce, [3D UI, URL], §4/8/1999)-.

- La segunda filosofía es la sistemática, más lenta y metódica, caracterizada por el estudio de las tareas del usuario, técnicas existentes, y las características del usuario, el sistema o el entorno que puedan afectar al rendimiento de la interfaz. Está basada en dos componentes: taxonomías, que clasifican y descomponen tareas y técnicas, y sirven de guía para nuevos diseños; y la evaluación, tanto formativa como sumativa, integrada en el ciclo de desarrollo (*sequential evaluation*, en [Gabbard, 1999]) o como herramienta de investigación con bancos de pruebas (*testbed evaluation*, como p. ej. en [Bowman, 1999]) –véase también [Bowman, 2002]-.

### 2.7.3 Diseño de controles

Una vez más, el diseñador de interfaces gráficas 2D tiene a su alcance un conjunto de widgets bien conocidos que pueden usarse en una aplicación, como botones, menús, etc. Incluso aunque su apariencia cambie de un sistema de ventanas a otro, su funcionalidad sigue siendo la misma.

En cambio, en el diseño de interfaces 3D no existe ningún conjunto de controles 3D estándar al que recurrir. No es extraño que cada cierto tiempo alguien pregunte en las listas de correo cuáles deberían ser los elementos que deberían integrar ese conjunto (véase p. ej. Sostein, [VRML Email, URL], §23/3/2004). Un camino que suele tomarse para llegar a ese conjunto estándar consiste en preguntarse cuáles son los equivalentes de los widgets en el espacio 3D. Pero las posibilidades que ofrecen las interfaces 3D son mayores, pudiendo mezclar –por ejemplo- widgets y modelos de controles reales en una misma interfaz. Más aún, pueden introducirse nuevos controles no vistos en el mundo real ni en las interfaces gráficas 2D, como por ejemplo el árbol de conos (*cone tree*) descrito en [Robertson, 1991].

Un ejemplo de toolkit para entornos virtuales es el desarrollado como parte del proyecto [INQUISITIVE, URL], que además de técnicas de interacción, implementa también un conjunto de objetos genéricos de interacción virtual (véase [Boyd, 1999], [Sastry, 2000] y [Sastry, 2001]).

En el apartado concreto de controles para Web 3D, cabe destacar las dos propuestas de arquitecturas para controles realizadas por el grupo de trabajo en widgets para VRML (VRML Widgets Working Group, [VRML WWG, URL]), una basada en niveles siguiendo las ideas de P. Isaacs y B. Becker, y otra en componentes según sugiere G. Seidman. También es de destacar el proyecto [CONTIGRA, URL], que propone un conjunto de técnicas, metáforas y

controles organizados de forma jerárquica, usando para su descripción XML y X3D, este último sólo para describir la forma (véase también [Dachselt, 2002]). Precisamente, una preocupación común a todas estas propuestas es la separación entre la forma del control y la lógica que gobierna el mismo, dando al diseñador libertad para definir la geometría y apariencia de los controles, esto es, su diseño 3D, labor que será abordada en el siguiente apartado.

En cuanto a toolkits comerciales, cabe destacar el de la herramienta SmartScene de la compañía Digital ArtForms para entornos virtuales, y la librería de objetos de Internet Scene Assembler de ParallelGraphics para Web 3D. Ambas ofrecen una colección de widgets 3D, esto es, widgets similares a los de una interfaz gráfica 2D, pero en tres dimensiones.

#### 2.7.4 Diseño de objetos 3D

Las interfaces de usuario 3D más conocidas son eminentemente visuales. Así es cómo, por ejemplo, describe [Méndez, 2001] a los entornos virtuales. Las imágenes que muestran son generadas gracias a los gráficos 3D. Pero implementar y usar software de gráficos 3D es muy laborioso, como ya se reconocía en [Herndon, 1994]. No es sólo la dificultad de programar, sino que aún con lenguajes de modelado o herramientas visuales tampoco resulta fácil el modelado de objetos 3D y su animación.

Por una parte, son necesarios conocimientos técnicos para entender el espacio 3D, las transformaciones geométricas, o los modelos de iluminación, por ejemplo. Por otra parte, también se necesitan habilidades artísticas para dar forma tridimensional a aquello que no existe, salvo tal vez en el papel o en la mente del diseñador. Y, aunque puede recurrirse a colecciones comerciales de modelos, al final también habrá que componer la escena con ellos y optimizarla para un buen rendimiento.

Este es uno de los aspectos que más diferencian a las aplicaciones gráficas 3D de las aplicaciones 2D de negocio, como procesadores de texto, hojas de cálculo o bases de datos, y las acercan más a las aplicaciones multimedia e hipermedia. Comparados con los típicos sistemas software, los sistemas multimedia suelen ser más interactivos e involucran un buen número de elementos que requieren una producción especial, como gráficos, sonidos y videos [Skov, 2001].

### 2.7.5 Librerías de programación

Los programadores siempre tendrán que crear de forma artesanal los nuevos estilos de interacción [Shneiderman, 1998]. Aunque para crear interfaces 3D, si bien es cierto que el diseñador no tiene tantas ayudas como en el caso de interfaces más convencionales, no es menos cierto que actualmente existen un buen número de librerías de programación a las que recurrir.

De nuevo, debido al carácter visual de la mayoría de estas interfaces, es usual recurrir a librerías de gráficos 3D que hacen de interfaz estándar con el hardware de gráficos –como OpenGL-, dependen del sistema operativo –como Direct3D en los sistemas Windows de Microsoft- o son independientes de la plataforma – como Java3D de Sun-. Además de representar las primitivas gráficas en la pantalla, también es preciso gestionar las estructuras de datos que guardan en memoria la escena, tarea que simplifican librerías de más alto nivel –como Performer u OpenSG- facilitando el uso de un grafo de la escena.

Pero aparte del hardware de gráficos, las interfaces 3D suelen involucrar otros dispositivos de salida, así como de entrada. Usualmente, estos dispositivos requieren el uso de librerías de programación específicas para cada uno de ellos. Aunque también hay librerías que tratan de unificar diferentes dispositivos bajo una misma interfaz, como por ejemplo Open Track y VRPN.

Además, es preciso incluir código que gestione las entradas y salidas, dando forma a la aplicación con la arquitectura más adecuada. En lugar de reinventar la rueda, lo más sensato suele ser valerse de librerías que proporcionan esa arquitectura, de modo que sólo hay que preocuparse por escribir el código de la aplicación. Ejemplos de estas librerías para Realidad Virtual son VR Juggler y DIVERSE, y para Realidad Aumentada cabe citar ARToolkit y Studierstube.

Por último, y aparte de las librerías y toolkits de técnicas de interacción y controles que se abordan en otros apartados, cabe citar aquí librerías que permiten embeber interfaces gráficas 2D en espacios 3D –como por ejemplo Tweek para VR Juggler-, una forma de solucionar cómo representar los widgets en tres dimensiones y de aprovechar el legado de aplicaciones 2D.

## 2.7.6 Prototipos y especificación de la interfaz

Para que los programadores hagan su trabajo, el diseñador debe especificar antes qué debe hacerse. Según [Shneiderman, 1998], los creadores y arquitectos de interfaces de usuario deben contar con:

- Métodos simples y rápidos para realizar bocetos que den a los clientes un medio de identificar sus necesidades y preferencias.
- Métodos precisos para: trabajar en el detalle (diagramas de transición, esquemas de pantalla, árboles de menús), coordinarse con los especialistas (diseñadores gráficos, escritores) y para comunicar a los ingenieros del software y programadores qué hacer.

Según ese mismo autor, los prototipos también pueden servir como especificación, a partir de los cuales los escritores crean los manuales y los ingenieros del software construyen el sistema. Para crear los prototipos (*mock-ups*), Shneiderman cita los siguientes medios: papel y lápiz, procesadores de texto, software de presentación de diapositivas, herramientas de creación multimedia y herramientas de desarrollo visual.

Cabe preguntarse entonces si esos medios sirven también para crear prototipos de interfaces 3D, o precisan de otros métodos, como por ejemplo el uso de maquetas, por las tres dimensiones. Más adelante se citarán algunas herramientas software para la creación de prototipos de interfaces 3D, basadas en editores gráficos que permiten al diseñador especificar la interfaz con algunas de las notaciones que se abordan en el siguiente apartado.

## 2.7.7 Notaciones para la especificación de la interfaz

Para [Shneiderman, 1998], la primera herramienta para confeccionar los diseños es una buena notación que permita plasmar y discutir diferentes alternativas. Este autor lista un conjunto de notaciones e indica para qué estilo de interacción resulta apropiado cada uno de ellos. Esa lista tiene elementos comunes con la que ofrecen [Foley, 1996] para la especificación de la secuencia del diálogo. A continuación se comentarán todas estas notaciones:

- **Especificación en lenguaje natural:** El lenguaje por defecto en cualquier campo, tiende a descripciones largas, vagas y ambiguas.

- **Gramáticas:** Notación formal que suele utilizarse para describir lenguajes de comandos. La notación más conocida es la BNF (*Backus-Naur Form* o *Backus Normal Form*); también puede representarse en forma de diagrama. Shneiderman describe también otras gramáticas cuya notación permite indicar quién produce la cadena (*multiparty grammars*), típicamente el usuario (U) o la computadora (C).
- **Árboles de selección de menús y cajas de diálogo:** Como un mapa, muestran relaciones de alto nivel y detalles de nivel más bajo, pero no revelan todas las posibles acciones del usuario, como por ejemplo la vuelta al menú anterior. Expuestos sobre una pared, permiten obtener una visión general de todo el sistema para su chequeo. Como es lógico pensar, esta notación no resulta apropiada si la interacción no está basada en menús o no forma una estructura de árbol.
- **Diagramas de estados:** Esta conocida notación recibe diferentes nombres según el autor (p. ej. *Finite-State Automaton* o FSA según [Conner, 1992], *transition networks* o *state diagrams* para Foley *et al.*, y *transition diagrams* para Shneiderman). Tienen una única variable que indica el estado, las acciones del usuario causan transiciones de uno a otro, transiciones que pueden tener asociadas llamadas a rutinas. Su mayor problema es la explosión del número de estados en interfaces más complejas. Algunas variantes:
  - Se puede mejorar su estudio reemplazando los estados por capturas de pantalla. Las interfaces con cientos de capturas son estudiadas mejor si se exponen sobre una pared, como se ha indicado antes.
  - **Subredes:** *subnetworks* según Foley *et al.*, *subgraphs* según Shneiderman. Este último autor habla también de *statecharts*, cuya característica es la de anidar unos diagramas en otros usando rectángulos redondeados.
  - **Redes de transición recursivas:** Redes de transición que pueden llamar a otras subredes de forma recursiva. Según Foley *et al.*, la notación BNF es equivalente en cuanto a poder de representación.
  - **Redes de Petri:** ayudan a representar la concurrencia y la sincronización.
- **Redes de transición aumentadas** (*Augmented Transition Networks* o ATNs según Conner *et al.* y Foley *et al.*): Más flexible, codifica el estado de la interfaz según el nodo activo de la red y los valores de un conjunto de variables, reduciendo el problema de la explosión del

número de estados. Esta notación también puede extenderse con subredes y con recursión.

- **Diagramas de flujo de datos** (*data-flow diagrams* según Foley *et al.*, también *flowcharts*): Otra forma de definir las interfaces de usuario, esta vez como módulos de procesamiento interconectados. Sufren el mismo problema de escala que el resto de diagramas, y es que cuando la complejidad aumenta, se hacen menos legibles y se prefieren otras notaciones más parecidas a programas.
- **Notación usuario-acción** (*User-Action Notation*, UAN): Permite describir las acciones del usuario utilizando para ello un conjunto de símbolos, elegidos de forma que recuerden la acción que representan, como por ejemplo  $v$  para pulsar un botón,  $\wedge$  para soltar un botón, o  $\sim$  para el movimiento del ratón. Lleva tiempo habituarse a ella, y no es conveniente para la especificación de interfaces ricas en gráficos.

Las anteriores notaciones resultan apropiadas para la especificación de interfaces convencionales. Para las interfaces 3D, sobre todo como interfaces post-WIMP y de Realidad Virtual, cabe añadir otras propuestas:

- [Jacob, 1996] considera que la esencia del diálogo no-WIMP es un conjunto de relaciones continuas, algunas permanentes y otras temporales, y que las acciones que engranan o desengranan esas relaciones son típicamente acciones discretas. En base a ello, propone un lenguaje de especificación cuya forma gráfica consiste en un grafo de flujo de datos –representando las relaciones entre las variables continuas- y un diagrama de estados –representando las transiciones entre estados que cambian las relaciones anteriores-, representados ambos de forma separada. Jacob apunta que, para la segunda parte, también sería válida cualquier otra forma de especificación de manejadores de eventos. También existe una propuesta de representación unificada de las dos componentes, ilustrada en [Jacob, 1999].
- En [Kim, 1998] también se recurre a diagramas de flujo de datos (*Data Flow Diagram*, DFD) y diagramas de estados (*statecharts*) para especificar la función y comportamiento de los objetos de una aplicación de Realidad Virtual. Así, un DFD puede tener su correspondiente diagrama de estados que especifica cuándo ejecutar los procesos según el estado del sistema. Antes, esos objetos habrán sido identificados en el comportamiento general del mundo virtual, descrito mediante diagramas de secuencia de mensajes (*Message Sequence Diagrams*, MSD).

- Los autores de [Smith, 1999] entienden que los entornos virtuales están formados por componentes discretos y continuos, son sistemas híbridos, una interpretación similar a la que Jacob hace de las interfaces no-WIMP en general. Estos autores parten de las redes de Petri y las extienden para representar esos dos tipos de componentes, resultando en una notación semiformal llamada *flownets* (redes de flujo) y basada en flujos de información que son controlados por medio de eventos discretos. A diferencia de Jacob, la representación gráfica combina en un mismo diagrama los dos aspectos.
- [Boyd, 1999] describe el uso del lenguaje UML (*Unified Modelling Language*) para el desarrollo de su toolkit de interacción. En concreto, para su especificación funcional se emplearon diagramas de casos de uso –átomos de interacción-, diagramas de clases –relaciones estáticas entre objetos-, diagramas de estado –vida de un objeto- y diagramas de interacción –colaboración entre objetos-.
- [McIntosh, 2000] también hace uso de UML, en este caso para describir la estructura, función y comportamiento de un mundo virtual. En el ejemplo que explica la autora especifica la estructura mediante diagramas entidad-relación, y la función mediante diagramas de caso de uso y de colaboración. Aparte, sugiere el uso de diagramas de secuencia para la función; también para el comportamiento, aunque para este añade además los diagramas de estado.
- [Figuerola, 2002] utiliza un diagrama de flujo de datos como representación visual de su lenguaje InTml para describir técnicas de interacción. Objetos, dispositivos y técnicas de interacción se representan como cajas en ese diagrama, dejando a la vista su interfaz pero ocultando los detalles. Las cajas que representan las técnicas actúan de filtros sobre el flujo de datos que llega de los dispositivos y se dirige a los objetos.

La mayoría de las notaciones gráficas propuestas suelen apoyarse en lenguajes de marcas para su representación textual, y en herramientas software como editores gráficos para facilitar su uso, ayudas que serán tratadas en los dos próximos apartados.

### 2.7.8 Herramientas software para el diseño de interfaces

Como apunta [Shneiderman, 1998], los diseños en papel están bien para empezar, pero para una especificación detallada es necesario contar con herramientas software.

Las herramientas que en [Foley, 1996] se denominan UIMS (*User-Interface Management System*) permiten a los desarrolladores de interfaces crear, probar, y modificar de forma rápida sus conceptos de interfaz, disminuyendo con ello el coste de esos pasos en el ciclo de desarrollo de la interfaz de usuario. Según estos autores, un UIMS ayuda en la implementación de, al menos, la forma de la interfaz, siendo un ingrediente esencial la especificación del diálogo de secuencia. Los elementos de la interfaz son especificados en algunos casos utilizando lenguajes de programación, y en otros por medio de editores gráficos interactivos.

En el caso de las interfaces de usuario 3D, un editor gráfico como VRED [Jacob, 1999] permite especificar la secuencia de diálogo, así como el comportamiento de los objetos [Tanriverdi, 2001], usando la notación gráfica propuesta Jacob y comentada en el apartado anterior. Además, esta herramienta permite generar prototipos a partir de la especificación, para su simulación en el entorno PMIW, también desarrollado por estos autores. Similar es el juego de herramientas (*toolset*) MARIGOLD [Willans, 2000], creado para facilitar el uso de flownets también para especificar tanto el diálogo como el comportamiento. En esta ocasión, estas herramientas generan código ejecutable para MAVERIK, un toolkit para aplicaciones de Realidad Virtual.

Sin embargo, la especificación de la geometría y apariencia de la interfaz, en especial si se trata de objetos en un mundo virtual, suele dejarse para herramientas de modelado 3D profesionales, como 3D Studio o Maya. Por ello, si bien en el caso de las interfaces gráficas 2D es fácil encontrar entornos de desarrollo visuales que integran todo el proceso en una única aplicación, en el caso de las 3D la norma es utilizar, más que un entorno integrado, un conjunto de diferentes herramientas.

### 2.7.9 Lenguajes de marcas para describir interfaces

Entre los lenguajes de descripción de interfaces de usuario (UIDL, *User Interface Description Language*) la tendencia en estos últimos años viene dada por el uso de lenguajes de marcas, en especial los basados en XML,

caracterizados por un conjunto de marcas de texto que estructuran los archivos que contienen la descripción de la interfaz. Como texto legible, puede ser manipulado con un simple editor de texto, pero su ventaja reside en la facilidad con la que su contenido puede ser extraído por diferentes herramientas, facilitando la interoperatividad. Un ejemplo de ello es UsiXML, un lenguaje UIDL basado en XML que captura de forma declarativa la esencia de la interfaz de usuario de forma independiente a su apariencia final [UsiXML, URL].

En el caso de las interfaces 3D, se ha comentado que las notaciones gráficas suelen basarse en lenguajes de marcas para su descripción textual. Ese es el caso de la notación propuesta por [Jacob, 1996] basada, en esta ocasión, no en XML, sino en un antecesor, SGML.

Precisamente, SGML fue la base para la definición del lenguaje HTML, que alcanzaría rápidamente gran popularidad como lenguaje para la creación de páginas Web. Tratando de seguir estos pasos, se concibió el lenguaje VRML, en un principio como lenguaje de marcas para mundos virtuales en la Web, aunque la especificación que fue finalmente estandarizada –VRML97 [VRML Spec, URL]- no usó marcas de texto, por lo que su denominación cambió a lenguaje de modelado. Sin embargo, resulta fácil para cualquier herramienta exportar objetos y escenas 3D usando ese lenguaje, lo que ha extendido su uso como lenguaje de intercambio. El problema se presentaba al importar ficheros con este formato, pues resulta difícil construir un intérprete que cumpla la especificación completa –incluyendo animación, interacción, scripts, etc.-, por lo que suele restringirse a geometría y apariencia.

La tercera versión de este lenguaje, el reciente estándar X3D (*eXtensible 3D graphics*) [X3D Spec, URL], sí hace uso de XML, lo que ha impulsado a varios autores a estudiar sus posibilidades como lenguaje de descripción de interfaces 3D. Por ejemplo, en el proyecto [CONTIGRA, URL] se hace uso de X3D para definir la geometría y apariencia de los componentes que integran una escena 3D, aunque para lo demás el autor propone nuevos nodos y marcas, pues indica que ciertos aspectos, como el audio o el comportamiento, no están suficientemente tratados en X3D [Dachselt, 2002]. Caso parecido es InTml [Figuroa, 2002], un lenguaje basado en XML cuyo autor también ha investigado la forma de dotar a X3D de la misma capacidad que tiene InTml para describir las técnicas de interacción de una interfaz 3D, de forma que la correspondencia entre ambos lenguajes sea posible.

Un último ejemplo de lenguaje basado en XML para la descripción de interfaces de usuario es VRXML, cuyos autores lo utilizan para diseñar interfaces gráficas 2D y describir su comportamiento en el espacio 3D que representa un entorno virtual [Cuppens, 2004].

### 2.7.10 Guías de diseño, usabilidad y evaluación

Como se recordará, dos de los tres pilares que citaba [Shneiderman, 1998] para un diseño con éxito de interfaces de usuario eran los documentos que describen las guías a seguir, y la revisión por expertos y realización de tests de usabilidad.

En el caso de las interfaces 3D, para empezar no existen muchos documentos que detallen el diseño de una interfaz 3D. Una solución, como se comenta [Herndon, 1994], es la de tomar prestadas ideas y técnicas de otras disciplinas, como el diseño industrial, arquitectura, diseño de interfaces 2D, diseño de juguetes, cinematografía y psicología. Por ejemplo, una referencia que se suele apuntar es el lenguaje de patrones de Christopher Alexander para el diseño de espacios habitados [Alexander, 1977].

Aún así, cabe citar algunos ejemplos relevantes. Uno de ellos es la guía RealPlaces [IBM, URL], para el diseño de un entorno 3D que reemplace al escritorio tradicional. Otro es [Barrilleaux, 2001], para el diseño de interfaces 3D para el tradicional PC, lo que el autor denomina POCS (*Plain Old Computer System*). Otro más es [Bowman, 2004], que recoge la experiencia de estos últimos años en el diseño de las interfaces de usuario 3D.

En cuanto a la usabilidad, este concepto ha alcanzado un gran éxito en estos últimos años, ligado a la productividad y satisfacción del usuario, más allá de la idea de interfaz amigable. Y en este caso las interfaces 3D no son una excepción. Es más, en la medida en que estas interfaces tienden a involucrar todos nuestros sentidos en el espacio tridimensional, los problemas de usabilidad requieren si cabe una mayor atención. En este campo, dos de los trabajos de investigación más relevantes son [Gabbard, 1997] y [Kaur, 1998], ambos en el campo de los entornos virtuales.

Gabbard clasifica 195 guías que cubren muchos de los aspectos de los entornos virtuales que pueden afectar a su usabilidad, incluyendo locomoción, selección y manipulación de objetos, objetivos del usuario, fidelidad de las imágenes, usos y modos de los dispositivos de entrada, metáforas y otros. Kaur, por su parte, describe 46 guías concretas de diseño a partir de otras tantas propiedades generales de diseño (*Generalized Design Properties* o GDPs, inferidas de teorías de la interacción) –véase también [Sutcliffe, 2003]–, y que cubren aspectos como la tarea del usuario, espacio, puntos de vista, representación del usuario, objetos, comportamiento del sistema, acciones y feedback.

Para presentar las guías y hacerlas más accesibles al diseñador, Kaur les dio forma de herramienta hipertexto usando para ello el lenguaje HTML. En este sentido, [Eastgate, 2001] opina que, más que ofrecer guías que puedan ser

usadas de forma retrospectiva para detectar los errores cometidos, resulta mucho más útil contar con guías que ayuden durante el propio proceso de diseño. Siguiendo su filosofía, Eastgate propone tres herramientas en diferentes formas: un diagrama de flujo para decidir cómo representar los objetos interactivos, una lista de verificación (*checklist*) para la reducción del número de objetos y el control de su nivel de detalle, y una tabla para guiar en el tema de la navegación. Sin duda, más herramientas como éstas ayudarían a un diseño con éxito de las interfaces de usuario 3D.

Finalmente, y en cuanto a la evaluación, cabe recordar lo apuntado en el apartado 2.7.2 acerca de su uso, bien integrado en el ciclo de desarrollo o bien como herramienta de investigación con bancos de pruebas [Bowman, 2002].

## 2.8 Resumen

En este segundo capítulo se ha realizado un primer estudio de las interfaces de usuario 3D abordando, en un primer momento, su definición y la identificación de los elementos que las integran, y profundizando, después, en diferentes aspectos del diseño de estas interfaces.

No ha sido fácil, sin embargo, dar con una definición de estas interfaces, pues usualmente no se presentan como tales, sino bajo otros nombres, como post-WIMP, non-WIMP o post-PC, nombres que aluden a interfaces que rompen con el estilo de interfaz WIMP que domina el campo de la interacción persona-ordenador hoy en día. También como interfaces de Realidad Virtual o entornos virtuales, e incluso de forma más general al hablar de gráficos 3D o, simplemente, de las “3-D”. Todos estos términos se relacionan con el uso de la tecnología 3D como interfaz entre el usuario y el ordenador, bien como entrada o como salida, característica que se ha utilizado entonces para proponer una definición formal de interfaz de usuario 3D.

Con la definición dada, se ha querido después profundizar en las aplicaciones de estas interfaces, proponiendo una lista de ocho diferentes usos de las interfaces de usuario 3D, recogida en la Tabla 2.5. Esta lista supone una recopilación de las diferentes formas en las que estas interfaces se han puesto en práctica, algunas con más éxito que otras como consecuencia del carácter experimental de esta tecnología, lo que también debe hacer pensar que la lista no está cerrada y que nuevos usos se añadirán en el futuro. La importancia de esta lista es que las experiencias pasadas sean tenidas en cuenta por los diseñadores de interfaces en cada nuevo proyecto, para explotar sus ventajas y no caer en los mismos errores.

Otra forma de mostrar al diseñador las interfaces de usuario 3D consiste en ubicarlas en el contexto de la evolución de la interacción persona-ordenador,

desde las primeras interfaces basadas en una línea de comandos o de instrucciones, esto es, desde las primeras interfaces de una sola dimensión. Así, partiendo del continuo Realidad-Virtualidad (RV) de Milgram y Kishino, en el apartado 2.4 se ha propuesto un nuevo espacio de diseño basado no en uno sino en dos ejes, uno para el número de dimensiones y otro para el grado de inmersión, y que ilustra el paso gradual de las interfaces de usuario desde 1D a 3D, y desde los mundos virtuales a la propia realidad, resultando en un nuevo continuo digital-virtual-real.

Definido qué es una interfaz de usuario 3D, sus aplicaciones y lugar en la interacción persona-ordenador, el siguiente paso en este capítulo ha sido el de identificar y describir los diferentes elementos de estas interfaces. Así, en el apartado 2.5 se han abordado cuatro elementos básicos: el espacio 3D, los objetos, el comportamiento y la interacción. Este último se ha ampliado en el apartado 2.6. El estudio de todos esos elementos ha puesto al descubierto el gran número de términos y clasificaciones existentes, pero sobre todo las diferencias entre los distintos autores, lo que hará necesario poner orden en este babel, en particular en el modelo de objetos y el modelo de interacción.

En último lugar, y siguiendo los estudios anteriores, en el apartado 2.7 se ha profundizado en el diseño mismo de estas interfaces. En él se han recordado los tres pilares sobre los que, según Shneiderman, se apoya el diseño con éxito de la interfaz de usuario, demostrado en particular por las interfaces 2D. Sin embargo, a lo largo de este apartado se ha podido comprobar que, si bien existen diferentes librerías de programación, notaciones y herramientas, el diseño de las interfaces 3D es más complejo y no ha alcanzado ese estado de madurez. Peor aún, se encuentra en pañales.

Con todo, aún conociendo los detalles de las interfaces de usuario 3D y contando con las herramientas apropiadas para su desarrollo, el diseño de las mismas, como cualquier otro artefacto, es más difícil sin un plan, sin un método que guíe el propio trabajo del diseñador, lo que será objeto del siguiente capítulo.



## 3

# Metodologías para el desarrollo de interfaces de usuario 3D

## 3.1 Introducción

Según [Shneiderman, 1998], el diseño es inherentemente creativo e impredecible. Para este autor, los diseñadores de sistemas interactivos deben conjugar un dominio de lo que es técnicamente factible con un místico sentido de la estética que atrae a los usuarios. Este autor cita a Carroll y Rosson, quienes caracterizan el diseño como un proceso, no jerárquico –no es estrictamente top-down ni bottom-up-, que puede conducirnos a soluciones que luego no sean usadas, o al descubrimiento de nuevos objetivos [Carroll, 1985].

Shneiderman también cita a W.H. Mayall, quien en su libro “Principles in Design” (1979) afirma que “...ningún producto ha sido creado por un simple momento de inspiración... los requisitos bien pueden surgir de ese momento de inspiración pero, casi con toda certeza, esa brillante idea que emerge será desarrollada siguiendo procesos iterativos...”. Sin embargo, la madurez alcanzada en el desarrollo de aplicaciones convencionales permite a los diseñadores realizar su trabajo en base a la experiencia previa, recogida en forma de guías y patrones, reduciendo el número de caminos a explorar hasta llegar al diseño final. En cambio, como se ha tratado de mostrar en el capítulo anterior, el campo de las interfaces de usuario 3D es aún muy inmaduro, lo que lleva a los diseñadores a explorar muchos caminos, haciendo del diseño iterativo una necesidad, como apunta [Stuart, 2001].

Aún así, como también explica Shneiderman, en cualquier campo creativo también puede haber una disciplina, técnicas refinadas, métodos correctos y erróneos, y medidas del éxito.

Por ejemplo, de esa misma referencia puede tomarse dos extractos que bien podrían aplicarse a cualquier metodología de desarrollo de interfaces de usuario. El primero dice que, una vez se ha llevado a cabo la primera recogida de datos y los requisitos preliminares han sido establecidos, es posible comenzar con el diseño más detallado y una primera implementación. El segundo, que resulta

sensato completar el diseño antes de comenzar la implementación, aún sabiendo que una vez empezada deberán llevarse a cabo nuevos cambios.

A partir de lo dicho, podría esbozarse una metodología de desarrollo basada en tres procesos consecutivos –requisitos, diseño e implementación–, que no empiezan hasta no terminar el anterior, y en donde se asume que el diseño estará compuesto por varios ciclos o iteraciones. Sin embargo, se trata de una metodología muy general, el diseño de interfaces de usuario 3D debería guiarse por procesos y actividades mucho más concretos.

En este capítulo se describirán y analizarán diferentes propuestas metodológicas que pueden encontrarse en la bibliografía, estudio que también se recoge de forma resumida en [Molina, 2005b]. En la gran mayoría de los casos, estas metodologías no apuntan al desarrollo general de interfaces de usuario 3D, sino que suelen enfocarse hacia aplicaciones más concretas de ese tipo de interfaces, como por ejemplo la Realidad Virtual o la Realidad Aumentada. Pero más que por su campo de aplicación, las metodologías revisadas han sido clasificadas en los siguientes grupos:

- Producción de cine de animación 3D.
- Creación de mundos virtuales para PCs.
- Proponen un desarrollo más participativo.
- Hacen uso del análisis de tareas.
- Se basan en la ingeniería del software.
- Van más allá del desarrollo para PCs.
- Sirven para desarrollar interfaces convencionales.

Los apartados que siguen abordarán uno por uno cada uno de esos grupos y las metodologías en ellos incluidas.

## **3.2 Producción de cine de animación 3D**

Como se comentaba en el apartado 2.7.10, ante la falta de documentación que detalle el diseño de una interfaz 3D se puede empezar por tomar prestadas ideas de otras disciplinas, tal y como se apuntaba en [Herndon, 1994]. Una de esas disciplinas era la cinematografía. En este apartado se abordará la producción de una película, sí, pero de animación 3D, por guardar mayores similitudes con la creación de mundos virtuales.

### 3.2.1 Cadena de producción de los estudios de animación Pixar

#### 3.2.1.1 Descripción

[Reddy, 2005] explica la cadena de producción de películas que se sigue en los estudios de animación Pixar, en la cual se mezclan técnicas tradicionales y comerciales con otras más propias de la animación por ordenador. Los pasos que hace mención son los siguientes:

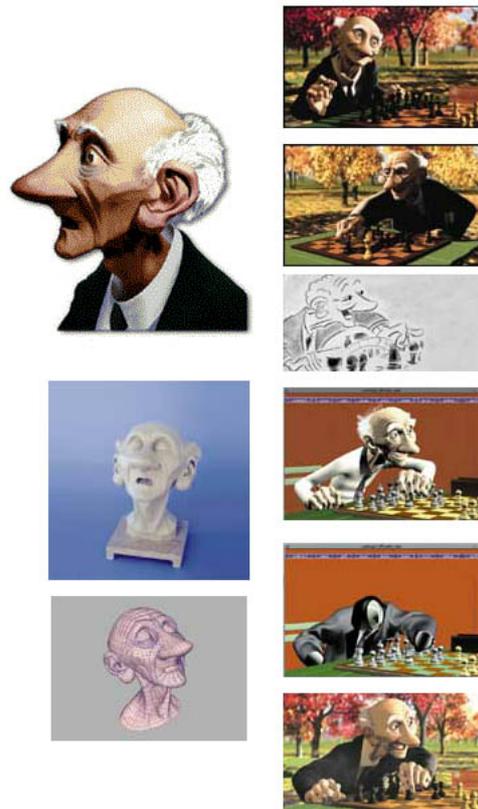
1. Storyboards.
2. Arte.
3. Modelado.
4. Sombreado, composición, vestuario, animación.
5. Iluminación, simulación, efectos.
6. Rendering.
7. Película.

La producción comienza con la tradicional creación de storyboards, que en las últimas producciones se acercaban a los 50.000. Desde el principio se lleva a cabo un proceso editorial que crea secuencias y tomas de la película y crea diferentes rollos (*reels*) para cada versión. Los primeros rollos son creados añadiendo voz a las viñetas de los storyboards. A partir también de los storyboards, y siguiendo con las técnicas tradicionales, el departamento artístico realiza estudios de los personajes para refinar su apariencia.

El paso siguiente es la creación de las versiones 3D por ordenador de los personajes, desde cero con la herramienta Maya. Como referencia se utilizan los modelos de arcilla producidos por el departamento artístico (Figura 3.1). Se definen las articulaciones de cada personaje, y se les añade color y textura. Se componen las escenas para nuevos rollos, sustituyendo los storyboards por modelos 3D, pero donde los personajes, aunque articulados, están formados por sencillas cajas. La ropa se les añade después, dándoles mayor realismo. La animación de los personajes se realiza a mano, sin hacer uso de técnicas de captura de movimientos a partir de actores reales.

Siguiendo la cadena, se añaden luces que iluminan las escenas y los personajes, produciendo nuevos rollos en los que aún se dejan fuera detalles como el movimiento del pelo o del agua. Esas animaciones se realizan a continuación mediante simulación. Otros efectos, como la nieve o las explosiones se dejan para el final.

El proceso por el que los ordenadores convierten toda geometría, apariencia y animaciones en la película se denomina “rendering”, y para ello se utilizan herramientas propias de este estudio de animación. Una vez terminada, la película está lista para su uso comercial.



**Figura 3.1** Bocetos, figuras, modelos 3D e instantáneas pertenecientes a la producción del corto de animación “El juego de Geri” (*Geri's game*, Pixar, 1997)

### 3.2.1.2 Análisis

Esta cadena de producción enumera los pasos para la creación y animación de figuras 3D por ordenador, desde su concepción artística hasta los últimos detalles de cada personaje, con el objeto final de crear una película de animación 3D. La similitud con la creación de mundos virtuales es grande, herramientas como Maya también se usan en el desarrollo de éstos, como por ejemplo en videojuegos, y por tanto podría pensarse que también pueden compartirse las técnicas utilizadas.

El primer aspecto a destacar es el propio orden que siguen los pasos, se empieza con el guión y como primera herramienta el storyboard, después artistas del lápiz y la arcilla profundizan en la apariencia que tendrán los personajes según su rol y, tras ello, otros artistas, esta vez del diseño por ordenador, adaptan esa apariencia a las posibilidades que ofrecen las primitivas gráficas y el software de “render”. Más concretamente, primero se define el rol de cada personaje, después una apariencia deseable para cada uno, y por último una apariencia concreta que poder “renderizar”.

Siguiendo con ese orden, también es de destacar el que se sigue al crear por ordenador los personajes y las escenas, empezando por la geometría, siguiendo por su apariencia, animación e iluminación, y terminando con la simulación y programación de los efectos más complejos. La creación así del modelo por ordenador se aborda entonces como un proceso incremental que parte de la geometría desnuda hasta alcanzar el acabado final, añadiendo en cada paso nuevos detalles.

Precisamente, el seguimiento de los modelos a lo largo de ese proceso incremental nos lleva a otro aspecto a destacar de esta metodología, y es el empleo de rollos de película en los que se van plasmando los progresos en cada escena de la película, usando primero los propios dibujos de los storyboards acompañados por voces, y posteriormente sustituyéndolos por imágenes por ordenador cuyo acabado va mejorando en cada nuevo rollo. Esto es, sin duda, una forma de prototipado del producto, que en este caso no es una aplicación software sino una película.

Sin embargo, existe una diferencia fundamental entre una película de animación 3D y un mundo virtual. Esta diferencia es la interactividad, que obliga al desarrollador a tener en cuenta las prestaciones de la máquina destino, así como las características del usuario final que se enfrentará a la interfaz. Por una parte, frente a las horas que pueden emplearse en la generación de una sola imagen de una película de animación, la simulación interactiva de un mundo virtual apenas deja una fracción de segundo para cada imagen. Con una restricción tan severa, el desarrollador debe cuidar mucho la complejidad del mundo virtual. Por otra parte, el usuario no observa de forma pasiva la escena 3D, sino que participa en ella a través de la interfaz. Como se explicó en el capítulo anterior, la creación de una interfaz de usuario 3D no es tarea fácil.

Aún así, puede aprenderse mucho de la cadena de producción de Pixar, en particular, como se ha destacado, de la forma en que los personajes saltan del papel y la arcilla al ordenador, y cómo se van refinando los rollos añadiendo cada vez más detalles, como un proceso de iterativo de creación de prototipos hasta obtener el producto final.

### 3.3 Creación de mundos virtuales para PC

En este apartado se recogen aquellas prácticas y propuestas más orientadas hacia la creación de mundos virtuales para PC, bien porque sus autores así lo subrayan o bien, si los autores no indican explícitamente tal cosa, porque los ejemplos de aplicación que publican sí muestran tal orientación.

#### 3.3.1 Creación en dos o tres pasos

##### 3.3.1.1 Descripción

En [Kim, 1998] se afirma que la práctica corriente en la construcción de mundos virtuales se basa en tres pasos. El primero consiste en crear los modelos virtuales usando herramientas de diseño asistido por ordenador (*CAD, Computer Assisted Design*). Después, los modelos son guardados en estructuras de datos o ficheros con el formato apropiado para ser utilizados en el entorno de ejecución del mundo virtual. Por último, las funciones y comportamientos son programados utilizando librerías gráficas de bajo nivel o paquetes de simulación de alto nivel.

[Smith, 2001] coincide con Kim *et al.* al entender esta práctica habitual, pero la describen en dos y no tres pasos. Así, en un primer paso se define el aspecto visual de los objetos del mundo mediante herramientas de creación de modelos 3D como por ejemplo 3D Studio. Esos objetos son entonces importados desde la aplicación de desarrollo de entornos virtuales, definiendo el comportamiento usando abstracciones a nivel de código.

La práctica descrita por los anteriores autores es en esencia la misma que recomienda, por ejemplo, la empresa EON Reality Inc. [EON, URL] como flujo de trabajo para la publicación de contenido 3D interactivo para Web con su herramienta EON Studio. Según la publicidad que ofrece dicha empresa, es tan fácil como seguir los siguientes pasos: 1) Importar los diferentes objetos 3D, usualmente creados con herramientas de modelado, como 3D Studio, o sistemas CAD, como ArchiCAD; 2) Asociar comportamientos interactivos a los modelos haciendo uso de la interfaz gráfica de la herramienta; y 3) Publicar en Internet o distribuir en CD-ROM.

### 3.3.1.2 Análisis

No cabe poner en duda lo sencillo que es seguir estos dos o tres pasos, aunque el hecho de que el método sea simple no significa que las tareas citadas en cada paso también lo sean. De hecho, la creación de los modelos 3D, incluso con la ayuda de herramientas visuales, suele ser un proceso laborioso que requiere, además de un dominio de la herramienta, ciertas dosis creativas. De forma similar, la asociación de comportamientos a los modelos puede parecer en principio fácil, sobre todo si la herramienta de desarrollo ofrece un conjunto de comportamientos predefinidos. Sin embargo, ese conjunto –si existe– suele ser reducido, su aplicación requiere el ajuste adecuado a los objetos, y para comportamientos más complejos suele ser necesario recurrir a programación.

Por otro lado, la aplicación de estos pasos tampoco asegura un buen resultado final. De hecho, Smith *et al.* apuntan que los objetos resultantes de la aplicación de un método así exhiben frecuentemente una apariencia visual compleja pero un limitado comportamiento asociado. Indican también que con ello suelen darse falsas pistas visuales acerca de la interacción que enlazan directamente con problemas de usabilidad. Esto es, un objeto creado con todo detalle puede hacer pensar que tiene igualmente asociados todos sus comportamientos, pero puede que no sea así y ello desconcierte al usuario. El problema, según Smith *et al.*, es que el comportamiento de un objeto del mundo no es considerado hasta después de que su geometría y apariencia visual hayan sido definidos.

La experiencia del que escribe con la herramienta EON Studio y el flujo de trabajo propuesto por la compañía EON Reality Inc. confirma las anteriores afirmaciones, como se comenta en [Molina, 2004]. Además, a partir de ella podemos decir que estos pasos no solamente no aseguran un producto final usable, sino que tampoco aseguran un buen rendimiento. Para que el mundo virtual pueda ser interactivo, el software y el hardware de gráficos disponen de un tiempo limitado para dibujar cada imagen. Si la apariencia visual de los objetos es compleja, puede que el sistema no pueda dibujar las imágenes a la cadencia adecuada, incurriendo en latencias que rompen las animaciones y acaban con la interactividad. Por esta razón, es necesario que el detalle de los objetos se adecue a las prestaciones del sistema de Realidad Virtual, lo que en la práctica significa cuidar que el número de polígonos que los forman sea bajo, bien usando las facilidades que ofrezca la herramienta de creación de modelos 3D o bien usando posteriormente otra herramienta especializada en la reducción del número de polígonos. Esta práctica requiere, además, un proceso iterativo de prueba y mejora hasta lograr el rendimiento adecuado, proceso que contrasta con la aplicación meramente secuencial de los pasos descritos.

Por último, Kim *et al.* critican además que, más allá de la creación de los objetos, estos pasos no representan un método ni una guía para el desarrollo del software de una aplicación de Realidad Virtual.

### 3.3.2 Metodología de diseño de EV's observada por Kaur

#### 3.3.2.1 Descripción

[Kaur, 1998] describe un estudio basado en entrevistas con diez diseñadores de tres organizaciones diferentes del Reino Unido, como representantes de la pequeña población de diseñadores de Entornos Virtuales (EV's).

A raíz de esas entrevistas, Kaur identificó hasta cinco procesos básicos comunes a la mayoría de los diseñadores, abstraídos de sus diferentes enfoques del diseño, y que son llevados a cabo de forma iterativa. Estos procesos son:

1. Especificación de requisitos.
2. Recogida de material de referencia de los modelos del mundo real.
3. Estructuración del modelo gráfico y, en ocasiones, división del mismo entre los diseñadores.
4. Creación de los objetos y su colocación en el EV.
5. Mejora del entorno con texturas, iluminación, sonido e interacción, y optimización del entorno.

Kaur apunta también que los diseñadores tienden a crear el entorno siguiendo bien un enfoque principalmente top-down o bottom-up. Así, seis de los diez diseñadores comenzaban con una estructura básica y le añadían detalle de forma gradual (top-down). Los otros cuatro creaban los objetos uno por uno y entonces los colocaban juntos en el EV (bottom-up). Independientemente del enfoque, todos los diseñadores mejoran el entorno después de haber creado los objetos y haberlos colocado.

El estudio también revela que los diseñadores crean y prueban de forma iterativa, pero que rara vez llevan a cabo tests con usuarios.

Por último, Kaur categoriza en tres áreas principales las preocupaciones de los diseñadores: balance entre rendimiento, detalle gráfico y realismo; comprensión del concepto de EV; e inmadurez de la tecnología. Kaur subraya que, a pesar de la importancia que otorga la bibliografía a los factores humanos, estos eran raramente mencionados por los diseñadores.

### 3.3.2.2 Análisis

Aunque los resultados de este estudio subrayan que la experiencia de estos diseñadores en la creación de mundos virtuales era pequeña, resultan significativos para mostrar una práctica real que difiere sustancialmente de la aplicación de los dos o tres pasos descritos en el apartado anterior.

En primer lugar, los diseñadores entrevistados revelan que antes de crear los objetos definen los requisitos del entorno, recogen material de referencia del mundo real y estructuran el modelo gráfico. Aunque el estudio no profundiza en estos procesos, ninguno de ellos se incluye entre los pasos del apartado anterior. Además, se habla también de realizar un reparto entre los diseñadores, prueba de que un mundo virtual suele ser el resultado del trabajo de un equipo, más que de una única persona. En suma, estos tres primeros puntos de la lista describen cómo los desarrolladores planifican y se preparan para construir el EV.

Entonces se aborda la creación de los modelos y su colocación en el EV, seguido por una mejora del entorno con texturas, iluminación, sonido e interacción, de forma entonces no muy diferente a la creación de escenas por ordenador en Pixar, salvo –muy especialmente- por la interacción. Precisamente, el estudio revela igualmente que es práctica habitual crear y probar de forma iterativa, lo que recuerda también a los sucesivos rollos de película, salvo –de nuevo- que el objetivo de los desarrolladores es el de obtener ese balance entre rendimiento, detalle y realismo. En cualquier caso, tampoco se refleja en las anteriores estrategias de dos o tres pasos.

A pesar de esas diferencias, las críticas anteriores siguen siendo igualmente válidas. Aunque persigue asegurar un buen rendimiento, esta práctica habitual no asegura un producto final usable como critican Smith *et al.*, lo que la propia Kaur también destaca y le motiva para plantear el flujo de trabajo que se verá en el apartado siguiente. Igualmente, esta práctica habitual se centra en los objetos del entorno virtual y, como apuntan Kim *et al.*, no guía para acometer la programación de una aplicación de Realidad Virtual.

## 3.3.3 Redefinición del proceso de desarrollo por Kaur

### 3.3.3.1 Descripción

A raíz del estudio realizado, y como parte de su trabajo doctoral, Kaur propone una redefinición del proceso de desarrollo que haga más énfasis en un diseño previo a la creación del mundo virtual, un diseño que tenga en cuenta los

factores humanos, buscando de este modo mayores garantías de que el producto final sea usable.

Kaur explica que las etapas de desarrollo fueron definidas usando conocimiento del proceso de diseño del EV –recogido en su estudio anterior-, y etapas de actividad comunes en métodos de desarrollo de sistemas, como el análisis de tareas o el diseño de presentación. Las siete etapas definidas son:

1. Definir requisitos.
2. Especificar componentes en el EV.
3. Especificar interacciones.
4. Diseñar componentes.
5. Diseñar interacciones.
6. Crear el entorno.
7. Evaluar el entorno.

Para dar soporte a esta metodología, Kaur propuso una herramienta hipertexto que recogía guías para cada una de las etapas propuestas, como ya se introdujo en el apartado 2.7.10. Cada guía está relacionada con una propiedad de diseño general (GPD), e incluye el propio consejo, la motivación, un contexto de uso y un ejemplo. Kaur implementó con el lenguaje HTML una versión de demostración de dicha herramienta que incluía un subconjunto de esas guías, 12 en total, distribuidas entre las dos etapas de diseño, y cubriendo tres elementos del EV (objetos, acciones y control del sistema). Finalmente, Kaur evaluó la utilidad de la herramienta mediante evaluación experta, pidiendo a cuatro diseñadores de una misma organización la realización de un storyboard para cada uno de los escenarios que les propuso, nueve en total, apoyando su diseño en la herramienta.

### 3.3.3.2 Análisis

La redefinición dada por Kaur subraya que la creación de un mundo virtual debe ir precedida no sólo de una definición de requisitos, sino de un buen diseño, coincidiendo con el consejo apuntado en [Shneiderman, 1998] para el diseño de la interfaz de usuario y que fue comentado al principio de este capítulo. Además, las etapas de desarrollo están soportadas por una herramienta de guía que ayuda al diseñador a plasmar las propiedades de diseño general (GDP's) en su mundo virtual, con el fin de que sea usable. Este énfasis en la usabilidad se observa también al incluir la evaluación como una etapa más del proceso.

Sin embargo, las siete etapas dadas no desvelan a simple vista actividades que sí formaban parte de la práctica habitual, como la recogida de material de

referencia de los modelos del mundo real, la estructuración del modelo gráfico o la división entre los diseñadores. Tampoco aparece la creación de los objetos y su colocación en el mundo, o el uso de estructuras y ficheros para intercambio entre las herramientas. Tal vez estas actividades quedan ocultas bajo alguno de los nombres de las etapas dadas, como por ejemplo la de creación del entorno. Pero Kaur no explica este punto, por lo que queda de nuevo al buen hacer del diseñador.

Por ello, si bien la redefinición de Kaur del proceso de desarrollo mejora a los métodos anteriores por ayudar a garantizar un producto final usable, no es un proceso completo al no capturar de forma total todo el desarrollo. En este sentido, la crítica de [Smith, 2001] ya no es aplicable, pero sigue teniendo validez la de [Kim, 1998], pues tampoco guía en la programación de una aplicación de Realidad Virtual.

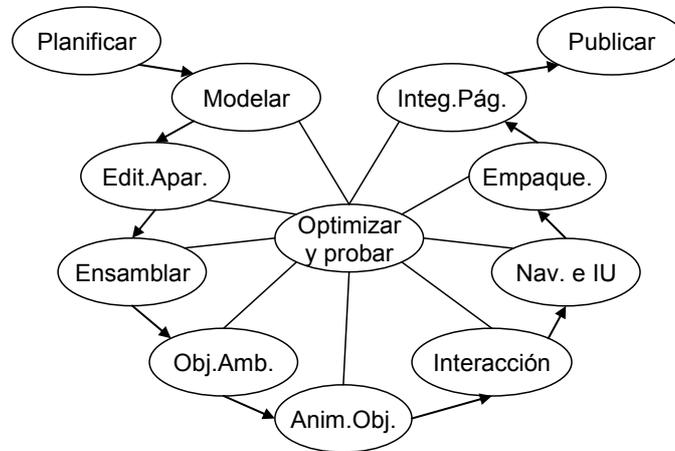
### 3.3.4 Creación de mundos virtuales con VRML/X3D

#### 3.3.4.1 Descripción

[Hay, URL] explica con gran detalle y claridad cómo crear un producto 3D interactivo usando el lenguaje VRML. El autor ilustra este proceso como una cadena de actividades que se realizan en secuencia, un total de 11 actividades que comienzan con la planificación del trabajo y terminan con la publicación del producto resultante en la Web. Estas actividades son:

1. Planificar.
2. Modelar objetos.
3. Editar apariencia.
4. Ensamblar la escena.
5. Objetos ambientales.
6. Animar objetos.
7. Interacciones complejas.
8. Navegación e interfaz de usuario.
9. Empaquetar.
10. Integración en la página Web.
11. Publicación.

Además, el autor añade una actividad más, “Optimizar y probar”, que interacciona con todas las actividades desde “Modelar objetos” a “Integración en la página Web”. Gráficamente:



**Figura 3.2** Creación de mundos virtuales con VRML según B. Hay

Al entrar en detalle, el autor describe también otras tareas incluidas en algunas de esas actividades. Por ejemplo, la actividad “Planificar” incluye tareas como definir la plataforma destino, comprender las limitaciones de la tecnología, y realizar un storyboard y una documentación del proyecto. En el caso de “Modelar objetos”, esta es ilustrada también como un nuevo flujo de actividades, incluyendo las siguientes:

- 2.1. Adquirir o crear el modelo 3D.
- 2.2. Convertir formato 3D.
- 2.3. Reducción del número de polígonos.
- 2.4. Convertir a VRML.
- 2.5. Importar desde la herramienta de autor VRML.

La actividad “Objetos ambientales” se refiere a la definición del fondo, y la inclusión de luces y sonido en la escena. Y como última actividad a destacar, la llamada “Navegación e interfaz de usuario” incluye la especificación del método o métodos de navegación por la escena, la definición de los puntos de vista y la creación de un HUD (Head Up Display) –como los instrumentos del cockpit de un avión-.

A lo largo de toda la explicación el autor utiliza como ejemplo la creación de un modelo 3D de una cámara fotográfica digital. Como herramienta de desarrollo, el autor utiliza principalmente Cosmo Worlds, aunque no es la única.

Por otra parte, en [Daly, 2002] también se explica cómo crear mundos virtuales para la Web, en esta ocasión haciendo uso del nuevo lenguaje estándar X3D, aunque con menor detalle que en el tutorial anterior. Según se extrae del material de este nuevo tutorial, el proceso de creación conlleva cuatro fases:

1. Fase de diseño.
2. Fase de modelado.
3. Fase de ensamblado.
4. Fase de optimización.

Los autores explican que en la fase de diseño se determinan las características de rendimiento, se realiza un boceto o croquis (*sketch*) del mundo –a papel y lápiz o por ordenador-, y se identifican los componentes de ese mundo. El resultado de esa fase es una vista del mundo que sirve de referencia para el resto de miembros del equipo y del cliente. En particular, el boceto define las dimensiones que se utilizarán en la fase de modelado y sirve también como plano maestro durante la fase de ensamblado, por lo que debe refinarse tanto como sea necesario. Para las animaciones complejas o inusuales se recomienda el uso de storyboards. Y en cuanto a la fase de optimización, esta se repite de forma iterativa hasta que se logran las características de rendimiento fijadas.

Una vez concluye la optimización, Daly *et al.* añaden que el mundo debe ser empaquetado para su distribución.

#### 3.3.4.2 Análisis

Ambos tutoriales presentan una metodología similar, si bien el de B. Hay resulta ser mucho más detallado. Los dos comienzan por proyectar el trabajo a desarrollar, actividad llamada “Planificar” en el caso del tutorial de B. Hay, y que coincide con lo que Daly *et al.* nombran como “Fase de diseño”. Aquí se identifica la plataforma destino que se utilizará como referencia en la optimización del rendimiento del producto. También se realizan bocetos de la escena a crear, una práctica muy común de diseño –como se ha visto en el cine de animación-, pero que parece haber sido pasada por alto por los anteriores métodos de esta sección.

Tras esa planificación, comienza el trabajo de desarrollo, modelando los objetos y ensamblando la escena. En este aspecto es donde más destaca el tutorial de B. Hay por su detalle, hablando por ejemplo de tareas como la importación de modelos, o la reducción de polígonos para ajustar su número a las prestaciones gráficas de la máquina, pero también de cómo mejorar la escena con fondos, luz y sonido, lo que Daly *et al.* no hacen. Más aún, B. Hay trata de forma explícita la interacción y la navegación en el mundo virtual, en definitiva la interfaz de usuario.

La inclusión de una planificación previa a la construcción y de un proceso iterativo de optimización hace que estas dos propuestas guíen adecuadamente hacia la obtención de un buen rendimiento en el producto final. Sin embargo, ninguna de ellas asegura que ese producto sea usable. Eso a pesar de los buenos consejos que ofrece B. Hay, pero que no son comparables con el conjunto de guías de diseño que se describen en [Kaur, 1998], ni en número ni en forma.

Por último, también les es aplicable la crítica que realiza [Kim, 1998], y es que no sirven de guía para la programación de una aplicación de Realidad Virtual. Tan sólo B. Hay habla del uso de scripts en la actividad “Interacciones complejas”. En cualquier caso, hay que tener en cuenta que estas metodologías se basan en la descripción de la escena mediante los lenguajes VRML y X3D, los cuales ya incluyen un conjunto predefinido de nodos para geometría, apariencia, animaciones e interacción, entre otros.

### 3.3.5 Análisis del grupo

En la creación de un mundo virtual bien se puede recurrir a técnicas y herramientas similares a las utilizadas en una película de animación por ordenador. Sin embargo, aunque sea así en el modelado geométrico y de la apariencia de los objetos del mundo virtual, incluso en la animación de los mismos, ya se ha advertido antes que la interacción marca una importante diferencia.

Las tres primeras metodologías en ser analizadas se basaban precisamente en importar de los modelos en la herramienta de desarrollo y ahí añadir los comportamientos, sin tener en cuenta ni al usuario ni a la máquina, ignorando posibles problemas de usabilidad y de rendimiento. Sin embargo, aunque esos métodos *naïve* son, según [Kim, 1998] y [Smith, 2001], práctica habitual, el proceso seguido por los desarrolladores de entornos virtuales entrevistados por Kaur era más bien diferente. Así, se ha visto que los desarrolladores primero planifican y se preparan para crear el entorno, después lo construyen y lo mejoran, y finalmente lo optimizan en un proceso iterativo, preocupados por conseguir un adecuado balance entre detalle y rendimiento.

A decir verdad, ambos modelos de la práctica habitual son complementarios, pues el estudio de Kaur no destaca que la creación de los objetos suele realizarse por herramientas de diseño diferentes al entorno de ejecución del mundo virtual, y que por ello es necesario guardar los objetos en estructuras o formatos de fichero que puedan servir de intercambio, en particular que puedan ser importados por el entorno de desarrollo.

Lo anterior ocurre, por ejemplo, en la creación de mundos virtuales para la Web, y así lo tienen en cuenta los autores de los dos tutoriales con VRML/X3D incluidos en este grupo. Al mismo tiempo, ambos tutoriales siguen un modelo de proceso basado en la planificación, creación del entorno y optimización, como el modelo del estudio de Kaur, e incluso siguen la estrategia bottom-up que se describe en ese estudio, esto es, primero modelar los objetos y después ensamblar la escena.

Sin embargo, mientras que el estudio de Kaur no profundiza en los primeros pasos –p. ej. la especificación de requisitos–, los autores de esos tutoriales sí dan más detalles de ese proceso previo a la construcción, que B. Hay llama “Planificar” y Daly *et al.* nombran como “Diseño”. Así, en ese punto se especifica la plataforma destino o las características de rendimiento a tener en cuenta luego al optimizar. También se crean bocetos y storyboards, como en los primeros pasos de una película de animación, sirviendo los primeros como planos del entorno virtual y los segundos, además de documentar el proyecto, para la descripción de las animaciones. Por el contrario, en ninguno de los tutoriales se plantea la existencia de un equipo de diseñadores y, por ello, no comentan reparto alguno del trabajo como sí se recoge en el estudio de Kaur.

Los procesos descritos en esos tutoriales, al igual que el proceso seguido por los desarrolladores que participaron en el estudio de Kaur, se centran en la propia construcción y optimización del entorno virtual, preocupándose más de la máquina y el rendimiento que del usuario y los problemas de usabilidad. Por ello, su estudio llevó a Kaur a proponer un nuevo modelo de proceso en el que no sólo se definían en primer lugar los requisitos, sino que a continuación se especificaban y luego se diseñaban los componentes e interacciones en el entorno virtual, siguiendo unas guías de usabilidad propuestas por la misma autora. Tras ello se crea y se evalúa el entorno. Sin embargo, en su intento, Kaur pasa de puntillas por el proceso de construcción y optimización antes descrito, ocultándolo bajo el título “Creación del entorno”, con lo que si bien ahora se tiene en cuenta la usabilidad, se deja a un lado el objetivo de conseguir un buen rendimiento final –aunque el rendimiento influye en la propia usabilidad–.

El estudio de las anteriores metodologías confirma, pues, que crear un mundo virtual no es sólo modelar y añadir comportamientos. Primero hay que especificar los requisitos (p. ej. la plataforma destino), y especificar y diseñar los componentes e interacciones en el entorno siguiendo unas guías de diseño como las de Kaur, documentando todo usando bocetos y storyboards. Tras ello hay que prepararse para construir, reuniendo el material de referencia de los modelos reales, estructurando el modelo gráfico y realizando un reparto entre los desarrolladores. Entonces se construye el entorno, modelando y ensamblando, añadiendo animaciones e interacciones, y mejorando y optimizando el entorno

virtual hasta lograr el acabado y el rendimiento deseado, produciendo prototipos de forma incremental. Finalmente, el entorno es evaluado.

El modelo de proceso anterior, fundamentado en las metodologías incluidas en este grupo, puede parecer suficiente para el desarrollo de mundos virtuales y, de forma más general, de interfaces de usuario 3D. Nada más lejos. Por una parte, dichas metodologías poseen una fuerte orientación hacia la creación de contenido 3D basado, sobre todo, en modelos reales, que el usuario explora sin un objetivo concreto. Suponen, además, que el desarrollador conoce ya todos los detalles de ese contenido 3D, pues el proceso comienza inmediatamente con el diseño y creación del entorno virtual, sin plantear siquiera la posibilidad de tener que discutir y decidir entre diseños alternativos. Precisamente, una de las discusiones que no se plantea es la elección de dispositivos, pues se asume que la plataforma es un PC convencional, con su teclado y ratón. Tampoco prestan especial atención a la programación, pues generalmente los comportamientos de los objetos son sencillos, y los más complejos se resuelven con unas pocas líneas escritas en un lenguaje de script, de ahí la falta de guía al programador. Por último, y no menos importante, aunque autores como Kaur se preocupan por el usuario y la usabilidad, la participación del propio usuario en el proceso es más bien escasa.

### **3.4 Desarrollo participativo de mundos virtuales**

Además del equipo de desarrollo existen otras personas que también juegan un papel en la creación de mundos virtuales, como es el caso de los clientes y los futuros usuarios, que juntos conforman la audiencia del producto, y los expertos en el dominio, los cuales asesoran al equipo de desarrollo en el campo de aplicación. La comunicación con esas otras personas es necesaria para poder realizar un producto que se ajuste a los requisitos del cliente, que los expertos aprueben y del que los usuarios puedan hacer un buen uso.

Sin embargo, los fallos de comunicación pueden propiciar errores de diseño que retrasan el desarrollo. Por esta razón, algunos autores abogan por estrategias que hagan que esas otras personas participen y se impliquen más en el desarrollo, como las propuestas que se describen y analizan en este apartado.

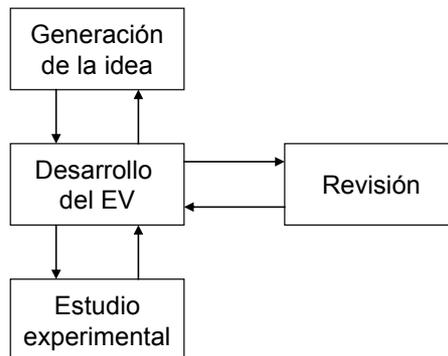
### 3.4.1 Desarrollo centrado en el usuario en el grupo VIRART

#### 3.4.1.1 Descripción

En [Neale, 2001] los autores explican los métodos seguidos para la creación de mundos virtuales por el grupo de investigación VIRART, un grupo multidisciplinar de la Universidad de Nottingham que incluye ergonomistas, psicólogos, ingenieros e informáticos. En particular, describe un proceso de desarrollo centrado en el usuario que es ilustrado con el desarrollo de una ciudad virtual. Esta propuesta rodea el desarrollo del entorno virtual con las siguientes tres actividades:

1. Generación de la idea.
2. Revisión.
3. Estudio experimental.

Gráficamente:



**Figura 3.3** Desarrollo de mundos virtuales centrado en el usuario según Neale y Nichols

Para la generación de la idea se definen dos grupos. Un primer grupo – “user group”- lleva a cabo tormentas de ideas, las discute y las plasma en storyboards. Un segundo grupo –“steering group”- revisa esas ideas, refina los storyboards y da prioridad a unos objetivos sobre otros.

En cuanto a la revisión, se llevan a cabo con usuarios y también por expertos. En las primeras fases del ciclo de vida del desarrollo, con el entorno virtual inacabado, los desarrolladores llevan a cabo revisiones más informales con los usuarios, con el fin de detectar las discrepancias más graves. En fases más

avanzadas se llevan a cabo revisiones más formales, con profesionales, usando cuestionarios y centrándose más en el contenido y no tanto en el diseño.

La última actividad que se cita es el estudio experimental acerca de la usabilidad, la diversión, el aprendizaje o la transferencia de conocimientos. El método seguido para este estudio incluye la recopilación de información de los usuarios, las medidas observadas, su análisis y la creación de informes.

### 3.4.1.2 Análisis

Dejando a un lado el trabajo más técnico que realiza el equipo de desarrollo, Neale y Nichols explican cómo el usuario puede participar aportando sus opiniones en tres momentos diferentes del desarrollo: al principio del mismo, durante la generación de las ideas que se concretarán en el entorno virtual; a lo largo del desarrollo, revisando con los diseñadores y profesionales el entorno; y a la conclusión del trabajo, participando en el estudio experimental que examina el resultado final.

Además de ese marcado carácter participativo, de esta metodología destaca, precisamente, esa etapa dedicada a la generación de la idea, no contemplada en ninguna de las metodologías analizadas hasta este punto, y que recuerda que el diseño no es un proceso mecánico sino creativo, y que en ese proceso de creación es importante revisar las diferentes alternativas antes de decidir el diseño a seguir.

Con la participación activa del usuario en el diseño, esta estrategia puede asegurar que el producto final sea de la satisfacción del usuario. La mera participación de este no asegura, sin embargo, que el contenido sea correcto, pues el usuario no siempre es quien puede ayudar en esa tarea, no tiene por qué ser un experto en el dominio. Sin otra ayuda, el equipo de desarrollo tendrá entonces que realizar un gran esfuerzo para familiarizarse con los conocimientos del dominio, esfuerzo que no asegura tampoco que el contenido sea correcto. Por ello es importante que participen expertos, y en este sentido se ve como un acierto que la propuesta promueva revisiones con profesionales, centradas no tanto en el diseño como sí en el contenido. Aunque cabe preguntarse por qué se dejan para fases más avanzadas y no se contemplan desde un principio, como se verá en la siguiente propuesta.

Con todo, lo que no puede asegurarse con este método es que el rendimiento sea el adecuado, objetivo que se marcaban la mayoría de las metodologías incluidas en el grupo precedente, lo que se deja entonces en manos del equipo de desarrollo.

### 3.4.2 El ciclo de diseño de interacción 3D según Celentano y Pittarello

#### 3.4.2.1 Descripción

En [Celentano, 2001a] y [Celentano, 2001b] se describe un ciclo de diseño de interacción 3D en el que se da mayor protagonismo a los expertos en el dominio, redefiniendo el significado de autor e introduciendo el rol de meta-autor.

Así, el autor pasa a ser una figura entre el usuario final y el creador de mundos virtuales (*world builder*), alguien conocedor del dominio pero que, sin tener grandes conocimientos en informática, también participa en la creación del mundo virtual dada la herramienta apropiada. La interfaz de esa herramienta es creada por el diseñador de interfaces, quien también colabora con el meta-autor. Este meta-autor, por su parte, es un gran conocedor del dominio, y se comunica con el usuario final.

El ciclo de diseño consta de las siguientes cuatro fases:

1. Fase conceptual.
2. Fase de implementación.
3. Fase de desarrollo del contenido.
4. Fase de interacción del usuario final.

La primera fase se caracteriza por la identificación del contenido y los requisitos de interacción. El meta-autor informa al diseñador de interfaces acerca del contenido, y con él debate la forma de tomar ventaja de la tecnología del momento. El resultado de esta fase es un conjunto de esquemas de interfaces de usuario (texto escrito o esquema visual) para definir lo que los autores llaman “clases de experiencias interactivas”, tanto para la interfaz del usuario final como la de los autores.

En una segunda fase, el diseñador de interfaces crea la interfaz de usuario final y la interfaz para el autor en base a los esquemas anteriores. Un resultado de esta fase es la herramienta que usará el autor, que podrá ser manipulada sin grandes conocimientos de informática, y que puede ser una personalización o un subconjunto de herramientas existentes, más que un desarrollo desde cero.

En la tercera fase, los autores eligen de entre las diferentes clases de experiencias interactivas y las concretan según sus necesidades (por ejemplo, una clase puede ser “Tour guiado”, y se concreta en caminos particulares). Durante esta fase, el trabajo de los autores es complementado por el escritor o el

artista de gráficos 2d (editores), el creador de modelos 3D y el creador de mundos virtuales.

Finalmente, en una cuarta fase el usuario final puede interactuar con los contenidos del mundo 3D resultado de la composición del autor, a través de la interfaz implementada por el diseñador de interfaces. La interacción del usuario final es monitorizada buscando mejorar tanto la usabilidad de la interfaz como la efectividad en la comunicación del contenido.

### 3.4.2.2 Análisis

Como ciclo de diseño de interacción 3D, de la propuesta de Celentano y Pittarello cabe subrayar el que se tenga en cuenta en una primera fase cuál es la tecnología disponible y qué oportunidades ofrece, que el experto en el dominio tenga un papel activo en la creación del contenido, y que evalúe la interacción del usuario con el producto final. Todo ello va encaminado a ofrecer un producto que se adecue a la tecnología, con un contenido correcto y que sea usable por el usuario.

Así, el debate que promueve entre el diseñador y el experto en el dominio sobre cómo aprovechar la tecnología actual ha sido pasado por alto en metodologías como las analizadas en el grupo anterior, con la única excepción del tutorial de B. Hay, cuya etapa de planificación incluye comprender las limitaciones de la tecnología.

En cuanto al experto en el dominio, este adquiere un papel protagonista al asumir los roles de autor y meta-autor. Para ello, se le facilitan herramientas cuya interfaz ha sido adecuada por el diseñador de interfaces para que puedan ser usadas sin grandes conocimientos de informática, y con un repertorio de clases de experiencias interactivas que poder concretar en el producto en desarrollo sin tener grandes conocimientos de interacción.

Sin embargo, más que promover la participación en el desarrollo del experto en el dominio, este ciclo parece girar en torno a esa figura, supeditando el trabajo del equipo de desarrollo a ella. Precisamente, ese equipo es el que trabaja para facilitar la labor al experto en el dominio, pero no se detalla ese trabajo, dejándose pues al buen hacer del equipo. Por ejemplo, no hay conexión entre la creación de la interfaz de usuario final por parte del diseñador de interfaces y el trabajo de los editores, el creador de modelos 3D o el creador del mundo.

En cualquier caso, esta propuesta pone de relieve el problema que supone para un desarrollador de mundos virtuales familiarizarse con el dominio, plasmarlo

adecuadamente en el diseño, y representarlo correctamente en el mundo virtual. Igualmente, saca a relucir los diferentes roles que intervienen en un desarrollo, cuando en otras metodologías sólo se ha hablado de diseñador o desarrollador, y casi indistintamente, y en realidad las tareas a realizar suelen ser tan diferentes que es necesario contar con profesionales especializados en cada una de ellas.

### 3.4.3 Análisis del grupo

El último comentario que se incluía en el análisis del grupo de metodologías anterior a este hacía referencia a la escasa participación del usuario, aspecto que se ha abordado en este nuevo grupo, en el cual se han incluido dos propuestas diferentes.

La primera metodología descrita y analizada en este grupo, y propuesta por Neale y Nichols, posee una fuerte orientación hacia el usuario y su participación, promoviendo ésta en tres etapas concretas del proceso de desarrollo. Una de ellas es la generación de la idea, que bien podría enlazarse con el debate sobre la tecnología y sus ventajas que se incluye en la primera fase de la segunda metodología, propuesta por Celentano y Pitarello. También coinciden en la evaluación final del entorno, donde también participa el usuario. Sin embargo, no es esa la mayor preocupación de esa segunda metodología, más dirigida hacia el experto en el dominio y su participación.

Puede pensarse, entonces, que ambos modelos de proceso son complementarios. Así, para asegurar que el producto final sea de la satisfacción del usuario y que su contenido sea correcto, debería facilitarse la participación del usuario y del experto en el dominio en diferentes momentos a lo largo del proceso de desarrollo. Sin embargo, estas metodologías se centran en esos dos roles y en sus interacciones con los demás miembros del equipo, no prestando mayor atención a las actividades propias de estos otros, por lo que resultan insuficientes.

## 3.5 Uso del análisis de tareas en la creación de mundos virtuales

En ninguna de las propuestas descritas hasta el momento se ha tratado de forma adecuada las tareas del usuario. Podría argumentarse que son abordadas en alguna de las etapas de requisitos, planificación o diseño que se incluyen en algunas de esas propuestas, pero en ningún caso se resalta su importancia.

En [Gabbard, 1999] se afirma que el análisis de tareas tal vez sea una de las fases que más se pasan por alto en la ingeniería de la usabilidad, y es una de las más importantes. Por lo visto hasta el momento, parece que esa afirmación

también es aplicable al desarrollo de mundos virtuales. En este apartado se verán propuestas metodológicas que sí tienen en cuenta el análisis de las tareas del usuario.

### 3.5.1 Evaluación secuencial según Gabbard

#### 3.5.1.1 Descripción

En [Gabbard, 1999] –y más tarde también en [Bowman, 2002]– se describe una aproximación a la evaluación de entornos virtuales basada en la realización de:

1. Análisis de las tareas del usuario.
2. Evaluación heurística (o experta basada en guías).
3. Evaluación formativa (o centrada en el usuario).
4. Evaluación sumativa y comparativa.

Los autores explican que esta aproximación puede aplicarse estrictamente en cadena o muchas veces de forma iterativa, bien realizando ciclos sobre el proceso en conjunto o sobre métodos individuales.

Con respecto a las tareas del usuario, Gabbard *et al.* entienden que son aquellas que el usuario necesita ser capaz de hacer. Para llevar a cabo el análisis de las mismas, antes es preciso:

- 0.1. Comprender al usuario, el flujo de trabajo social y de la organización.
- 0.2. Realizar un modelado del usuario y el análisis de requisitos.
- 0.3. Definir las clases de usuario, los objetivos del sistema y las características básicas.

A partir de lo obtenido, el análisis de las tareas de usuario genera:

- 1.1. Una descomposición top-down de descripciones detalladas de tareas.
- 1.2. Secuencias de tareas y su semántica.
- 1.3. Ordenación, relaciones, e interdependencias entre las tareas del usuario.
- 1.4. Trabajo del usuario y flujo de información.

El análisis de las tareas del usuario también da forma a los escenarios representativos de las tareas de usuario, útiles para la realización de las evaluaciones que siguen. Con respecto a esas evaluaciones, cabe remitir al lector a los apartados 2.7.2 y 2.7.10 del capítulo anterior.

### 3.5.1.2 Análisis

Se trata de un enfoque para la evaluación de entornos virtuales basada en el análisis de las tareas de usuario y tres tipos de evaluación, con lo que se pretende conocer en profundidad al usuario, sus tareas y su contexto, y cuidar que el producto final sea usable, todo ello en un proceso secuencial pero que también puede ser cíclico.

Así, el análisis de tareas da forma a los escenarios representativos con los que, posteriormente, llevar a cabo los diferentes tipos de evaluación que propone. Los autores explican qué pasos dar previos al análisis de tareas, y qué se obtiene como resultado del mismo, aunque no detallan cómo se lleva a cabo.

En cuanto a los tres tipos de evaluación propuestos, la evaluación experta persigue detectar los errores más graves –que violan las guías descritas en [Gabbard, 1997] y comentadas en el apartado 2.7.10-, la formativa los más sutiles, y la sumativa permite comparar diferentes configuraciones de la interfaz (paradigmas, dispositivos, etc.).

Sin embargo, como enfoque para la evaluación que es, no representa todo el proceso necesario para la creación de un entorno virtual. En [Bowman, 2002] se indica que debería usarse desde el principio del ciclo de diseño y de forma continuada a lo largo del mismo, pero no detalla ese ciclo de diseño. El análisis de tareas puede ser un buen comienzo, y la evaluación continuada una garantía para lograr un buen producto, pero el resto queda al buen hacer del desarrollador.

## 3.5.2 Metodología para la creación de ambientes virtuales 3D

### 3.5.2.1 Descripción

En [Pereira, 2000] y en [García, 2001] se describe una metodología para la creación de entornos virtuales que propone el Laboratorio de Realidad Virtual de la Universidad Federal de Santa Catarina, y más concretamente para el diseño de interfaces para esos entornos. Según sus autores, las técnicas para el desarrollo de este tipo de interfaces son semejantes a las utilizadas en el desarrollo de otras interfaces de ordenador. Los autores afirman incluso que muchos criterios de diseño de interfaces 2D pueden ser fácilmente aplicados a las interfaces 3D, aunque destacan que la principal diferencia es la distribución de contenido y de elementos.

La metodología describe una secuencia de actividades agrupadas en cuatro fases:

1. Análisis (diagramas de flujo).
2. Concepción (árbol del sistema).
3. Proyecto de interfaz (diseño).
4. Desarrollo (implementación).

En la fase de análisis se identifica el público al que va dirigido el entorno virtual, se determinan y se analizan las necesidades, se validan los requisitos y se realizan sesiones de ajustes y clasificaciones. Los autores aconsejan el uso de diagramas de flujo (*fluxogramas*) como una buena forma de trazar los objetivos, indicando tanto los caminos a ser seguidos como las tareas a ser ejecutadas.

En la fase de concepción se especifican y se reparten las funciones hombre-máquina. También se especifican los caminos de navegación, así como las actividades y las tareas interactivas.

En la fase de diseño o, como la denominan los autores, de proyecto de interfaz, se aborda la metáfora del proyecto gráfico, se conciben las tareas a ser elaboradas por el usuario en el entorno, se crea un storyboard, se disponen los contenidos y se conciben los elementos de la interfaz.

Finalmente, en la fase de desarrollo o implementación se crean maquetas, prototipos y versiones crecientes, se producen los elementos de la interfaz y se programan las tareas a ser ejecutadas dentro del ambiente virtual.

### 3.5.2.2 Análisis

Para los autores de esta metodología, las tareas del usuario representan uno de los tres factores principales en los que debe basarse el desarrollo de una interfaz de un entorno virtual, junto a las necesidades de los usuarios –novatos y más experimentados- y los criterios ergonómicos de la IPO.

Así, puede verse como esas tareas y actividades del usuario se abordan explícitamente en momentos distintos de la metodología: en la etapa de análisis, se indican las tareas a ser ejecutadas; en la etapa de concepción, se aborda la especificación de las tareas interactivas; en la etapa de diseño, la concepción de las tareas a ser realizadas en el entorno; y en la etapa de desarrollo, la programación de esas tareas. En este reparto, resulta curioso que las tareas se conciban en la etapa de diseño y no en la de concepción, aunque más que curioso es confuso, y una muestra de los problemas que crea la terminología utilizada por los distintos autores al describir sus propuestas. Más interesante

aún, es que en la etapa de diseño se puntualice que las tareas que se conciben son las que serán realizadas en el entorno, y cabe preguntarse si con ello los autores persiguen distinguir entre estas y las que se indican en el análisis, como se verá luego en la metodología VEDS.

También muy interesante, y es por ello que se incluye esta metodología en esta sección del capítulo, es que los autores insisten en que las tareas y las actividades que serán realizadas por los usuarios deben pasar por un proceso de análisis que determine la necesidad y viabilidad de representación por medio de un entorno virtual. Y es que, como luego diría Sutcliffe, muchas las aplicaciones no son apropiadas para la Realidad Virtual [Sutcliffe, 2003], o como se afirmaría en [D’Cruz, 2003], es importante reconocer que la Realidad Virtual puede no ser la única solución o la más apropiada, y que su tecnología es todavía relativamente nueva.

Sin embargo, las mayores dudas que deja la metodología brasileña están en los contenidos 3D. Según los autores, un entorno virtual es un entorno donde el usuario navega para buscar él mismo la información, ejecutar las actividades y desarrollar las tareas. Pero la principal diferencia con las interfaces 2D es, según ellos, la disposición de esos contenidos. Volviendo entonces a la metodología, los contenidos se disponen en la etapa de diseño, pero no se dan mayores detalles. También se habla de especificar los caminos de navegación en la etapa de concepción, lo que podría entenderse como caminos en el espacio 3D, pero a falta de una aclaración lo más probable es que atienda a la navegación por pantallas típica de las interfaces 2D.

Con todo, se trata de una metodología que enumera un buen número de actividades en una secuencia similar a la que se emplea en interfaces más convencionales, pero que los autores no llegan a describir por completo, dejando muchas dudas acerca del proceso real que hay detrás de los nombres de esas actividades.

### **3.5.3 VEDS: Virtual Environment Development Structure**

#### **3.5.3.1 Descripción**

VEDS es el resultado de años de trabajo en el campo de los entornos virtuales en el grupo VIRART de la Universidad de Nottingham, grupo del que ya se ha descrito una propuesta de desarrollo centrado en el usuario –[Neale, 2001], apartado 3.4.1-. VEDS se describe en [Eastgate, 2001], [Wilson, 2002] y [D’Cruz, 2003], por citar algunas referencias. A lo largo de estos años,

diferentes autores han contribuido a refinar VEDS. Así, Eastgate explica que la primera versión data de 1994, aunque el autor toma como punto de partida la versión de 1998, e introduce nuevos cambios como resultado de su trabajo doctoral.

D’Cruz *et al.* definen VEDS como un marco de trabajo (*framework*), holístico – el todo es distinto a la suma de las partes-, y centrado en el usuario, para la especificación, desarrollo y evaluación de aplicaciones de EV’s. Estos mismos autores resumen VEDS en los siguientes aspectos:

1. Requisitos.
2. Atributos del EV y objetivos de la aplicación.
3. Prioridades y restricciones.
4. Análisis funcional de tareas y análisis del usuario.
5. Objetivos del EV.
6. Diseño del concepto.
7. Análisis de las tareas en el EV.
8. Configuración del sistema.
9. Especificación de la apariencia y la interacción, y especificación de claves y realimentación.
10. Construcción.
11. Evaluación.

D’Cruz *et al.* explican que las primeras etapas del desarrollo del EV conllevan determinar qué tareas y funciones deben ser completadas en el EV, cuáles son las características del usuario y sus necesidades, y ubicar y dividir funciones dentro del EV. Para ello se utilizan métodos como las entrevistas, grupos de trabajo (*focus groups*) y el análisis de las tareas en el mundo real.

Así, a través de reuniones con miembros relevantes del personal, pueden definirse un conjunto de escenarios que el usuario reconozca como actividades clave. El análisis del usuario y de las tareas se realiza entonces revisando paso a paso cada escenario, preferiblemente con usuarios que realizan esas tareas, anotando qué hacen y piensan, y usando técnicas como la observación directa, análisis de videos y fotografías, protocolos verbales, entrevistas o cuestionarios.

Los objetivos del EV son proporcionados por los implicados tras la etapa de análisis y, además de ser la principal influencia en el diseño del EV, también pueden usarse como criterio del éxito del EV. Para una mejor comprensión de esos objetivos, en el diseño del concepto y en el análisis de las tareas en el EV se pueden usar métodos tales como la creación de storyboards. También pueden celebrarse sesiones de trabajo (*focus groups*) con miembros clave del personal, expertos IPO y desarrolladores. El storyboard es revisado de forma iterativa prestando especial atención a aspectos de interacción y usabilidad.

Las siguientes etapas antes de la evaluación final no son descritas por D’Cruz *et al.*, pero sí por Eastgate y con gran detalle. De hecho, fue aquí donde se centró su trabajo doctoral y donde introdujo más modificaciones sobre la versión anterior de VEDS. Según Eastgate, tras las etapas de preparación (pasos del 1 al 3), análisis (paso 4) y especificación (pasos del 5 al 8) ya descritas, le siguen las siguientes, que vendrían a corresponderse con los pasos 9 y 10 del anterior listado:

- Diseño de conjunto del EV.
- Adquisición de recursos.
- Diseño de los detalles del EV.
- Construcción del EV.
- Prueba del EV.
- Implementación.

Eastgate no se queda en el nombre de las etapas sino que describe también las actividades que cada una de ellas incluyen. Por ejemplo, la etapa de construcción del EV incluye la creación de los modelos 3D y su ensamblado, algo común en otras propuestas anteriores, pero también la programación de la dinámica y los comportamientos, asociación de sonidos, correspondencia de texturas (*texture mapping*), cajas de texto y botones, y puntos de vista. Además, Eastgate ayuda al desarrollador en la realización de ciertas etapas con herramientas de guía que él mismo propone, y que fueron comentadas en el apartado 2.7.10.

Es importante aclarar que la etapa que Eastgate nombra como implementación no hace referencia a ningún trabajo de programación, sino a la implantación o despliegue del sistema creado, tras su construcción y prueba.

Con respecto a la última etapa, la de evaluación (paso 11), D’Cruz *et al.* explican que debe realizarse tanto del propio entorno como de su uso y utilidad, pudiendo dividir dicha evaluación en un examen de la validez, los resultados, la experiencia del usuario y el proceso.

### 3.5.3.2 Análisis

Esta propuesta ha sido incluida en esta sección por la importancia que otorga al análisis de tareas, y muy especialmente por la distinción que realiza entre las tareas en el mundo real y las tareas en el EV, lo que lleva a incluir no una sino dos etapas de análisis de tareas. Esta distinción es fundamental, pues recuerda al desarrollador que, dependiendo de los objetivos de la aplicación, las tareas en el EV pueden diferir de las tareas en el mundo real, y aunque se pretenda que sean

iguales no debe olvidarse que la Realidad Virtual otorga nuevos poderes pero también impone limitaciones, como decía [Kaur, 1998].

Precisamente, sobre esas limitaciones, los autores afirman que con su propuesta pueden ser afrontadas o ver sus efectos reducidos gracias a una apropiada selección de la tecnología y una cuidada especificación del EV que cumpla los objetivos. También afirman que puede resolver ese equilibrio entre fidelidad visual e interactividad que en anteriores apartados se vio que preocupaba a los creadores de mundos virtuales. Y la usabilidad puede ser mejorada no sólo a través de la selección del hardware más apropiado, sino también con la inclusión de ayudas en la interfaz.

Para ello, este enfoque estructurado se apoya también en el uso de técnicas de interacción persona-ordenador, como en las ya descritas anteriormente por Neale y Nichols para un diseño centrado en el usuario, y en herramientas que guían al desarrollador, como las que propone Eastgate en su trabajo doctoral. Cabe subrayar que, en el caso de las herramientas de Eastgate, el autor utiliza formatos que faciliten su uso por parte del desarrollador –igual propósito que Kaur para su herramienta [Kaur, 1998]-, ya sea un diagrama de flujo, una lista de verificación o una tabla.

Con todo, lo que más destaca de esta propuesta es la profundidad con la que estructura todo el proceso, sobre todo tal y como lo describe Eastgate, enumerando más actividades que cualquier otra metodología vista.

Así, esta metodología comienza con el análisis de los requisitos, los objetivos de la aplicación, las tareas y el usuario, de forma similar a las dos metodologías ya descritas en este grupo. En relación a la propuesta de Gabbard *et al.*, en VEDS el análisis de tareas no da forma a los escenarios, sino que los escenarios se identifican primero para proceder a ese análisis, enumerando además D’Cruz *et al.* las herramientas necesarias para llevarlo a cabo. En relación a la propuesta brasileña, en VEDS también se cuestiona si un EV es la solución más apropiada.

Sin embargo, a diferencia de esas propuestas anteriores, en VEDS se distingue explícitamente entre las tareas en el mundo real y las tareas en el mundo virtual, como ya se ha comentado. Así, tras los análisis anteriores la metodología continúa fijando los objetivos del EV, se realiza entonces un diseño del concepto –el cual bien podría relacionarse con la etapa de generación de la idea que describen Neale y Nichols, apartado 3.4.1- y se analizan las tareas tal y como se llevan a cabo en el EV. Seguidamente, se configura el sistema, se diseña y construye el EV, seguido por la prueba del mismo, despliegue (“implementación” según Eastgate) y evaluación. Del diseño cabe destacar la distinción entre el diseño del conjunto y el diseño de los detalles, y entre medias la adquisición de los recursos –recordando la recogida del material de referencia

que apuntaban los desarrolladores entrevistados por Kaur, apartado 3.3.2-. Precisamente, ese diseño, junto con la construcción del EV, es en sí ya comparable a todo el proceso que se describe en una metodología bien detallada como es el tutorial de B. Hay comentado anteriormente –véase [Hay, URL] y el apartado 3.3.4-, y que ridiculiza otras propuestas basadas sólo en dos o tres pasos.

No obstante, aún hay aspectos que se echan en falta en un enfoque estructurado para el desarrollo de aplicaciones de Realidad Virtual, y es que los casos de estudio que se describen –especialmente en Eastgate- están basados en PC, por lo que la atención a dispositivos hardware más exóticos y sus técnicas de interacción asociadas es mucho menor, y al ser realizados con entornos de desarrollo como Superscape VRT, tampoco aborda la problemática de la programación de los componentes software. Igualmente, no resuelve la gestión del tiempo y el presupuesto, algo fundamental en un proyecto de estas características.

### 3.5.4 Ingeniería de entornos virtuales con X3D según Polys

#### 3.5.4.1 Descripción

En [Polys, 2005], y más concretamente en la primera de las cuatro partes de ese tutorial y cuyo autor es N.F. Polys, se esboza la creación de mundos virtuales con el lenguaje estándar X3D apoyándose tanto en el diseño basado en escenarios como en estructuras tarea-conocimiento (TKS, *Task-Knowledge Structure*).

Polys sitúa los escenarios como herramienta en la ingeniería de la usabilidad y en la ingeniería del contenido –el autor hace referencia a [Rosson, 2002] y [Norman, 1986]-, relacionando actividades, información e interacción. Con respecto a la estructura tarea-conocimiento, ésta modela la información que necesita el usuario para completar sus actividades como diagramas Entidad-Relación (E-R) para tareas y medios –Polys hace referencia en esta ocasión a [Sutcliffe, 1994]-.

Con los escenarios y las estructuras TKS, el proceso de producción de una escena típica seguirá, según este autor, los siguientes pasos:

1. Definir entorno y lugares.
2. Definir la interfaz de usuario y los puntos de vista.
3. Definir las interacciones.
4. Organizar el grafo de la escena declarativo.

5. Modelar los objetos.
6. Construir prototipos.
7. Transformar datos y componer marcas visuales.
8. Entregar al usuario.

#### 3.5.4.2 Análisis

El uso de escenarios ya ha sido introducido en propuestas anteriores de este mismo grupo, tanto por Gabbard *et al.* como en la metodología VEDS, de modo que la novedad del enfoque que propone Polys radica en el uso también de estructuras TKS para el análisis de las tareas del entorno virtual. Como novedad, el empleo de esas estructuras TKS, al fin y al cabo diagramas E-R, contrasta además con las herramientas de diseño usadas en las metodologías hasta aquí analizadas, normalmente bocetos y storyboards.

Con esas armas, Polys presenta entonces un proceso de producción basado en definir y organizar antes de modelar los objetos y construir los prototipos, de forma similar a otras propuestas en donde una planificación precede a la construcción, como por ejemplo en los tutoriales para la creación de mundos virtuales con VRML/X3D descritos en el apartado 3.3.4, comparación muy apropiada pues no en vano la propuesta de Polys está dirigida a la creación de entornos virtuales con el lenguaje X3D. Igualmente, la organización del grafo de la escena recuerda a la estructuración del modelo gráfico que observó Kaur en su estudio –véase el apartado 3.3.2-. Más aún, los tres primeros pasos del proceso que propone Polys –definir el entorno y los lugares, la interfaz de usuario, los puntos de vista y las interacciones- recuerdan a la especificación de los componentes y las interacciones de la redefinición del proceso que propuso la propia Kaur –véase el apartado 3.3.3-.

Sin embargo, no queda claro cómo se relacionan los escenarios, las estructuras TKS y el proceso. Puede pensarse que el análisis de tareas es un paso previo al proceso de producción, pero Polys no lo indica de forma explícita. Y dentro del proceso, no queda claro el paso dedicado a la construcción de prototipos. Al hablar de la ingeniería de la usabilidad, Polys indica que el proceso de diseño es repetido hasta lograr un objetivo predefinido de rendimiento o satisfacción. Puede entonces pensarse que la producción de la escena se hará de forma iterativa, pero Polys no indica dónde se define ese objetivo de rendimiento o satisfacción, ni tampoco en base a qué. Por estas y otras razones cabe concluir que, aunque la propuesta es sensata, su aplicación dejaría muchos aspectos en manos del buen hacer del desarrollador.

### 3.5.5 Análisis del grupo

Frente al énfasis que se ha dado al contenido en los grupos precedentes de metodologías, este nuevo grupo se ha caracterizado por su orientación hacia las tareas que debe realizar el usuario en el entorno virtual, en general sumando el análisis de tareas al modelo de proceso.

La primera metodología descrita y analizada ha sido la de Gabbard *et al.*, los cuales no abordan el diseño o construcción del entorno virtual, pero sí el análisis previo y la evaluación posterior. Ese análisis da forma a los escenarios representativos que luego son utilizados en las distintas evaluaciones del entorno virtual. Se puede deducir entonces que el diseño y construcción del entorno no cambian esos escenarios. Sin embargo, la forma en que se realizan las tareas en el entorno virtual bien puede diferir de cómo se observan en la realidad, de modo que si ese análisis corresponde al mundo real, las evaluaciones posteriores del entorno virtual podrán observar tareas bien diferentes.

La anterior diferencia no es advertida por estos autores, pero es de gran importancia, como sí se destaca en la metodología VEDS al incluirse dos procesos de análisis de tareas, un primer análisis de las tareas en el mundo real y otro análisis, más adelante en el desarrollo, de las tareas en el mundo virtual. En otra de las metodologías estudiadas en este apartado, y propuesta esta vez por un grupo brasileño, parece existir también esta diferencia, entre las tareas identificadas en la etapa de análisis y la concepción, ya en la etapa de diseño, de las tareas a ser elaboradas por el usuario en el entorno. Sin embargo, la breve descripción de estos autores brasileños deja muchas dudas al respecto.

Con todo, era en los contenidos 3D donde más dudas dejaba esa metodología brasileña. Como se vio en el apartado 3.5.2, esta metodología indica que los contenidos se disponen en la etapa de Diseño, pero no se dan mayores detalles. Y es que el problema no es sólo distinguir entre las tareas en el mundo real y las tareas en el mundo virtual, sino también relacionar tareas y contenido, pues si las metodologías de grupos anteriores estaban fuertemente dirigidas hacia el contenido, propuestas como esta parecen dejarlo a un lado para dirigir su atención hacia las tareas.

Así, en la metodología VEDS es en el proceso de construcción y en el previo de diseño –también en el diseño del concepto– donde se aborda el contenido 3D, aunque tal vez la relación entre tareas y contenido 3D queda más clara en la metodología propuesta por Polys, basándose en el uso de escenarios y estructuras TKS. Sin embargo, lo que no deja muy claro este último autor es la forma en que se relaciona el uso de esas herramientas con el diseño y construcción del entorno.

En cualquier caso, las metodologías incluidas en este grupo son una buena muestra de que el análisis de tareas también tiene su lugar en el proceso de desarrollo de entornos virtuales. Por un lado, incluido en una etapa de análisis previa al diseño del EV, en la que se analizan los requisitos del sistema, las tareas y el propio usuario. Por otro lado, incluido en una etapa de diseño del EV, en la que fijan los objetivos para el EV, se realiza un diseño del concepto y entonces se analizan las tareas tal y cómo se llevarán a cabo en ese entorno virtual, para después proceder al diseño del conjunto del EV y de cada uno de sus detalles. Una vez definido el EV y con todo preparado para construirlo, se procede a ello creando prototipos y versiones crecientes del mismo hasta que, una vez terminado, es entregado. A lo largo de todo este desarrollo el EV es continuamente evaluado y, por supuesto, también al final.

El anterior modelo de proceso tiene su fundamento en las cuatro metodologías estudiadas en este apartado, y aunque pueda parecer suficiente para el desarrollo de mundos virtuales y, con ello, de interfaces de usuario 3D en general, de nuevo no es así. Por una parte, el cambio de orientación de estas metodologías hacia las tareas destapa un nuevo problema, y es la forma en que esas tareas se relacionan con el contenido. Puede que el uso de escenarios y estructuras TKS como propone Polys sea parte de la solución, aunque la introducción de esas herramientas también destapa otro problema, y es precisamente la escasez que hasta ahora han mostrado las metodologías en herramientas, usualmente bocetos y storyboards. Por último, sigue asumiéndose que la plataforma es un PC convencional, y sigue sin prestarse mayor atención a la programación.

### **3.6 Desarrollo basado en ingeniería del software**

Las propuestas más ingenuas (naïve) de creación de mundos virtuales centraban su atención sobre la estética de los modelos, lo que generalmente llevaba a objetos visualmente complejos que ni eran apropiados para una representación en tiempo real ni estaban libres de problemas de usabilidad. Por ello, las siguientes propuestas ponían más énfasis en conseguir el equilibrio entre complejidad y rendimiento, y en lograr una interfaz usable.

Sin embargo, otro de los problemas que presentaban esas propuestas naïve era que no ofrecían ningún marco metodológico para el desarrollador más allá de la creación de los modelos y su importación en el entorno de desarrollo, como se apuntaba en [Kim, 1998]. Por ello, algunos autores ven en la ingeniería del software una solución a esta carencia, proporcionando además herramientas que, como los diagramas, pueden ayudar al diseñador de mundos virtuales a capturar

los detalles del proyecto. A continuación se describirán las propuestas de estos autores.

### 3.6.1 Metodología de diseño para EV's según Fencott

#### 3.6.1.1 Descripción

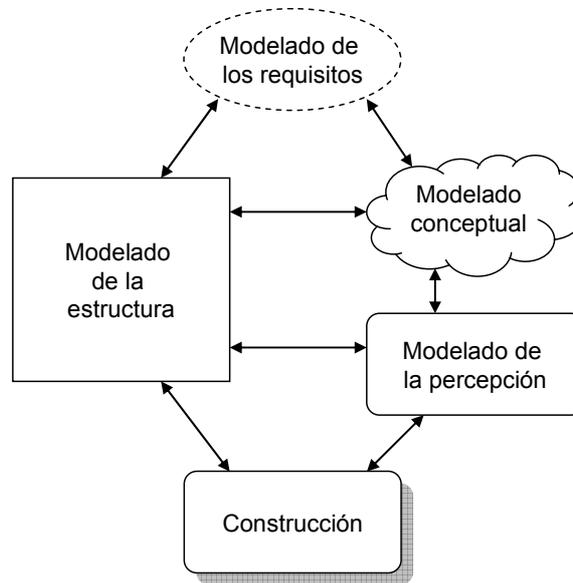
En [Fencott, 1999] -también en [Fencott, 2001], y más recientemente en [Fencott, 2005a] y [Fencott, 2005b]- se describe una metodología basada en la práctica de diseño de EV's observada por Kaur y que resumió en cinco etapas – véase apartado 3.3.2 y también [Kaur, 1998]-.

Fencott redefine el proceso de diseño de EV's basándose en su percepción del mismo como una tensión entre la estética (*aesthetics*) y el diseño ingenieril, en términos de percepción y de estructura. En este sentido, Fencott opina que si bien la ingeniería del software no puede ayudarnos con el modelado de esa percepción, sí puede hacerlo con el modelado estructural.

A semejanza de la práctica observada por Kaur, la metodología de diseño que propone Fencott también consta de cinco fases, que son las siguientes:

1. Modelado de los requisitos.
2. Modelado conceptual.
3. Modelado de la estructura.
4. Modelado de la percepción.
5. Construcción.

El primer punto –el modelado de requisitos- se corresponde con la primera etapa que describe Kaur –la especificación de requisitos-, indicando Fencott que se asemeja mucho al concepto que se tiene en la ingeniería del software. Los tres pasos siguientes no son una sucesión estricta, sino que el modelado de la estructura comienza en paralelo al modelado conceptual, y seguiría también en paralelo al modelado de la percepción. Gráficamente:



**Figura 3.4** Diseño de entornos virtuales según Fencott

Según Fencott, el modelado conceptual equivale también al segundo punto que describe Kaur –recogida de material de referencia del mundo real-. Se trata, según el autor, de la actividad de estudio común a muchos proyectos de diseño pero en especial aquellos con un componente estético. Se recoge material, se toman fotografías, se hacen bocetos, se graba sonido y video, y se construyen tiras de viñetas y storyboards. Las técnicas de los artistas y animadores son aplicables aquí. Fencott subraya que un resultado importante de esta fase es la elección del género (*genre*) más apropiado para cumplir con los requisitos, y que tendrá su influencia en la fase de modelado de la percepción.

En la fase de modelado de la estructura se empieza tomando decisiones acerca de las dimensiones, y se construyen planos y diagramas. Esta fase equivale al punto tres que describe Kaur –estructurar el modelo gráfico y dividir entre los diseñadores-, y es aquí donde, según explica Fencott, puede incorporarse la práctica de la ingeniería del software, haciendo uso por ejemplo del lenguaje UML. Así, Fencott indica que en las etapas más tempranas pueden emplearse diagramas de casos de uso para identificar las relaciones entre el usuario y el entorno virtual. Más adelante, modelos de los objetos pueden mostrar su estructura de nodos en el grafo de la escena, así como diagramas de clases para los componentes programados. El resultado de esta fase es un diagrama del grafo de la escena que muestra la estructura del código del EV y sus componentes de comportamiento.

La fase de modelado de la percepción equivale a grandes rasgos al punto cinco que describe Kaur –mejora del entorno-, y su objetivo es modelar la experiencia que se pretende que tenga el usuario del EV. Para ello, Fencott propone utilizar en esta fase Mapas de Percepción (*Perceptual Maps*) para asegurar que el orden en el tiempo de las atenciones y actividades del usuario en el EV es el adecuado. Estos mapas levantan una estructura meta-narrativa que se continúa en la fase de modelado de la estructura, pero cuya naturaleza depende del género elegido en la fase de modelado conceptual. Más concretamente, estos mapas estructuran lo que Fencott llama Oportunidades de Percepción (*Perceptual Opportunities*, POs), que se clasifican según el papel de cada objeto en la experiencia. El autor propone utilizar entonces su Modelo de Oportunidades de Percepción (*Perceptual Opportunities Model*), descrito junto a Isdale en [Fencott, 2001], y que consiste, como se vio en el apartado 2.5.2, en un conjunto de categorías sintácticas que pueden verse como posibles atributos de cualquier objeto de un EV –véase Figura 2.3-. Por ejemplo, en el primer nivel de este modelo de clasificación encontramos: certezas –*sureties*, lo que el usuario da por seguro y que confiere credibilidad al EV-, sorpresas, y sobresaltos (*shocks*).

La última fase de esta metodología es la construcción, y que Fencott describe como el proceso de utilizar herramientas para la codificación del grafo de la escena y del propio código del programa, ya sea con Java/VRML, VRToolkit, etc.

### 3.6.1.2 Análisis

Esta metodología está basada en el estudio realizado por Kaur de la práctica habitual de los desarrolladores, y por ello pueden identificarse similitudes que el propio Fencott subraya. Así, el autor establece una correspondencia uno a uno entre las etapas que describe Kaur y las fases de esta nueva metodología. Sin embargo, existen notables diferencias. Empezando por el orden, mientras las etapas descritas por Kaur pueden entenderse como una secuencia estricta, las fases de Fencott no lo son, y el autor indica además relaciones bidireccionales entre ellas. Además, la última fase descrita por Kaur es la mejora del entorno, mientras que para Fencott es la construcción, en consonancia con otras propuestas que abogan por realizar un buen diseño antes de comenzar el modelado y ensamblaje del mundo.

Podría decirse que la redefinición que lleva a cabo Fencott plasma su percepción del diseño como una tensión entre la ingeniería y la estética, lo que recuerda a Shneiderman diciendo que los diseñadores deben conjugar lo técnico y lo estético –véase apartado 3.1 y [Shneiderman, 1998]-. Más concretamente, el autor destaca dos tensiones básicas: en la primera sitúa en un extremo la

representación de un objeto (simulación, connotación), y en el otro lo que se pretende con él (comunicación simbólica, denotación); en la segunda, compara la estructura del grafo de la escena, jerárquica, con la de los mapas de percepción, que es espacio-temporal.

Así, por una parte, Fencott sitúa el papel de la ingeniería del software en dos fases: el modelado de requisitos y el de la estructura. En esta última sugiere el uso de diagramas UML, haciendo referencia al trabajo de P.G. McIntosh, quien utilizaba UML para enseñar VRML a alumnos de arquitectura. Aunque la página Web a la que Fencott hacía referencia ya no está disponible, ese trabajo puede verse en [McIntosh, 2000], referencia ya comentada en el apartado 2.7.7, y en donde McIntosh hace uso del lenguaje UML para analizar un entorno virtual VRML siguiendo el modelo SBF (estructura, comportamiento, función). Sin embargo, la propuesta de Fencott va en el otro sentido, esto es, modelar primero con UML para después obtener el grafo de la escena, lo que sí se ilustra en [Huda, URL2]. Huda, continuando las enseñanzas de McIntosh, concluye que ese camino también es posible, utilizando diagramas UML en una primera fase de transformación del modelo conceptual a formas gráficas, y comparándolo con los esquemas del proceso de diseño arquitectónico. Sin embargo, Huda se limita a usar diagramas de clases para la forma y de casos de uso para la función.

En cuanto a la experiencia del usuario en el EV, el Modelo de Oportunidades de Percepción es sin duda la principal contribución de este autor. En [Fencott, 2001], junto a Isdale, estos autores proponen además lo que denominan “la estética Church-Murray de la RV”, a partir de estudios previos en medios digitales y juegos de ordenador, y que persigue proporcionar una perspectiva de lo que es exactamente necesario diseñar cuando se crean EV’s. Las Oportunidades de Percepción (PO’s) pueden verse entonces como el siguiente paso a dar en el modelado del contenido. Fencott indica además que las PO’s y su estructuración en mapas de percepción podrían ser útiles también para evaluar la usabilidad, llevando a cabo experimentos con los que comprobar si las predicciones se corresponden con el comportamiento real del usuario.

Aún con todo, el propio Fencott reconoce que todavía hay aspectos de la metodología que requieren mayor investigación. Uno de ellos es la relación entre el grafo de la escena y los mapas de percepción. Otro es el significado del proceso de verificación y validación en el caso de los EV’s, y también su relación con lo que en la industria de los videojuegos se conoce por “beta-testing”.

### 3.6.2 CLEVR: Concurrent and LLevel by level development of VR systems

#### 3.6.2.1 Descripción

En [Seo, 2001] se da este nombre a una metodología estructurada para la especificación o diseño temprano de sistemas de Realidad Virtual, basada en la idea de desarrollo jerárquico e incremental así como en la simulación. Seo *et al.* ilustran la metodología CLEVR como una espiral en la que se identifican tres etapas:

1. Etapa de especificación: objetos, comportamiento burdo, arquitectura del sistema.
2. Rendimiento, descomposición de tareas, modelo de interacción, refinamiento de los modelos de forma, función y comportamiento.
3. Presencia, efectos especiales.

Según explican los autores, las primeras etapas se centran en aspectos usuales como la identificación de objetos y características, comportamiento burdo del sistema, modelado de las tareas del usuario, y la arquitectura general del software.

En una segunda etapa se abordan aspectos más ligados a la Realidad Virtual, en particular el rendimiento, conforme los objetos y los módulos software identificados en la primera etapa van siendo refinados. Pueden realizarse simulaciones o ejecuciones con el propósito de validar y revisar los comportamientos de los objetos importantes, predecir el rendimiento y tomar decisiones más precisas en cuanto a distribución de procesos.

Finalmente, una vez que ese aspecto que es el rendimiento ha sido tratado hasta cierto punto, en una tercera etapa de iteraciones se abordan cuestiones como los efectos especiales o mejorar la presencia.

De la explicación que dan los autores destaca la distinción que hacen entre forma, función y comportamiento, y es que una de las ideas en las que se basa CLEVR es en el principio de la ingeniería del software de la separación de vistas. La forma es la apariencia externa de un objeto virtual, sus atributos físicos y su estructura, mientras que las funciones son acciones primitivas que precisa el objeto virtual para llevar a cabo comportamientos de alto nivel.

Para modelar pues los objetos, los autores comienzan con la creación de diagramas de secuencia de mensajes (*Message Sequence Diagrams*, MSDs) con

los que, primero, especificar las interacciones entre esos objetos y, después, ayudar a completar el siguiente diagrama, el de clases. Entonces utilizan diagramas de flujo de datos (*Dataflow Diagrams*, DFDs) y lenguajes de script para representar las funciones de los objetos, diagramas de estados (*statecharts*) para el comportamiento de los objetos, y para la forma de estos introducen una representación orientada a objetos llamada VOS (*Visual Object Specification*) que incluye atributos, métodos para las funciones/comportamiento y una estructura similar a un grafo de escena.

### 3.6.2.2 Análisis

Esta metodología se basa en un trabajo previo descrito en [Kim, 1998], en donde ya se presentaba el proceso de especificación de la forma, función y comportamiento como un modelado concurrente e iterativo que permite al desarrollador moverse libremente entre esos tres espacios, y refinarlos de la manera apropiada. Incluso en esa referencia anterior también se describen los diagramas utilizados para modelar esos tres aspectos de los objetos virtuales, algunos de los cuales también fueron comentados en el apartado 2.7.7.

La contribución pues de Seo *et al.* sobre ese trabajo previo es la de orientar el proceso hacia una rápida creación de prototipos del sistema basada en un modelo en espiral y en la reutilización de componentes, apoyándose en una herramienta llamada CCTV (*Component Combination Tool for VR*). Esa herramienta se une además al conjunto P-VoT (Postech-Virtual reality system develOpment Tools) del laboratorio POSTECH de la Universidad de Ciencia y Tecnología de Pohang.

Con todo, resulta interesante observar el parecido entre la separación de forma, función y comportamiento que proponen Kim *et al.*, y el modelo estructura, comportamiento y función al que se recurre en [McIntosh, 2000], y ya comentado en el apartado anterior y también en el 2.7.7. Sobre todo cabe fijarse en las representaciones que se utilizan en cada caso para especificar esos conceptos.

Así, ambos coinciden en utilizar diagramas de estados para describir el comportamiento. McIntosh sugiere utilizar también diagramas de secuencia, diagramas que Kim *et al.* utilizan al comienzo del modelado para describir el comportamiento externo del mundo virtual.

McIntosh también propone utilizar diagramas de secuencia para describir la función, además de diagramas de casos de uso y de colaboración, aunque en su caso de estudio sólo utiliza estos dos últimos. En cambio, Kim *et al.* recurren a

los diagramas de flujo de datos y lenguajes como el CSL (*Computation Specification Language*) para la función.

Los autores tampoco coinciden en la representación de la forma, que Kim *et al.* describen usando una representación orientada a objetos a la que llaman VOS. McIntosh, por su parte, no habla de forma sino de estructura, para la que emplea diagramas Entidad-Relación.

Por último, Kim *et al.* crean diagramas de clases antes de detallar la función, comportamiento y forma de los objetos. Para McIntosh, el uso de diagramas de clases se justifica después de haber modelado el entorno virtual con diagramas UML de instancias de objetos, si en estos se observa la posibilidad de generalizar. No debe olvidarse, sin embargo, que el propósito de McIntosh no es el de construir un mundo virtual, sino el de analizar uno ya creado, por lo que sigue una estrategia de ingeniería inversa.

En conclusión, podría decirse que tanto Kim *et al.* y McIntosh recurren a modelos similares, si bien los primeros añaden representaciones con las que especificar al detalle –como los diagramas de flujo, el lenguaje CSL y la especificación VOS–, en tanto que McIntosh ofrece otras representaciones de más alto nivel –como los casos de uso o los diagramas de colaboración–. La Tabla 3.1 resume la comparación aquí hecha.

	[Kim, 1998]	[McIntosh, 2000]
Función	Diagramas de flujo de datos y lenguaje CSL	Diagramas de casos de uso, de secuencia y de colaboración
Comportamiento	Diagramas de estado y diagramas de secuencia	Diagramas de estado y diagramas de secuencia
Forma / Estructura	Visual Object Specification (VOS)	Diagramas Entidad-Relación
Diagramas de clases	Antes de modelar las vistas	Después del modelado SBF

**Tabla 3.1** Diagramas en la separación de vistas de Kim *et al.* frente al modelo SBF de McIntosh

### 3.6.3 SENDA: Metodología de desarrollo de mundos virtuales habitados

#### 3.6.3.1 Descripción

La metodología SENDA es el resultado del trabajo doctoral de M.I. Sánchez, y cuya descripción se desgrana en múltiples publicaciones. Así, en [Méndez, 2001] se describe el modelo de proceso, el diseño se describe en [Sánchez, 2001]

y también junto al análisis en [Sánchez, 2003], análisis que se describe más en detalle en [Sánchez, 2005a], y finalmente podemos encontrar una descripción completa de la metodología en [Sánchez, 2005b], así como en la propia tesis doctoral de M.I. Sánchez.

El objetivo final de los autores es definir una metodología especialmente diseñada para entornos virtuales pero que no suponga una ruptura con las experiencias previas de los diseñadores. Para ello, esta metodología trata de conjugar tres disciplinas: el corazón del proceso de desarrollo es dado por la ingeniería del software; las técnicas específicas para el diseño de la interfaz, por la interacción persona-ordenador; y la inteligencia artificial proporciona las técnicas para diseñar e implementar el conocimiento del sistema.

La metodología está fundamentada en un modelo de procesos agrupados en: procesos de gestión, procesos orientados al desarrollo del software y procesos integrales. Los propios autores reconocen no aportar nada nuevo a los procesos de gestión –estimación, planificación, seguimiento, control y calidad- ni a los integrales –gestión de la configuración, verificación y validación-. Sin embargo, es en los procesos orientados al desarrollo del software donde se encuentran sus principales contribuciones, redefiniendo los procesos de:

1. Análisis (A).
2. Diseño (3DD, AD, SD).
3. Implementación (SCI, CI).

Así, el análisis está compuesto por cinco actividades:

- A1. Pre-conceptualización
- A2. Definición de los requisitos específicos.
- A3. Conceptualización.
- A4. Modelado estático.
- A5. Modelado dinámico.

Para la extracción de los requisitos durante el proceso de análisis, los autores emplean casos de uso, según UML, e introducen lo que denominan conceptos de uso. Los primeros para los requisitos que implican interacción del usuario con el sistema, y los segundos para aquellas acciones automáticas que realizará el avatar por el usuario. Además, se emplean diagramas de clases para el modelado estático, y diagramas de transición para el dinámico.

Continuando con el proceso de diseño, en él se cuentan tres procesos:

- 3DD. Proceso de Diseño 3D.
- AD. Proceso de Diseño de las Acciones.

SD. Proceso de Diseño del Sistema.

En [Méndez, 2001] también se nombra un cuarto proceso, que es el de Diseño de Elementos Multimedia, que sin embargo no aparece en publicaciones posteriores. Además, el proceso AD o Diseño de las Acciones también puede leerse en [Sánchez, 2001] como CIA o Diseño de la Arquitectura Interna de los Componentes.

El proceso 3DD o Diseño 3D incluye el diseño gráfico de los escenarios, objetos decorativos, avatares, etc. Incluye dos tareas:

- 3DD1. Modelado del Entorno Virtual Habitado.
- 3DD2. Modelado de los Avatares.

Los autores entienden que el diseñador del sistema y el diseñador gráfico son dos roles diferentes, por lo que el resultado de esas tareas son los requisitos con los que explicar al diseñador gráfico cómo debe ser el entorno y sus elementos, empleando para ello mapas y un conjunto de formularios (del entorno, del avatar, de la jerarquía del avatar).

El proceso de Diseño de las Acciones, o AD, tiene por objeto definir las acciones que realizarán los avatares y el resto de elementos en el entorno virtual. Se descompone en las siguientes tareas, en las cuales se utilizan técnicas de inteligencia artificial:

- AD1. Modelado de la Percepción.
- AD2. Modelado de la Personalidad.
- AD3. Modelado de las Acciones Físicas.
- AD4. Modelado de las Reacciones.

El siguiente proceso es el de Diseño del Sistema, o SD, aunque no debe verse como el último de la secuencia, pues los tres procesos de diseño se llevan a cabo en paralelo e interrelacionándose, según describen los autores. En cualquier caso, este último proceso es similar al seguido en el diseño tradicional orientado a objetos, y cuenta con las siguientes tareas:

- SD1. Modelado Estático Expandido.
- SD2. Modelado Dinámico Expandido.
- SD3. Descripción Detallada de los Métodos.
- SD4. Diseño de la Arquitectura del Sistema.
- SD5. Diseño de la Persistencia de Datos.
- SD6. Diseño de la Interfaz.

Con respecto al Diseño de la Interfaz, los autores explican que una vez se tiene el diseño de los elementos relacionados con el aspecto, debe comprobarse que satisfacen al cliente, usando maquetas o videos y mostrando distintas alternativas de navegación por el EV. Los propios autores señalan la utilidad del diseño participativo o centrado en el usuario para esta tarea.

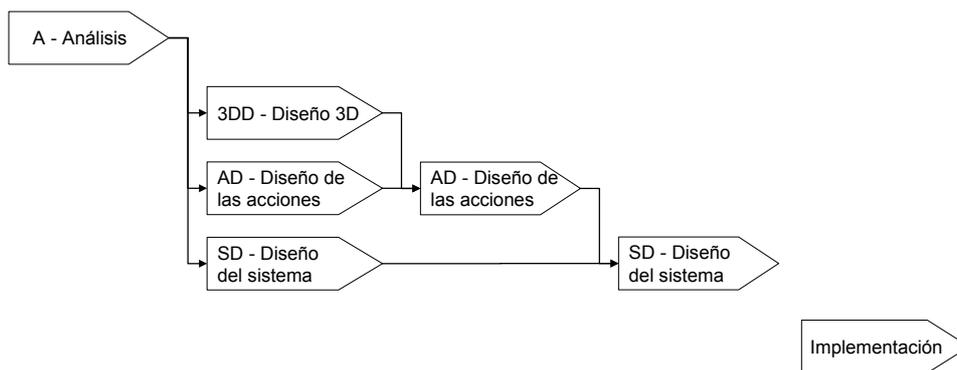
Por último, el proceso de implementación se divide en las siguientes tareas:

SCI. Implementación de los Componentes de Soporte.

CI. Implementación del Módulo Principal.

La propuesta de los autores para este proceso es que cada una de las muchas partes que componen el entorno virtual se implemente de forma separada (SCI) para luego integrarlas todas en el sistema (CI).

La siguiente figura ilustra, de forma gráfica y resumida, esta metodología:



**Figura 3.5** Desarrollo de mundos virtuales habitados según SENDA

### 3.6.3.2 Análisis

Sin duda, uno de los primeros aspectos que destacan de SENDA como marco metodológico es el gran número de procesos y actividades que enumera, sus relaciones y dependencias, sólo comparable a VEDS –tal y como se describe en [Eastgate, 2001]- y ya algo más lejos el tutorial de B. Hay –descrito en el apartado 3.3.4-. Más aún, por cada actividad los autores indican los productos de entrada y de salida, las técnicas y los participantes involucrados, lo que permite también comparar esta metodología con otra más comercial como es LUCID, que será analizada más adelante.

SENDA trata además de aprovechar la enorme experiencia ya existente en ingeniería del software, y ofrece una solución adaptada a raíz de experiencias previas en la aplicación de métodos para la creación de software convencional – particularmente la aplicación del método de Larman al desarrollo de entornos virtuales habitados, según se describe en [Méndez, 2001]-. En este sentido, SENDA es más que una simple adaptación de una metodología de desarrollo de interfaces de usuario 2D, como lo era la propuesta brasileña descrita y analizada en el apartado 3.5.2.

Precisamente, esas experiencias previas impulsaron a los autores a proponer nuevas herramientas para modelar el sistema y comunicarse entre los distintos miembros del equipo de desarrollo. Así, los autores introducen los conceptos de uso, una breve descripción de la funcionalidad del avatar que incluye: propósito, modo de operación y frecuencia. Igualmente, los autores describen formularios que sirven para que el diseñador del sistema pueda informar al diseñador gráfico cómo debe ser el entorno y los elementos que lo componen. Estas contribuciones de los autores también están relacionadas con otra característica de SENDA que la diferencia de todos los métodos anteriores, y es el desarrollo de mundos virtuales como espacios habitados por avatares.

El conjunto de herramientas propuestas en SENDA se completa con otras ya vistas en anteriores metodologías de este grupo. Así, los conceptos de uso se emplean junto a los diagramas de caso de uso para la extracción de requisitos, diagramas que Fencott propone para identificar las relaciones entre el usuario y el EV. Para el modelado estático, en SENDA se utilizan diagramas de clases, herramienta que Fencott sugiere para los componentes programados, y a la que también se recurre en CLEVR. Por último, para el modelado dinámico, en SENDA se opta por los diagramas de transición de estados, igualmente empleados en CLEVR para especificar el comportamiento de los objetos.

Dejando a un lado las herramientas y centrando la discusión en el propio modelo de procesos, resulta ineludible destacar la inclusión del diseño 3D (3DD) como un proceso más en la metodología, sobre todo frente a propuestas como la ya citada brasileña, en la que el diseño del contenido 3D parecía quedar escondido bajo la actividad de disposición de los contenidos. Más aún, de aquella propuesta también nos quedaba la duda de si los caminos de los que se hablaba se referían o no a caminos en el entorno 3D, mientras que SENDA no deja duda al respecto. En SENDA, y más concretamente en el diseño de la interfaz (SD6), se indica la necesidad de mostrar distintas alternativas de navegación por el EV. Y es que, como los propios autores brasileños reconocían, un entorno virtual es un entorno por el que el usuario navega para buscar él mismo la información, ejecutar las actividades y desarrollar las tareas.

Aún con todo, es la opinión del que escribe que esta metodología podría beneficiarse aún más de las técnicas de diseño de la interfaz de usuario, como el diseño participativo y centrado en el usuario que apuntan al respecto de esa tarea SD6, teniendo aún más en cuenta las preferencias del usuario y prestando atención a los aspectos de usabilidad como, en cambio, sí hacen otros enfoques anteriormente analizados.

### 3.6.4 Análisis del grupo

Las metodologías descritas y analizadas en este apartado se caracterizan por confiar en la ingeniería del software bien para dar estructura al proceso de creación de un entorno virtual, bien para proporcionar herramientas con las que llevar a cabo ese proceso, o ambas cosas.

En cuanto a la estructura, los autores de la metodología CLEVR apuestan por un ciclo de vida en espiral, en el que el entorno virtual es desarrollado de forma iterativa e incremental. En cambio, la metodología SENDA parece ajustarse más a un clásico ciclo de vida en cascada en el que se suceden tres grandes procesos –análisis, diseño y implementación–, si bien en cada uno de ellos se cuentan variadas actividades que se desarrollan en paralelo, como por ejemplo el diseño del sistema (SD) y el diseño 3D (3DD). El caso de la propuesta de Fencott se asemeja al anterior, con el modelado conceptual y el de la percepción realizándose en paralelo al modelado de la estructura.

En lo que respecta a las herramientas, todas estas propuestas apuestan por el uso de diagramas en alguna de las formas que se recogen en la especificación UML, aunque hay autores que no olvidan otras herramientas menos formales pero muy extendidas, como es el caso de Fencott y los storyboards en el modelado conceptual de su propuesta, y hay otros que encuentran insuficientes los diagramas UML y optan por proponer nuevas herramientas, como es el caso de los conceptos de uso en la metodología SENDA.

Así, uno de esos diagramas UML utilizados es el de casos de uso, propuesto por Fencott en su modelado estructural para identificar las relaciones entre el usuario y el EV, diagrama también utilizado en SENDA para la extracción de requisitos en el proceso de análisis, junto con los conceptos de uso. Al elaborar su propuesta, Fencott se basa en el trabajo de McIntosh, quien propone el uso de los diagramas de caso de uso para describir la función, según el modelo SBF (estructura, comportamiento y función). En cambio, en CLEVR se recurre a diagramas de flujo y lenguajes de script para describir esa función, siguiendo en este caso otro modelo basado en la distinción entre forma, función y comportamiento.

En lugar de los casos de uso, los primeros diagramas que se utilizan en CLEVR son los diagramas de secuencia y diagramas de clases. Estos últimos son utilizados en SENDA para el modelado estático, en tanto que Fencott los propone para los componentes programados.

Otro diagrama utilizado en SENDA es el de transición de estados, esta vez para el modelado dinámico. Ese diagrama también es utilizado en CLEVR para especificar el comportamiento. Aunque Fencott no comenta nada acerca de ese diagrama, sí es propuesto en el trabajo de McIntosh, también para describir el comportamiento de los objetos en el entorno.

El último diagrama UML que propone Fencott es el de objetos, con el fin de mostrar la estructura de nodos en el grafo de la escena. En el caso de CLEVR, esa estructura es parte de la forma, para lo que se utiliza no un diagrama sino una especificación que sus autores denominan VOS, y que incluye, precisamente, una estructura similar a la de un grafo de escena. Tampoco SENDA utiliza un diagrama en este caso, sino un conjunto de formularios.

La Tabla 3.2 resume el uso que se da a los diferentes diagramas en cada una de las tres propuestas incluidas en este grupo.

	Fencott	CLEVR	SENDA
Diagrama de casos de uso	Modelado de la estructura, relaciones usuario-EV		Análisis, extracción de requisitos junto con los conceptos de uso
Diagrama de secuencia		Escenarios, especifican interacciones entre objetos	
Diagrama de objetos	Modelado de la estructura, nodos en el grafo de la escena		
Diagrama de clases	Modelado de la estructura, componentes programados	Tras el de secuencia, y previo al modelado de las vistas	Modelado estático
Diagrama de flujo de datos		Función, junto con el lenguaje CSL	
Diagrama de estados		Comportamiento	Modelado dinámico

**Tabla 3.2** Uso de diagramas en las propuestas de desarrollo basadas en ingeniería del software

Con todo, se reconoce que la ingeniería del software, si bien es de gran ayuda en los aspectos más técnicos, no lo es tanto para abordar la estética del entorno virtual, para lo que se proponen otras herramientas, como por ejemplo los Mapas de Percepción que introducen Fencott e Isdale.

Aún así, las metodologías incluidas en este grupo resultan también insuficientes para abordar todos los procesos que se incluyen en el desarrollo de un entorno virtual y, por extensión, a cualquier interfaz de usuario 3D en general. Si bien destacan por una aproximación al diseño más ingenieril, con las técnicas y herramientas del ingeniero del software y del programador, no abordan los demás procesos con la misma profundidad con la que lo hacen otros autores, como por ejemplo D'Cruz *et al.* al tratar el análisis de la tareas –véase apartado 3.5.3-, B. Hay al describir el proceso de construcción del mundo virtual –véase apartado 3.3.4-, o Gabbard *et al.* y la evaluación –véase apartado 3.5.1-. Además, aún considerando sólo el diseño, este sigue orientado hacia el PC, dejando a un lado el problema de diseñar una interfaz cuando los dispositivos no son los propios del ordenador personal.

### **3.7 Creación de interfaces de usuario más allá del PC**

Hasta el momento, los métodos vistos se han centrado en aspectos como las tareas del usuario, la apariencia del espacio 3D, el rendimiento de la aplicación o la usabilidad de la interfaz. En cambio, rara vez se ha hecho referencia a los dispositivos que empleará el usuario para interactuar con la máquina, pues en la mayoría esa máquina –bien porque esa es la intención de los autores o bien porque sus casos de estudio así nos lo hacen ver- es un PC. O, mejor dicho, un POCS, utilizando el término que, como se recordará del apartado 2.7.10, fue introducido en [Barrilleaux, 2001] para describir al tan extendido PC con su pantalla, teclado y ratón.

Aunque el desarrollo de interfaces de usuario 3D para POCS no es sencillo – como el propio Barrilleaux demuestra-, este desarrollo se apoya en las técnicas de interacción ya conocidas en estos ordenadores, como apuntar o arrastrar. Sin embargo, la introducción de otros dispositivos menos convencionales supone un reto al que el desarrollador debe enfrentarse, como ya se comentó en el apartado 2.7.1. En este apartado se describen entonces métodos propuestos para el desarrollo de interfaces de usuario 3D que se alejan del PC convencional.

### 3.7.1 El proyecto INQUISITIVE

#### 3.7.1.1 Descripción

El proyecto [INQUISITIVE, URL] tenía como propósito el estudio de métodos y principios para el diseño y evaluación de interfaces de EV's. Aunque acerca de este proyecto pueden encontrarse muchas publicaciones, cabe destacar aquí [Smith, 2001], en donde se expone el proceso de diseño en base a los métodos propuestos en el proyecto, y cuyo flujo, partiendo de unos objetivos dados para la aplicación, es el siguiente:

1. Definición de los requisitos.
  - a. Cuestiones relativas a los objetos.
  - b. Requisitos de interacción del usuario.
2. Refinamiento del diseño.
  - a. Concordar los requisitos con las capacidades del toolkit.
  - b. Realizar la correspondencia entre el diseño y el toolkit.
3. Evaluación.

Según los autores, el modelado de los objetos precisa conocer su apariencia, geometría y comportamiento. Para extraer los requisitos de los objetos, Smith y Duke apuntan dos técnicas: representar las tareas mediante storyboard para identificar los objetos y sus componentes, comportamientos en el entorno y los principales objetos interactivos; estructurar el entorno mediante un grafo de escena anotado (*annotated scene graph*). En este proceso los autores no olvidan el compromiso entre realismo y coste computacional, pero también entre apariencia y comportamiento, subrayando que la desproporción entre estos últimos puede ser la causa de problemas de usabilidad.

En cuanto a los requisitos de interacción del usuario, Smith y Duke comentan que son satisfechos por medio de técnicas de interacción, para cuya especificación los autores apuntan dos técnicas: descripciones informales en texto o vídeo; redes de flujo (*flownets*), una extensión de las redes de Petri para modelar tanto los procesos discretos como continuos de la interacción en el EV, y ya comentada en el apartado 2.7.7.

A partir del objetivo de la aplicación y de los requisitos del diseño –realismo, objetos, tareas, interacción–, se evalúa si las características del toolkit dado permiten cumplirlos. Si no es así, se cambia de toolkit si es posible. Con el toolkit elegido, se procede a un prototipado rápido. Si no se tiene éxito, de nuevo se cambia de toolkit si es posible. Cuando no es posible cambiar de toolkit

entonces es necesario revisar los requisitos del diseño. El diseño es refinado teniendo en cuenta los compromisos antes citados.

En cuanto a la evaluación, esta se centra en la usabilidad y en la experiencia que proporciona el EV. Se trata de una evaluación empírica que se apoya en guías propuestas para el diseño y evaluación de espacios virtuales 3D, así como en la teoría de la actividad para identificar interrupciones en la ilusión de la Realidad Virtual y problemas de usabilidad.

### 3.7.1.2 Análisis

El proceso de diseño descrito por Smith y Duke pone especial énfasis en dos aspectos. El primero de ellos es lograr un adecuado equilibrio entre apariencia y comportamiento, pues los objetos que tienen una apariencia compleja pero cuyo comportamiento asociado es limitado pueden engañar al usuario, lo que enlaza con problemas de usabilidad. Como se recordará del apartado 3.3.1, esta era la crítica de estos autores a los métodos más ingenuos. La propuesta de los autores es la de identificar las necesidades de interacción mediante storyboard. También usan –como novedad– un grafo de escena anotado, aunque cabe preguntarse si una representación tan ligada a la implementación es apropiada en la etapa de extracción de requisitos.

El segundo aspecto que los autores enfatizan es la interacción, y más concretamente la especificación de técnicas de interacción, una tarea apenas comentada en este capítulo hasta este apartado. En este sentido, una de las contribuciones del proyecto INQUISITIVE son las redes de flujo [Smith, 1999], basadas en redes de Petri para representar los procesos discretos de la interacción, pero añadiendo nuevos elementos a esa notación para representar los flujos continuos de datos. O, en otras palabras, representan sistemas continuos cuyo control se especifica mediante redes de Petri.

En [Willans, 2000] se hace uso también de las redes de flujo para especificar el comportamiento de los objetos. Parece que, a la hora de especificar interacción y comportamiento, diversos autores coinciden en distinguir también entre flujo y control, aunque difieren en las notaciones utilizadas. Así, en [Kim, 1998] –véanse los apartados 2.7.7 y 3.6.2– también se utiliza diagramas de flujo –aunque para representar funciones, y con una notación diferente–, y se complementan con diagramas de estados –para representar el comportamiento que controla las funciones–. Por su parte, [McIntosh, 2000] –véanse apartados 2.7.7 y 3.6.1– propone los diagramas de secuencia y de estados.

El proceso de diseño se basa además en la elección del toolkit más apropiado y en un prototipado rápido a modo de análisis previo a la implementación, con el fin de asegurar que es posible cumplir los requisitos. Sin embargo, los autores pueden beneficiarse de este enfoque pues cuentan con diferentes plataformas – VRML, Division, DIVE, MAVERIK- y herramientas como MARIGOLD –véase apartado 2.7.8 y [Willans, 2000]- o el toolkit creado en el marco del proyecto – véase apartado 2.7.3 y [Sastry, 2000]-. Sin embargo, este no es siempre el caso, aunque es deseable.

Finalmente, sobre el proceso de evaluación los autores no entran en gran detalle, y aunque no es muy diferente al de otras propuestas, sería interesante conocer cuáles son esas guías de diseño a las que hace referencia o cómo hacen uso de la teoría de actividad. Con todo, el proceso esbozado por Smith y Duke deja otras lagunas más importantes para el desarrollo de un EV, como pueda ser la participación del usuario o la especificación de otros componentes software del sistema.

### **3.7.2 Estrategias dirigidas por el contenido o por la interacción**

#### **3.7.2.1 Descripción**

En [Parés, 2001] se describen dos estrategias de desarrollo. En primer lugar, se explica la estrategia dirigida por el contenido, en donde el propio contenido define un contexto que determina además la metáfora a utilizar. En contraste con la primera, los autores introducen una segunda estrategia que se centra en el diseño de la interacción del usuario, independientemente del contenido específico de la aplicación, lo que según los autores es útil en desarrollos más artísticos o creativos, o cuando se quiere evaluar ciertas teorías de interacción de forma experimental.

La Tabla 3.3 recoge las dos secuencias de etapas correspondientes a las dos estrategias descritas por los autores. Aunque los propios autores indican que algunas de las etapas pueden no ser necesariamente secuenciales sino analizadas en paralelo a otras próximas en la lista, existe un notable cambio en el orden de las etapas entre las dos estrategias. Para resaltar esos cambios, a las etapas listadas se les ha asociado el mismo número en ambas columnas de la tabla.

Según los autores, Parés y Parés, independientemente de la estrategia seguida, las etapas de desarrollo de una aplicación de Realidad Virtual están relacionadas con: el bucle de simulación, las interfaces, el modelado de los objetos, el diseño del comportamiento de los objetos y el diseño del estímulo.

Estrategia dirigida por el contenido	Estrategia dirigida por la interacción
1. Definir el tema de la aplicación	7. Identificar interfaces de entrada
2. Definir el tipo de aplicación	8. Identificar interfaces de salida
3. Definir el tipo de usuario	3. Definir el tipo de usuario
4. Identificar los objetos virtuales necesarios	2. Definir el tipo de aplicación
5. Identificar los datos implicados	1. Definir el tema de la aplicación
6. Identificar procesos	6. Identificar procesos
7. Identificar interfaces de entrada	4. Identificar los objetos virtuales necesarios
8. Identificar interfaces de salida	5. Identificar los datos implicados
9. Identificar las herramientas de modelado de objeto necesarias	9. Identificar las herramientas de modelado de objeto necesarias
10. Identificar las herramientas de desarrollador de aplicaciones	10. Identificar las herramientas de desarrollador de aplicaciones

**Tabla 3.3** Secuencia de etapas según el tipo de estrategia

Empezando por las interfaces, los autores indican que las etapas del diseño de las interfaces se dividen en tres partes esenciales: decidir qué canales externos se comunicarán con qué canales internos (*mapping*); qué elementos harán de enlaces externos de la aplicación (interfaces físicos); y qué elementos harán de enlaces internos (interfaces software o lógicos).

Siguiendo con los objetos, su modelado debe ser cuidado para ajustarse al poder de computación, los requisitos visuales y la interacción definida. Según Parés y Parés, el diseño del comportamiento de los objetos es una tarea compleja debida a sus dificultades inherentes (inteligencia, reacciones, acciones espontáneas) y a la falta de herramientas generales y homogéneas para su desarrollo, aunque indican que los algoritmos pueden implementarse como autómatas finitos.

En cuanto al diseño del estímulo, los autores la describen como una etapa especialmente delicada, pues tanto los estímulos generados como los capturados deben alcanzar unas condiciones acordes con el ser humano, ya que de no ser así pueden causar malestar en el usuario.

Finalmente, el bucle de simulación será el que gestione la evolución del sistema repitiendo una serie de acciones en cada ciclo, que incluyen: gestión de los sensores, actualización del estado del usuario y de los objetos, determinar las interacciones y gestionar las salidas. Entre las etapas relacionadas con este bucle los autores subrayan la identificación de los procesos y de datos implicados (etapas 5 y 6 en la Tabla 3.3).

### 3.7.2.2 Análisis

El proceso de desarrollo descrito por Parés y Parés en sus dos estrategias enumera un conjunto de actividades que sirven para definir e identificar diferentes aspectos de los elementos que integran una aplicación de Realidad Virtual: el usuario, las interfaces, los objetos y el bucle de simulación. Entre las contribuciones de los autores cabe destacar la clasificación que proponen de tipos de aplicación –de exploración, de manipulación o de interacción contributiva- y la clasificación entre tipos de usuario –experto, público en general, etc.-, distinciones apenas tenidas en cuenta en los métodos analizados hasta aquí –p. ej. sólo la metodología descrita en el apartado 3.5.2 llega a distinguir entre usuarios novatos y expertos-.

También cabe destacar que el diseño de las interfaces se basa en unos enlaces externos –interfaces físicas-, unos enlaces internos –interfaces lógicas- y la comunicación entre canales externos e internos, en lugar de abordar el tema con el concepto de técnica de interacción que, como se vio en el capítulo anterior, puede resultar más impreciso. Igualmente, en la propuesta de estos autores destaca el diseño del estímulo, lo que podría relacionarla con propuestas como la de Fencott –véase apartado 3.6.1- y alguna más, pero también destaca el bucle de simulación, un elemento fundamental que sorprendentemente no se ha tenido en cuenta hasta aquí, salvo tal vez en SENDA –véase apartado 3.6.3-.

Sin embargo, una de las limitaciones de esta propuesta es que enumera un conjunto pasos de definición e identificación, pero en la mayoría de ellos no se entra en mayor detalle, y deja la duda de cómo seguir el desarrollo una vez definido e identificado lo que apunta cada paso. Así, aunque los autores hablan de etapas del desarrollo de una aplicación de Realidad Virtual al presentar los elementos con los que suelen relacionarse esas etapas, tampoco se da nombre ni orden a las mismas. Y, si bien se habla del diseño de esos elementos, también se echan en falta herramientas para este fin. Por ejemplo, para la implementación del comportamiento de los objetos se apunta el uso de autómatas finitos, pero no se sugiere ninguna herramienta para plasmar su diseño, y ya se ha visto a lo largo de este capítulo y el anterior que no hay una única opción.

Así, todo y que los autores presentan nada menos que dos estrategias en una misma propuesta, ninguna de las dos representa un modelo completo de procesos y herramientas.

### 3.7.3 VRID: Virtual Reality Interface Design

#### 3.7.3.1 Descripción

En [Tanriverdi, 2001] se describe la metodología VRID para el diseño de una interfaz de Realidad Virtual como un proceso iterativo en el que los requisitos de la interfaz son transformados en una especificación de diseño que pueda ser implementada por los desarrolladores de software.

A partir de una descripción en lenguaje natural del sistema de Realidad Virtual, la metodología VRID arranca dividiendo el proceso de diseño en fases de alto nivel (*High-Level*, HL) y de bajo nivel (*Low-Level*, LL). Según los autores, los diseñadores deberían iterar entre pasos de diseño de alto y bajo nivel para refinar la especificación hasta que estén convencidos de que el diseño tiene sentido y que las especificaciones son implementables. La Tabla 3.4 recoge estas dos fases de diseño.

Fase de diseño de alto nivel (HL)	Fase de diseño de bajo nivel (LL)
HL1. Identificar datos	
HL2. Identificar objetos	
HL3. Modelar los objetos	
HL3.1. Gráficos	LL1. Gráficos
HL3.2. Comportamientos	LL2. Comportamientos
HL3.3. Interacciones	LL3. Interacciones
HL3.4. Comunicaciones internas	LL4. Comunicaciones internas
HL3.5. Comunicaciones externas	LL5. Comunicaciones externas

**Tabla 3.4** Las dos fases de diseño de VRID

Así, en el paso HL1 se identifican los flujos de datos entrantes a la interfaz de Realidad Virtual. En el paso HL2, se identifican los objetos que tienen papeles bien definidos en la interfaz, para lo cual los autores sugieren el uso de técnicas de análisis y diseño orientado a objetos. También apuntan que los objetos físicos no requieren ser modelados a menos que exhiban comportamientos mágicos.

El paso HL3 comprende el modelado de los objetos según el modelo de software VRID, que diferencia entre gráficos, comportamientos, interacciones y comunicaciones –internas si son entre cualquiera de estos componentes, externas en otro caso-, todo ello recogido en los pasos HL3.1 al HL3.5.

El modelado de los objetos en el paso anterior es a alto nivel, los detalles se describen en los pasos LL1 a LL2, que repiten los cinco pasos de modelado pero a bajo nivel. El componente de gráficos es tratado en VRID como una caja negra, su especificación se realiza de modo que el diseñador gráfico pueda comprender la apariencia y animaciones de los objetos. Para el comportamiento, los autores proponen el uso de PMIW, en origen una herramienta para especificar interacciones en base a diagramas de flujo de datos –para representar la parte continua- y diagramas de transición de estados –para representar la parte discreta-. En cuanto a la interacción, los autores consideran que lo apropiado es usar PMIW, aunque dejan libertad al diseñador para escoger otro lenguaje de diseño que crea más apropiado. Finalmente, en el apartado de comunicación se definen los mecanismos más apropiados de atención (*scheduling*) y de paso de mensajes.

### 3.7.3.2 Análisis

A la hora de analizar esta propuesta, debe tenerse en cuenta que el objetivo de los autores es el diseño de la interfaz, entendida como la composición de los objetos de la interfaz y el flujo de datos que llega a la misma como entrada de los usuarios u otras fuentes externas, dejando a un lado el control del diálogo y la propia aplicación de Realidad Virtual.

Desde este punto de vista, una de la diferencias de VRID frente a otra propuestas es que distingue entre objetos físicos y virtuales, y entre comportamientos físicos y mágicos. Esta distinción supone un primer paso desde la Realidad Virtual hacia la Realidad Aumentada, según el continuo descrito en el apartado 2.3.8.

En cuanto al proceso de diseño en sí, si lo comparamos con las estrategias de Parés y Parés vistas en el punto anterior, coinciden al incluir pasos en los que se identifican los datos (paso HL1) y los objetos (paso HL2). Además, el modelado de gráficos, comportamientos e interacciones que se incluye en la propuesta de Tanriverdi y Jacob también puede corresponderse con el diseño de los objetos, sus comportamientos y las interfaces que abordan Parés y Parés en su artículo. Sin embargo, estos últimos no explican cómo especificar, en particular, los comportamientos y las interfaces, lo que Tanriverdi y Jacob sí hacen al proponer el uso de PMIW.

Precisamente, sobre PMIW ya se habló en el apartado 2.7.8 como herramienta de apoyo para el lenguaje visual introducido en [Jacob, 1996] y que combina diagramas de flujo y diagramas de transición de estado para describir las relaciones continuas que componen los diálogos non-WIMP y las operaciones discretas que actúan sobre esas relaciones –véase el apartado 2.7.7-.

Al igual que se vio con las redes de flujo al hablar del proceso de diseño en el proyecto INQUISITIVE –apartado 3.7.1-, el lenguaje de Jacob fue inicialmente concebido para la especificación de la interacción, pero ahora se propone también para especificar los comportamientos de los objetos. Y no es su único parecido. La concepción de la interfaz de Jacob como una parte continua y otra discreta es similar al modelo híbrido descrito en [Smith, 1999]. Sin embargo, la representación es diferente, en parte porque la notación de los flujos de datos continuos no es exactamente igual, pero sobre todo porque las redes de flujo integran las operaciones discretas en el mismo diagrama usando la notación de las redes de Petri, mientras que Jacob utiliza aparte diagramas de transición de estados.

Y del mismo modo que en el apartado 3.7.1 se comparó las redes de flujo con los trabajos de [Kim, 1998] y [McIntosh, 2000], cabe aquí hacerlo también. Así, Kim *et al.* ya propusieron antes que Tanriverdi y Jacob el uso de diagramas de flujo y de transición de estados para especificar el comportamiento y función de los objetos, aunque PMIW –además de ser utilizado también para describir la interacción- hace explícita la relación entre ambas representaciones a través de un nuevo elemento –el enlace y su condición- en la notación de flujo de datos. Y con respecto a McIntosh, también coinciden con esta autora, aunque sólo en el uso de los diagramas de transición de estados.

Las coincidencias con los anteriores autores no se limitan sólo a los diagramas, pues también se encuentran grandes similitudes en los modelos que siguen. Así, Kim *et al.* siguen el principio de separación entre forma, función y comportamiento. McIntosh, por su parte, se basa en el modelo SBF (estructura, comportamiento y función). Y la propuesta que aquí se analiza distingue entre gráficos, comportamientos, interacciones y comunicaciones. La Tabla 3.5 recoge estos tres modelos. Observándolos, llama la atención que en uno se hable de forma, en el otro de estructura y en este último de gráficos, para referirse los tres a la imagen visual de los objetos.

Kim <i>et al.</i> , 1998	McIntosh, 2000	Tanriverdi <i>et al.</i> , 2001
<ul style="list-style-type: none"> <li>• Forma</li> <li>• Función</li> <li>• Comportamiento</li> </ul>	<ul style="list-style-type: none"> <li>• Estructura</li> <li>• Comportamiento</li> <li>• Función</li> </ul>	<ul style="list-style-type: none"> <li>• Gráficos</li> <li>• Comportamientos</li> <li>• Interacciones</li> <li>• Comunicaciones</li> </ul>

**Tabla 3.5** Similitudes entre los modelos que proponen Tanriverdi *et al.* y otros autores

Justamente, y en relación a los gráficos, cabe destacar el tratamiento que se hace en VRID de los mismos, como una caja negra desde la perspectiva del diseñador

de la interfaz, recordando en este sentido a SENDA al diferenciar entre la labor del diseñador del sistema y el trabajo del diseñador gráfico.

Con todo, aunque esta propuesta se limita al diseño de la interfaz, deja a un lado cuestiones que sí han preocupado a otros autores, como el equilibrio entre rendimiento y usabilidad o los problemas de usabilidad de la interfaz.

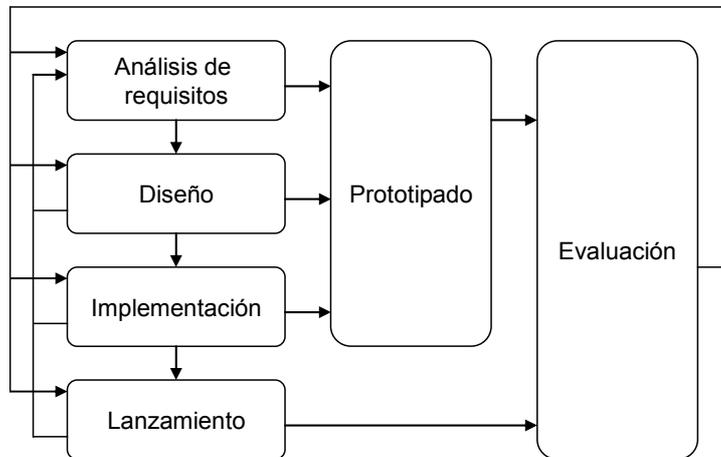
### **3.7.4 MPIUA: Modelo de Proceso de la Ingeniería de la Usabilidad y de la Accesibilidad**

#### **3.7.4.1 Descripción**

En [Granollers, 2002] se describe este modelo de proceso para el diseño de sistemas interactivos que a su vez está basado en el "DUTCH Model". Se trata de un modelo de proceso centrado en el usuario, que implica activamente a los mismos desde las etapas iniciales del diseño. Seis son las etapas en las que se divide este modelo de proceso:

1. Análisis de requisitos
2. Diseño
3. Implementación
4. Prototipado
5. Evaluación
6. Lanzamiento

No debe entenderse el proceso como una secuencia estricta de esas etapas, de hecho el modelo anima a realizar una evaluación continuada con los usuarios incluso desde el diseño de los primeros prototipos simulando partes del sistema. La Figura 3.6 ilustra gráficamente el proceso, y en ella se pueden observar las múltiples realimentaciones que se contemplan entre unas y otras etapas.



**Figura 3.6** Diseño de sistemas interactivos según MPIUA

En la referencia dada, Granollers *et al.* nos describen su experiencia al aplicar este modelo a la concepción y diseño de un sistema de Realidad Aumentada -en concreto, una visita "aumentada" a los restos de la fortaleza "Els Vilars"-, y es el proceso que se analizará aquí.

Así, los autores comienzan con el análisis de los requisitos, realizando estudios etnográficos del yacimiento, estudios de los diferentes perfiles de los futuros usuarios (modelos mentales), de las actuales visitas guiadas, y del urbanismo y vida social en aquella antigua fortaleza. Además, los autores también analizan el nuevo paradigma, la Realidad Aumentada, imaginando cómo ayudará al visitante del yacimiento, y preguntándose las diferencias con respecto al paradigma de sobremesa.

En el diseño, Granollers *et al.* abordan las siguientes etapas:

- 2.1. Análisis de tareas
- 2.2. Modelo arquitectónico.
- 2.3. Modelo conceptual.

La primera de esas etapas es definida como el estudio de las tareas que deben realizarse en términos de acciones y/o procesos cognitivos, entendiendo que una tarea es la actividad necesaria para conseguir un objetivo y que un objetivo es el resultado o logro que el usuario quiere alcanzar dentro de una aplicación. El método utilizado por los autores es el análisis de tareas jerárquico (*Hierarchical Task Analysis*, HTA).

Para la segunda de esas etapas, el modelo arquitectónico –correspondiente al apartado “User Virtual Machine (UVM), funcionalidad” del DUTCH Model-, los autores han optado por definir las principales zonas de interés y hacerles corresponder una serie de puntos –lógicos- en el yacimiento. Cada punto cuenta entonces con una zona de influencia asociada, y también con una serie de puntos de observación (P.O.), que son orientaciones representativas o vistas concretas del punto de interés.

El modelo conceptual se divide en modelo de diálogo y modelo de diálogo aumentado. El primero –correspondiente al apartado “UVM, diálogo” del DUTCH Model-, representa la estructura del diálogo, la comunicación con el usuario, y el método que aconsejan los autores es el diagrama de transición de estados (*State Transition Notation*, STN). El segundo modelo –correspondiente al apartado “UVM, representación” del DUTCH Model- es introducido por los autores para tener en cuenta la localización del usuario en el diálogo, proponiendo el uso de unos “apuntadores” que conectan los P.O. –del paradigma de la realidad aumentada- con su modelo de diálogo –del paradigma de sobremesa- correspondiente.

Por último, y dejando ya el diseño, los autores también dan cuenta de los prototipos y evaluaciones realizadas. Como se ha dicho, según este modelo de proceso no es necesario esperar a concluir el diseño para producir los primeros prototipos y realizar ya evaluaciones con los usuarios. Así, en cuanto a los prototipos, se indican una tableta (de madera), una reconstrucción 3D de la fortaleza y dos vídeos, y en cuanto a las evaluaciones, cuatro del tipo “focus group”.

### 3.7.4.2 Análisis

El modelo de proceso propuesto por Granollers *et al.* bien podría incluirse en el grupo de metodologías que se caracterizan por su diseño participativo, dado su enfoque centrado en el usuario. Igualmente, también podría haber sido incluido en el grupo de metodologías basadas en el análisis de tareas, dado el uso que hacen del mismo en el diseño; de hecho, sus definiciones de tarea y de objetivo son las utilizadas en el apartado 5.3 al definir el nuevo modelo de la interacción.

Sin embargo, la verdadera contribución de los autores se encuentra en el modelo arquitectónico y en el modelo de diálogo aumentado, al incluir la localización del usuario como un nuevo aspecto que no se tenía en cuenta en el paradigma de sobremesa pero que ahora sí es necesario al abordar el nuevo paradigma de Realidad Aumentada. Ello les lleva a introducir también el concepto de apuntadores para relacionar lo que los autores llaman puntos de observación

(P.O.) con los diálogos tradicionales asociados a cada uno de esos puntos. Aunque tanto la posición de los puntos de interés como los puntos de observación se localizan en el plano bidimensional de la fortaleza, no deja de ser interesante como aproximación al modelado del mundo real tridimensional y su aumento mediante interfaces digitales.

Con todo, sorprende que se aconseje el simple uso de diagramas de transición de estados para el diálogo, cuando otras propuestas anteriores los complementan con diagramas de flujo o de secuencia. La sorpresa no es tanta pues el empleo de los diagramas de transición de estados es una de las notaciones que suelen utilizarse para interfaces convencionales, como se vio en el apartado 2.7.7, pero en el caso de otras aplicaciones de Realidad Aumentada –como la aplicación de cirugía que se describe en [Tanriverdi, 2001] y que ilustra la metodología VRID vista antes- puede no ser lo más apropiado.

### **3.7.5 Diseño de entornos virtuales según Sutcliffe**

#### **3.7.5.1 Descripción**

En [Sutcliffe, 2003] se describe un proceso de diseño de aplicaciones de Realidad Virtual que tiene sus orígenes en el trabajo doctoral de Kaur –véase [Kaur, 1998] y los apartados 3.3.2 y 3.3.3-. El proceso de diseño que detalla Sutcliffe está compuesto por cinco etapas:

1. Análisis de requisitos
2. Análisis de tareas y de dominio
3. Diseño del EV
4. Prototipado del EV
5. Evaluación de la usabilidad

Se trata de un enfoque centrado en el usuario, pues involucra al mismo en el análisis de requisitos y la evaluación. Precisamente, el proceso comienza con ese análisis de requisitos, así como el enfoque del sistema. Para esta especificación inicial, el autor propone tres aproximaciones diferentes: métodos convencionales como los modelos de tareas, modelado de casos de uso, y escenarios.

En el primer caso, Sutcliffe nombra dos métodos: análisis jerárquico de tareas (*Hierarchical Task Analysis*, HTA) y estructuras tarea-conocimiento (*Task Knowledge Structure*, TKS). Estos métodos llevan a cabo una descomposición top-down de las actividades del usuario como objetivos y, según el autor, permiten la especificación del comportamiento de los agentes en el EV, pero no

expresan interacción. En cambio, los casos de uso del lenguaje UML permiten representar de forma explícita esa interacción como paso de eventos entre agentes y objetos, ofreciendo ese enfoque inicial del sistema, y a partir de ellos puede procederse a analizar las clases de objetos y agentes y describir la actividad en mayor detalle con diagramas de secuencia de actividades, siguiendo la notación UML. Pero los casos de uso no representan las tareas que no son interactivas, por lo que Sutcliffe recomienda una combinación de análisis de tareas orientado a objetivos y casos de usos. De hecho, el autor explica que los casos de uso comparten esa orientación a objetivos de las tareas, y cada caso de uso debe lograr un objetivo particular, describir una tarea o servicio de un agente interactivo contribuyendo así a la especificación de su comportamiento.

El tercer método que apunta Sutcliffe para el análisis de requisitos son los escenarios, simples descripciones narrativas que o bien ilustran la forma en la que el usuario interactuaría con el sistema o bien un problema a ser resuelto en el diseño. Según el autor, estos escenarios pueden ser transformados en casos de uso y en especificaciones como diagramas de secuencia de actividades, o ser usados para motivar el desarrollo de primeros prototipos.

Siguiendo con el proceso de diseño, el análisis de requisitos se acompaña con el análisis de las tareas y del dominio. En esta segunda etapa es importante el grado de naturalidad deseado para la representación del entorno y la realización de las tareas del usuario. Para averiguarlo, el autor lista cuatro preguntas encaminadas a conocer la naturalidad de la interacción en el EV.

El análisis de tareas tiene como resultado descripciones de las tareas o de los casos de uso como servicios ligados a agentes, su localización en el mundo virtual, y los objetos y herramientas que involucran. Estas descripciones documentan las acciones físicas del usuario y sus demandas de percepción, un conjunto de propiedades de la interacción –partes del cuerpo que intervienen, feedback, etc.- que tendrán una gran influencia en el diseño de la presencia del usuario y de los controles. Para este análisis, Sutcliffe aconseja seguir las prácticas IPO, como es tener en cuenta el conocimiento, experiencia, aptitudes o el género y la edad del usuario.

En cuanto al dominio, Sutcliffe lo define como la parte del mundo real que debe ser representada en el EV, y su análisis se lleva a cabo realizando bocetos (*sketches*) del entorno a modelar, tomando fotografías, vídeos y observando a la gente. El análisis de quién hace qué, con qué y dónde nos lleva a describir los agentes, objetos, estructuras espaciales y áreas físicas que constituyen el mundo virtual. Según Sutcliffe, las descripciones son importantes en el caso de los objetos interactivos, en tanto que en el caso de las estructuras físicas sólo son necesarias en el caso de ayudas para la navegación o la manipulación, o si pueden llegar a ser agentes en futuros diseños.

El diseño es la tercera etapa de este proceso. La primera decisión que debe tomarse es elegir entre mundos de escritorio o inmersivos, considerando las diferentes opciones y sus costes. Una vez esa decisión ha sido tomada, el diseño se subdivide en las siguientes actividades:

- diseñar la representación gráfica del mundo virtual y de los objetos y agentes que en él hay;
- seleccionar los dispositivos interactivos y la presencia del usuario;
- y el diseño de la interacción y la manipulación en el mundo virtual.

Sutcliffe apunta que, dado que la interacción influye en la representación, en la práctica el diseño de ambos aspectos se entremezcla.

En cuanto a la representación gráfica, el autor lista los siguientes pasos:

- crear el fondo;
- diseñar los objetos interactivos, agentes y estructuras;
- añadir caminos y señales para la navegación;
- planificar la integración del EV con la interfaz de usuario completa –p. ej. componentes GUI emergentes o que rodeen la ventana que muestra en mundo virtual-.

Con respecto a la última actividad de esa lista, Sutcliffe comenta que conseguir una representación totalmente realista no es posible pues el limitado poder de proceso lleva a tiempos de respuesta bajos y, con ello, a problemas de usabilidad.

Sobre la elección de dispositivos interactivos, el autor comenta diferentes aspectos: acciones del usuario, elección de las modalidades, sustituciones modales, correspondencia entre la entrada y los dispositivos, dispositivos software y reales, usuarios profesionales y público general. El grado de naturalidad puede aconsejar el uso de dispositivos tangibles, lo que según el autor nos acerca a la Realidad Aumentada.

Relacionado con lo anterior es la presencia del usuario, que incluye: su representación (*self*) –que depende de la naturaleza de las tareas (desde una simple flecha de cursor hasta una representación de cuerpo entero) y del punto de vista escogido-; los controles de navegación y movimiento; y la comunicación con otras personas. Esta presencia es una parte del diseño de los agentes interactivos de la aplicación de Realidad Virtual, y que incluye además: agentes automatizados, objetos que responden (objetos activos), y agentes

interactivos conversacionales. La presencia de estos otros agentes inteligentes también debe ser diseñada.

Tomando el diseño del EV y la especificación de la presencia del usuario y de los agentes interactivos, se completa todo ello con el diseño en detalle del diálogo y los controles, así como las facilidades que dan soporte a la interacción.

Tras el diseño, la cuarta etapa del proceso es la creación del prototipo del EV, seguido por la quinta etapa que corresponde a la evaluación de la usabilidad. Esta evaluación lleva a volver una y otra vez sobre el diseño hasta que el prototipo es mejorado y lanzado como producto final. A la hora de evaluar la usabilidad, el autor subraya que deben tenerse en cuenta no sólo los objetivos de la aplicación sino también los límites de la tecnología.

### 3.7.5.2 Análisis

Comparando el proceso descrito por Sutcliffe con el que propone Kaur en su Tesis y que fue analizado en el apartado 3.3.3, la primera diferencia que se observa es el número de etapas, pasando de siete a sólo cinco, aunque podría explicarse si se piensa que las dos etapas de especificación originales se han fundido en una, así como las otras dos etapas de diseño. Sin embargo, la diferencia más notable es que Sutcliffe sí describe en detalle cada una de esas etapas, en tanto que de las etapas de Kaur poco se sabía más allá del nombre.

La descripción que realiza Sutcliffe del proceso es extensa, tanto como pueda ser la de VEDS o SENDA –véanse los apartados 3.5.3 y 3.6.3, respectivamente-, aunque como guía metodológica no llega a ser tan precisa como las citadas. De hecho, podría dividirse el proceso de Sutcliffe en dos partes. La primera corresponde al análisis, en donde además de enumerar las actividades a realizar lista también las técnicas y métodos para su realización. La segunda parte corresponde al diseño, donde describe el gran número de aspectos a tratar, y aunque trata de darles orden, no ofrece ninguna técnica o método para abordarlos, por lo que queda al buen hacer del desarrollador. Aparte quedaría la creación del prototipo de EV y su evaluación, en lo que no entra en mayor detalle.

Reparando en esa primera parte, el uso del análisis de tareas permite compararlo con otras metodologías ya vistas, en particular aquellas que se caracterizaban precisamente por ese análisis y que se estudiaron en el apartado 3.5. Una de esas metodologías era propuesta por Polys para la creación de entornos virtuales con X3D, y que comparte con Sutcliffe el uso de escenarios y estructuras TKS, no en vano Polys se basa en trabajos previos del propio Sutcliffe. En este aspecto

también se puede comparar con metodologías de su mismo grupo, como por ejemplo la que le precede en este texto, MPIUA, con la que comparte en este caso el análisis jerárquico de tareas (HTA). Es importante destacar, sin embargo, que Sutcliffe diferencia entre los modelos de las tareas y el análisis de las tareas. Los modelos representan la descomposición de las tareas, el análisis describe cada tarea en particular.

Además de las tareas, Sutcliffe también propone el empleo de casos de uso en el análisis, como en la metodología SENDA –véase apartado 3.6.3-, aunque advierte que tienden a incluir detalles físicos, y en el análisis debe evitarse cualquier referencia al diseño. Sin embargo, los diagramas de caso de uso que muestra Sutcliffe corresponden en realidad a diagramas de secuencia de mensajes como los que se utilizan en CLEVR –véase apartado 3.6.2-. De hecho, Sutcliffe indica que a partir de ellos puede procederse al análisis de clases de objetos y agentes del mismo modo que se hace en CLEVR, con lo que ambos procesos coincidirían en este comienzo del desarrollo. En CLEVR, sin embargo, el uso de diagramas va más allá de estos primeros pasos.

Otro aspecto a destacar de las etapas de análisis descritas por Sutcliffe es la importancia que da al grado de naturalidad deseado, y que influye en la descripción de las tareas y en el diseño posterior de la interacción. Como se recordará del apartado 2.3.1, Sutcliffe distinguía entre tres tipos de aplicaciones según ese grado: EV's naturales, híbridos y artificiales. Para averiguarlo, el autor propone un cuestionario que podría relacionarse con otro similar propuesto en la metodología SENDA, también para el análisis, y descrito en [Sánchez, 2005a].

Con el grado de naturalidad definido, la metodología continúa con el análisis de las tareas y el análisis del dominio. El primero, que no es lo mismo que el modelado de las tareas, como ya se ha advertido, tiene entre otros fines localizar las tareas en el mundo virtual, algo que recuerda la asociación de tareas y lugares en RealPlaces [IBM, URL] y también al grafo de escena anotado visto en el apartado 3.7.1. El análisis del dominio, por su parte, recuerda al proceso de recogida de material de referencia de los modelos del mundo real que observó Kaur –véase apartado 3.3.2- y al modelado conceptual que propone Fencott en su metodología –no en vano este último está basado en el primero, como se vió en el apartado 3.3.3-, por el uso de medios como bocetos, fotografías y vídeos para el análisis y descripción de la parte del mundo a ser representada en el EV.

En el diseño destaca la decisión entre mundos de escritorio e inmersivos, de forma que el diseño se lleva a cabo dentro de los límites de la tecnología elegida. Tras esa decisión, Sutcliffe lista muchos aspectos a tratar del EV, aunque de todos ellos cabe subrayar tres, por su relación con otras metodologías anteriores. Por una parte, plantea como posibilidad el uso de dispositivos reales, lo que recuerda a la distinción entre objetos reales y virtuales de VRID. Según el autor,

su elección nos acerca a la Realidad Aumentada, aunque más concretamente hablaríamos de Virtualidad Aumentada, según el continuo descrito en el apartado 2.3.8. El segundo aspecto a destacar es el diseño de la presencia del usuario y de los demás, siendo la segunda metodología, después de SENDA, que trata la problemática de los avatares. Y el tercer aspecto a destacar es el soporte de la interacción, que lo relaciona con la metodología VEDS.

Con todo, se trata de un proceso de diseño muy exhaustivo en su descripción pero que, precisamente en la etapa de diseño, se le echa en falta un orden más exacto de las actividades a realizar, pero sobre todo una descripción de las guías y herramientas a utilizar.

### 3.7.6 Análisis del grupo

Al hablar de desarrollo de interfaces de usuario que van más allá del PC se pretende dejar atrás la típica combinación de pantalla, teclado y ratón, con la introducción de dispositivos no convencionales como lo son, por ejemplo, los de Realidad Virtual. Estos requieren nuevas técnicas de interacción, y también un software que soporte su desarrollo. La elección del hardware, el diseño de las técnicas y el software para el desarrollo son abordados en las metodologías incluidas en este grupo, aunque de diferentes formas.

Así, respecto a la elección del hardware, podría argumentarse que ya metodologías anteriores se planteaban las posibilidades de la tecnología –p. ej. en la propuesta de Celentano y Pitarello descrito en el apartado 3.4.2- o las limitaciones de la misma –p. ej. en el tutorial de B. Hay analizado en el apartado 3.3.4-, pero bien diferente es el hecho de tener que definir qué elementos actuarán de interfaces físicos, cuáles de interfaces lógicos y cómo será la correspondencia, como plantean Parés y Parés en su artículo. En VRID, por ejemplo, se distingue entre objetos virtuales y reales, estos últimos formando parte de la interfaz física, como por ejemplo la aguja de cirujano en el sistema quirúrgico de Realidad Aumentada que los autores de esta metodología utilizan como caso de estudio en su artículo. En el modelo MPIUA, por su parte, se evalúan diferentes dispositivos gracias al prototipado, como por ejemplo una tableta de madera para representar una ventana de Realidad Aumentada. Y en el caso de la propuesta de Sutcliffe, uno de los pasos a dar consiste en elegir entre mundos de escritorio o inmersivos, y en el diseño se tienen en cuenta no sólo las características de los dispositivos, sino también las diferentes modalidades, como ya se comentó en el apartado 2.6.4.

En cuanto al diseño, se recurre a diagramas y notaciones que permitan representar no sólo entradas discretas, sino también continuas –p. ej. la posición

de la mano del usuario seguida por un sistema electromagnético-. Pero, más que entradas independientes, se entiende que entre ambos tipos existe una relación, y es que los flujos de datos son controlados por acciones discretas. Para estos sistemas híbridos, diversos autores coinciden en distinguir en sus propuestas, precisamente, entre flujo y control. Este es el caso del proyecto INQUISITIVE y las redes de flujo que en él se introducen, y el del formalismo introducido por Jacob y utilizado en VRID. Pero esa separación también se vio antes en CLEVR, por lo que en los análisis realizados se ha aprovechado para compararlas con esta otra. Curiosamente, en todos esos casos las notaciones fueron primero propuestas para especificar la interfaz de usuario –véase [Jacob, 1996], [Kim, 1998], [Smith, 1999]-, y después se vio que también podían servir para describir la función y comportamiento de los objetos –véase [Willans, 2000], [Tanriverdi, 2001], [Seo, 2001]-. Más aún, en las tres el diseño está soportado por herramientas gráficas, como el editor VRED, o los conjuntos MARIGOLD y P-VoT. La Tabla 3.6 resume estas tres propuestas.

	VRID	CLEVR	INQUISITIVE
Flujo de datos	Grafo de flujo de datos	Diagrama de flujo de datos (DFDs)	Extensión de las redes de Petri a flujos de datos
Control	Diagrama de estados	Diagrama de estados	Redes de Petri
Representación	Relacionadas a través de enlaces y condiciones	Relacionadas	Combinada: redes de flujo o “flownets”
Herramienta visual	VRED Editor	P-VoT	MARIGOLD Toolset

**Tabla 3.6** Distinción entre flujo y control en varias propuestas

Distinto a esas tres propuestas es el diseño en la metodología MPIUA, en la que se combina un mapa de posiciones de interés con diagramas de estados, representando estos últimos el diálogo en cada una de esas posiciones. Sutcliffe, por su parte, no recurre a ninguna notación en particular, aunque sí deja claro que en el análisis de las tareas deben indicarse las propiedades de interacción que deberán tenerse en cuenta en propio diseño, según el grado de naturalidad.

Y, en lo que se refiere al software de desarrollo, precisamente el proceso de diseño propuesto para el proyecto INQUISITIVE gira en torno a la elección de un toolkit con el que cumplir los requisitos del diseño. En el caso de Parés y Parés, los dos últimos pasos de cualquiera de sus dos estrategias identifican las herramientas necesarias, entre ellas las de desarrollo de aplicaciones. En VRID, por último, se confía en PMIW para la simulación y prueba de los diseños.

Con todo, podría decirse que las descripciones de Parés y Parés, así como la de Sutcliffe, dan buena forma a los pasos iniciales del desarrollo, en tanto que se muestran menos precisas en el diseño. Para ese diseño, la propuesta del proyecto INQUISITIVE y la metodología VRID parecen más adecuadas, sobre todo por las herramientas que las acompañan. Aunque cualquiera de ellas bien podría beneficiarse del punto de vista que, de la usabilidad y la accesibilidad, ofrece la metodología MPIUA. En realidad, los autores de esta última metodología la han aplicado en mayor medida en la creación de interfaces 2D, por ejemplo sitios Web. Precisamente, el siguiente grupo de metodologías que se analiza en este capítulo está orientado hacia interfaces convencionales como esas.

### 3.8 Métodos para el desarrollo de interfaces convencionales

En [Pereira, 2000] se afirmaba que las técnicas para el desarrollo de proyectos de interfaz para ambientes virtuales son semejantes a los procesos utilizados en la construcción de otras interfaces computacionales. Sin embargo, en [Méndez, 2001] se puso de relieve las carencias de un método de desarrollo de interfaces convencionales –en particular, el de Larman- al ser aplicado en la creación de un entorno virtual.

A pesar de las diferencias que obligan a proponer una metodología específica para el desarrollo de interfaces de usuario 3D, en este apartado se analizarán cuatro metodologías de desarrollo de interfaces convencionales con un doble propósito: en primer lugar, identificar aquellas actividades comunes al desarrollo de cualquier tipo de interfaz que no hayan sido tenidas en cuenta en las metodologías vistas hasta el momento; en segundo lugar, analizar otros enfoques utilizados en el desarrollo de interfaces de usuario que puedan ser también útiles en el caso de las tridimensionales.

#### 3.8.1 Metodología de diseño según Foley *et al.*

##### 3.8.1.1 Descripción

En [Foley, 1996] se describe una metodología de diseño de interfaces de usuario basada en la distinción entre dos componentes –significado (*meaning*) y forma (*form, look and feel*)-, en cada uno de cuales se distinguen a su vez dos elementos –conceptual (modelo del usuario) y funcional (semántico) en el primero, y secuencia (sintáctico) y vinculación con el hardware (léxico) en el segundo-, conformando así el modelo de cuatro niveles que proponen los autores. La metodología sigue entonces los siguientes pasos:

1. Definición de requisitos
2. Diseño conceptual
3. Diseño funcional
4. Diseños de secuencia y vinculación
5. Prototipado de la interfaz de usuario

El primer paso en el diseño de la interfaz es decidir qué se pretende con ella, para lo cual los autores indican que: hay que comprender los requisitos del usuario, qué tareas hacen actualmente y porqué; también deben identificarse las características del usuario, sus habilidades y conocimientos, su actitud y la frecuencia con la que hará uso de la interfaz.

Una vez definidos los requisitos, se procede a diseñar cada uno de los niveles de la interfaz siguiendo un diseño top-down, abordando primero las cuestiones de diseño más generales y más tarde las de más bajo nivel. Así, se empieza por el nivel conceptual, desarrollando varios diseños conceptuales alternativos y evaluándolos en función de cómo de bien permitirá a los usuarios llevar a cabo las tareas identificadas en la definición de requisitos –p. ej., las tareas más frecuentes deberían ser especialmente sencillas-.

El diseño funcional se centra en las órdenes, la información que precisan, sus efectos, la información que se presenta al usuario y las posibles condiciones de error. Uno de los objetivos aquí es minimizar el número de errores posibles, definiendo las órdenes apropiadamente.

Los siguientes diseños definen juntos la forma de la interfaz y por ello, según los propios autores, es mejor abordarlos en conjunto. En primer lugar se selecciona un conjunto apropiado de estilos de diálogo, y entonces se aplican estos estilos a la funcionalidad en cuestión. Para detallar las secuencias usuario-acción los autores aconsejan el uso de diagramas de estados, en tanto que para definir los aspectos visuales y algunos de los temporales se puede recurrir a dibujar secuencias de pantallas como storyboards.

Los autores también comentan que la forma de la interfaz puede definirse mediante una guía de estilo, una codificación escrita de los muchos elementos de la interfaz de usuario, algunos de los cuales pueden ser implementados en librerías de técnicas de interacción (*toolkits*).

En cuanto al prototipado, Foley *et al.* indican que puede comenzar tan pronto como esté listo el diseño conceptual, de modo que el diseño funcional y el estilo de diálogo puedan ser desarrollados concurrentemente. Los autores opinan que el prototipado ayuda enormemente al diseño, ofreciendo a los usuarios un marco concreto de referencia sobre el que hablar de sus necesidades. Precisamente, los

autores añaden que, tan pronto como algunos de los elementos de la interfaz estén ya desarrollados, los usuarios deberían verlos de modo que puedan sugerir cómo mejorar la interfaz. Una vez se hagan las modificaciones y el prototipo crezca, los usuarios deberían volver a trabajar con el sistema, transformando el proceso en un ciclo iterativo.

### 3.8.1.2 Análisis

Como metodología de desarrollo, podría decirse que la propuesta de Foley *et al.* se basa en tres pilares. El primero de ellos es que antes de diseñar la interfaz es preciso tener bien claro saber cuál es su propósito, lo que coincide con lo que luego apuntaría Shneiderman y que se comentó al principio de este capítulo, esto es, que sólo después de recoger los datos y requisitos preliminares es posible comenzar con el diseño [Shneiderman, 1998].

El segundo pilar es la separación entre forma y función, y más concretamente que la forma debe seguir a la función. No seguir este principio es precisamente el error de las propuestas más ingenuas de desarrollo de mundos virtuales, y que como hemos visto en este capítulo se trató de corregir en propuestas posteriores ajustando la apariencia de los objetos a la interactividad que ofrecen –véase apartado 2.3.1-.

El tercer pilar es el usuario. Se tienen en cuenta sus necesidades y características antes del diseño, y durante el mismo se lleva cabo un proceso iterativo en el que se crea un prototipo de la interfaz que se evalúa con el usuario para mejorarlo en la siguiente iteración, haciendo de este proceso algo participativo. El resultado es una metodología que bien puede calificarse de centrada en el usuario.

Con todo, no cabe duda de que la metodología descansa sobre el modelo de cuatro niveles que los autores proponen para entender las interfaces de usuario: conceptual, funcional, secuencia y vinculación con el hardware. Así, el primero da lugar al diseño conceptual, el cual podría relacionarse con otras etapas en varias metodologías anteriores, como la generación de la idea de Neale y Nichols –véase apartado 3.4.1-, la fase conceptual de Celentano y Pitarello –apartado 3.4.2-, la etapa de concepción de la propuesta brasileña –apartado 3.5.2-, el diseño del concepto en VEDS –apartado 3.5.3-, o el modelado conceptual de Fencott –apartado 3.6.1-. El segundo nivel da lugar al diseño funcional, que también podría relacionarse con otras etapas de metodologías previas, en concreto con aquellas dedicadas al análisis de tareas, pero especialmente según lo describe Sutcliffe, es decir, abordando no la jerarquía de tareas sino cada tarea en particular, como un servicio –según el propio Sutcliffe- o función –según Foley *et al.*-. Los dos últimos niveles dan lugar al cuarto paso

de esta metodología de diseño, en donde la descripción del diálogo por medio de diagramas de estados recuerda al modelo de procesos MPIUA visto en el apartado 3.7.4, y el uso de storyboards para definir aspectos visuales y temporales recuerda al proceso de producción de una película de animación como el que se vio en el apartado 3.2.1, aunque tampoco es la única propuesta en recurrir a esa herramienta.

### 3.8.2 LUCID: Logical User-Centred Interaction Design

#### 3.8.2.1 Descripción

En [Cognetics, URL] se presenta la metodología LUCID como un marco de trabajo que proporciona una estructura para conducir las actividades de producto, diseño de interfaz y usabilidad, junto con las herramientas necesarias para gestionar esas actividades, y de la que se dice además que combina con las metodologías de desarrollo más populares. De ella se dice también que es el resultado de una validación y refinamiento a través de múltiples proyectos, y aunque se reconoce que cada proyecto tiene sus propias necesidades, todos ellos pasan por seis etapas, cada una con su propio conjunto de actividades de gestión y diseño, así como herramientas:

1. Visionar (*Envision*, desarrollo del concepto de producto)
2. Descubrimiento (*Discovery*, realizar estudio y análisis de requisitos)
3. Cimientos del diseño (*Design Foundation*, diseño de conceptos y prototipo de pantallas clave –*key-screen prototype*-)
4. Diseño detallado (*Design Detail*, realizar diseño y refinado iterativo)
5. Construcción (*Build*, implementar el software)
6. Lanzamiento (*Release*, proporcionar soporte durante su implantación)

Con la primera etapa (Visionar) se persigue que tanto el equipo del proyecto como el resto de implicados (*stakeholders*) compartan una misma visión del producto, así como de sus objetivos y limitaciones. El centro es pues la creación del concepto de alto nivel del producto (*high concept*), un breve texto que define los objetivos, la funcionalidad y los beneficios del producto (*envision statement*). Además, en esta etapa se ensambla el equipo del proyecto, se asignan roles y responsabilidades, se dispone un plan de proyecto y un presupuesto. También se identifican los grupos de usuarios principales, los objetivos de usabilidad preliminares, y posibles limitaciones técnicas, de entorno o legales. Las ideas de diseño se plasman en bocetos (*concept sketches*, como prototipos en papel o en pantalla –*early paper/on-screen prototypes*-).

En la segunda etapa (Descubrimiento) se persigue comprender las necesidades y competencias de los usuarios, el proceso de negocio a soportar y los requisitos funcionales del sistema. Se conduce entonces un análisis de usuario y necesidades a través de diseño participativo (entrevistas, tests de usabilidad, etc.). Se elaboran “personas” (perfiles de usuarios) y escenarios de trabajo que muestran sus tareas y a partir de ellos se derivan los requisitos de alto nivel. Además, se resuelven problemas técnicos y otras limitaciones y se revisan los objetivos de usabilidad.

En la tercera etapa (Cimientos del diseño) se crea el diseño conceptual de la interfaz de usuario, sus objetos y sus metáforas, para lo que se selecciona un modelo de navegación, se identifican los tipos básicos de pantalla, se perfila el diseño de las pantallas clave, se empieza la creación de guías de estilo, y se documenta el modelo mental bajo la interfaz de usuario. Con una herramienta de prototipado rápido se crea el prototipo de pantallas clave –*key-screen prototype*, desarrollado usualmente por un programador del equipo de desarrollo de software- que incluye los principales caminos de navegación y que es primero probado con usuarios y después aprobado por la dirección del proyecto.

A esa tercera etapa le sigue el Diseño detallado, donde se completa el diseño de la interfaz de usuario, sirviendo el prototipo como parte de las especificaciones que se entregan a los desarrolladores, además de una guía de estilo, diseños de las ventanas y una especificación uno-por-uno de cada elemento. En este diseño y refinado iterativo se realizan revisiones heurísticas y expertas, evaluaciones de usabilidad de partes concretas del diseño o del todo.

Con las especificaciones en mano, los desarrolladores llevan a cabo la construcción o implementación del software en una quinta etapa, según la práctica estándar, pero trabajando en conjunto con los diseñadores para resolver las cuestiones de última hora que puedan surgir y que puedan suponer un rediseño de pantallas o la realización de nuevas evaluaciones si fuera preciso.

Finalmente, en una sexta y última etapa se produce el lanzamiento del producto, diseñando y probando la experiencia inicial del cliente con el propio producto, realizando evaluaciones de usabilidad del mismo, y midiendo la satisfacción del usuario.

### 3.8.2.2 Análisis

Shneiderman describe LUCID como una metodología comercial que, además de los elementos propios de las soluciones académicas, resalta las estrategias para cumplir el plan y el presupuesto, especifica resultados concretos para diferentes etapas del diseño, e incorpora análisis de coste/beneficio y retorno de la

inversión para facilitar la toma de decisiones [Shneiderman, 1998]. Estas características destacan a LUCID sobre cualquier otra metodología aquí analizada, pues aunque pueden establecerse similitudes –por ejemplo, las etapas tres y cuatro recuerdan al diseño general y diseño detallado de la metodología VEDS según la describe [Eastgate, 2001]-, la mayoría de ellas proceden del mundo académico y no tienen en cuenta el aspecto de negocio.

Además, Shneiderman considera que LUCID también es un marco en el que pueden incorporarse diferentes estrategias, como la observación etnográfica, el diseño participativo, el desarrollo de escenarios o la revisión de informes de impacto social. Estas estrategias son de gran interés en el campo de la interacción persona-ordenador, por lo que la posibilidad de incorporarlas también es un punto a favor de LUCID.

Con todo, hay un detalle en LUCID que, cuando menos, resulta extraño, y es que en la primera etapa –Visionar- se planteen ya bocetos del sistema –aunque a alto nivel- cuando aún no se conoce realmente para qué servirá el sistema, lo cual sólo se sabrá tras completar la etapa siguiente –Descubrimiento-. En este sentido, parece más sensata la propuesta de Foley *et al.* antes descrita, y en la que se afirmaba que lo primero en el diseño de una interfaz es averiguar qué se pretende con ella.

### **3.8.3 OVID: Object View Interaction Design**

#### **3.8.3.1 Descripción**

En [Bardon, 2001] se desarrolla un enfoque de creación de interfaces de usuario basado en la metodología OVID de IBM, pero que no se limita tan sólo a la fase de diseño, sino que extiende el enfoque centrado en el usuario y las técnicas y herramientas de ingeniería del software (principalmente diagramas UML) a todas las fases, al proceso completo de crear una experiencia de usuario (*total user experience*) desde el conocimiento inicial de los usuarios y su entorno hasta el despliegue y la evaluación de la aceptación por parte del usuario.

Este enfoque distingue tres modelos: el del diseñador, el del implementador y el de los usuarios. Si la implementación es fiel al diseño y las deducciones de los usuarios son correctas, entonces el modelo de los usuarios debe corresponderse con lo que se pretendía, esto es, el modelo del diseñador. Se entiende además que los usuarios interactúan con el sistema a través de vistas (*views*) de los objetos, siendo la separación entre esas vistas –que encapsulan el “look and feel”- y el modelo de objetos del usuario el principio del modelo del diseñador.

Más aún, esas vistas se diseñan a dos niveles: uno abstracto, especificando qué se presenta al usuario, independientemente de la plataforma o el dispositivo; y otro de presentación, específico para una plataforma o dispositivo particular.

En cuanto a las fases que distingue este enfoque, son cuatro, todas ellas empezando por la letra ‘D’:

1. Descubrir (*Discover*)
2. Diseñar (*Design*)
3. Desarrollar (*Develop*)
4. Desplegar (*Deploy*)

En primer lugar, se escriben historias –*stories*, escenarios que describen cómo los usuarios logran sus objetivos, pero adornados con atributos del usuario y del entorno- como entrada para la primera fase. En esta fase –Descubrir- se determina quiénes son los usuarios reales y se documenta en un diagrama de actores UML representando grupos de usuarios y sus atributos. Después se determina cuáles son sus objetivos y sus criterios de éxito, y se documenta esta vez como clases y atributos de un diagrama de clases. Seguidamente, se estudia cómo esos objetivos son logrados hasta el momento por los usuarios, y las tareas se plasman en diagramas de actividades. Tras todo ello, se especifica la funcionalidad que deberá ofrecer el sistema como un conjunto de casos de uso.

En la segunda fase –Diseño-, se deja a un lado el análisis para centrarse en el modelo del diseñador, empezando por el diseño abstracto (Diseño I), y continuando con el diseño a nivel de presentación (Diseño II).

Así, primero se decide para quién se está realmente diseñando el sistema, produciendo un nuevo diagrama de actores basándonos en los casos de uso. Después se revisan los diagramas de actividad en busca de los objetos que los usuarios necesitan para realizar las tareas –esto es, el modelo mental del usuario-, y se crea un diagrama de clases con sus relaciones, atributos y operaciones. A continuación, se proponen esquemas para las tareas usando nuevos diagramas de actividad que separan lo que hace el usuario de lo que hace el sistema. Tras ello, se determinan qué objetos deben ver los usuarios para realizar cada tarea, y se añaden vistas al diagrama de clases. De forma más detallada, se especifica la interacción entre el usuario y el sistema mediante diagramas de secuencia que reúnen a usuarios y vistas. Por último, se documenta lo que ocurre con los objetos cuando el usuario interacciona con ellos como diagramas y tablas de estados. El diseño abstracto termina cuando se han procesado todos los casos de uso, no se han encontrado nuevos objetos y el modelo ha sido validado con usuarios –usando prototipos de baja fidelidad o “low fidelity prototypes”-.

El diseño continúa ya a nivel de presentación (Diseño II), lo que los autores llaman también como fase de realización (*realization*), con la que se persigue concretar el diseño abstracto en un entorno específico, teniendo en cuenta la tecnología –plataforma hardware (dispositivos de entrada y presentación) y software-, y la accesibilidad, cultura y geografía. Todo ello configura la paleta gráfica para el diseño visual, que se recomienda sea abordado desde tres puntos de vista: identidad visual de cada objeto, diseño de las vistas de presentación, y la experiencia completa del usuario. Se presentan bocetos de diferentes alternativas de diseño, decidiendo cuándo embeber unas vistas en otras y cuándo decantarse por usar navegación de una a otra, y determinando cuándo y cómo deben advertirse las transiciones de estado de los objetos.

Tras sucesivas iteraciones sobre el diseño, realizando y probando prototipos, se alcanza el final del proceso, y se entrega a los desarrolladores el modelo de vistas de presentación (*Presentation View Model*) que refleja la estructura del diseño final. Entonces comienza la tercera fase –Desarrollo-, donde se crea la experiencia deseada para el usuario, cuidando que la implementación sea fiel al diseño, aunque a veces puede requerirse iterar de nuevo sobre el diseño.

Terminada esa tercera fase, en la cuarta –Despliegue- se evalúa el rendimiento y la satisfacción del usuario, usualmente mediante cuestionarios.

### 3.8.3.2 Análisis

El enfoque que nos proponen estos autores guarda muchas similitudes con las dos propuestas previamente analizadas, la metodología de diseño de Foley *et al.* y LUCID. Con respecto a la primera, comparte perseguir en un primer momento conocer quién y para qué utilizará la interfaz de usuario y en el diseño posterior separar el qué y el cómo –el significado y la apariencia o "look and feel"-. Así, la primera fase –Descubrir- puede corresponderse en la metodología de Foley *et al.* con la etapa de definición de requisitos, el diseño abstracto con el diseño conceptual y funcional, y la realización con los diseños de secuencia y vinculación. Por su parte, aunque LUCID no se basa en esa separación de vistas, también guarda similar paralelismo. Así, la fase Descubrir puede corresponderse también en la metodología LUCID con la etapa Descubrimiento, la fase Desarrollar con la de Construir, y la de Desplegar con la de Lanzamiento.

Pero el enfoque de Bardón *et al.*, basado en OVID, también recuerda a otras metodologías vistas anteriormente. Así, la distinción explícita entre cómo se realizan las tareas en la realidad –modeladas mediante diagramas de actividad en la fase Descubrir- y cómo se llevarán a cabo con el nuevo sistema –plasmadas en diagramas de actividad y de secuencia en la fase Diseño I- es comparable a la

distinción que se realiza en la metodología VEDS –véase apartado 3.5.3- entre tareas reales y tareas virtuales. Precisamente, una de las diferencias importantes del enfoque de Bardon *et al.* frente a esas otras metodologías es el uso de técnicas y herramientas de ingeniería del software, y más concretamente diagramas UML, para modelar cada aspecto de la realidad o del sistema. El uso de esos diagramas acerca más este enfoque a metodologías como CLEVR o SENDA analizadas anteriormente –véanse los apartados 3.6.2 y 3.6.3-.

Aún y todo, si bien las fases Descubrir y Diseño I (diseño abstracto) poseen una estructura bien definida dada por un conjunto de actividades en secuencia, y para cada una de las cuales se indica con qué diagrama UML documentar, a partir de ahí el desarrollo es menos sistemático. Por ejemplo, en la fase Diseño II (realización) se indican tres puntos de vista diferentes desde los que abordar el diseño visual, pero que los propios autores subrayan que no deben tomarse como pasos en un proceso lineal. Precisamente, este diseño de presentación tendría gran importancia en el desarrollo de una interfaz de usuario 3D, pues es donde se tiene en cuenta la tecnología y se realiza el diseño de interacción. Sin embargo, si bien los autores demuestran preocupación por la tecnología –con comentarios como que el dispositivo de presentación puede influir en las decisiones sobre diseño y partición de las vistas, o que los dispositivos de entrada pueden imponer ciertos mecanismos de interacción-, la propuesta se centra en el diseño visual, como ya ocurriera en LUCID.

### 3.8.4 IDEAS: Interface Development Environment within OASIS

#### 3.8.4.1 Descripción

IDEAS [Lozano, 2001] pretende ser un sistema de desarrollo automático de interfaces integrado en el modelo orientado a objetos OASIS [Letelier, 1998] para permitir la producción automática de interfaces de usuario de calidad, y ha sido probada con éxito en el desarrollo de aplicaciones WIMP, tanto para entornos de escritorio como para la Web –véase, p. ej., [Lozano, 2002a]-. IDEAS está basado en modelos abstractos que son empleados para modelar y comprender la realidad. El proceso de desarrollo de la interfaz de usuario está estructurado en cuatro niveles de abstracción:

1. Requisitos
2. Análisis
3. Diseño
4. Implementación

Esta estructuración vertical muestra el proceso de reificación que se sigue desde el primer y más abstracto nivel hasta alcanzar finalmente la implementación del sistema, que constituye el último nivel. En cada nivel de abstracción, diferentes modelos se utilizan para capturar y modelar los múltiples aspectos de la interfaz gráfica de usuario.

Así, a nivel de requisitos se crean tres modelos: Modelo de Casos de Uso, Modelo de Tareas y Modelo de Usuario. El primero captura los casos de uso que deberá soportar el sistema, y por cada uno de ellos habrá una o más tareas que serán documentadas en el Modelo de Tareas. Este segundo modelo define el conjunto ordenado de actividades y acciones que el usuario realiza para lograr un objetivo concreto, descritas en lenguaje natural siguiendo una plantilla dada. Por último, el Modelo de Usuario describe las características de los diferentes tipos de usuarios.

A continuación, a nivel de análisis se crea el Modelo de Dominio, formado por diagramas de secuencias que definen el comportamiento del sistema y los objetos participantes se definen en los correspondientes diagramas de clases junto con las relaciones entre ellas y el rol de cada una de ellas (lo que define otro modelo, el Modelo de Roles).

El siguiente nivel es el de diseño, y en él se crea el Modelo de Diálogo, a partir del cual la metodología comienza a abordar los aspectos gráficos de la interfaz de usuario final. En este caso, el propósito es describir la estructura sintáctica de la interacción usuario-ordenador, definiendo cuándo puede el usuario ejecutar órdenes, seleccionar o especificar los datos de entrada y cuándo el ordenador debe solicitar nuevos datos al usuario o presentar los datos de salida. Se definen diagramas para describir las transiciones entre ventanas, y diagramas para especificar los elementos de cada ventana y su comportamiento. A este nivel todos los elementos son Objetos Abstractos de Interacción (AIOs, *Abstract Interaction Objects*).

El nivel de más baja abstracción es el de implementación, donde se crea el Modelo de Presentación, el cual describe los objetos de interacción concretos (CIOs, *Concrete Interaction Objects*) que compondrán la interfaz de usuario final. Esta interfaz de usuario no es programada de forma manual, sino que es generada de forma automática según un conjunto de reglas que traducen los objetos abstractos en objetos concretos, teniendo en cuenta las guías de estilo de la plataforma final.

Todo el proceso descrito está soportado por la herramienta IDEAS-CASE que Lozano también presenta en su Tesis –véase también [Lozano, 2002b]-.

### 3.8.4.2 Análisis

La propuesta de Lozano también posee puntos en común con las propuestas precedentes, como por ejemplo la separación entre el nivel de diseño abstracto y el de presentación que ya se vio en OVID. También en esta ocasión se hace uso de diagramas UML para capturar los diferentes aspectos de la realidad y el sistema, si bien en IDEAS se definen nuevos diagramas para representar la navegación de una ventana a otra y la interacción dentro de cada ventana que no se encuentran en las propuestas anteriores.

Pero más allá de proponer un nuevo conjunto de diagramas, IDEAS representa dos importantes novedades. La primera de ellas es un enfoque fuertemente basado en modelos. No es sólo contar con diferentes tipos de diagramas que sirvan para modelar cada cuestión particular, sino que se hacen explícitas las conexiones entre unos diagramas y otros y el conjunto se reúne en un modelo, y la suma de los modelos representa el sistema. La segunda de esas novedades es la generación automática de la interfaz de usuario frente al diseño heurístico que se proponía, por ejemplo, en OVID. Con la asistencia de la herramienta IDEAS-CASE, el desarrollador se centra en modelar los casos de uso, las tareas, los usuarios, el dominio y el diálogo, delegando la generación de código sobre esa herramienta.

En conjunto, IDEAS e IDEAS-CASE es un binomio que representa el grado de maduración alcanzado en las interfaces convencionales, donde los desarrolladores apenas tienen que preocuparse por cuestiones de tecnología o interacción que afortunadamente han sido resueltas a lo largo de todos estos años de investigación. El hecho de que las interfaces de usuario 3D estén lejos aún de alcanzar ese mismo estadio basta para imaginar las dificultades que existen para aplicar en su desarrollo un enfoque como el que propone Lozano.

### 3.8.5 Análisis del grupo

Las metodologías incluidas en este último grupo no están dirigidas a la creación de mundos virtuales o de interfaces de Realidad Aumentada como venía siendo la tónica a lo largo de este capítulo. En su lugar, se trata de metodologías claramente orientadas hacia la creación de interfaces de usuario más convencionales, interfaces bidimensionales basadas en pantallas y ventanas. Como en otros grupos, estas metodologías presentan muchas similitudes entre sí, pero también notables diferencias.

Así, uno de los parecidos que guardan estas cuatro metodologías es su preocupación por conocer mejor al usuario y sus tareas. Esta preocupación se

concreta, según Foley *et al.*, en la etapa de definición de requisitos, la primera de la metodología propuesta por estos autores, coincidiendo con IDEAS pues en esta última también se trata en la primera etapa, llamada “requisitos”. En el caso de la metodología OVID, también es la primera etapa, llamada “descubrir”, similar ahora al nombre de etapa que lo aborda en la metodología LUCID, “descubrimiento”, aunque en esta última es la segunda y no la primera.

Otra de las similitudes es la atención que prestan al prototipado y la evaluación. Empezando por Foley *et al.*, estos autores animan a crear prototipos tan pronto esté listo el diseño conceptual, y a mostrarlo a los usuarios para que puedan contribuir al desarrollo con sus comentarios. En el caso de LUCID, la primera de las dos etapas de diseño –llamada “cimientos del diseño”- ya involucra la creación de un primer prototipo de pantallas clave. En OVID también se divide el diseño en dos, a nivel abstracto y de presentación, y tras cada una se crean prototipos que son evaluados, destacando la creación de prototipos de baja fidelidad tras el primero con el fin de validar el diseño. Por último, la evaluación de las interfaces producidas siguiendo la metodología IDEAS lleva a realimentar el proceso de desarrollo en cualquiera de sus cuatro niveles.

La diferencia más notable se encuentra en la herramienta en la que los autores de cada metodología confían el desarrollo. Así, Foley *et al.* apuntan a un UIMS – véase apartado 2.7.8-. En el caso de LUCID se recurre a una herramienta de prototipado rápido operada por un programador del equipo de desarrollo. OVID, por su parte, se apoya en la herramienta Rational Rose para la creación de los diagramas, al menos hasta llegar a la fase de realización. Por último, la metodología IDEAS está soportada por su propia herramienta CASE y de compilación de modelos, llamada IDEAS-CASE.

En cualquier caso, la suerte de estas metodologías está ligada a la madurez de las interfaces bidimensionales, formadas por un conjunto de elementos bien conocidos con los que el usuario interactúa de forma también bien conocida, por lo que el interés está en la disposición de esos elementos en la pantalla y en la navegación de pantalla a pantalla. Además, no preocupa el rendimiento del hardware, pues es de suponer que cualquier máquina actual es capaz de ejecutar, sobradamente, la interfaz bidimensional resultante. Por el contrario, las interfaces de usuario 3D no tienen estas ventajas, por lo que si bien estas metodologías presentan características muy interesantes, también es cierto que resultan insuficientes para abordar el desarrollo de este otro tipo de interfaces.

### 3.9 La metodología IDEAS-3D

La metodología IDEAS que se estudió en el apartado 3.8.4 ofrece un buen número de útiles herramientas de abstracción, incluyendo aquellas que ayudan al desarrollador a recoger los requisitos de usuario y clasificar tareas y subtareas, y ha sido aplicada con éxito en el desarrollo de aplicaciones de escritorio y Web. Esas razones, unidas a la experiencia previa del autor de esta Tesis con dicha metodología, llevaron a pensar que IDEAS también podía ser un buen punto de partida para el desarrollo de interfaces de usuario 3D de forma sistemática.

Esta idea fue esbozada en un primer momento en [Molina, 2002]. Para hacerla posible, se proponía extender la metodología de tal modo que el modelo de diálogo incluyese no sólo ventanas sino también conceptos de los entornos 3D como puedan ser las habitaciones o lugares, y que la generación final de la interfaz gráfica acomodara también lenguajes de descripción de escenas 3D, como el estándar VRML97 o, su sucesor, el lenguaje X3D –por entonces aún no estandarizado-. Posteriormente, la idea fue desarrollada y plasmada en varias publicaciones –véanse [Molina, 2003a], [Molina, 2003b] y [Molina, 2003c]-. El resultado fue una metodología orientada no sólo al desarrollo de interfaces de usuario para diferentes plataformas, ya sean ordenadores personales o dispositivos móviles, sino también al desarrollo de versiones 2D y 3D de la interfaz de una misma aplicación, esto es, produciendo una interfaz gráfica WIMP similar a la de otras aplicaciones convencionales o una interfaz 3D similar a la de escritorios 3D o entornos Web tridimensionales. Para ello, se propusieron una serie de modificaciones a la metodología IDEAS original, a lo largo de sus cuatro niveles de abstracción –requisitos, análisis, diseño e implementación-, cambios que se comentan en los siguientes párrafos, dando como resultado la metodología IDEAS-3D ilustrada en la Figura 3.7.

Así, y comenzando a nivel de requisitos, el Modelo de Casos de Uso se mantiene tal cual, pero en el caso del Modelo de Tareas y el Modelo de Usuario se apuntaba que el desarrollador de la interfaz de usuario pudiera recoger nueva información relacionada con la tercera dimensión del espacio. En lo que respecta a las tareas, esta información se refiere al lugar donde las tareas son llevadas a cabo, dato que puede ser añadido a la plantilla de tareas en el Modelo de Tareas. Por otra parte, los usuarios también pueden expresar sus preferencias acerca del tipo de visualización, ya sean gráficos 2D o 3D, información que el desarrollador recopila en el Modelo de Usuario.

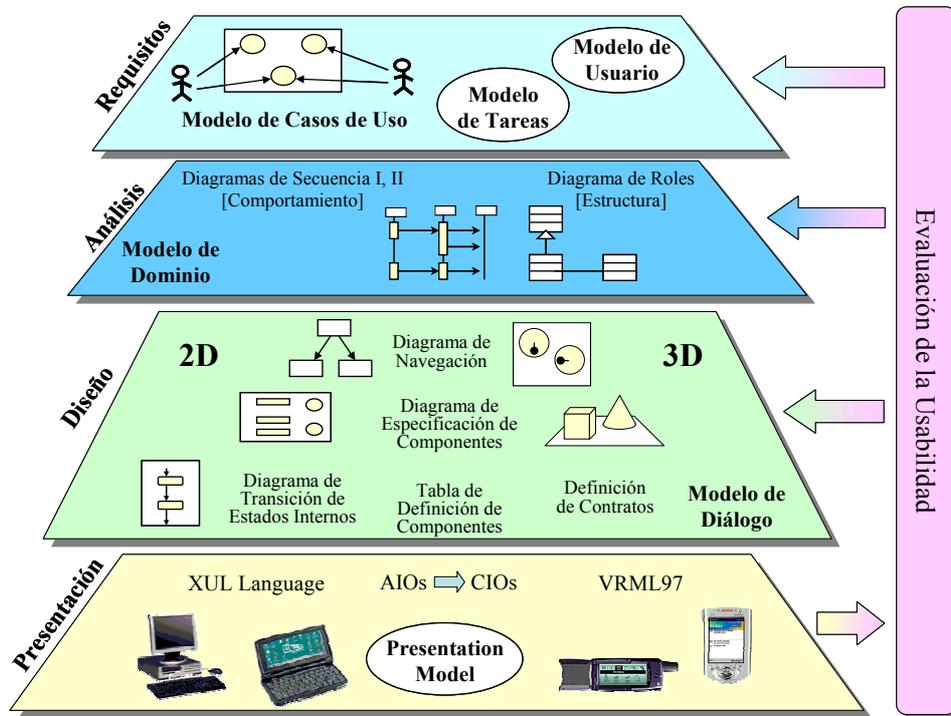


Figura 3.7 Los cuatro niveles de la metodología IDEAS-3D

A nivel de análisis también se mantienen los dos diagramas propuestos en IDEAS para este nivel, esto es, el diagrama de secuencia y el de clases, con los que se da forma al Modelo de Dominio. Tanto este modelo como los del anterior nivel son iguales para todas las posibles versiones de interfaz de usuario de una misma aplicación, pues se asume que la funcionalidad o los objetos de la aplicación no dependen de que la interfaz de usuario final sea 2D o 3D. Ninguno de esos modelos recoge por ello el aspecto gráfico de la interfaz de usuario, cuestión que comienza a ser tratada entonces a partir del siguiente nivel.

Así, en el nivel de diseño se describe la estructura sintáctica del diálogo entre el usuario y la máquina, por medio de diferentes modelos con los que se desarrolla el Modelo de Diálogo. Y es a este nivel donde ya se distingue entre una interfaz 2D y otra 3D, con representaciones diferentes del diagrama de especificación de componentes –aunque se mantiene la distinción entre tres tipos de AIO’s, estos son, controles, presentadores y componentes- y del diagrama de navegación. Este último, en el caso de una interfaz 3D, es representado entonces como un “Mapa” que relaciona tareas con lugares en el espacio 3D. Este mapa representa el entorno 3D de forma esquemática, como una proyección ortográfica de planta o una imagen tomada a vista de pájaro. La elección de un mapa bidimensional

para este cometido se fundamentó en el hecho de que muchos entornos 3D restringían el movimiento del usuario a navegar sobre el plano horizontal, la mayoría simulando así un entorno real donde la gravedad mantiene al usuario pegado al suelo, haciendo de la navegación en el espacio 3D una tarea mucho más sencilla. En este diagrama, un “Lugar” representa una región del espacio donde se lleva a cabo una tarea. Cada uno de estos lugares se hace corresponder con un diálogo o conjunto de diálogos del Modelo de Diálogo. Un lugar también posee una “Zona de Influencia”, que representa la región del espacio donde el usuario es capaz de manipular los elementos interactivos del diálogo asociado a ese lugar, y también un “Punto de vista preferido”, que representa la posición óptima del observador para interactuar con ese diálogo. Además, el Modelo de Diálogo de la metodología IDEAS representa una jerarquía de tareas y subtareas que puede ser también reflejada en el mapa anidando unos lugares en otros.

Finalmente, a nivel de implementación, la información recopilada en los niveles superiores puede servir ahora para generar una clásica aplicación WIMP, pero también puede servir para obtener una interfaz de usuario 3D. En este último caso, los objetos de interacción abstractos (AIO's) especificados en el Modelo de Diálogo deben ser traducidos a widgets de interacción concretos (CIO's) que dependen de la plataforma final. En particular, el lenguaje soportado por la herramienta IDEAS-CASE para la presentación de la interfaz de usuario 2D es XUL, a lo que se añadió el lenguaje VRML97 y Java/Javascript para dar soporte a la generación de interfaces de usuario 3D. Así, un componente con controles puede ser traducido en un panel con botones si es una interfaz de escritorio, un marco con hiperenlaces si es una página Web, o un objeto con botones 3D en una interfaz de usuario 3D. Tanto en uno como en otro caso la presentación final puede ser refinada por el desarrollador, por ejemplo recolocando controles o personalizando los colores, según las guías y directrices de estética y usabilidad que se sigan.

En las publicaciones antes mencionadas se ilustra esta metodología con la creación de la interfaz de usuario de una “Zona Multimedia”, compuesta por un proyector de diapositivas y un reproductor de música, caso de estudio inspirado en una parte del escritorio Win3D [ClockWise, URL]. En la interfaz final 3D, los diálogos de esta aplicación se muestran al usuario como paneles interactivos levantados en posición vertical sobre el suelo del entorno 3D, a los que el usuario se desplaza con sólo hacer un clic en la zona de influencia, a imitación también del propio escritorio Win3D.

Con todo, la propuesta aquí descrita perseguía llenar el vacío que existía entre el desarrollo de interfaces de usuario 2D y las interfaces de Realidad Virtual, dando soporte al desarrollo de interfaces de usuario similares a las proporcionadas por los escritorios 3D, que en el continuo que se introdujo en el apartado 2.4.2 se situaban precisamente a medio camino entre uno y otro tipo de interfaz. En esas

mismas publicaciones ya se apuntaba el interés por ampliar aún más la metodología, avanzando así en el continuo para abarcar también el desarrollo de aplicaciones de Realidad Virtual y de Realidad Aumentada. De hecho, las modificaciones introducidas en esta propuesta en la metodología IDEAS no son muy diferentes a las introducidas por Granollers *et al.* al aplicar la metodología MPIUA al desarrollo de una aplicación de Realidad Aumentada –véase apartado 3.7.4-. Los esfuerzos por aprender de otros métodos y llevar la metodología IDEAS más allá de ese punto en el continuo desembocaron, sin embargo, en la definición un nuevo modelo de procesos, que es el que representa esta Tesis.

### 3.10 Resumen

A lo largo de este capítulo se han descrito y analizado un total de veintiséis metodologías. Estas han sido clasificadas en siete categorías distintas, con el siguiente reparto: una cadena de producción de cine de animación 3D, siete propuestas para crear mundos virtuales para PC, dos métodos con un desarrollo más participativo, cuatro que hacen uso del análisis de tareas, tres más que se basan en la ingeniería del software, cinco métodos para el desarrollo más allá del PC, y cuatro metodologías de desarrollo de interfaces convencionales. El estudio se ha completado con la descripción de una última metodología, IDEAS-3D, como primera propuesta del autor de esta Tesis.

Podría añadirse, incluso, una metodología más a la anterior lista, y es que al principio del propio capítulo ya se esbozó un primer modelo de tres procesos que se suceden en cascada –requisitos, diseño e implementación-, según se desprendía de las enseñanzas de Shneiderman. Dejando a un lado esas meras directrices, la mayoría de las propuestas recogidas en este capítulo son métodos para la creación de mundos virtuales o aplicaciones de Realidad Aumentada. Esto es, más que metodologías para el desarrollo, en general, de interfaces de usuario 3D, los autores centran su atención en aplicaciones concretas de estas. Además, es común acercarse al problema desde el punto de vista de una disciplina particular, lo que resulta en diferentes coberturas según las actividades de cada rol, como se ha tratado de resumir en la Tabla 3.7. Este hecho tiene su reflejo en las herramientas que se citan en cada propuesta, como muestra la Tabla 3.8.

	Diseñador IU	Ingeniero de software	Programador	Artista	Creador de contenidos
<b>Producción de cine de animación 3D</b>					
Cadena de producción de Pixar, según (Reddy, 2005)	.	.	/	X	X
<b>Creación de mundos virtuales para PC</b>					
Desarrollo actual de aplicaciones de RV, según (Kim, 1998)	.	.	/	.	/
Desarrollo actual de EV's, según (Smith, 2001)	.	.	/	.	/
Flujo de trabajo con EON Studio (EON Reality Inc.)	.	.	.	.	/
Metodología de diseño de EV's observada por (Kaur, 1998)	.	.	/	.	X
Redefinición del proceso de desarrollo por (Kaur, 1998)	X	.	.	.	.
Creación de un producto 3D interactivo con VRML (Hay, URL)	X	.	/	/	X
Construcción de mundos X3D (Daly, 2002)	X	.	.	/	X
<b>Desarrollo participativo de mundos virtuales</b>					
Desarrollo centrado en el usuario, grupo VIRART (Neale, 01)	X	.	.	.	.
Metodología centrada en el contenido 3D (Celentano, 2001)	X	.	.	.	X
<b>Análisis de tareas en la creación de mundos virtuales</b>					
Evaluación secuencial de EV's (Gabbard, 1999)	X	/	.	.	.
Creación de ambientes virtuales 3D (Pereira, 00), (García, 01)	X	/	.	.	.
VEDS (Eastgate, 2001), (Wilson, 2002), (D'Cruz, 2003)	X	/	X	/	X
Ingeniería de entornos virtuales con X3D (Polys, 2005)	X	/	/	.	/
<b>Desarrollo basado en ingeniería del software</b>					
Metodología de diseño para EV's según (Fencott, 1999)	X	X	/	X	.
CLEVR (Seo, 2001)	X	X	X	.	.
SENDA (Méndez, 2001), (Sánchez, 2001, 2003, 2005)	X	X	X	.	/
<b>Creación de interfaces de usuario más allá del PC</b>					
Proceso de diseño de EV's, proyecto INQUISITIVE (Smith, 01)	X	/	X	.	.
Diseño dirigido por el contenido o la interacción (Parés, 01)	X	.	X	.	.
VRID (Tanriverdi, 2001)	X	/	.	.	.
MPIUA (Granollers, 2002)	X	/	.	.	.
El proceso de diseño de RV según (Sutcliffe, 2003)	X	X	.	/	.
<b>Desarrollo de interfaces de usuario convencionales</b>					
Metodología de diseño según (Foley, 1996)	X	/	.	.	.
LUCID (Cognetics Corp.)	X	.	.	.	.
OVID (Bardon, 2001)	X	X	.	.	.
IDEAS (Lozano, 2001)	X	X	.	.	.

**Tabla 3.7** Cobertura de las actividades según rol y propuesta: . - pobre, / - parcial, X - amplia

Cad. de producción de Pixar (Reddy, 05)	Storyboards, arte, bocetos, arcilla, sw. de mod. 3D
Des. actual de apl. de RV (Kim, 1998)	Herr. CAD, librerías de programación
Des. actual de EV's (Smith, 2001)	Sw. de modelado 3D, sw. de desarrollo de EV's
Fujo con EON Studio (EON Reality Inc.)	Sw. de autor 3D
Diseño de EV's observado por (Kaur, 98)	Material de ref., sw. de desarrollo de EV's
Redefinición del proceso por (Kaur, 98)	Herr. hipertexto para consejos de diseño
Creación de un producto ... (Hay, URL)	Doc. del proyecto, storyboard, sw. de autor 3D
Construcción de mundos X3D (Daly, 02)	Bocetos, storyboards, sw. de modelado 3D
Des. centrado en el usuario (Neale, 01)	Tormenta de ideas, storyboards, informes
Metodología centrada en el contenido 3D (Celentano, 2001)	Clases de experiencias 3D, sw. de autor 3D personalizado
Eval. secuencial de EV's (Gabbard, 99)	Escenarios de tareas de usuario
Des. EV's (Pereira, 2000), (García, 2001)	Diagramas de flujo de datos, storyboards
VEDS (Eastgate, 2001), (Wilson, 2002), (D'Cruz, 2003)	Entrevistas, focus groups, técnicas etnográficas, storyboards, listas de tareas, material de ref., taxonomías, herr. de guía, sw. de desarrollo de EV's
Ingeniería de EV's con X3D (Polys, 05)	Escenarios, Task-Knowledge Structures (TKS)
Metodología de diseño para EV's según (Fencott, 1999)	Arte, storyboards, planos, diag. de casos de uso y de clases, Perceptual Maps
CLEVR (Seo, 2001)	Diag. de secuencia, de clases, de flujo, leng. CSL, diag. de estados, Visual Object Spec. (VOS), componentes sw., sw. de desarrollo de EV's
SENDA (Méndez, 2001), (Sánchez, 2001, 2003, 2005)	Diag. de casos de uso, conceptos de uso, diag. de clases y de estados, trazas de eventos, cuestionarios, mapas y formularios
Proceso de diseño de EV's, proyecto INQUISITIVE (Smith, 2001)	Storyboards, grafo de escena anotado, redes de flujo, sw. de desarrollo de EV's, toolkits para EV's
Diseño dirigido por el contenido o la interacción (Parés, 2001)	Taxonomías, sw. de modelado 3D, sw. de desarrollo de aplicaciones
VRID (Tanriverdi, 2001)	Desc. en leng. natural, taxonomías, PMIW: diag. de flujo y de estados, sw. de desarrollo de VR
MPIUA (Granollers, 2002)	Técnicas etnográficas, focus groups, Hierarchical Task Analysis (HTA), Puntos de Observación (P.O.), State Transition Notation (STN)
El proceso de diseño de RV según (Sutcliffe, 2003)	HTA, TKS, diag. de casos de uso, escenarios, cuestionarios, mat. de ref., bocetos, taxonomías
Metodología de diseño según (Foley, 1996)	Diag. de estado, storyboards, guías de estilo, toolkits para IU's
LUCID (Cognetics Corp.)	Envision stat., bocetos, personas, escenarios, proto. de pantalla ppal., guía de estilo, sw. de des. de IU's
OVID (Bardon, 2001)	Historias, diag. UML, herr. CASE, bocetos, sw. de desarrollo de IU's
IDEAS (Lozano, 2001)	Diag. de casos de uso, plantillas de tareas, diag. de secuencia, de clases, de diálogo y de espec. de componentes, UIDL, herr. CASE

**Tabla 3.8** Herramientas más representativas citadas en cada propuesta

La primera metodología en ser descrita y analizada, sin embargo, no tenía por objeto el desarrollo de ninguna interfaz de usuario, sino una película de animación 3D. Aún así, de ella destaca el orden seguido para llevar las escenas del papel al ordenador, las herramientas utilizadas, y la revisión continua de rollos en los que se van plasmando los avances en cada escena.

La inclusión de la anterior metodología en este estudio no es casual, ya que las herramientas y técnicas utilizadas en animación son similares a las empleadas en la creación de mundos virtuales, como se comprobó en el segundo grupo de metodologías analizado. La diferencia se encuentra en la interacción, que introduce la preocupación por lograr un adecuado rendimiento y evitar problemas de usabilidad. Lo primero se resuelve con un proceso de optimización que persigue ajustar la complejidad del mundo virtual al hardware gráfico de la plataforma, referencia que algunos autores apuntan debe fijarse en una etapa de planificación o diseño previo a la construcción. Lo segundo se trata de evitar siguiendo en el diseño unas guías como las propuestas en [Kaur, 1998].

Las metodologías del grupo anterior parecían estar pensadas para un único desarrollador, sólo una de ellas planteaba la posibilidad de un equipo de desarrolladores. Más aún, parecen suponer que el desarrollador conoce el contenido del mundo virtual y los gustos del usuario, cuando lo más probable es que no esté familiarizado con ese contenido y que se equivoque si cree que el usuario compartirá con él sus mismos gustos. Con ello se llegó al tercer grupo de metodologías, caracterizadas por promover una mayor participación en el desarrollo tanto del usuario como del experto en el dominio. Así, para lograr un producto que sea de la satisfacción del usuario, se propone que este participe al principio del desarrollo, durante el curso del mismo, y al final. Y para lograr un producto cuyo contenido sea correcto, el experto en el dominio también deberá participar al principio y durante el desarrollo, asistiendo, supervisando o participando como un autor más.

Hasta ese momento los métodos presentados estaban orientados hacia el contenido del mundo virtual, suponiendo que la labor del usuario en dicho mundo no sería otra más que explorarlo y observar las reacciones de los objetos a sus acciones, pero sin desarrollar ninguna tarea más allá de esas. Por el contrario, en otros entornos el usuario sí debe llevar a cabo ciertas tareas, por lo que el desarrollo debe contemplar el análisis de las mismas. Precisamente, el cuarto grupo de propuestas gira en torno a este tema, de él se destaca la diferencia entre las tareas en el mundo real y las tareas en el mundo virtual, y que se traduce no sólo en uno, sino en dos procesos de análisis de tareas en un mismo desarrollo. Se plantea, sin embargo, la duda de cómo se relacionan esas tareas con el contenido, lo que en metodologías como VEDS no parece mayor problema si esas tareas resultan ser nada más que una lista de acciones sobre los objetos, como se lee en [Eastgate, 2001].

El hecho de que la lista de acciones sea reducida, que sólo unos pocos objetos reaccionen a ellas y que sus comportamientos sean limitados, parece dejar la programación, y con ello a la ingeniería del software, en un segundo plano. El quinto grupo de metodologías demostró, por el contrario, que las técnicas y herramientas de la ingeniería del software también pueden aplicarse a los aspectos estructurales del entorno virtual, y por supuesto a los componentes del mismo que haya que programar, sobre todo cuando el desarrollo está basado en librerías software y no en un entorno visual. Aunque ciertos aspectos de un entorno virtual requieren la introducción de nuevos diagramas, como los conceptos de uso en SENDA, y otros simplemente quedan fuera de estas prácticas, como la estética del entorno según [Fencott, 1999].

Aún con sus diferencias, las metodologías de los anteriores grupos guardaban una característica en común, y es que todos ellos suponen que el usuario se enfrentará al ordenador a través de simples técnicas basadas en el teclado y ratón de un PC. El empleo de dispositivos no convencionales requiere técnicas de interacción diferentes que deben ser plasmadas en el diseño, pero también un software que lo soporte. Las metodologías del penúltimo grupo estaban entonces orientadas, unas, al desarrollo de entornos virtuales desde los de escritorio hasta los inmersivos, y otras, al desarrollo de interfaces de Realidad Aumentada. El resultado es una mayor atención a la propia interfaz, en unos casos recordando la teoría, como medio físico y lógico entre el usuario y los objetos de la aplicación, y en otros aportando soluciones más prácticas, como por ejemplo las redes de flujo del proyecto INQUISITIVE.

El último grupo de metodologías también dirigen su atención hacia la interfaz, pero a la interfaz gráfica de usuario 2D. Y no a una interfaz multimedia que bien podría relacionarse con la creación de mundos virtuales, sino hacia interfaces para aplicaciones de negocios. La atención, por tanto, se centra en analizar las tareas del usuario, modelar ese usuario, diseñar los caminos de navegación de unas pantallas o ventanas a otras, y especificar los componentes que se incluyen en cada una de ellas. Al confiar además en componentes bien conocidos, estas metodologías tienen la suerte de contar con herramientas que facilitan el prototipado rápido de la interfaz, como en LUCID, e incluso la compilación de la propia interfaz a partir de los modelos que describen la aplicación y la interacción, como en IDEAS.

Precisamente, con el propósito de aprovechar las ventajas de estas últimas metodologías y llevarlas también al desarrollo de interfaces de usuario 3D, se propuso la metodología IDEAS-3D, con la que se llenó parte del espacio que une las 2D con las 3D según el continuo introducido en el capítulo anterior. Sin embargo, esa experiencia también pone en evidencia la dificultad que representa extender ese mismo método a todo el espacio de diseño de las interfaces de

usuario 3D y que, a tenor de lo estudiado en este capítulo, precisa mayores cambios en el modelo de proceso, los cuales son abordados en el siguiente capítulo con la presentación de una nueva metodología.



## 4

# Hacia la metodología TRES-D

### 4.1 Introducción

La interfaz de usuario es el medio que, a través de un lenguaje de salida, permite al usuario acceder a la información que el ordenador guarda y, a través de un lenguaje de entrada, manipular dicha información. En las interfaces de usuario 3D esos lenguajes de entrada y/o salida están basados en las tres dimensiones del espacio, implementados sobre una tecnología poco convencional y en gran parte aún inmadura. Precisamente, la inmadurez del campo contrasta con la de otras interfaces convencionales como las interfaces WIMP basadas en gráficos 2D, teclado y ratón, y a diferencia de éstas, los pilares del diseño de las interfaces de usuario 3D aún están en construcción.

Como se recogió en el segundo capítulo, esas interfaces de usuario 3D representan un campo experimental en el que se reúne una notable variedad de tecnologías, tanto hardware como software, de diferente evolución y difusión. Los experimentos ayudan a conocer más este campo resumiendo la experiencia en guías para los desarrolladores, como las que recoge [Bowman, 2004], si bien otros autores también han inferido guías a partir de modelos teóricos, como [Kaur, 1998]. Aún con esas guías, no existe un conjunto de controles y técnicas de interacción que pueda llamarse estándar, y la aparición de nuevos dispositivos obliga a imaginar nuevas formas de interacción. Por ello, la ayuda que prestan las herramientas software suele orientarse hacia el diseño en general, supeditada sin embargo a la disponibilidad de hardware compatible.

Esta falta de experiencia y ayuda la debe suplir el desarrollador con su ingenio e intuición, haciendo del diseño lo que es, una actividad no sólo técnica sino creativa. Como campo creativo, el diseño es un proceso impredecible, pero es un proceso iterativo para el que también puede haber una disciplina y diferentes técnicas, métodos y medidas. Precisamente, una de las lecciones fundamentales aprendidas en la ingeniería del software es que las mejoras en los procesos de diseño y de calidad requieren prácticas sistemáticas que involucran métodos bien fundados [Sommerville, 1999].

Así se llegó entonces al tercer capítulo, en el que se presentó un conjunto de metodologías y se analizó la utilidad de cada una de ellas como guía en el desarrollo de interfaces de usuario 3D. Algunas de ellas no fueron concebidas

para tal fin, como es el caso de la producción de una película de animación 3D – en donde no hay interacción- o el desarrollo de interfaces gráficas de usuario 2D –apoyado sobre un conjunto estándar de widgets 2D-, y los cambios necesarios para trasladarlas a este otro campo no son menores, pero de ellas también hay lecciones que aprender. Las restantes metodologías, la mayoría, se orientan hacia una aplicación concreta de las interfaces de usuario 3D, como son los entornos virtuales y la Realidad Aumentada, cada una con una estrategia y/o técnicas diferentes a las demás pero que, en cualquier caso, ninguna llegaba a ser lo suficientemente completa y general como para guiar el desarrollo de cualquier interfaz de usuario 3D.

Ese reto se asume entonces a partir de este cuarto capítulo, presentándose la metodología TRES-D, nombre con el que se hace una clara referencia a las tres dimensiones del espacio, pero en donde la “D” se asocia también a “Desarrollo”. Dicho en inglés, “ThRee dimEnsional uSer interface Development”. La filosofía que guía este nuevo enfoque persigue una orientación por igual hacia las tareas y hacia el contenido. La primera versión de la metodología TRES-D se recoge en [Molina, 2005b], junto al modelo de interacción en [Molina, 2006b], y junto a diferentes casos de estudio en [Molina, 2006d].

## 4.2 Propósito y alcance

La metodología que se ha llamado TRES-D y que es el principal resultado de esta Tesis, persigue varios propósitos. Estos son:

- En primer lugar, promover el uso de un enfoque estructurado en el desarrollo de las interfaces de usuario 3D, dando al proceso la forma de un conjunto ordenado de pasos, identificando las actividades a realizar así como los roles implicados en las mismas.
- En segundo lugar, ofrecer un modelo de las interfaces de usuario 3D y sus elementos que, asociado a la metodología, sea fácil de asimilar, ayude a entender los diseños previos y estimule nuevas ideas, apoyado en un lenguaje que, además, facilite la comunicación.
- Por último, el tercer propósito es proporcionar un marco de trabajo que promueva el uso de buenas prácticas ya probadas, principalmente de los campos de la ingeniería del software y la interacción persona-ordenador, en el que encajar herramientas de guía y herramientas de desarrollo ya existentes, y evolucionar con el campo de las interfaces de usuario 3D añadiendo paulatinamente otras nuevas conforme se van asentando los pilares de este campo.

En cuanto al objeto de esta metodología, este es el desarrollo de interfaces de usuario 3D en general, en cualquiera de las manifestaciones de estas en las aplicaciones que se enumeraron en el segundo capítulo –véase apartado 2.3-, tanto aquellas más marcadas por el contenido como las que ponen el énfasis en las tareas. Ambos aspectos se contemplan por igual en la metodología TRES-D, facilitando la adaptación de la misma a cada desarrollo concreto, esto es, será el desarrollador quien otorgue mayor importancia a unas u otras actividades según la aplicación, pero en cualquier caso no será limitado por la metodología.

### **4.3 A partir de aquí...**

Como respuesta a los problemas observados en la creación de las interfaces de usuario 3D y que han sido destacados en los anteriores capítulos, en esta Tesis se propone el desarrollo de una nueva metodología. Aquí se le ha dado un nombre a la misma, TRES-D, y junto a este se ha fijado el alcance de dicha metodología, listando entre los objetivos no sólo el dar un conjunto ordenado de pasos para el desarrollo de estas interfaces, sino también proporcionar unos modelos que faciliten la comprensión de dichas interfaces, así como un marco de trabajo en el que encajar prácticas y herramientas de diseño y construcción.

A partir de aquí se desarrollará entonces esta metodología TRES-D, cuya principal seña de identidad es la de conjugar dos corrientes de desarrollo de la interfaz de usuario, la de las tareas y la del contenido, en un mismo proceso estructurado. Sin embargo, los detalles de ese proceso no se darán aún. Antes serán presentados tres modelos, uno de objetos, otro de elementos de la interacción y un último del espacio. Con estos modelos o, mejor dicho, “meta-modelos”, se persigue ese lenguaje que facilite la comunicación entre los desarrolladores de estas interfaces, utilizando un vocabulario que les es familiar pues está basado en la literatura existente en el campo, pero con los necesarios cambios en términos y significados para resolver las ambigüedades y contradicciones encontradas en ella, y estableciendo un conjunto de relaciones que ayuden a entender mejor los conceptos y su uso dentro de la propia metodología.



## 5

# Meta-modelos para una nueva metodología

## 5.1 Introducción

En el segundo capítulo se puso de relieve la dificultad para definir qué es una interfaz de usuario 3D y cuáles son sus elementos por las diferencias existentes entre los distintos autores. Ya entonces se advirtió que sería necesario poner orden a ese babel de términos, significados, clasificaciones y modelos. Ahora, para cumplir con uno de los propósitos dados en el capítulo anterior y ofrecer no sólo un marco metodológico sino también un lenguaje con el que los desarrolladores de este tipo de interfaces puedan comunicarse con claridad, esa necesidad se hace más realidad si cabe. Así, se tomará la definición de interfaz de usuario 3D dada ya en aquel segundo capítulo, pero se propondrán tres nuevos meta-modelos, uno de objetos, otro de elementos de la interacción, y un último sobre el espacio, los cuales se exponen y explican a continuación, como paso previo a la presentación de la metodología.

## 5.2 Meta-modelo de objetos

En el capítulo dos se recogieron diferentes clasificaciones de los objetos en una interfaz de usuario 3D. Más concretamente en el apartado 2.5, al abordar los elementos que integran dichas interfaces. Entonces se mostró cómo diferentes autores clasifican los objetos siguiendo diferentes criterios, distinguiendo por ejemplo entre objetos reales y virtuales, con y sin comportamiento asociado, y con un comportamiento real o mágico, por citar algunos.

En realidad, muchas son las clasificaciones que pueden encontrarse en la bibliografía y muchas más las que podrían proponerse, pero desde el punto de vista del desarrollador sólo serán interesantes aquellas que faciliten la labor del mismo, por ejemplo ayudando en las decisiones de diseño o implementación. Las herramientas guía que propone Eastgate son un ejemplo de ello [Eastgate, 2001].

En el caso particular de las interfaces de usuario 3D, es importante clasificar los objetos según su comportamiento e interacción, pero las clases resultantes varían según el autor consultado. Este será entonces el primer tema a resolver en este apartado, pero no el único, ya que relacionado con el mismo se encuentra el conjunto de elementos de un objeto. Y en este segundo tema también se ha visto una falta de consenso en la bibliografía, pues diferentes autores basan sus trabajos en modelos diferentes, como por ejemplo el modelo SBF en [McIntosh, 2000] o la separación de vistas en [Kim, 1998], de nuevo por citar algunos. Así, en segundo lugar, se propondrá también un modelo de objetos para la metodología que aquí se desarrolla.

### **5.2.1 Clasificación de objetos según comportamiento e interacción**

La taxonomía que aquí se propone no nace con la pretensión de ser la única a utilizar por el desarrollador. Otras más sencillas siguen siendo útiles, como la distinción entre objetos reales y virtuales antes comentada, y que se introdujo en [Tanriverdi, 2001]. Tampoco pretende desplazar a otras más elaboradas e igualmente útiles para el desarrollador, como las clasificaciones de objetos para la recogida de material de referencia y para la reducción del número de polígonos, ambas propuestas en [Eastgate, 2001], o el conjunto de categorías del Modelo de Oportunidades de Percepción descrito en [Fencott, 2001]. Ninguna de ellas ha sido puesta en duda en la bibliografía consultada, ni ningún otro autor ha presentado clasificación alternativa a las mismas.

Sin embargo, sí es frecuente clasificar los objetos según su comportamiento e interacción, y es en este empeño donde sí se encuentran diferencias entre autores. Así, como se recogió en el apartado 2.5.3, el propio Eastgate distingue entre cuatro clases de objetos según su comportamiento –interactivo, autónomo, conectado, sin comportamiento-, tantas como Sutcliffe en el análisis del dominio –estructuras principales, objetos pasivos, objetos activos y agentes-. En este caso, las diferencias se encuentran principalmente en los nombres usados, pues es posible hacer corresponder interactivo con activo, autónomo con agente, y sin comportamiento con pasivo, quedando tan sólo las clases conectado y estructuras principales sin correspondencia. El problema se complica al contar también las subclases –hasta ocho variantes de las clases expuestas- que propone Eastgate en su clasificación, y la distinta clasificación a la que recurre Sutcliffe en el diseño de agentes interactivos, distinguiendo entre cuatro clases –automatizados, interactivos, conversacionales y el usuario o presencia-. A todo ello hay que añadir la jerarquía propuesta en la guía de diseño RealPlaces, con ocho nodos –sumando categorías y sub-categorías, véase Tabla 2.7-, cinco si se atiende únicamente a los nodos hoja –ambientales (sin comportamiento),

decorativos, objetos de las tareas, herramientas basadas en apuntador y dispositivos-.

Con el propósito de unificar los diferentes criterios, se propone entonces la siguiente clasificación de objetos según comportamiento e interacción, que no sólo no es una simple unión de las clasificación anteriores sino que pretende además huir de complejas jerarquías, limitándose a las seis clases que se recogen en la Figura 5.1 .

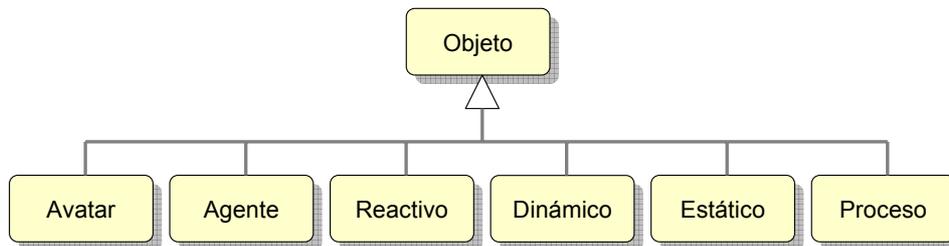


Figura 5.1 Clasificación propuesta de objetos

A continuación, se describen una a una todas esas clases:

- **Procesos:** Son algoritmos y otros procesos del sistema que no tienen una representación en la interfaz de usuario, aunque sí pueden guardar relación con la misma.
- **Objetos estáticos:** Son objetos que sí se representan en la interfaz, aunque carecen de comportamiento asociado. Pueden jugar un papel meramente decorativo, pero también otros más importantes, como los que juegan los objetos que dan forma al espacio o guían al usuario en su navegación por el mismo.
- **Objetos dinámicos:** Son objetos que se representan en la interfaz y sí tienen un comportamiento asociado, aunque éste se reduce al cambio de alguna o algunas de sus propiedades y condiciones con el continuo paso del tiempo. Un reloj es el ejemplo más representativo de esta clase de objetos, el movimiento de sus manecillas depende sólo de ese paso del tiempo. Muchos de los objetos animados que pueblan los mundos virtuales también se clasificarían de esta forma.
- **Objetos reactivos:** Estos objetos también se representan en la interfaz y tienen un comportamiento asociado, si bien dicho comportamiento se manifiesta sólo como respuesta a la acción del usuario u otro agente sobre el objeto. Los usuales controles de la interfaz son ejemplo de ello, tanto los conocidos widgets como los obgets, término este último que se

definirá en breve. Un objeto reactivo puede también provocar la respuesta de otro objeto reactivo, siendo esa acción parte de su comportamiento o consecuencia de la ejecución de dicho comportamiento, resultando entonces en un comportamiento encadenado, como ocurre con una bombilla y su interruptor. La acción del usuario no tiene que ser explícita, puede también ser implícita, como una puerta que se abre al paso del avatar del usuario.

- **Agentes:** También se representan en la interfaz, aunque no manifiestan su comportamiento sólo en respuesta a una acción externa –con la excepción del paso del tiempo–, sino que toman la iniciativa y actúan sobre otros objetos, posiblemente exhibiendo ciertos rasgos de inteligencia humana. Esta última característica suele ligarse al empleo de mecanismos de inteligencia artificial. Los agentes conversacionales son un ejemplo de ello, usualmente representados en la interfaz con formas humanas, aunque no todas las figuras antropomórficas que pueblan los mundos virtuales merecen el calificativo de agente conversacional, muchas de ellas son simples objetos reactivos.
- **Avatar del usuario:** Es la representación del usuario en la interfaz. Puede ser algo tan simple como la flecha del cursor en aplicaciones de escritorio, una mano y/o cabeza virtual, o incluso algo más completo como una representación de cuerpo entero. A través de este avatar, el usuario transmite sus acciones sobre los objetos de la interfaz. Como también se verá en breve dentro de este mismo capítulo, el usuario mueve este avatar como si de una marioneta se tratara, siendo los dispositivos de entrada y su correspondencia con el avatar la madera y los hilos que la sujetan.

### 5.2.2 Elementos de un objeto

Clasificar los objetos en base a su comportamiento e interacción puede ayudar al desarrollador a determinar, en un primer momento, las necesidades de diseño que tendrá cada objeto. Pero la mera clasificación no es suficiente, pues el desarrollador necesita también conocer qué debe especificar y cómo. Además del comportamiento e interacción, también tendrá que contar otras propiedades del objeto. Por ejemplo, en [Eastgate, 2001] se cita la forma, el tamaño y la apariencia.

Con este propósito, diferentes autores recurren a modelos que separan las diferentes vistas de los objetos. Sin embargo, el modelo utilizado es diferente para cada autor, como se ha comprobado en los capítulos precedentes. Así, en

[Conner, 1992] se distingue entre geometría y comportamiento, el trabajo descrito en [McIntosh, 2000] se basa en el modelo SBF –estructura, comportamiento y función-, para la metodología CLEVR –véase apartado 3.6.2- se recurre a la separación en forma, comportamiento y función, en el proyecto INQUISITIVE –véase apartado 3.7.1- se diferencia entre apariencia y geometría, comportamiento e interacción, y los autores de VRID –véase apartado 3.7.3- hablan de gráficos, comportamiento e interacción. En todos ellos, la palabra que se repite es comportamiento, y bien puede suponerse que su significado es el mismo en todos los modelos. Atendiendo a las semejanzas entre las notaciones propuestas por los diferentes autores, tampoco parece haber diferencia entre función e interacción. Lo que resta es un variado surtido de términos –estructura, geometría, apariencia, forma y gráficos- que sin duda hacen referencia a la representación visual del objeto, aunque cada uno aporta un matiz distinto, fácil de entender si se piensa en un lenguaje para la descripción de mundos virtuales como es VRML, en donde cada objeto se describe como una jerarquía de formas, cada una con su geometría y apariencia particular. Pero del mismo modo que si faltase un tipo de nodo VRML no podría describirse el objeto, prescindir de alguno de esos términos deja incompleto el modelo, y así se observa en las propuestas de esos autores.

El problema no sólo es una falta de consenso. Los modelos propuestos por esos autores están además fuertemente basados en objetos cuya representación es puramente gráfica, un reflejo del dominio de la imagen visual en las interfaces de usuario 3D, principalmente en mundos virtuales, en detrimento de otros canales sensoriales. Sin embargo, el modelo de objetos no debería limitarse únicamente a esa faceta. De hecho, más que de un único modelo debería hablarse de un conjunto de modelos, como los modelos geométrico, cinemático y físico que se explican en [Burdea, 1996]. Y es que el modelado y simulación de un objeto es más fácil de abordar si se divide el problema, del mismo modo que en gráficos por ordenador la luz que reflejan los objetos se divide en cinco componentes que simulan cinco efectos visuales diferentes, pero cuyas contribuciones se suman para dar el resultado final –véase, por ejemplo, [Foley, 1996]-.

Por todo ello, en esta Tesis se propone como modelo la composición de función, comportamiento y un conjunto de sub-modelos, entre los que se encuentra el modelo gráfico, como se describe a continuación:

- **Función:** Comprende las operaciones que puedan realizarse sobre el objeto, procedimientos que pueda ejecutar, y cualquier otro servicio que ofrezca a través de una interfaz de comunicación con el resto de procesos, objetos y agentes del sistema. La función de un objeto determina entonces las posibilidades de interacción con dicho objeto, y debe entenderse entonces como sinónimo de esta.

- **Comportamiento:** Es la manera de comportarse del objeto, la cual depende en todo momento del estado del objeto y, posiblemente, de otras variables del entorno –p. ej. el tiempo-. La distinción que realizan los autores de VRID entre comportamientos físicos y mágicos resulta muy acertada, con el fin de distinguir entre aquellos comportamientos que por naturales requieren poca explicación y aquellos otros que por poco habituales precisan una descripción más detallada.
  
- **Sub-modelo gráfico:** Engloba las descripciones del objeto relacionadas con su representación gráfica en la interfaz. Debe tenerse en cuenta que no todos los objetos del sistema se manifiestan en la interfaz, y de hacerlo puede que su imagen no sea visual, y en esos casos este sub-modelo no será necesario. También debe tenerse en cuenta que algunos objetos precisan este sub-modelo aun cuando estos sean invisibles, una propiedad innatural según Eastgate pero posible con gráficos por ordenador –p. ej. lindes que impiden acercarse al decorado de fondo-. En cualquier caso, este sub-modelo consta de las siguientes descripciones:
  - **Estructura:** Representa la distribución y orden de las diferentes formas que componen el objeto. La apariencia visual suele revelar unas relaciones espaciales entre esas formas, ordenándolas de forma jerárquica siguiendo como criterio las posiciones relativas entre ellas. Sin embargo, como bien se apunta en [Huda, URL1], el grafo de la escena puede describir una jerarquía distinta a la que se muestra a la vista. Esta diferencia es importante, y se tendrá presente al describir la metodología.
  - **Geometría:** Define las medidas del objeto, cuya su figura se dibuja a partir de la geometría de las formas que lo componen y su distribución según la estructura del propio objeto.
  - **Apariencia:** Describe el aspecto externo del objeto, o las características del propio objeto que marcan dicho aspecto, pues la imagen del objeto que se represente en la interfaz puede depender también de otras variables del entorno –p. ej. fuentes de luz y otros objetos a su alrededor-.
  - **Percepción:** La diferencia entre los anteriores apartados y este último es la misma que se explica en [Fencott, 2001] entre simulación (denotación) y comunicación simbólica (connotación). Se trata entonces de describir el objeto no cómo es, sino cómo lo percibirá el usuario, y para ello bien puede servir el propio Modelo de Oportunidades de Percepción que proponen esos autores –véase apartado 2.5.2-.

- **Otros sub-modelos:** El modelo que aquí se propone no es cerrado sino ampliable a través de nuevos sub-modelos que describan otras facetas de los objetos no recogidas anteriormente, como la imagen sonora o háptica de un objeto. Por ejemplo, propiedades físicas como el tacto y peso del objeto podrían describirse en un nuevo sub-modelo háptico.

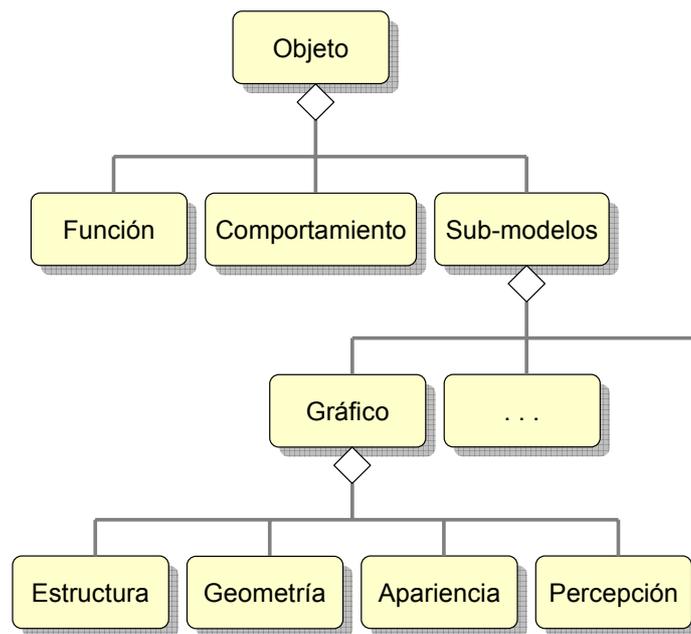


Figura 5.2 Modelo propuesto para los objetos

La Figura 5.2 resume el modelo. Con lo aquí expuesto el desarrollador podría hacerse una mejor idea acerca de qué debe especificar de cada objeto, pero sigue sin detallarse cómo, lo que sí abordan los autores de las propuestas citadas en este apartado. Esta cuestión, sin embargo, se dejará a un lado hasta haber presentado la metodología.

### 5.3 Meta-modelo de elementos de la interacción

Como se expuso en el apartado 2.6, no todos los autores utilizan los mismos términos para referirse a los diferentes elementos de la interacción y, a veces, cuando sí coinciden en los términos, se refieren en realidad a elementos diferentes. Como se ha resaltado antes, resulta de gran importancia que los diseñadores de interfaces que colaboran en un mismo desarrollo utilicen un

vocabulario común, y por ello es preciso decidir qué términos son los más apropiados y definir sin ambigüedad cuál es el significado de cada uno de ellos.

Por ello, en los siguientes párrafos se explica un modelo de elementos de la interacción que persigue poner de acuerdo las diferentes versiones. La estructura de este modelo parte de las explicaciones de [Foley, 1996], y es ampliada para basarse en los siguientes conceptos: diálogo, tareas, operaciones, tareas de interacción, técnicas de interacción, acciones, controles y dispositivos físicos. La explicación del modelo y la definición de estos conceptos también puede encontrarse en [Molina, 2006b].

### 5.3.1 El diálogo

El **diálogo** es la comunicación entre el usuario y la máquina, y puede ser visto a diferentes niveles de abstracción y medidas de extensión. Puede estudiarse desde el nivel conceptual de las tareas del usuario, o desde el nivel físico de las acciones de dispositivo. Puede extenderse a lo largo de todo un escenario o restringirse a una subtarea concreta.

El diálogo, como herramienta del diseñador, debe servir para poder decidir qué feedback debe proporcionarse al usuario, entendiéndose en los términos que explica [Barrilleaux, 2001], esto es, la guía que proporciona el diseñador al usuario sin estar presente. Y debe decidirse a qué nivel, y esos niveles son los que aquí se explican.

El diálogo también debe servir para verificar que la secuencia de tareas puede realizarse con las técnicas y controles dados, que no existan incompatibilidades. De existir, debería plantearse la creación de modos (control de la interacción) entre los cuales el usuario cambie, por ejemplo, para usar un mismo dispositivo en diferentes técnicas de interacción.

### 5.3.2 Las tareas y las operaciones

Tomando la definición de tarea dada en [Granollers, 2002], en este modelo se identifica una **tarea de interacción** con la actividad necesaria para conseguir un objetivo, siendo el objetivo un resultado o logro que el usuario quiere alcanzar dentro de la aplicación.

Además, atendiendo a las explicaciones de Foley *et al.*, se entiende que una tarea de interacción requiere la introducción de una o más unidades de información, distinguiéndose entre tareas básicas –transmiten una unidad de un tipo- y

compuestas –transmiten varias unidades de tipos iguales o diferentes-. Sin embargo, se propone cambiar los acrónimos BIT y CIT, introducidos por esos autores –recuérdese el apartado 2.6.2-, por BITa y CITa, para hacer referencia a las tareas de interacción básicas y compuestas, respectivamente, ya que las iniciales IT pueden asociarse también a las técnicas de interacción y crear confusión.

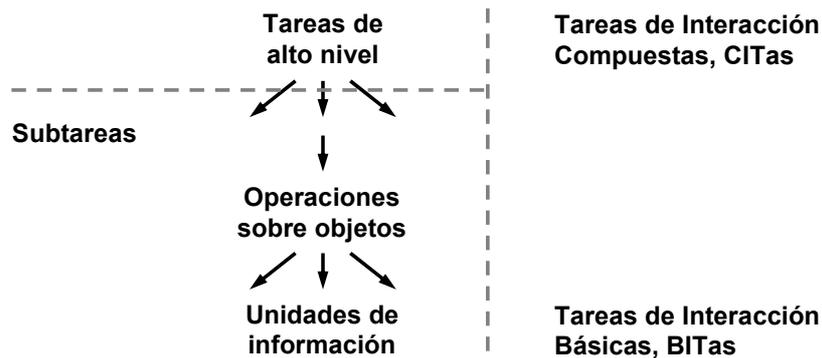


Figura 5.3 Descomposición de las tareas de interacción

De este modo, las tareas de alto nivel se descomponen en subtareas más sencillas, como explican otros autores, pero entendiéndose a partir de lo dicho en el párrafo anterior que las primeras corresponden a tareas compuestas, mientras que aquellas subtareas que no pueden dividirse en otras son las llamadas tareas básicas. Con independencia de ello, en esa descomposición se destaca aquí el nivel de **operaciones** individuales sobre los objetos –véase Figura 5.3-, dado que las operaciones que puedan realizarse sobre un objeto definirán su funcionalidad, y de esta forma se relaciona el modelo de objetos y el de elementos de la interacción.

“Intrínsecamente gráficos”	“No intrínsecamente gráficos”
<ul style="list-style-type: none"> <li>• Manipulación directa                             <ul style="list-style-type: none"> <li>○ Diálogo basado en formularios</li> </ul> </li> <li>• What You See Is What You Get (WYSIWYG)</li> <li>• Interfaces de usuario basadas en iconos</li> </ul>	<ul style="list-style-type: none"> <li>• Menús</li> <li>• Lenguaje de comandos</li> <li>• Diálogo de pregunta-respuesta</li> <li>• Diálogo en lenguaje natural                             <ul style="list-style-type: none"> <li>○ Interacción deíctica (p. ej. “pon eso allí”)</li> </ul> </li> </ul>

Tabla 5.1 Diferentes estilos de interacción según [Foley, 1996]

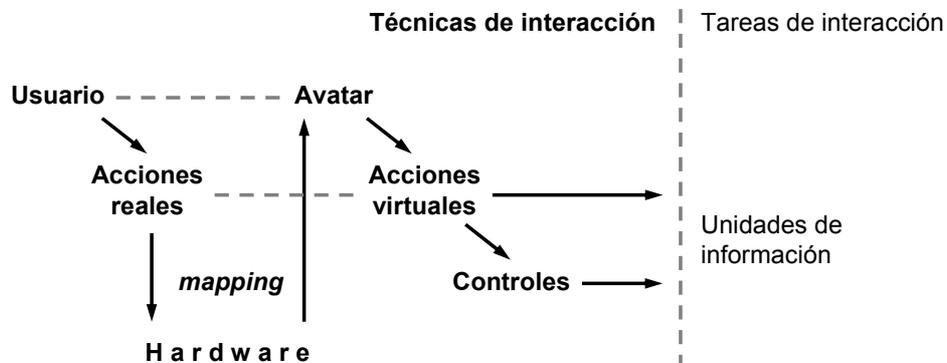
Para cada operación, de nuevo según Foley *et al.*, se deben definir las pre- y post-condiciones, los parámetros, resultados y situaciones de error. La ejecución de una operación dependerá del estilo concreto de interacción (véase Tabla 5.1 sobre estilos de interacción), pudiendo requerir por ejemplo la marcación explícita de inicio y fin, así como cada uno de los parámetros. Como cada

unidad de información representa una tarea básica, una operación que requiera varias de ellas deberá verse, de este modo, como una tarea compuesta. Si sólo precisa la introducción de una unidad, la operación será una tarea básica, esto es, una operación atómica que no podrá descomponerse en otras tareas más sencillas.

### 5.3.3 Técnicas de interacción, acciones y controles

La principal contribución de este modelo es que define que las tareas de interacción se implementan con técnicas de interacción y controles, distinguiéndose entre ambos conceptos, y añadiendo otros a la ecuación.

Así, una **técnica de interacción** (*Interaction Technique*, ITe, en lugar de IT, o de InT como en [Figuroa, 2002]) especifica una forma de llevar a cabo una tarea de interacción, centrada alrededor de una o varias acciones virtuales, pero cuya descripción también incluye, a un lado, las acciones reales del usuario, los dispositivos físicos que las capturan y el avatar que finalmente ejecuta esas acciones, y a otro lado, los controles –en su caso- que son objeto de esas acciones. De esta forma, el concepto de control se refiere sólo a una parte de la técnica, no al todo. Además, la técnica también describe un comportamiento y la metáfora en la que se basa, qué puede hacerse y qué no, y el feedback asociado.



**Figura 5.4** Técnicas de interacción: acciones virtuales y controles, y su relación con las tareas

Las **acciones virtuales** tienen significado y efecto en el mundo virtual de la aplicación y sus objetos y es allí donde son ejecutadas, no directamente por el usuario, sino por un agente informático en su lugar, siendo este el avatar del usuario, su “yo” virtual. Una técnica de interacción define entonces una o varias acciones virtuales con las que se lleva a cabo la tarea –esto es, con ellas se transmite la información que representa esa tarea-, o bien una o más acciones virtuales y los controles que son objeto de las anteriores y que llevan a cabo la

propia tarea –ahora son estos controles los que, a modo de transductores, transmiten la información que representa dicha tarea-, como se ilustra en la Figura 5.4.

Además, la técnica de interacción describe la forma en que las **acciones reales** del usuario en el mundo físico son trasladadas a la aplicación, convirtiéndolas en acciones virtuales. Para ello, la técnica de interacción incluye también los dispositivos físicos que capturan esas acciones del usuario en el mundo real (caracterizados por el número de grados de libertad, tipo de entrada, etc., de ello se hablará más en el siguiente apartado), la correspondencia de las lecturas de esos dispositivos con el avatar del usuario, y las partes de ese “yo” virtual que son objeto de esa correspondencia y que ejecutan finalmente, como una marioneta del usuario, las acciones virtuales –véase Figura 5.5. En este sentido, esta parte del modelo bien recuerda a las tres partes que identifican Parés y Parés del diseño de interfaces: canales externos (interfaces físicos), canales internos (interfaces lógicos) y comunicación entre ellos (*mapping*) –apartado 3.7.2-.

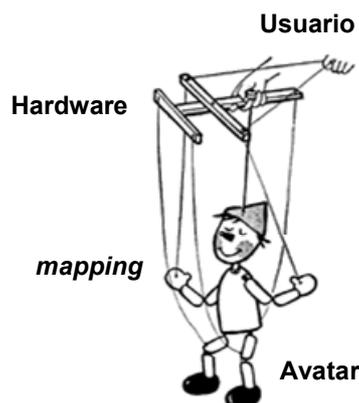


Figura 5.5 El avatar como una marioneta del usuario

Sobre ese avatar del usuario al que aquí se liga la técnica de interacción ya se comentó, al presentar el modelo de objetos –véase apartado 5.2.1-, que podía tomar la forma completa de una persona, o reducirse a una sola mano virtual, o a la simple flecha del cursor. Más aún, a lo dicho entonces cabe añadir que puede no tener siquiera representación gráfica en la aplicación, aunque eso no significa que el “yo” virtual no esté presente de algún modo en la aplicación.

En cuanto a los **controles**, estos son objetos reactivos que reciben las acciones virtuales y generan las unidades de información que representan las tareas de interacción. Un control encapsula comportamiento y geometría, generalmente acompañado de una representación visual, aunque puede no tenerla. Como

recoge la Figura 5.6, los controles pueden, además, combinarse formando un **componente**.

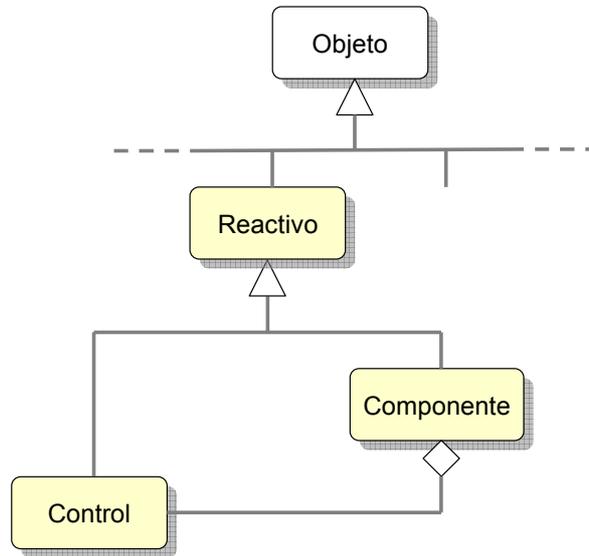


Figura 5.6 Relación entre componente y control

El término control y componente es tomado de la guía RealPlaces [IBM, URL], aunque la definición parte de la idea de widget 3D de [Conner, 1992]. Precisamente, el término control ha sido elegido para no sobrecargar el de widget. Un **widget** es entonces un ejemplo de control pero, al contrario que autores como Conner *et al.*, un widget no será considerado una técnica de interacción, ni viceversa.

Más aún, se propone aquí restringir el término widget a los elementos de las interfaces gráficas de usuario 2D, y el término **widget 3D** a la representación en 3D de esos elementos, ya sea con gráficos ráster o con polígonos. La Figura 5.7 da una muestra de estos últimos, en este caso de la librería VUIToolkit, desarrollada también en el marco de esta Tesis –véase Anexo A-.

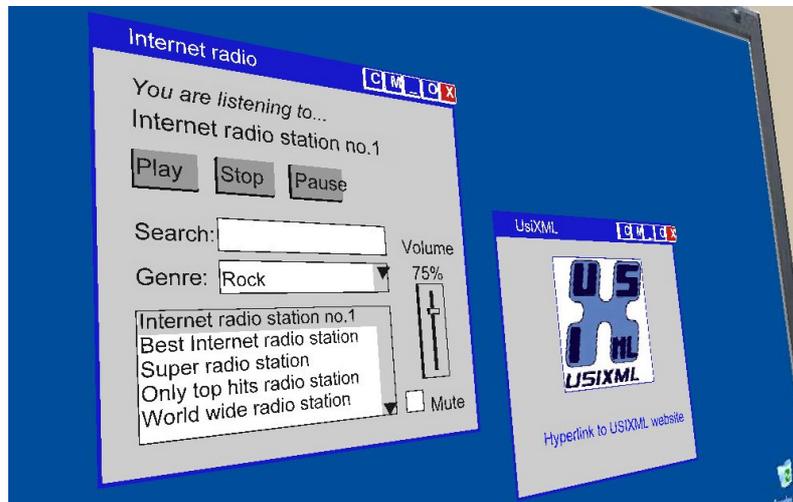


Figura 5.7 Widgets 3D de la librería VUIToolkit

Otros ejemplos de controles son los elementos interactivos de los objetos virtuales. Para designar esos elementos podría acuñarse entonces otro término, por ejemplo “**obget**” (formado a partir de las palabras “object gadget”, a semejanza de widget, que proviene de “window gadget”). Estos obgets pueden ser controles reales del objeto, modelos virtuales de los mismos, abstracciones de ellos, o controles no existentes en el objeto original añadidos en el mundo virtual. Por ejemplo, son obgets el volante o la palanca de cambio de marchas del juguete virtual que se muestra en la Figura 5.8.

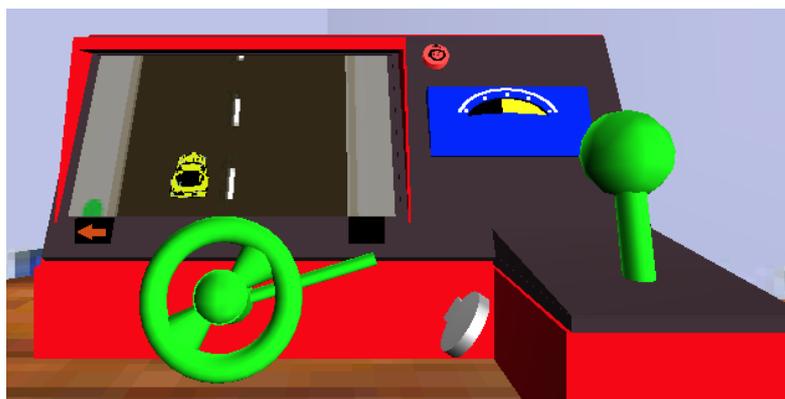


Figura 5.8 Obgets en un juguete virtual (autor: Fernando Royo)

Es muy fácil que existan obgets similares a widgets, pues muchos de los widgets son metáforas de los controles que encontramos en la realidad. Y viceversa, muchos obgets añadidos a los objetos 3D derivan de controles 2D, como por

ejemplo los que permiten redimensionar un objeto (*handles*). A modo de resumen, la Figura 5.9 recoge los tres tipos de controles aquí identificados.

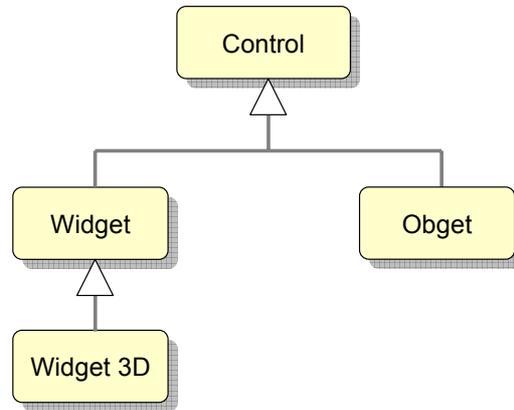


Figura 5.9 Tipos de controles

Volviendo a la definición de una técnica de interacción, esta posee también un comportamiento, basado usualmente en una metáfora –como se comentó en el apartado 2.6.3- en la que se inspira al diseñador, y que sirve de ayuda al usuario cuando construye su propio modelo mental sobre qué puede hacerse (*affordances*) y qué no (*constraints*). Además, para cada parte de la técnica, de forma individual o todas ellas en conjunto, puede definirse también el feedback que el propio desarrollador deja en forma de ayudas para guiar al usuario en el uso de la técnica.

Por último, de la misma forma que una tarea de interacción puede componerse con otras para constituir una tarea compuesta de interacción o CITA, una técnica también, pudiéndose entonces distinguir entre una técnica de interacción básica –con el acrónimo BITE- y una técnica de interacción compuesta –o CITE-. De esta forma, una operación que requiera varias tareas de interacción puede concretarse en una CITE, esto es, en un conjunto de técnicas que, a través de las acciones virtuales que especifican y, en su caso, los controles sobre los que actúan, transmiten la información necesaria para llevar a cabo esa operación.

#### 5.3.4 Dispositivos físicos

Los **dispositivos de entrada** son las herramientas físicas sobre las que se apoya la implementación de las técnicas de interacción. Estas técnicas, sin embargo, no deberían restringirse a un dispositivo concreto. Puede hacerse uso del concepto de **dispositivo de entrada abstracto** que, en principio, sólo debe definir el número de grados de libertad del dispositivo y los tipos de entrada. Los

dispositivos que cumplan esas características deberían ser utilizados sin problemas en la implementación de la técnica de interacción.

Si el número de grados de libertad o el tipo de entrada del dispositivo no coincide, entonces debería ser posible implementar un adaptador software que permita el uso de esa técnica de interacción con ese dispositivo, tal vez añadiendo restricciones, conversores de tipo, también nuevos controles y modos de interacción en la interfaz. La introducción de estos elementos puede apartar al usuario de la tarea principal, haciéndola más compleja, por lo que según el caso puede ser mejor implementar técnicas de interacción apropiadas para las características de ese otro dispositivo.

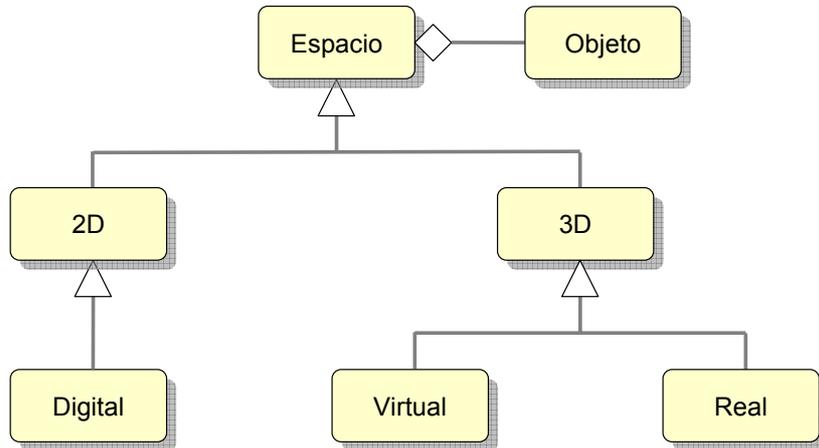
La modalidad de entrada también podría ser una característica que se defina en el dispositivo de entrada abstracto, y de nuevo los adaptadores software deberían permitir el cambio de modalidad para una técnica de interacción dada. Aunque en el caso de la entrada, sólo cabe distinguir dos modalidades: motor y habla –si bien también cabe añadir una tercera, el pensamiento, como una modalidad superior a esas dos usada en las incipientes interfaces cerebro-computador o, en inglés, *brain-computer interfaces*-

Y en cuanto a los **dispositivos de salida**, estos deben ser igualmente tenidos en cuenta en el modelo de elementos de la interacción, escogiendo los dispositivos que más se ajusten a la modalidad deseada para la representación de los objetos y el feedback. De nuevo, se podría hablar de **dispositivos de salida abstractos** que, en principio, definiría la modalidad que debe usarse para comunicar la salida al usuario. Al igual que antes, la implementación de adaptadores software debería hacer posible el cambio de modalidad –trasposición de los sentidos-.

## 5.4 Meta-modelo del espacio

Resulta inevitable introducir también este otro modelo cuando se trata del desarrollo de interfaces de usuario tan fuertemente ligadas al espacio como las tridimensionales. Y no porque los modelos precedentes hayan ignorado este aspecto. De hecho, el modelo de objetos, a través de sub-modelos como el gráfico, permite describir cada objeto en las tres dimensiones del espacio, ya sea este virtual o real. Y de esas descripciones, aquellas de los objetos que participan en las tareas también se tienen en cuenta en el modelo de interacción, así como las acciones del usuario en el espacio real que las técnicas de interacción trasladan a la aplicación. Sin embargo, aunque este último modelo aborda los dispositivos usados en la interacción, no incluye una descripción de los mismos en el espacio, especialmente necesaria en este tipo de interfaces donde se conectan dos mundos diferentes, el de los objetos digitales o virtuales y el del

usuario, superponiendo las dimensiones de uno sobre el otro, creando con ello un continuo espacial por el que el usuario navega y manipula los objetos que lo pueblan.



**Figura 5.10** Los espacios digital, virtual y real en el meta-modelo

Como cabría esperar, este tercer meta-modelo viene marcado por el continuo digital-virtual-real introducido en el capítulo 2, distinguiéndose en primer lugar entre esos tres espacios, como muestra el diagrama de la Figura 5.10. En dicho diagrama también se destaca la relación de agregación de objetos en el espacio, pues son los objetos los que dan forma al propio espacio.

A partir de aquí, los siguientes apartados detallarán el modelo avanzando a través cada uno de esos espacios, abordando en primer lugar el espacio bidimensional de las interfaces gráficas WIMP, sumando una dimensión más para abordar el espacio de los mundos virtuales, y terminando con las tres dimensiones del espacio real que rodea al usuario. El interés, no obstante, no se centrará sólo en la descripción de cada espacio sino, como ya se ha ilustrado en el párrafo anterior, en la forma en que se relacionan unos con otros. Todo ello parte de las ideas plasmadas en varias publicaciones previas, en concreto las que describen IDEAS-3D –véase apartado 3.9- y también [Molina, 2006c].

#### 5.4.1 El espacio 2D digital

La necesidad de contar con un modelo del espacio no es exclusiva del diseño de las interfaces de usuario tridimensionales, pues metodologías como LUCID, OVID o IDEAS –todas ellas descritas en el capítulo 3- también lo introducen a su manera para el desarrollo de interfaces gráficas de usuario 2D. No en vano

estas últimas se muestran sobre dispositivos de presentación usualmente planos y rectangulares, con un área finita dada por la resolución en pixels de la pantalla, y en la que es usual que no puedan dibujarse a la vez todos los elementos gráficos de la interfaz, por su número y tamaño. Ese número depende de las tareas a realizar por medio de la interfaz, y el tamaño suele variar no sólo para ajustarse a diferentes tamaños de pantalla sino también al tipo de dispositivo apuntador -sea este un ratón, un lápiz o el dedo del usuario-. Además de esas consideraciones técnicas, también existen razones estéticas que llevan a variar el tamaño de los widgets y su disposición en la pantalla, así como dividir los diálogos interactivos en varias pantallas. Por ello, no es extraño que las anteriores metodologías recurran a la creación de diagramas o prototipos que capturen la disposición en el espacio de la pantalla de esos elementos y las opciones de navegación a través de las distintas ventanas.

Un ejemplo de ello es el diagrama de especificación de componentes que introduce la metodología IDEAS en su Modelo de Diálogo –véase el apartado 3.8.4-, con el que se especifica de forma gráfica la disposición de los diferentes controles de una ventana en las dos dimensiones del espacio de la misma. Aunque el propósito de este diagrama es el de obtener una especificación abstracta de la interfaz, usando para ello simples círculos y rectángulos que representan los diferentes controles, el propio diagrama ya hace uso del espacio de la misma forma que la interfaz final y, de hecho, el despliegue y agrupación de controles en el mismo sirve de base para la generación de esa interfaz, cuyo aspecto definitivo es ajustado manualmente.

En general, la disposición de los controles sobre la ventana puede formalizarse definiendo un conjunto de relaciones espaciales. Así, en [Trevisan, 2004] se parte de las 13 relaciones temporales de Allen para, tras generalizarlas al espacio 2D y construir con ellas una matriz (**matriz 2D de Allen**), formalizar las posibles relaciones entre dos controles en una interfaz gráfica 2D. Una formalización así bien sirve para cubrir esta parte del modelo del espacio que aquí se propone. No obstante, la anterior matriz también formaliza las posibles relaciones entre dos elementos de una interfaz de Realidad Mixta en la que un objeto digital puede superponerse a otro real. Y es que, una vez más, el desarrollo de una interfaz de usuario 3D no excluye la utilización de elementos 2D sino que, como precisamente se quiso expresar al proponer el continuo digital-virtual-real, pueden encontrarse diferentes combinaciones a lo largo del mismo. Precisamente, entre los diferentes controles bidimensionales que pueden especificarse cabe destacar aquí uno en especial, el puerto de vista, no contemplado en la propuesta de Trevisan *et al.*, pero que, como se verá a continuación, será uno de los nexos de unión entre ambos espacios.

### 5.4.2 El espacio 3D virtual

Cuando se planteó la extensión de la metodología IDEAS al desarrollo de interfaces de usuario 3D se pensó que un punto de vista del mundo virtual podía asimilarse a una ventana en una interfaz 2D, y los controles que quedaban dentro de esa vista podían describirse como los elementos de tal ventana. Al fin y al cabo, esos controles en 3D suelen ser proyectados sobre una superficie plana como la propia ventana, aunque la perspectiva los hace más pequeños con la distancia –lo que, por otra parte, y como se vio en el capítulo 2, es también una solución al problema de espacio en la pantalla-. Y del mismo modo que el usuario de una interfaz 2D navega entre ventanas, también el usuario de un mundo virtual puede saltar de un **punto de vista** a otro.

Sin embargo, más que una vista, era una zona del mundo virtual la que se asimilaba a una ventana, siguiendo así la idea propuesta en la guía de diseño RealPlaces [IBM, URL] de estructurar el espacio virtual en lugares (*places* según la guía) de forma jerárquica, asignando a cada lugar un punto de vista que lo abarca. Más aún, esa jerarquía de zonas venía a ordenar en parcelas el suelo que pisaba el avatar del usuario, pensando en un mundo horizontal, plano, que el usuario recorrería pegado al terreno, despegándose sólo para saltar de uno a otro punto vista, de una a otra zona. En cualquier caso, la división en zonas no era casual sino un reflejo de la jerarquía de tareas que el usuario lleva a cabo a través de la interfaz, relacionándose a través de esas tareas con el resto de modelos de la metodología IDEAS-3D.

Sin embargo, una concepción así del espacio 3D resulta limitada para el desarrollo en general de interfaces de usuario 3D, empezando por los propios mundos virtuales. Por una parte, si bien el concepto de **zona**, **lugar** o **locus** – véase apartado 2.5.1- se estima necesario, no siempre se corresponde con un espacio abierto como se propone en la guía RealPlaces –y tras ella la metodología IDEAS-3D-, donde las construcciones ayudan a diferenciar unos espacios de otros pero sin encerrarlos, sino que esas estructuras pueden levantarse hasta dividir por completo el espacio y, sin embargo, no hallar diferencia entre una estancia y otra.

Por otra parte, no siempre es posible compartimentar el espacio por tareas concretas o asignar un único punto de vista significativo a cada una de las divisiones resultantes, pues ocurre en muchos mundos virtuales que el usuario simplemente explora el espacio, desplazando libremente su avatar y llevando a cabo acciones aquí y allá. Al mismo tiempo, la distinción de regiones en el espacio puede ser igualmente útil para otros usos, por ejemplo para especificar los límites del mundo virtual que el avatar del usuario no debe abandonar, el

alcance de los rayos de una fuente de luz o las ondas de una fuente de sonido, o la pirámide de vista asociada a un punto de observación.

Por último, porque muchos mundos virtuales no son simples planos sobre los que se levantan las diferentes construcciones que observa el usuario al andar, sino que la topografía de cada mundo puede ser de muy distinta naturaleza, y el avatar del usuario puede incluso volar u orbitar. En muchos casos, además, no se puede hablar de un único mundo virtual sino de varios conectados entre sí, por lo que no sólo es preciso modelar cada uno de ellos sino también la forma en que se comunican.

En definitiva, es necesario librarse de las anteriores restricciones para hacer que esta parte del modelo describa un espacio 3D más general, en donde las zonas circulares de IDEAS-3D o las esferas de RealPlaces asociadas a tareas dejan paso a una ordenación más libre que sirva a los diferentes propósitos de las interfaces de usuario 3D.

Como ilustra la Figura 5.11, este concepto de zona, lugar o locus es común a todos los espacios, si bien es en el virtual que ahora se describe y el real que se abordará después donde más protagonismo tiene. La figura también plasma la relación que un lugar puede tener con objetos, tareas y puntos de vista. También con otros lugares, como por ejemplo una relación jerárquica como la que se plantea en RealPlaces, y haciendo extensible dicha jerarquía a los objetos, tareas y puntos de vista con los que se relacionan.

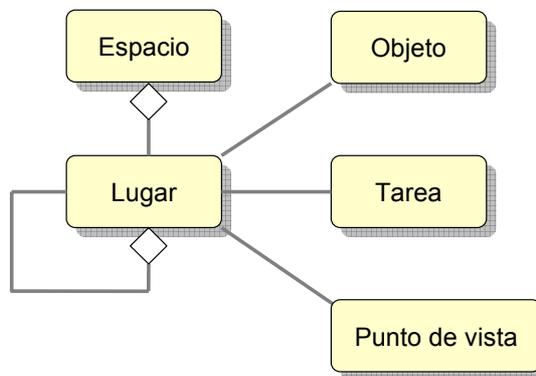
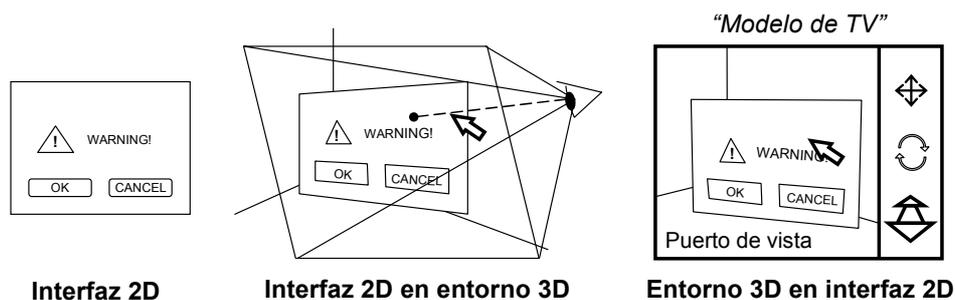


Figura 5.11 El concepto de lugar es común a todos los espacios

La libertad a la que antes se hacía referencia no significa que no se pueda formalizar este espacio, de hecho podría seguirse la idea de Trevisan *et al.* y, generalizando las relaciones de Allen al espacio 3D, construir una nueva matriz tridimensional que formalice las relaciones espaciales entre dos objetos del mundo virtual (lo que podría llamarse la **matriz 3D de Allen**).

Es más, podría decirse que esta parte del modelo no sólo abarca el espacio del mundo virtual con sus tres dimensiones, sino también el de las ventanas y widgets bidimensionales, pues no en vano se podría describir este último como un subespacio del anterior. Y es que, como ya se expuso en el apartado 2.4.1, los entornos 3D no están reñidos con las interfaces 2D (véase Figura 5.12), aunque en ocasiones hay obstáculos que sortear para conseguir tal integración. Por ejemplo, a diferencia de los componentes de una ventana, los objetos de un mundo virtual no se miden en unidades de pantalla –pixels–, sino en unidades físicas como, por ejemplo, el metro. Una solución consiste en mostrar las interfaces 2D sobre una versión virtual de un monitor de ordenador, con una resolución de pantalla dada en pixels pero también con sus correspondientes dimensiones físicas en pulgadas, solución adoptada en [Molina, 2005a], y formalizada después en [Molina, 2006c] con la representación virtual de plataformas hardware y superficies de interacción, concepto este último sobre el que se extenderá el próximo apartado.



**Figura 5.12** Una interfaz de usuario 2D en un entorno virtual 3D y viceversa

En el sentido opuesto, también los entornos 3D se integran en las interfaces 2D, en este caso a través del llamado **puerto de vista** –véase la Figura 5.13–, nombrado en el apartado anterior, un componente 2D que sirve para presentar al usuario la proyección del mundo virtual sobre un plano tomando como proyección un punto de vista dado. El puerto de vista es, de esta forma, como la retina del ojo del avatar del usuario y, por tanto, forma parte de él, como el resto de su ser virtual. Pero va más allá, dado que el usuario puede observar el mundo incluso desde puntos de vista de tercera persona, y en este sentido el puerto de vista puede verse como una ventana que el usuario abre en el mundo virtual, rasgando su espacio para asomarse a él, y comunicando sus dimensiones con las del mundo real.

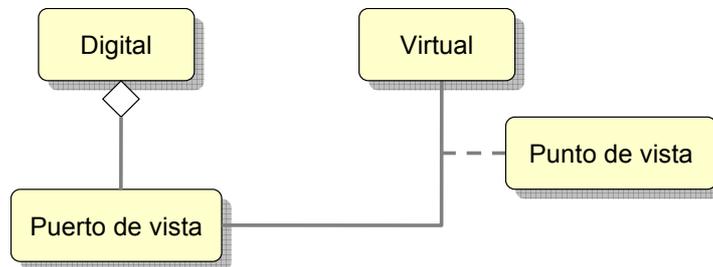


Figura 5.13 El puerto de vista relaciona el espacio virtual con el digital

Como componente 2D, el puerto de vista se integra en una interfaz de usuario 2D, siendo normal que en las aplicaciones de escritorio esté rodeado por otros controles 2D, lo que van Dam llamaba el modelo de TV, como se recordará del apartado 2.1 –véase también, de nuevo, la Figura 5.12-. En otras ocasiones, especialmente en las aplicaciones inmersivas, ese componente llega a llenar toda la pantalla, y esos controles 2D que le rodeaban pueden aparecer dentro del propio puerto de vista –lo que se denomina HUD (*Head Up Display*)- usando diferentes técnicas, desde superposición y transparencias hasta aparecer como controles 3D asociados a la vista del usuario y formando, de esta manera, también parte del avatar del usuario como pueda serlo, por ejemplo, una mano virtual. Además, una misma interfaz 2D puede contar con varios puertos de vista que muestren vistas simultáneas del mismo mundo virtual, incluso en diferentes pantallas, lo que también se verá en el siguiente apartado.

### 5.4.3 El espacio 3D real

Hasta este momento se ha abordado la ordenación en el espacio digital de widgets y objetos virtuales, lo cual puede ser suficiente para interfaces de usuario 2D y mundos virtuales, pero no lo es cuando la realidad alcanza el protagonismo que tiene en las aplicaciones del paradigma Realidad Aumentada. Y es que, en este último caso, lo que percibe el usuario es el mundo real, y éste es aumentado a través de ciertos dispositivos. Pero no sólo en ese caso se necesita un modelo del mundo real, pues los periféricos del ordenador que sirven de medio físico en interfaces de usuario 2D y mundos virtuales también ocupan un lugar en el mundo real, y las propias acciones del usuario se llevan a cabo en el espacio de ese mundo, aunque los dispositivos y las técnicas de interacción las transformen en acciones virtuales dentro de la aplicación.

Así, en una aplicación de Realidad Aumentada, el espacio bien puede corresponderse con el de una edificación –p. ej. un museo- o un lugar al aire libre –p. ej. un yacimiento arqueológico-. En el caso de la Realidad Virtual o la Virtualidad Aumentada, es más fácil que se corresponda con lugares más

reducidos, como el que ocupa una instalación tipo CAVE o incluso un simple ordenador de escritorio. Como en los mundos virtuales, el concepto de **zona**, **lugar** o **locus** resulta también necesario, describiendo espacios a los que dan forma las propias construcciones físicas –p. ej. las salas de un museo-, o bien espacios definidos no tanto a ese nivel físico sino a otro nivel lógico, de la aplicación –p. ej. las proximidades a una vitrina expuesta en una de las salas-. Volviendo aquí al caso de estudio que se presentaba en [Granollers, 2002], serían las zonas de influencia de los puntos de interés que se marcaban sobre el plano del yacimiento.

En el caso de estudio al que se ha hecho referencia en el párrafo anterior, los autores también marcaban, dentro de la zona de influencia, unos **puntos de observación** como vistas concretas de cada punto de interés. Sin embargo, a diferencia de los mundos virtuales, en donde el avatar puede situarse de forma exacta sobre un punto de vista nada más pulsar el usuario un botón, en el paradigma de la Realidad Aumentada es el usuario el que se suele desplazar por sus propios medios a tal posición o girar su propia cabeza para mirar en tal dirección, con lo que el resultado es más bien una aproximación a las posiciones y/u orientaciones marcadas. A pesar de diferencias como esta, el modelo del espacio real es, hasta aquí, muy parecido al modelo del mundo virtual. Y como entonces, lo que se pretende en esta parte del modelo es dotar al mismo de la suficiente generalidad como para hacerlo útil a cualquier desarrollo de interfaz de usuario 3D.

De este modo, la simple división en parcelas del mundo real debe dejar paso también a una ordenación más versátil que tenga en cuenta, por ejemplo, el volumen de trabajo de un determinado dispositivo de entrada –p. ej. un sistema de localización- o de salida –p. ej. un sistema de presentación estereoscópico-, lo que dará pie, como se verá a continuación, al concepto de volumen de interacción.

Precisamente, son los **dispositivos físicos** el nexo de unión entre el mundo real y el digital o virtual, y en este punto el modelo se nutre también de otras ideas ya presentadas en [Molina, 2006c], en concreto del modelo del entorno para el lenguaje UsiXML que en él se recoge –véase también el apartado 2.7.9-. A grandes rasgos, este otro modelo describe las plataformas hardware que maneja el usuario en su entorno, y en particular las superficies asociadas a las mismas, como la pantalla de un monitor o de un equipo de proyección, aunque sin detenerse tanto en el tipo de estas como, más bien, en su condición de **superficie de interacción** –véase Figura 5.14 y Figura 5.15-. Este último término fue introducido en [Coutaz, 2003], definiéndose como cualquier superficie física que puede ser observada (p. ej. mirando a la pantalla) o sobre la que se puede actuar (p. ej. a través del puntero del ratón) y que permite la interacción con un sistema, ya sea visible o embebido.

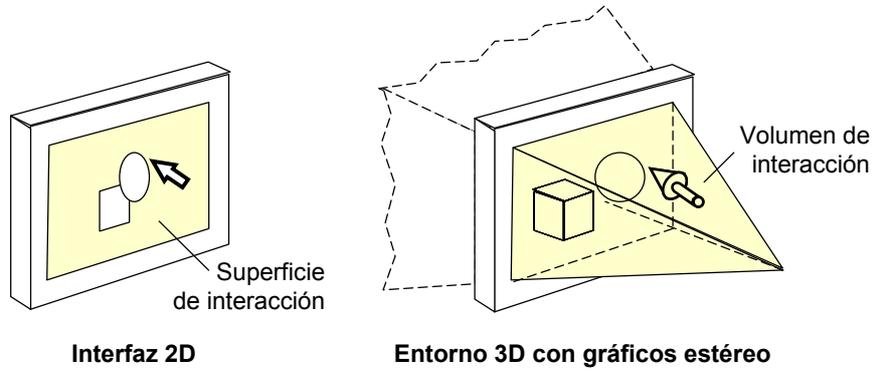
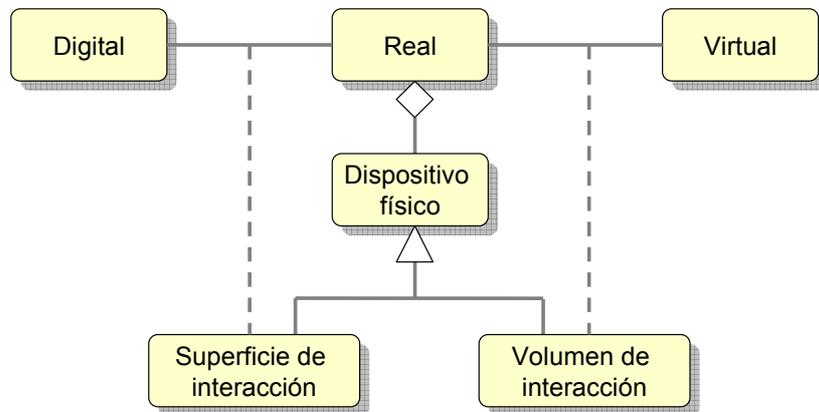


Figura 5.14 Superficie y volumen de interacción en una misma pantalla de ordenador

Sin embargo, no cabe duda que ese concepto de superficie de interacción fue introducido a propósito del tipo de interfaz más extendido: la interfaz gráfica 2D que se representa en pantallas planas. En cambio, aquí es necesario extenderlo más allá de las dos dimensiones para introducir, como se adelantado antes, el concepto de **volumen de interacción**, más apropiado para las interfaces 3D en donde la interacción no se restringe al plano sino que tiene lugar en las tres dimensiones del espacio real –véase, de nuevo, Figura 5.14 y Figura 5.15-. Así, como también se ha apuntado antes, un volumen de interacción puede venir definido por el alcance de un sistema de localización, pero también un monitor de ordenador puede crear un volumen de interacción frente al mismo al representar imágenes 3D estéreo, solapándose las dimensiones virtuales con las reales. De hecho, la representación de imágenes 3D sintéticas en un puerto de vista sobre la pantalla de un monitor también crea un volumen de interacción tras esa pantalla, aunque no pueda alcanzarse alargando la mano. Más aún, dichas imágenes son calculadas en base a un punto de vista que tiene también su correspondencia con un punto en el mundo real, de modo que el usuario verá mejor dicha imagen cuanto más ajuste su posición a dicho punto, del mismo modo que para experimentar sonido 3D con dos altavoces debe situarse en una posición dada frente a los mismos.



**Figura 5.15** Las superficies y volúmenes de interacción relacionan el espacio real con los espacios digital y virtual

#### 5.4.4 Transiciones intra-, inter- y trans-espaciales

Los espacios digital y virtual son potencialmente infinitos, sólo limitados por la tecnología del momento, y los límites del espacio real van más allá de nuestro mundo. Sin embargo, los contenidos digitales y virtuales suelen estar bien acotados dentro de sus espacios, así como los objetos en el mundo real que participan en la interacción. A pesar de ello, la vista que el usuario tiene de los espacios anteriores y sus contenidos suele ser sólo parcial, incompleta. Algunas de las razones para ello son físicas, tienen que ver con la propia percepción humana –p. ej. el campo de visión del sentido de la vista-, aunque es más normal que sean los dispositivos los que, por sus limitadas prestaciones, sólo puedan abarcar una porción de los contenidos de esos espacios. Otras razones derivan de la práctica en el diseño de interfaces, más relacionadas con la secuencia de las tareas a realizar a través de ellas, pero también con la capacidad de atención del ser humano y el riesgo de sobrecarga de información. Sea como fuere, para obtener una vista completa del espacio el usuario debe entonces navegar a través de los contenidos, dejando una vista atrás para avanzar a otra, lo cual, por razones como las anteriores, suele traducirse en el reemplazo de unos contenidos por otros en el espacio de los dispositivos de presentación.

Para el caso del espacio 2D digital, las metodologías de desarrollo de interfaces bidimensionales también introducen diagramas específicos con los que modelar estas transiciones. Por ejemplo, en la metodología IDEAS el diagrama concreto es el de navegación, con el que se describe de forma gráfica las diferentes posibilidades de transición entre ventanas, complementando así a otro diagrama, el de especificación de componentes, en la descripción sintáctica de un diálogo persona-ordenador basado en una interfaz gráfica 2D. Esta es la navegación **inter-ventanas**, aunque el concepto de ventana no debe entenderse en el sentido

estricto de un sistema operativo basado –precisamente- en ventanas, sino en un sentido más amplio, pudiéndose hablar también, por ejemplo, de pantallas en lugar de ventanas.

Seguramente, lo que representa el anterior diagrama de navegación coincide con los caminos de navegación que se incluían en la propuesta metodológica brasileña descrita y analizada en el apartado 3.5.2, como saltos de unas ventanas a otras y no como sendas que recorre el usuario en un mundo virtual. En este último caso, el del espacio 3D virtual, aunque el usuario pueda navegar libremente por el mundo virtual y con ello explorarlo desde muy diferentes puntos de vista, también es común definir un conjunto significativo de los mismos, incluso un orden en el que se toman esos puntos de vista predefinidos cuando se desea programar una visita guiada por el mundo. Esta es la navegación **intra-mundo**, aunque no es la única en este tipo de espacios.

Y es que la naturaleza inmaterial de los mundos virtuales permite jugar con el espacio de formas imposibles en el mundo real, como las estructuras tipo “tardis” que se comentaron en el apartado 2.5.1. Así, otra de las posibilidades que ofrece el espacio 3D virtual es la de comunicar dos mundos a través de unas singulares puertas que conectan las dimensiones de uno con las del otro –lo que en InterSpace se denominan puertas super- o subespaciales [Leftwich, 1993], y que en Croquet se denominan portales o hiperenlaces 3D [Croquet, URL]- permitiendo así viajar entre ellos con un solo paso. Los puntos de vista predefinidos bien pueden servir como puntos de entrada al mundo destino. Esta es la navegación **inter-mundos**, mecanismo que recuerda en parte a las teorías de los agujeros de gusano en el Universo, por lo que puede pensarse en el mismo como atajos espacio-temporales dentro de un mismo Universo o entre varios Universos (Multiverso).

El último tipo de navegación que aquí se contempla es el **trans-espacial**. Con él se pretende modelar las transiciones que se producen en un espacio por estar ligado a otro. Un ejemplo de este tipo de navegación es el cambio de punto de vista sobre un mundo virtual producido por el propio movimiento del usuario en el mundo real, gracias al empleo de ciertos dispositivos de seguimiento. Otro ejemplo más, esta vez de una aplicación de Realidad Aumentada, es el diálogo aumentado que se presenta en [Granollers, 2002], en el que se liga la presentación de ciertas interfaces bidimensionales a la llegada del visitante de un yacimiento arqueológico a diferentes puntos del mismo.

De esta forma, se completa este modelo del espacio que, sumado a los otros dos que le precedían en este capítulo, el de los objetos y el de la interacción, juntos constituyen la base del lenguaje que se empleará en la metodología TRES-D y que, ahora sí, se irá desarrollando en los siguientes apartados.

## 5.5 Resumen

A lo largo de este capítulo se han descrito tres modelos con los que se ha querido limpiar de ambigüedades y contradicciones el vocabulario propio de las interfaces de usuario 3D, afinando el significado de unos términos o introduciendo otros nuevos cuando se ha creído necesario, y reafirmando o redefiniendo las relaciones entre unos conceptos y otros. El fin de estos modelos es el de ayudar al desarrollador a construir su propio modelo mental de dichas interfaces, a modelar los diferentes aspectos que involucran y comunicar dichos modelos a otros desarrolladores. Por eso, más que modelos, lo que en este capítulo se han descrito son “meta-modelos”.

Empezando entonces con el meta-modelo de objetos, con él se introdujo una nueva clasificación de estos tratando de reunir en una sencilla jerarquía las diferentes clases identificadas por otros autores, distinguiendo entre proceso, objeto estático, dinámico, reactivo, agente y avatar. Con esta jerarquía se persigue que el desarrollador pueda hacerse una idea de las necesidades de diseño que tendrá cada objeto, pero una vez clasificados debe también saber qué especificar y cómo. En este sentido, y tras una nueva revisión de los diferentes modelos de autores anteriores, se propuso para esta metodología la composición de función, comportamiento y un conjunto de sub-modelos. Entre estos sub-modelos se destacó el gráfico, identificando en el mismo otros cuatro elementos: estructura, geometría, apariencia y percepción.

Al meta-modelo de objetos le siguió el de elementos de la interacción, basado en parte en las explicaciones de Foley *et al.*, pero sobre el que se ha hecho un gran esfuerzo por aclarar algunos términos muy ampliamente utilizados en la literatura, pero en cuyos significados los autores no siempre se ponen de acuerdo. Los conceptos que incluye este modelo son: diálogo, tareas, operaciones, tareas de interacción, técnicas de interacción, acciones, controles y dispositivos físicos. De todo ello cabe destacar cómo, en el caso de las tareas, se distingue entre tareas de alto nivel y subtareas, y dentro de éstas las de nivel más bajo son las operaciones, seguidas por las unidades de información que estas precisan. Estas últimas representan entonces las Tareas de Interacción Básicas o BITa's, y conforme se sube en la jerarquía lo que se encuentra son Tareas de Interacción Compuestas o CITa's. También cabe destacar la distinción entre técnicas de interacción y controles, participando estos últimos en las primeras, de modo que un widget, como control que es, no se considera aquí una técnica de interacción, sino sólo una parte de ella. Además, se aclaró el concepto de widget 3D y se introdujo el de objeto. Y por paralelismo con las tareas de interacción, también se puede hablar de Técnicas de Interacción Básicas o BITE's y Técnicas de Interacción Compuestas o CITE's. Con estos acrónimos se

trata de huir de las siglas IT, tan utilizadas en la bibliografía, pero que llevan a confusión al poder representar tanto a tareas como a técnicas de interacción.

El tercer y último meta-modelo en ser desarrollado fue el del espacio que, basado en el continuo digital-virtual-real presentado en el segundo capítulo, distinguía entre el espacio 2D digital, el espacio 3D virtual y el espacio 3D real. En estos dos últimos se subrayaron los conceptos de zona, lugar o locus, así como el de punto de vista o punto de observación, como elementos importantes en la ordenación de esos espacios. En los tres se destacaron las relaciones que los unían, como embeber un mundo virtual en una interfaz 2D a través de un puerto de vista, sumergir una interfaz 2D en un entorno 3D, y conectar todo ello con el mundo real a través de dispositivos físicos que crean superficies y volúmenes de interacción. Todo ello se completó con la definición de la navegación inter-ventana, intra-mundo, inter-mundos –a través de portales- y trans-espacial.

Con todo, y como se habrá podido observar, los tres meta-modelos también se relacionan entre sí, las tareas se descomponen en acciones concretas sobre unos objetos, y tanto tareas como objetos tienen una fuerte relación con el espacio, algo de esperar en este tipo de interfaces de usuario. Esta relación a tres bandas marcará el modelo de proceso de la metodología TRES-D, como se verá en el siguiente capítulo.



## 6

# El modelo de proceso de la metodología TRES-D

## 6.1 Introducción

Un modelo de proceso puede ser tan simple como la sucesión requisitos, diseño e implementación que se derivaba de las palabras de Shneiderman con las que se abrió el tercer capítulo. Estas etapas son comunes en los proyectos de ingeniería del software, y en aquel capítulo podían verse formando la espina dorsal de las metodologías que apoyaban su solución en esta disciplina, acompañadas por otras etapas también conocidas, como por ejemplo la evaluación y el lanzamiento o despliegue. Este era el caso de metodologías tales como LUCID, OVID o IDEAS –recuérdense del apartado 3.8-, en las cuales las técnicas de ingeniería del software se combinan con las propias de la interacción persona-ordenador para identificar, a lo largo del proceso, las interacciones entre el usuario y los objetos de la aplicación, para así desarrollar la funcionalidad de la interfaz de usuario. Orientadas hacia interfaces más convencionales, los objetos de la aplicación que tratan suelen ser encapsulados de datos abstractos para cuya representación en la interfaz y manipulación a través de la misma se recurre a un conjunto estándar de widgets. Algunas de estas metodologías aprovechan esta circunstancia para abordar el desarrollo a dos niveles, uno abstracto y otro concreto o de presentación, como es el caso de OVID e IDEAS, y en general de muchas de las últimas metodologías de desarrollo de interfaces de usuario.

En la creación de mundos virtuales no se observa, en cambio, esa distinción entre lo abstracto y lo concreto, al menos no en las metodologías para este propósito revisadas en el capítulo anterior –véase, en particular, el apartado 3.3-. Estas se vuelcan, en su lugar, hacia técnicas más propias de los artistas, como el dibujo de bocetos y de tiras de viñetas que describan el mundo virtual y los objetos que lo pueblan. Y la etapa que sigue al diseño no es la implementación, sino la construcción o, más concretamente, el modelado, ensamblado y la optimización, destacándose ahora estas como la espina dorsal de estos otros modelos de proceso. Y es que dichos procesos no suelen tratar con datos abstractos sino con objetos que tienen una apariencia propia que es la que debe mostrarse, en la medida que el hardware de gráficos lo haga posible, a través de la interfaz de usuario.

Sin embargo, para que el mundo virtual creado sea usable, es preciso combinar las técnicas y procesos de los creadores de mundos tridimensionales con lo propio de los diseñadores de interfaces de usuario, como ya apuntara Kaur en su trabajo doctoral [Kaur, 1998]. En este sentido, las soluciones propuestas por esta y otros autores para aunar todo el desarrollo bajo un mismo modelo suelen pasar bien por una unión de los diferentes procesos en secuencia, bien por una unión en paralelo. En el primer caso, las primeras etapas se llevan a cabo siguiendo los métodos de la interacción persona-ordenador que aseguren un correcto diseño, dejando a continuación la construcción en manos de los artistas que crearán los objetos tridimensionales y les darán vida en el mundo virtual. Ejemplos de ello son la metodología propuesta por la propia Kaur –apartado 3.3.3- y la metodología VEDS –apartado 3.5.3-. En el segundo caso, la creación de los modelos tridimensionales suele tratarse como una caja negra, dando la solución sólo desde la perspectiva del diseñador de interfaces de usuario, y limitándose a las especificaciones que este debe dar al creador de los modelos en determinados momentos del desarrollo. Ejemplos de ello son las metodologías SENDA –apartado 3.6.3- y VRID –apartado 3.7.3-.

Siendo los mundos virtuales una de las aplicaciones de las interfaces de usuario 3D, podría pensarse entonces que cualquiera de estas últimas metodologías sirve igualmente para el desarrollo general de dichas interfaces. Pero como ya se discutiera en el capítulo anterior, propuestas como la de Kaur o metodologías como VEDS no contemplan la importante labor de programación que puede acarrear la implementación de una interfaz de usuario 3D, incluyendo muchos entornos virtuales, dejando la construcción final en manos de los artistas y sus herramientas. La propuesta que se presenta en esta Tesis opta entonces por integrar en el desarrollo los procesos de los diferentes roles en paralelo, resaltando también los momentos en los que deben comunicarse y cómo deben hacerlo, pero sin volcarse tampoco en un rol concreto como ocurre en SENDA o VRID, sino conservando las perspectivas de cada uno de ellos sobre el desarrollo. En este sentido, se sigue más el camino abierto por Fencott –recuérdese el apartado 3.6.1- quien resalta la necesidad de combinar lo artístico con lo ingenieril, y para ello propone un modelo en el que se desarrollan, en paralelo, un modelado conceptual y otro de la percepción a un lado, y un modelado estructural a otro.

## 6.2 Fases y etapas del modelo de proceso

La metodología TRES-D que aquí se propone es un modelo de proceso que parte de una primera división, en horizontal, en una etapa de **diseño** y otra de **implementación**. La etapa de diseño se divide, a su vez, y también en

horizontal, en dos niveles: un primer nivel más alejado de la implementación, en la cual podría enmarcarse el diseño abstracto de las metodologías más generales de creación de interfaces de usuario, pero también el diseño conceptual o general de los mundos virtuales; y un segundo nivel más próximo a la implementación, en el cual se situaría el nivel concreto o de presentación de las interfaces de usuario, y también el diseño detallado de los mundos virtuales. El paso del primero al segundo vendrá marcado por la decisión que se tome respecto al software y hardware –sobre todo este último- que se utilizará para la implementación. Ya en esa etapa de implementación, se distingue entonces entre programación, por una parte, y creación de los modelos tridimensionales y su ensamblado para formar la escena, por otro.

Como se puede observar, se establece también una división vertical del modelo entre los procesos más ligados al desarrollo ingenieril de una interfaz de usuario, y aquellos otros que responden a las prácticas de los artistas. Esta división vertical produce entonces dos caminos a seguir en el desarrollo, uno orientado hacia las tareas y la interacción, y otro hacia los objetos y el mundo. Ambos caminos se cruzan entre sí, y en las interfaces de usuario 3D especialmente a través de un tercer protagonista, el espacio. Cada uno de ellos se corresponde con uno de los **tres meta-modelos** presentados anteriormente en este mismo capítulo, formando estos la piedra angular de la metodología. Así, desde la perspectiva de cada uno de esos tres modelos, la metodología guía entonces hacia la creación de diferentes **vistas de diseño** de la interfaz de usuario 3D desplazando el foco de atención sobre dos ejes, uno que diferencia entre vistas *individuales* y *de conjunto*, y otro que distingue entre vistas *estáticas* y *dinámicas*.

Sobre esta base, la metodología TRES-D se levanta incorporando etapas previas al diseño que tienen por fin identificar los requisitos que debe cumplir la interfaz de usuario, y etapas siguientes a la implementación que abordan la explotación de la interfaz creada. Aún con todas ellas, el desarrollo de una interfaz de usuario 3D presenta grandes retos debidos, como se trató de transmitir en el segundo capítulo, al estado inmaduro en el que actualmente se encuentra este campo. Estos retos requieren la toma de una serie de decisiones, más difíciles de resolver por la falta de experiencia. Y con estas decisiones se asumen unos riesgos, más elevados al apostar por soluciones menos convencionales. Sobre estas arenas movedizas, el desarrollador debe moverse con pies de plomo.

Por esta razón, para la metodología TRES-D se ha optado por dividir el conjunto del desarrollo en dos grandes *fases*, un **estudio previo** y un **estudio detallado**, repartiéndose las diferentes *etapas* entre ellos –véase Figura 6.1-. Así, en el estudio previo se presenta el problema, se analiza y se llevan a cabo diferentes estudios de diseño para proponer una solución que sea factible. Si se acepta su desarrollo, sólo entonces toma el relevo el estudio detallado, profundizando en el

diseño de la solución hasta completarla, siguiendo después con la implementación, el despliegue y el mantenimiento.

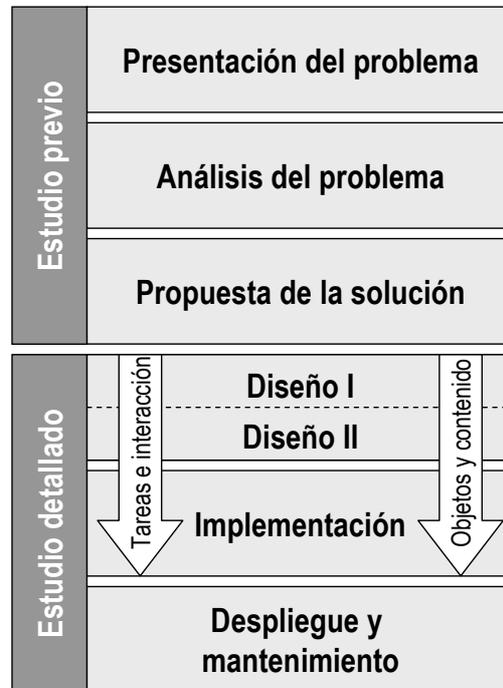


Figura 6.1 Modelo de proceso de la metodología TRES-D

Puede parecer que esto último va en contra de la corriente dominante en la producción del software actual, en la cual se prima la rapidez con la que se ejecuta el desarrollo apostando por técnicas más agresivas. Sin embargo, debe tenerse presente que estas últimas suelen beneficiarse de la seguridad que da la experiencia, experiencia que en el caso de las interfaces de usuario 3D escasea. Por esta razón, es opinión del autor de esta Tesis volver la mirada a soluciones de épocas pretéritas en las que los riesgos en la producción de software eran mucho mayores, pues entonces el estado en el que se encontraba ese campo era también inmaduro, como lo es ahora el de las interfaces de usuario 3D.

No debe, sin embargo, confundirse esta división en fases y etapas con un desarrollo fuertemente dirigido en el que no haya lugar para el comienzo de una etapa si no se ha completado antes la anterior, lo que en la práctica puede resultar pesado de ejecutar pero, sobre todo, puede carecer de todo sentido. Nada más lejos. Precisamente, para poder avanzar sobre un terreno que no se conoce, lo sensato es tantearlo antes de pisarlo con fuerza. En este sentido, en la definición de la metodología TRES-D se ha tenido muy presente la necesidad de **adelantarse** a los posibles problemas que puedan surgir durante el desarrollo.

Para ello, a lo largo del mismo se contempla la posibilidad de avanzar en actividades que correspondan no a la etapa actual sino a otras posteriores, con el propósito de comprobar si una determinada solución es factible y apoyar así la toma de decisiones. El ejemplo más claro de ello se encuentra en la etapa de propuesta de la solución, última del estudio previo, en la cual se avanza en diferentes actividades que corresponden a las etapas presentes en el estudio detallado, buscando el respaldo técnico de la solución cuyo desarrollo se quiere proponer. Dejando a un lado esta característica del modelo de proceso de la metodología TRES-D, esta se beneficia también de otras estrategias en diferentes momentos del desarrollo, como el ciclo en espiral de Boehm [Boehm, 1988] para dar solución a los retos que se van presentando, ciclos iterativos para refinar las distintas soluciones, construcción top-down y bottom-up, etc.

Con la metodología TRES-D se persigue, además, ofrecer un marco de referencia en el que acomodar las diferentes **herramientas** que los creadores de las interfaces de usuario 3D puedan necesitar a lo largo del desarrollo, tales como aquellas que sirven para guiar en el diseño de las soluciones, para documentarlas o para plasmarlas en una implementación concreta. En este caso, la metodología TRES-D no se liga a ningún conjunto particular de herramientas, pero sí se dan las recomendaciones oportunas acerca de cuáles pueden ser las más apropiadas en cada momento del proceso. De esta forma, la metodología puede adaptarse mejor a la variable experiencia que puedan tener los diferentes creadores según qué herramientas, y a las preferencias que puedan tener esos creadores por unas u otras. No obstante, frente a la mayor atención que han prestado otros autores a las notaciones para la especificación de diferentes aspectos de las interfaces de usuario 3D, es la intención del que escribe hacer más hincapié en las herramientas que guían al creador de esas interfaces.

Con todo ello, la metodología TRES-D puede verse como una propuesta con la que se busca reunir, en un nuevo modelo de proceso, lo mejor de las metodologías anteriormente publicadas, para poder así adaptarse a la variable complejidad de las diferentes aplicaciones que pueden plantearse en el campo de las interfaces de usuario 3D. Este modelo de proceso se describe en profundidad en los siguientes apartados, siguiendo el orden dado para las fases y etapas que lo componen.

### 6.3 Estudio previo

En la metodología TRES-D se distingue entre dos tiempos del desarrollo de una interfaz de usuario 3D, uno en el que se trabaja para comprender el problema y proponer una solución, y otro en el que se trabaja por hacer realidad esa solución. Esta distinción tiene su reflejo en la división, en horizontal, del modelo

de proceso en dos fases, un estudio previo y un estudio detallado, tal y como se ha visto en el apartado precedente. Entre la primera y la segunda fase media una decisión vital, que es la de aceptar la solución propuesta y dar con ello luz verde a la segunda fase. Con ello se pretende, por un lado, ahorrar esfuerzos cuando no se da con una solución convincente y, por otro, que se den las mayores garantías cuando se decida llevar una solución hasta el final.

El estudio previo se enmarca, de esta forma, en ese primer tiempo del desarrollo. Abarca tres etapas, empezando con la presentación del problema, siguiendo con su análisis y terminando con la propuesta de solución. Tras esas etapas, el resultado es una propuesta que, sin llegar a ser desarrollada como solución en toda su extensión, se detalla lo suficiente como para permitir, por una parte, evaluar los beneficios e inconvenientes de dicha propuesta y fijar, por otra parte, el compromiso y los límites que con ella asume el equipo de desarrollo.

Es importante destacar que la metodología no precipita el desarrollo hacia una interfaz de usuario 3D. Antes al contrario, parte del trabajo a realizar en esta primera fase consiste en evaluar si una interfaz de ese tipo es la solución más apropiada al problema que se presenta, advirtiendo del riesgo que conlleva dejarse llevar por el simple atractivo inicial que puedan ofrecer estas interfaces. Resulta inevitable, sin embargo, que el simple de hecho de seguir esta metodología pueda interpretarse como una declaración de intenciones aunque, más que eso, debe considerarse como una apuesta sensata.

Así, los primeros pasos de la metodología bien pueden asimilarse a los que puedan darse con otros enfoques estructurados para el desarrollo de otras formas de interfaces de usuario, por lo que llegado el momento el equipo de desarrollo podría abandonar la metodología TRES-D, pero sin perder con ello el trabajo realizado hasta entonces. Más aún, dado que esta metodología contempla también otras formas de interfaces de usuario que se presentan en muchas aplicaciones junto a las tridimensionales, como las convencionales de escritorio, también cabe la posibilidad de continuar el trabajo con la propia metodología TRES-D, dejando a un lado las características propias de la interacción en el espacio 3D.

Con todo, esta fase involucra personas con muy diferentes roles. Comienza con reuniones entre el cliente y el arquitecto del sistema, siguiendo el trabajo analistas, expertos en el dominio del problema y usuarios actuales o futuros, para finalmente involucrar a todos ellos junto a diseñadores, ingenieros, programadores y artistas en la búsqueda de una solución.

En este sentido, cabe subrayar que tanto aquí como en la explicación que resta de la metodología, toda vez que se haga referencia a un tipo de participante deberá entenderse que se trata del rol que representa, no de una persona o

personas concretas, con independencia de que en el texto se emplee el singular o el plural al citar dicho rol, pues dependerá ya de cada proyecto particular que ese rol sea asumido por una o varias personas, o que una misma persona pueda asumir varios de ellos.

En cuanto al tiempo que llevará esta fase, el plazo estimado para la misma dependerá también de la envergadura de cada proyecto en concreto, pero no debería extenderse más allá de unos meses, repartidos entre las tres etapas que se detallan a continuación.

### 6.3.1 Presentación del problema

Esta etapa marca el comienzo de la metodología TRES-D y en ella el equipo de desarrollo tiene su primer contacto con el problema, normalmente presentado por un **cliente**. En lo que sigue se supondrá que es así, aunque también es posible que no exista tal cliente sino que el equipo se plantee el reto de dar solución a un problema por propia iniciativa.

El equipo de desarrollo o, mejor dicho, una representación del mismo, se reúne entonces con ese cliente. De esa representación destaca el **arquitecto**, nombre que aquí se da a la figura principal del equipo, siguiendo la idea de Shneiderman de ese arquitecto de la interfaz de usuario –“UI architect”- que debería tener todo proyecto, con las habilidades necesarias para planificar y preparar presupuestos, gestionar el trabajo de otra gente y coordinarlo con profesionales internos y externos [Shneiderman, 1998].

El propósito de esa reunión –o reuniones- es que el cliente exponga el problema y, con ello, los **objetivos** –en inglés, “goals”- que debe cumplir la solución y los **límites** –en inglés, “constraints”- dentro de los cuales debe plantearse dicha solución –p. ej. en el caso de un mundo virtual para la Web, este deberá ser optimizado para su distribución a través de Internet-. La labor del arquitecto es confeccionar con ello la lista de requisitos iniciales.

No debe esperarse, sin embargo, que de la exposición del cliente se deriven fácilmente esos requisitos, pues como se apuntaba en [Eastgate, 2001] el cliente es quien decide el fin principal, pero este puede ser vago o en parte impracticable. Por ello, el arquitecto debe hacer uso de su experiencia y habilidad para discutir y aclarar en este primer momento los **requisitos principales**, así como los **criterios** con los que se evaluará el cumplimiento de los mismos. Como herramientas, el arquitecto puede recurrir a los *casos de uso* del lenguaje UML, o en su defecto a *listas o tablas*.

La propia experiencia del arquitecto le ayudará a estimar, ya en este momento, hasta qué punto el problema es practicable. Si el arquitecto juzga que no lo es, el trabajo acabaría aquí. Por el contrario, si el arquitecto juzga que el desarrollo es viable, centra entonces su atención en la fase que está en marcha y planifica las dos siguientes etapas, estas son, el análisis del problema y la propuesta de solución. Para ello, el arquitecto se plantea qué recursos humanos y materiales serán necesarios, el tipo de colaboración que deberá prestar el cliente o que pueda necesitarse de terceras personas, y los plazos de ejecución. Con ello elabora un **presupuesto** y, en su caso, redacta los términos en los que se solicita la colaboración del cliente.

Tras los arreglos necesarios para llegar a un acuerdo con el cliente, se da paso entonces a la segunda etapa, en la que se profundizará en los requisitos listados y otros que se hayan podido dejar a un lado por ser menores, más los nuevos que puedan surgir a raíz de ese estudio.

### 6.3.2 Análisis del problema

Como bien se apuntaba en [Foley, 1996], antes de llevar a cabo ningún diseño es necesario saber bien qué se pretende conseguir, labor que es realizada en esta etapa por miembros del equipo de desarrollo bajo el rol de analistas. Al término de la misma deberemos saber qué tipo de aplicación se desea crear, cuáles son las características de las personas que harán uso de la misma, qué tareas llevarán a cabo y cuáles serán los objetos de esas tareas.

Para el correcto desarrollo de esta etapa, los analistas deben primero familiarizarse con el problema, recopilando información al respecto. En este sentido, la colaboración del cliente resulta muy valiosa, bien por ser él quien tenga esa información o bien por las facilidades que pueda dar para acceder a ella. En cualquier caso, en la mayoría de los casos los analistas se encontrarán con un campo que les es ajeno y, por ello, además de la documentación a la que puedan acceder, resulta también muy interesante contar con la colaboración de expertos en el dominio del problema, por ejemplo a través de entrevistas que puedan conceder al equipo de analistas. El grado de implicación de estos expertos puede, sin embargo, llegar a ser mucho mayor, hasta el punto de adquirir un papel protagonista como ocurría en la metodología descrita por Celentano y Pitarello –apartado 3.4.2-.

Como punto de partida, los analistas toman los requisitos principales que recogió el arquitecto en la etapa anterior, tratando de profundizar en los cuatro frentes antes citados. Cualquier plan de ataque es válido con tal de que, al final, se hayan batido todos ellos. Ahora bien, si existe ya una solución al problema, parte

del trabajo de entender los requisitos estará hecho si se empieza por estudiar esa solución. En ese caso, los analistas deben fijarse en los objetivos incluidos entre los requisitos principales. Aquellos que esa solución cumpla, debe estudiarse cómo lo hace. Aquellos que no, debe estudiarse el porqué.

### 6.3.2.1 El tipo de aplicación

Por seguir un orden en este apartado, se tratará en primer lugar el tipo de aplicación. Con él se pretende definir la aplicación que demanda el problema asociándola a alguna de las diferentes clases en las que pueden catalogarse las aplicaciones ya existentes. Parés y Parés, por ejemplo, también incluían en su metodología el paso de definir el campo y el tipo de la aplicación en cualquiera de sus dos estrategias –apartado 3.7.2-, y en este sentido su clasificación de aplicaciones bien puede ser de utilidad en este punto. También la de Sutcliffe, atendiendo en su caso al grado de naturalidad –recuérdese el apartado 2.3.1-. Una buena definición de la aplicación ayudará al equipo, ya en la siguiente etapa, a decidir hacia qué tipos de interfaces de usuario dirigir sus esfuerzos, aunque para concretar las diferentes propuestas necesitarán más información.

### 6.3.2.2 El perfil del usuario

Siguiendo con el análisis, otro frente a estudiar es el usuario, la persona o personas que harán uso de la aplicación una vez terminada. Más que eso, el usuario es aquél para el que debe diseñarse la aplicación. Y es que la aplicación no sólo debe ser funcionalmente correcta sino que, a través de su interfaz, el usuario debe poder hacer uso de sus funciones de forma satisfactoria. Por ello es obligado tener presente al usuario a lo largo del desarrollo, y esta obligación empieza aquí mismo, por los analistas, quienes deben recopilar toda la información posible sobre él, quién es y cómo es.

Así, por un lado, los analistas deberán informar si ese usuario será una persona o muchas, si son conocidas o se trata de un público indeterminado. Por otra parte, qué conocimientos o habilidades tiene o se esperan de él –tanto del campo propio del problema como del uso de aplicaciones informáticas-, cuál será la frecuencia con la que hará uso de la aplicación, y –no menos importante- cuál es la actitud del usuario hacia la futura aplicación.

La realización de entrevistas con los implicados puede servir para recoger o completar esos datos. Con ellos, los analistas estudian si los usuarios forman un todo homogéneo o, al menos, si pueden dividirse en grupos de características similares, dando como resultado un conjunto de **perfiles de usuario**. Los

perfiles pueden describirse en *fichas* o en diagramas UML, utilizando en este último caso el concepto de *actor*. En cualquier caso, serán usados entonces por los diseñadores para ajustar aún más sus soluciones.

Por ejemplo, es habitual encontrar en la bibliografía la distinción entre dos perfiles particulares, uno el del usuario con conocimientos especializados y un uso continuado de la aplicación, y otro el del usuario con conocimientos más generales y un uso más bien esporádico. En el primer caso, puede sacrificarse tiempo del usuario para que este aprenda un conjunto más amplio de funciones, y/o atajos que reduzcan el coste de la interacción. En el segundo caso, se busca una interfaz que no requiera apenas aprendizaje previo a costa de reducir el número de funciones disponibles, y/o aumentar el coste de la interacción con menús y diálogos.

### 6.3.2.3 Las tareas y los objetos

Los últimos dos frentes en los que los analistas deben avanzar son las tareas y los objetos de las mismas, y por la relación que los une si bien pueden afrontarse por separado, a través de un **análisis de las tareas** del usuario, por una parte, y de un **análisis del dominio**, por otra, más sensato parece hacerlo de forma conjunta, comenzando por uno u otro frente, y cambiando según se avance. Precisamente, el peso de uno u otro análisis dará ya una idea de la orientación de la aplicación, si cae del lado de las tareas –esto es, de la funcionalidad- o, por el contrario, de los objetos -esto es, del contenido-. En cualquier caso, los analistas parten de los objetivos de la aplicación incluidos en los requisitos iniciales, preguntándose qué necesita el usuario para lograrlos. Más concretamente, deben preguntarse qué tareas necesita completar para lograr esos objetivos, y qué objetos necesita para completar esas tareas.

El resultado de estos últimos análisis será, por un lado, una descripción de cada tarea –indicando, en particular, quién la ejecuta y qué objetos involucra-, y el orden temporal, relaciones y dependencias de unas con otras. Por otro lado, también debe darse una descripción de cada objeto –indicando, en particular, en qué tareas participa-, y la relación de unos con otros, por ejemplo de agregación o todo-parte. Estas descripciones serían similares a las que se plantean en metodologías para interfaces más convencionales.

Sin embargo, pensando también en una posible interfaz tridimensional, aquí se pide a los analistas que no pasen por alto las **relaciones espaciales** que puedan existir tanto en las tareas como en los objetos, incluyendo en sus descripciones dónde se ejecutan las tareas y dónde se encuentran los objetos. De esta forma,

desde una etapa temprana como es esta se piensa ya en la relación entre los tres modelos sobre los que se apoya la metodología TRES-D.

Siguiendo con las tareas y sus objetos, la realización de entrevistas con los implicados puede ser de gran ayuda, identificando las actividades que estos consideren claves y dando forma con ellas a un conjunto de **escenarios** que guíen no sólo el análisis, sino también el diseño y la evaluación. Como herramientas para plasmar todo ello se recomienda recurrir a *árboles de tareas*, *diagramas de actividad* y *diagramas de casos de uso*, en el caso de las tareas, y *diagramas de clases*, en el caso de los objetos. Estas son, en definitiva, herramientas usuales en proyectos de ingeniería del software e incluidas en el lenguaje UML.

Pero, pensando de nuevo en la dimensión espacial, aquí se recomienda añadir a las anteriores el uso de representaciones gráficas de los lugares y objetos de las tareas, como *bocetos*, *planos* y *mapas*. Más aún, también se recomienda tomar *fotografías*, *grabaciones de audio* o *videos* de los mismos, recopilando así un material gráfico y sonoro que podrá ser utilizado por los diseñadores en sus propuestas, e incluso después, ya en el estudio detallado.

Si para el problema que se plantea ya existía una solución previa, parte del estudio de los analistas deberá centrarse en ella, tal y como se apuntaba al principio. Para acercarse a dicha solución, además de entrevistas y reuniones con los implicados, se recomienda la realización de un estudio etnográfico que permita conocer de primera mano cómo se logran los objetivos en la actualidad, qué es lo que hacen sus usuarios y –como destacan Foley *et al.*– por qué. De nuevo, las herramientas antes señaladas pueden servir para documentar esta solución –como, por ejemplo, se describe en la metodología OVID, apartado 3.8.3–, si bien debe diferenciarse entre los usuarios de ésta y los de la aplicación que se proyecta desarrollar, pues es posible que no sean los mismos –como también se destaca en OVID–. La realización del estudio etnográfico puede ser, además, una buena oportunidad para recopilar el material gráfico y sonoro del que antes se hablaba, más aún si se tiene en cuenta que, como se advierte en [Eastgate, 2001], no siempre se tienen muchas oportunidades para acudir a los lugares de interés.

Pero estudiar la solución previa no sólo debe tomarse como un medio para conocer mejor el problema, sino para analizar, de forma crítica, esa solución, destacando aquellos aspectos en los que se puede cambiar a mejor, justificando así la necesidad de su rediseño. En este sentido, los analistas pueden estimar la necesidad de llevar a cabo evaluaciones que les permitan profundizar en los puntos más críticos de dicha solución.

En cualquier caso, ya sea en las entrevistas, en el estudio etnográfico o en las evaluaciones, la colaboración del usuario es esencial. Precisamente, esta primera colaboración del usuario puede ayudar a identificar quién o quiénes son las personas más apropiadas para participar, de forma más activa, en el diseño de la aplicación, empezando ya en la siguiente etapa.

### 6.3.3 Propuesta de la solución

Hasta llegar aquí, el proceso comenzó dando forma a una lista de requisitos iniciales, entre los cuales se encontraban los objetivos de la aplicación, y siguió con el análisis del problema, profundizando en el tipo de aplicación, los perfiles de los usuarios, las tareas que deben poder realizar y los objetos sobre los que estas se llevan a cabo. Si, además, ya existía una solución al problema, los analistas también la han estudiado, destacando los aspectos en los que se puede mejorar.

Con toda esa colección de datos, es tiempo ya para que empiece esta tercera etapa del estudio previo, en la cual los diseñadores tomarán el relevo de los analistas para, ahora sí, dar forma a una propuesta de solución al problema planteado. No se trata aquí de desarrollar completamente la solución, sino de dar con la idea que mejor resuelve el problema en los términos dados, que prometa ser **satisfactoria**, desde el punto de vista del usuario, y **viable**, desde el punto de vista técnico, y elaborar con ella una oferta de desarrollo que el cliente pueda valorar y decidir si continuar o no con la siguiente fase, el estudio detallado.

La entidad del problema y los recursos de los que se dispongan determinarán el esfuerzo que puede realizarse en esta etapa para profundizar en un mayor o menor número de soluciones alternativas. Aunque ya se ha comentado anteriormente, se insiste de nuevo que no es el propósito de la metodología TRES-D empujar al equipo hacia el uso de interfaces de usuario 3D, y de la misma forma que sí se recomienda al equipo plantearse si este tipo de interfaces y su tecnología asociada pueden dar solución al problema o a partes del mismo, también se recomienda que, en cualquier caso, se planteen también soluciones alternativas que no hagan uso de la tercera dimensión. En todo momento, la prioridad debe ser sacar partido de la tecnología actualmente disponible, o de aquella que lo estará dentro del plazo de ejecución del proyecto, para, con ella, dar la mejor solución al problema. Ahora bien, desde esa perspectiva, es la opinión del autor de esta Tesis que no debe pasarse por alto el impulso que han experimentado en estos últimos años las tecnologías asociadas a las interfaces 3D –y no sólo los gráficos por ordenador- y las posibilidades que ello ofrece.

No es ninguna sorpresa pensar que la dinámica de esta etapa va a estar marcada por ciclos en los que se va alternando la generación de ideas con la validación de las mismas, y en la que si bien son los diseñadores los que asumen el papel protagonista, en ella deben involucrarse todos los roles, pues de su éxito dependerá la continuación del proyecto. No en vano el fin de esta etapa es **visionar** cómo será la aplicación una vez terminada, y sentar las bases para que su realización, ya en la fase de estudio detallado, se desarrolle sin contratiempos a través de las etapas de diseño, implementación y posterior despliegue y mantenimiento, en las que todos los roles están implicados. Para ello, el equipo de desarrollo no sólo debe dar con la idea que de solución al problema, sino también adelantarse al trabajo de esas etapas posteriores, avanzando ya en algunas de las actividades que en ellas se incluyen, y que más tarde se describirán, y utilizando para ello las mismas herramientas que para dichas etapas se recomiendan.

#### 6.3.3.1 Generación de ideas

Con ese papel protagonista, son entonces los diseñadores quienes comienzan a dar forma a esa visión de la aplicación, para concretarla a lo largo de esta etapa – y con la ayuda de los demás roles involucrados- en un **nuevo conjunto de escenarios** que describa la solución propuesta, detallando cómo los usuarios lograrán con ella los objetivos fijados. Una de sus decisiones más importantes es la de escoger el tipo de interfaz de usuario que exhibirá tal aplicación, decisión que se apoyará en la información recabada en la etapa anterior, en particular el tipo de aplicación, los perfiles de usuario y la orientación de la aplicación, bien hacia las tareas o hacia los objetos, así como la dimensión espacial de estos. La decisión a tomar puede variar según cada escenario, e incluso dentro de cada uno de ellos, según la tarea o el objeto del que se trate.

Precisamente, el plan a seguir consiste en revisar uno a uno los escenarios identificados durante el análisis, proponiendo ideas para cada uno de ellos, y en particular para cada tarea y objeto. Las ideas giran sobre la forma de llevar a cabo esas tareas y la forma de presentar esos objetos, pensando en unas interfaces de entrada y de salida particulares, y concretando los **requisitos de interacción** –p. ej. el grado de naturalidad-, de **contenido** –p. ej. el grado de realismo en la presentación- y de **tecnología** –software y hardware- de la aplicación para una solución dada. Más aún, la solución puede redefinir las tareas, los objetos e incluso los escenarios, haciéndose necesaria la distinción entre aquello que resultó del análisis del problema, y lo que ahora se propone como solución. Así, si la solución pasa por la creación de un entorno virtual, lo que debe concretarse ahora son las tareas que realizará el usuario en ese entorno,

distinguiéndose de las tareas que pudieran haberse identificado en etapas anteriores, como bien se diferenciaba, por ejemplo, en la metodología VEDS.

Este es un proceso indiscutiblemente creativo, en donde se mezcla experiencia, técnica, arte e ingenio. Existen, eso sí, ciertas reglas generales que los diseñadores bien pueden seguir, como hacer las tareas frecuentes simples y directas, y tener en cuenta las características del usuario, no sólo su grado de experiencia –novato o experto-, sino también su género, edad y aptitudes físicas, siguiendo las reglas de ergonomía más apropiadas en cada caso.

Como recurso, es común en estos casos realizar reuniones en las que se de rienda suelta a las ideas que puedan tener los diferentes miembros del equipo, utilizando técnicas como la tormenta de ideas o *brainstorming*. Como herramientas para plasmar los escenarios propuestos en una solución, es también común recurrir a *documentos escritos* que los narren, *bocetos* y *tiras de viñetas* que los ilustren, e incluso *videos* que les den vida con imágenes y sonido. Para profundizar en aquellos aspectos de la aplicación que se consideren más importantes, pueden utilizarse también las técnicas y herramientas que se enumerarán en la etapa de diseño, avanzando de esta forma en el estudio detallado para adelantar decisiones y resolver de antemano aquellos problemas que puedan preverse, asegurando con ello una solución factible.

### 6.3.3.2 Validación de ideas

Como se ha apuntado ya, a lo largo de esta etapa debe alternarse la generación de las ideas con la validación de las mismas. Así, por una parte, los escenarios que van produciendo los diseñadores deben contar con el visto bueno de los expertos en el campo del problema, y también con el de los futuros usuarios. En el caso de los expertos, estos deben dar su opinión acerca de la **corrección** de la propuesta con respecto a ese campo del problema, y en el caso del usuario, su opinión debe servir para profundizar en aspectos de **usabilidad** de la interfaz. Para ello deben programarse reuniones con unos y otros según se avance en esta etapa, haciéndoles partícipes del diseño –“participatory design”, en inglés-, refinando los escenarios y dando más prioridad a unos u otros aspectos de la solución.

Otro aspecto a validar, y que también se ha apuntado ya, es que la solución sea factible. Para ello, los diseñadores deben reunirse con una representación de aquellos miembros del equipo de desarrollo que llevarían a cabo la implementación final de la aplicación, como programadores y creadores de contenido digital. Los diseñadores les presentan los requisitos de interacción, de contenido y tecnológicos, y en esas reuniones deciden sobre la **viabilidad** del conjunto, y qué software –entornos de programación, librerías, herramientas de

autor, etc.- y hardware –máquinas, dispositivos de entrada y de salida, etc.- harán falta para llevar a cabo su desarrollo. Para comprobar la viabilidad de la solución, se llevan a cabo –siempre que sea posible- las *pruebas técnicas* necesarias, realizándose unos primeros desarrollos que dan lugar a unos prototipos tempranos, y con los cuales se persigue validar partes concretas de esa solución. De nuevo, esto representa un avance del trabajo a realizar en la fase de estudio detallado, en este caso de la etapa de implementación.

### 6.3.3.3 Presentación de la propuesta

El final de esta etapa viene marcada por la elección, por parte del arquitecto de la aplicación, de al menos una de las propuestas de solución desarrolladas hasta aquí, y su presentación al cliente, a la espera de su aprobación final. Para llegar a esa elección, antes habrán tenido que analizarse los pros y contras de cada solución, esto es, se habrá realizado un **análisis de riesgos**, de modo que el arquitecto pueda tomar una decisión informada.

Una vez tomada la decisión, se prepara entonces el *documento de propuesta de la solución*, en el cual se describa la aplicación, sus objetivos, funcionalidad y contenidos, se resalten sus beneficios pero se informe igualmente de los posibles problemas que podrá afrontar su desarrollo –no sólo técnicos, sino también de otra índole, como de entorno, sociales, legales, etc., al estilo del “social impact statement review” descrito en [Shneiderman, 1998]- y, finalmente, se detalle un plan para dicho desarrollo, en el que se especifique un presupuesto y un plazo de ejecución.

## 6.4 Estudio detallado

La aceptación, por parte del cliente, de la propuesta de solución, da comienzo a la segunda fase contemplada en esta metodología TRES-D que aquí se propone. Este estudio detallado parte entonces de esa propuesta, para ahora completar su diseño y hacerla realidad con su correspondiente implementación, despliegue y mantenimiento posterior.

Ahora bien, en la fase previa se huía de imponer una interfaz de usuario 3D a la aplicación, insistiendo en que una interfaz así podría no ser la más apropiada para el problema y que, por ello, se estudiara bien su adecuación a dicha aplicación y que, en cualquier caso, se tuvieran en cuenta otras alternativas. En esta nueva fase, bien al contrario, sí se asumirá que la interfaz de usuario es tridimensional, pues no en vano ese es el objeto de la metodología TRES-D. No obstante, lo que resta del proceso también podría servir para el desarrollo de

otros tipos de interfaces, dejando a un lado la relación de tareas y objetos con el espacio 3D, como ya se ha apuntado en alguna ocasión anterior. Y aún tratándose de una interfaz de usuario 3D, también cabe la posibilidad de continuar el proceso siguiendo alguna otra metodología de entre las analizadas en esta Tesis, pues aunque se destacaron las carencias de unas y otras, no es menos cierto que algunas de ellas sí han demostrado ser útiles en aquellas aplicaciones a las que se dirigen. Sin embargo, con la metodología TRES-D se pretende huir de la especialización en una aplicación concreta, aprendiendo de todas esas otras metodologías para ofrecer las técnicas y herramientas con las que abordar el desarrollo de cualquier interfaz de este tipo.

También se asume que, antes de llegar a esta fase, se han tomado ya las decisiones más importantes que afectan al desarrollo de la interfaz, no sólo el tipo, que ya sabemos que será 3D, sino también el grado de naturalidad en la interacción, el grado de realismo en la presentación, y el software y hardware en los que se basará la aplicación, su interfaz y el desarrollo de ambos. En realidad, lo que se ha hecho en la fase de estudio previo es avanzar a través del estudio detallado hasta aquellos momentos en los que es necesario enfrentarse a esas decisiones, adelantándose a los problemas que pudieran surgir. Precisamente, para esta fase se dispone, de partida, de mucha más documentación que la simple propuesta de solución, y no sólo por la descripción de los escenarios o los prototipos tempranos que hayan podido crearse, sino también por toda la documentación técnica correspondiente a esta nueva fase que se tuvo que adelantar en la fase anterior, para trazar con ello las líneas maestras del desarrollo de una interfaz que ahora se completa hasta su fin.

Con todo, esta fase se divide, al igual que la anterior, en tres etapas, una de diseño, otra de implementación, y una última de despliegue y mantenimiento. Cada una de ellas será detallada en profundidad en los siguientes apartados.

#### **6.4.1 Diseño**

Tras haber dejado atrás la etapa de propuesta de la solución, en la cual ya se llevó a cabo una labor de diseño muy importante, y en la que incluso se comentaba que el equipo de desarrollo podía adelantar trabajo del estudio detallado, podría pensarse que el trabajo de diseño ya está hecho y que, por tanto, no tiene sentido dedicarle otra etapa más. Pero no debe olvidarse que en aquella etapa no se realizó un diseño completo, sino que se limitó a un diseño general, plasmando el equipo sus ideas en escenarios, y profundizando sólo lo necesario para asegurar los pilares de dicho diseño. Y al profundizar se estaba, en realidad, adelantando trabajo de esta etapa que ahora se presenta, siguiendo la

estructura y utilizando las herramientas que se describen aquí, estructura y herramientas que servirán para obtener, ahora sí, un diseño completo.

Como ya se comentó al principio de esta descripción de la metodología TRES-D, el diseño que ahora nos ocupa se divide en horizontal en **dos niveles**, por una parte un primer diseño alejado de la implementación, y por otra un segundo diseño ligado a la propia implementación, mediando entre ambas la decisión de qué software y hardware se empleará en dicha implementación. Esta división responde a la tendencia que domina el desarrollo actual de interfaces de usuario y que, pensando en el desarrollo de una misma aplicación hacia múltiples y diferentes plataformas, persigue reutilizar aquella parte del diseño que no dependa de la plataforma final. De este modo, los diseñadores se centran en un primer momento en la función *–el qué–*, y sólo cuando se ha completado su definición, se aborda entonces la forma *–el cómo–*.

Sin embargo, hablar de forma en el caso de interfaces como las tridimensionales puede resultar confuso. De hecho, desde un principio se empezará a detallar la forma de aquellos objetos que tengan una apariencia, geometría y estructura propia. Lo que ocurrirá es que habrá que esperar a tener una definición completa de la función para dar la forma definitiva de esos objetos, con lo que se consigue que el diseño en la metodología TRES-D se mantenga fiel al principio de que la forma sigue a la función, y que se vio en la metodología de [Foley, 1996].

De este diseño también se comentó que no sólo se dividía en horizontal, sino que en vertical se distinguían **dos líneas de trabajo paralelas** pero relacionadas entre sí, una dirigida hacia las tareas, la interacción, y otra hacia los objetos, el contenido. Estas dos líneas de trabajo van más allá de esta etapa, continuando así también en la de implementación. Estas líneas responden, además, a dos de los tres modelos sobre los que se asienta la metodología TRES-D, siendo el tercero el modelo del espacio. La labor de los diseñadores será obtener una descripción de la interfaz en cada nivel de acuerdo a cada modelo, y desde diferentes vistas de diseño, estáticas o dinámicas, individuales o de conjunto, con ayudas y herramientas como las indicadas, por ejemplo, en [Molina, 2004].

Por último, en esta etapa se tienen muy en cuenta esos **compromisos** –en inglés, “trade-offs”– que traían de cabeza a los desarrolladores, especialmente de mundos virtuales, buscando ese equilibrio entre los requisitos visuales –detalle, realismo–, los de interacción –comportamiento, naturalidad–, la tecnología –rendimiento, disponibilidad– y la usabilidad.

### 6.4.1.1 Diseño I

Este primer nivel de la etapa de diseño se ha dado en llamar “diseño I” con el propósito de huir de otros nombres usados en otras metodologías, tales como “diseño abstracto” o “modelado conceptual”. Y, si bien podrán establecerse similitudes entre lo que aquí se presenta y lo que en esas otras metodologías se describe bajo esos nombres, la correspondencia no es total. Más aún, es una mezcla de ambos diseños, ya que aquí se conjuga el diseño abstracto de las interfaces de usuario con los modelos conceptuales de los artistas.

En cualquier caso, el punto de vista que se tiene a este nivel del diseño es uno alejado de la implementación, que ignora el software y hardware que se empleará en ella, y que se centra en **el qué** –el significado del mensaje, su contenido- y no en el cómo –la forma en que se transmite el mensaje, el continente-. En otras palabras, aquí el diseñador de interfaces de usuario debe abstraerse de los posibles dispositivos de entrada o de salida y de artificios como widgets u otros controles, y el artista debe despreocuparse de cuestiones de más bajo nivel tales como la representación en polígonos y el número de estos.

Los diseñadores parten entonces de los escenarios confeccionados en la fase anterior y desarrollan su trabajo en dos líneas paralelas, como se ha mencionado ya para la presente etapa y por ende también a este nivel, con el propósito de especificar qué podrá hacer el usuario, en una, y sobre qué, en la otra. Más concretamente, el resultado de la primera será un conjunto de operaciones que el usuario podrá realizar sobre los objetos a través de la interfaz, y el de la segunda será una descripción de esos y otros objetos de la interfaz y la aplicación.

Como también se ha comentado, puede empezarse por cualquiera de las dos líneas de trabajo, incluso seguir ambas en paralelo, si bien tarde o temprano ambas se entrecruzan. En particular las operaciones que podrá realizar el usuario definen la función de los objetos, y viceversa, la función de los objetos dependerá de las operaciones que el usuario deba realizar sobre ellos. Además está la dimensión espacial de tareas y objetos, que deberá plasmarse en bocetos y mapas que servirán de planos maestros en este desarrollo.

#### 6.4.1.1.1 *Las tareas*

Empezando, por ejemplo, por las tareas, ya se ha citado en otras ocasiones a Eastgate, quien apuntaba, al hablar de los entornos virtuales, que las tareas en estos solía reducirse a una simple lista de operaciones [Eastgate, 2001], que es el

resultado que se persigue aquí. De no ser así, a partir de los escenarios dados pueden seguirse técnicas como las descritas en la etapa de análisis para describir las tareas que en ellos se han plasmado, pero teniendo muy presente que las tareas que ahora se analizan son las que se proponen en la solución, no las que se encontraron en el problema.

Así, se puede recurrir al *análisis jerárquico de tareas* (HTA), o el empleo de *diagramas de secuencia* de UML. Con estos últimos se captura mejor la dimensión temporal de las tareas y operaciones, si bien algunas notaciones de jerarquías de tareas también permiten plasmar esa ordenación temporal –p. ej. Jackson Structured Design (JSD) [Murray, 1999] o ConcurTaskTrees (CTT) [Paternò, 1999]-. También hay que tener en cuenta su dimensión espacial, localizando actividades en zonas particulares del espacio, así como los objetos, herramientas y agentes involucrados. Precisamente, con respecto a quién ejecuta la tarea –los agentes-, en este análisis debe distinguirse entre aquellas que realiza el sistema, y que deberán traducirse en objetos que encapsulen dichos **procesos**, y aquellas que lleva a cabo el usuario, las llamadas **tareas de interacción** o ITas, según el modelo propuesto en esta Tesis –recuérdese apartado 5.3-.

Siguiendo ese mismo modelo, el diálogo a alto nivel entre el usuario y los objetos se obtiene descomponiendo las tareas de interacción en otras más simples hasta llegar al nivel de **operación sobre un objeto**. Para cada operación aislada deberá entonces documentarse qué condiciones deben darse para su ejecución –precondiciones-, qué **unidades de información** debe proporcionar el usuario para llevarla a cabo –parámetros-, cómo reacciona el objeto ante la acción del usuario –comportamiento-, y las condiciones de error que pueden darse. Todo ello será muy útil en el siguiente nivel de diseño. Para documentarlas se pueden recurrir a *plantillas*, pero también a *diagramas de flujo* y *diagramas de estado* que especifiquen, como veremos a continuación, la función y el comportamiento del objeto.

#### 6.4.1.1.2 Los objetos

En el anterior análisis de tareas se habrán encontrado objetos, sí, pero no tienen por qué ser todos los que compongan la interfaz, por ello tanto si se ha realizado ya ese análisis como si no, la línea de trabajo dirigida hacia los objetos comienza también revisando los escenarios en busca de esos objetos.

Los objetos encontrados serán clasificados de acuerdo al modelo también propuesto en esta Tesis –recuérdese apartado 5.2.1-, y detallados también de acuerdo al mismo. Así, para cada objeto habrá que especificar su **función** y **comportamiento**, por ejemplo con los diagramas antes citados, aunque la

necesidad de este punto dependerá del tipo de objeto del que se trate. También dependerá del tipo los sub-modelos que deben completarse si bien, como entonces, aquí nos fijaremos en uno en particular, el **sub-modelo gráfico**.

Siguiendo entonces ese sub-modelo, habrá que especificar la estructura, geometría y apariencia, así como la percepción que el usuario tendrá del objeto. Este es trabajo principalmente de artistas, creando *bocetos*, *tiras de viñetas*, *modelos de arcilla*, e incluso *modelos de alta resolución* creados con herramientas informáticas de diseño 3D, todo ello producto de su propia creación a partir de las descripciones dadas en los escenarios o como réplicas de objetos reales a partir de material de referencia, bien recopilado en la fase anterior o bien recogido ahora en esta etapa.

En cuanto a la estructura que muestran los objetos a la vista, en particular las relaciones espaciales entre las geometrías que los forman, esta puede describirse con *diagramas E-R*, como se proponía en [McIntosh, 2000], o con *diagramas de objetos* de UML, tal y como se usaba en [Huda, URL2].

E independientemente de su apariencia, la percepción que se considere que el usuario tendrá del objeto bien puede plasmarse recurriendo al Modelo de Oportunidades de Percepción propuesto por Fencott e Isdale [Fencott, 2001].

Pero volviendo a esa representación en diagramas de objetos, usada para especificar las sub-partes que componen el objeto, esta también se puede utilizar para dar, junto a los *diagramas de clases* de UML, una visión de conjunto de los objetos y clases identificadas. Por esta coincidencia, es previsible que al diseñador le quede la duda acerca de qué debe considerar como objeto y qué como parte de él, aunque en este caso la experiencia será su mejor consejera. Aún y todo, esos diagramas de objetos y de clases no son tan apropiados para expresar la dimensión espacial de los objetos, para lo cual lo mejor es recurrir a *mapas* y *planos*, de objetos individuales, de conjuntos, o de la totalidad de ellos.

#### 6.4.1.1.3 El espacio

Hablando de mapas y planos, se ha visto que tanto en una línea como en otra se hace hincapié en la localización de tareas y objetos en el espacio, siguiendo en este caso el modelo del espacio también propuesto en este capítulo –recuérdese apartado 5.4-. Entonces se distinguió entre tres espacios, el digital, el virtual y el real, si bien a este nivel del diseño sólo se abordan los dos últimos.

Así, los mapas y planos pueden hacer referencia a un **espacio real**, cuando se describen objetos tangibles, o a un **espacio virtual**, cuando esos objetos son

fruto de la imaginación de artistas y diseñadores. A la hora de crear nuevos espacios mucho puede aprenderse de la experiencia de otros arquitectos, no de interfaces de usuario, sino de edificios, como por ejemplo la de Alexander condensada en sus famosos patrones [Alexander, 1977]. En cualquier caso, la descripción espacial de los objetos puede ser aumentada, siguiendo el modelo propuesto, con la introducción del concepto de zona o lugar, el de punto de vista o de observación, y la relación entre ellos así como con las tareas y operaciones.

Eso sí, lo que aún no se incluye en esos mapas y planos son los dispositivos que, a modo de interfaz, median entre el espacio real y el virtual, dado que el uso de unos u otros dispositivos es una decisión ligada a la implementación y, por tanto, no corresponde a este nivel de abstracción.

#### 6.4.1.1.4 Validación

El hecho de tener una visión alejada de la implementación no impide, sin embargo, que a este nivel ya se pueda realizar una validación del diseño realizado hasta ahora, incluso fabricar algún prototipo de baja fidelidad, por ejemplo en papel. En este sentido, los diseñadores y artistas colaboran con los expertos en el dominio y los futuros usuarios, como ya hicieron en la fase anterior, haciéndoles partícipes del proceso de desarrollo. En particular, las reuniones con unos y los tests con otros tienen por fin comprobar que los conceptos, el qué, está correctamente plasmado en el modelo del diseñador.

#### 6.4.1.2 Diseño II

Tras completar el diseño desde una perspectiva ajena a la implementación, el trabajo en esta etapa continúa ahora concretando ese diseño para un software y hardware particular, lo que se ha dado en llamar “diseño II”, siendo la razón más simple para ello el continuar con la numeración dada para el anterior nivel de diseño. Aquí los diseñadores se centran entonces en la forma en que se transmite el mensaje, **el cómo**, especificando en particular cómo el usuario ejecuta las operaciones y cómo se presentan los objetos al mismo. Con ello, los diseñadores estarán especificando la interfaz de usuario, una interfaz que será del tipo tridimensional, tal y como se ha supuesto desde el principio de esta fase, lo cual, de acuerdo a la definición que se ha dado en esta Tesis, significa que o bien las acciones del usuario tienen lugar en las tres dimensiones del espacio, o bien los objetos se presentan en tres dimensiones, o ambas cosas. Es decir, será una interfaz en la que el espacio 3D jugará un papel muy importante, y ello requerirá una adecuada composición del espacio real con el digital y virtual, actuando los dispositivos físicos como ventanas entre esos mundos.

De nuevo, el trabajo continúa a este nivel siguiendo las dos líneas de trabajo destacadas en el anterior, una dirigida hacia las tareas y la interacción, y otra hacia los objetos y el contenido. Y, de nuevo otra vez, se destacarán las relaciones de ambas líneas entre sí, y de estas con la dimensión espacial. Así, por una parte, los diseñadores tomarán la descomposición de las tareas en operaciones sobre los objetos y en unidades de información para concretar todo ello ahora en acciones reales y virtuales, controles, avatar y hardware, esto es, en técnicas de interacción o ITes. Por otra parte, tomarán la descripción de los objetos y concretarán su función y comportamiento, su estructura y nivel de detalle, y planificarán el reparto de trabajo entre los creadores de modelos 3D. Por último, completarán los mapas del espacio real y virtual, a los cuales sumarán ahora los del espacio digital, especificando para todos ellos no sólo el orden espacial sino también la secuencia temporal, y resaltando ahora los dispositivos físicos que comunican unos espacios con otros. Todo esto siguiendo, como es de suponer, los modelos de los objetos, de la interacción y del espacio ya propuestos.

Como se ha destacado anteriormente, es precisamente la **elección del software y hardware** –especialmente este último- lo que marca la diferencia entre el anterior nivel de diseño y este, y en esa elección pesan muchos factores. En [Sutcliffe, 2003], por ejemplo, el debate se centra –para una aplicación de Realidad Virtual- entre un equipo de escritorio o uno inmersivo. La situación ideal sería que, a partir de los requisitos de interacción y de presentación, el equipo de desarrollo tuviera total libertad para escoger el software y el hardware que mejor se adapta a dichos requisitos, lo cual, como ya se comentaba en [Foley, 1996] no siempre es posible. En el peor de los casos, la elección vendrá dada por los requisitos de la aplicación, por lo que el trabajo de los diseñadores se centrará en refinar el diseño hasta hacerlo practicable con dicha imposición. En general, se seguirá un algoritmo similar al que se exponía en [Smith, 2001], donde dado un diseño se pregunta si es realizable con un cierto toolkit, si no entonces se cambia de toolkit, y si no se puede cambiar de toolkit entonces se cambia el diseño. Aún y todo, llegados a esta fase, ese hardware y software ya habrá sido elegido, pues en la etapa de propuesta de la solución se debió avanzar por esta y siguientes etapas lo suficiente como para tomar dicha decisión e incluirla en la propia propuesta.

#### 6.4.1.2.1 *Las tareas*

Empezando, como ya se hizo en el anterior nivel, por la línea de trabajo que se centra en las tareas y la interacción, el objetivo aquí es, tal y como se ha adelantado, concretar las tareas en un conjunto de **técnicas de interacción** o

ITes, y con ellas los controles, la representación para el “yo” virtual del usuario, el hardware de entrada y de salida, y la correspondencia entre las acciones reales del usuario y las virtuales. La elección de una técnica de interacción, aunque motivada por los requisitos de interacción de una operación, debe tener también en cuenta el resto de operaciones. Esto es porque en el diálogo interactivo unas técnicas sucederán a otras. Además, como en cualquier interfaz de usuario, los diseñadores deben ser coherentes en sus elecciones, pensando también en el conjunto del escenario y de la aplicación. Así, estos diseñadores pueden revisar los árboles de tareas del nivel anterior empezando por las hojas, atendiendo a las operaciones y las unidades de información que estas requieren.

Para cada **operación**, deben recordarse los requisitos de la aplicación – naturalidad en la interacción, realismo en la presentación- y del usuario – habilidades motoras, modalidades preferidas- y conjugarse con los propios de esa operación.

En primer lugar, el diseñador repasa en las precondiciones, en particular qué necesita el usuario ver para llevar a cabo esa operación y cómo –modalidad, hardware de salida-. Por ejemplo, si el objeto no es abstracto, sino que tiene una forma propia tridimensional que presentar al usuario, esta puede mostrarse a través de un puerto de vista en el dispositivo de presentación; entonces el diseñador debe añadir ese puerto como un objeto más de la interfaz –un widget- y pensar cuánto espacio ocupará ese puerto de vista en el dispositivo, y ajustarlo a las dimensiones de este o buscar un dispositivo mayor.

A continuación, el diseñador se centra en cada una de las **unidades de información** que el usuario debe transmitir a la máquina y busca la técnica de interacción más apropiada para cada una de ellas –varias BITE’s- o para todas en conjunto –una CITE-, bien a partir experiencias anteriores o bien ingeniando otras nuevas. En el primer caso, en el siguiente apartado se presentarán un par de herramientas para guiar al diseñador en la elección de técnicas para dos tareas universales, siendo una de ellas la selección de objetos y la otra la introducción de texto. En el segundo caso, el diseñador debe decidir cuál será el estilo de interacción más apropiado –debatiendo, p. ej., entre directo e indirecto- y a partir de ahí cuáles serán las acciones reales del usuario, cuáles las virtuales, y cómo hará corresponder unas con otras a través de los dispositivos de entrada, de la representación del avatar, y de los controles adicionales que necesite; tanto el avatar como los controles son también objetos nuevos a añadir a la interfaz, y si el diseñador no puede escoger el dispositivo de entrada que mejor se ajusta a las características de la técnica, tendrá entonces que adaptar esta a aquellos de los que disponga.

Para terminar con la operación, hay que sumar la realimentación con la que se informa al usuario del proceso en ejecución y su resultado final, en unos casos

ligada a la propia técnica o técnicas, y en otros al propio objeto sobre el que se realiza la operación.

Para especificar todo esto, la metodología TRES-D tampoco impone unas herramientas concretas, dando a los diseñadores la libertad de elegir la que mejor se ajuste en cada momento, desde descripciones más informales utilizando *texto*, *viñetas* e incluso *videos*, al uso de otras herramientas más refinadas, como PMIW y su editor VRED [Jacob, 1999], flownets y el toolset MARIGOLD [Willans, 2000], o InTml y su editor visual [Figuerola, 2002], siendo cualquiera de ellas igualmente válidas para especificar el comportamiento de las técnicas y producir un prototipo de rápida evaluación.

Siguiendo el diálogo, los diseñadores deben también **componer unas técnicas con otras**, subiendo a través de las ramas del árbol de tareas y comprobando no sólo que no haya problemas que impidan concatenar en el tiempo una técnica con la siguiente, sino que el conjunto resulte coherente.

Así, desde el punto de vista de la entrada, se debe observar qué comparte y qué no una técnica con la siguiente, por ejemplo si el dispositivo de entrada es el mismo o no, y siendo el mismo si es necesario conmutar el modo de funcionamiento como cuando faltan teclas para asignar funciones.

Desde el punto de vista de la presentación, los diseñadores deben **ordenar el espacio**, no sólo la distribución de los objetos dentro de sus límites, si no también la sucesión en el tiempo de unos objetos sobre otros dentro del mismo espacio de presentación. En el caso de las interfaces convencionales, esto último es lo que hemos llamado navegación inter-ventana. Si hablamos de mundos virtuales, un mismo puerto de vista puede servir de ventana a todos los objetos de un mismo mundo, por lo que un cambio de atención de uno a otro se traduce en un cambio del punto de vista o navegación intra-mundo. Incluso, el mismo puerto de vista puede servir para mostrar, alternativamente, diferentes mundos, lo que se corresponde con la navegación inter-mundos.

#### 6.4.1.2.2 *Los objetos*

A lo largo de esta línea de trabajo dirigida a las tareas y la interacción se ha visto cómo el diseño de las operaciones sobre los objetos, en la forma de técnicas de interacción, se relaciona forzosamente con el diseño a este nivel de objetos ya identificados en el anterior, al tiempo que introduce nuevos objetos cuyo diseño también debe ser abordado aquí. A ellos hay que sumar el resto de objetos no interactivos, pero que son igualmente necesarios en la interfaz y la aplicación. Todos ellos son el centro de atención de la segunda línea de trabajo que se

desarrolla en paralelo a lo largo de esta etapa de diseño y, más tarde, de la implementación. En este momento, con esa próxima implementación en mente, los diseñadores completan la descripción de los objetos en cada una de sus facetas, tanto su función y comportamiento como el resto de sub-modelos.

Destacando una vez más el **sub-modelo gráfico**, los diseñadores deben especificar la presentación final que deben tener los objetos, con atención al nivel de detalle de aquellos que se plasmarán en modelos 3D formados a partir de polígonos, colores y texturas.

Así, conocidos los límites del hardware de aceleración de gráficos 3D, el **nivel de detalle** de unos y otros deberá equilibrarse para poder obtener un buen rendimiento en la generación de imágenes, esto es, una velocidad de síntesis acorde a una aplicación interactiva. Sin embargo, con independencia del hardware que se escoja o se disponga, es muy importante recordar que el nivel de detalle debe ajustarse también al grado de funcionalidad y comportamiento del objeto, para evitar falsas pistas al usuario y, con ello, problemas de usabilidad. De esta forma, el diseñador debe dar con el equilibrio entre realismo, rendimiento y usabilidad.

Junto a la geometría y apariencia del objeto también debe completarse la especificación de su estructura, si bien a este nivel del diseño esa estructura puede ser muy diferente a la que se dio en el diseño I. La razón para ello es que la estructura que entonces se dio correspondía a la que los objetos muestran a la vista, y esta puede no ser la misma que requiera su implementación, por ejemplo en forma de grafo de escena. Este hecho ya fue advertido en [Huda, URL1], y es una diferencia importante que debe tenerse muy presente. Así, aunque se sigan utilizando *diagramas de objetos* de UML, la estructura que reflejen será, en su caso, la de ese grafo de escena, o mejor dicho de una parte del mismo, ya que el grafo de escena se construirá componiendo las estructuras de todos los objetos que forman dicha escena.

Y completando el sub-modelo, no debe olvidarse la percepción que se espera que el usuario tenga del objeto, continuando el trabajo ya iniciado en el anterior nivel.

Con el diseño de todos los objetos convenientemente especificado, el equipo realiza el reparto de modelos 3D a realizar entre los creadores de contenido, quienes abordarán su desarrollo en la fase de implementación, pero que en este punto se preocupan en recoger aquel material de referencia que les pueda hacer falta para ese cometido y que no haya sido recogido hasta ahora.

#### 6.4.1.2.3 *El espacio*

Una vez más, en ambas líneas de trabajo se ha podido comprobar la relación tanto de tareas como de objetos con el espacio, completándose también a este nivel la especificación de la interfaz desde el modelo del espacio, aunque en esta ocasión se aborda tanto el digital como el virtual y el real.

Así, empezando por el **espacio digital**, se debe poner orden en el espacio de cada ventana, distribuyendo los widgets en cada una de la forma más conveniente, agrupando y dividiendo según convenga, p. ej. de acuerdo al espacio disponible en el dispositivo de presentación. Esta descripción puede llevarse a cabo, como se vio en las metodologías orientadas hacia interfaces convencionales, utilizando diagramas que muestren la disposición de los widgets dentro de cada ventana, y árboles o grafos que representen la navegación de unas a otras o navegación inter-ventana. Esos diagramas pueden llegar a ser incluso prototipos de cada ventana, creados por ejemplo con una herramienta visual, y esos árboles o grafos pueden ilustrar cada ventana con una instantánea de su correspondiente prototipo.

Siguiendo con el **espacio virtual**, aquí se completan los planos y mapas ya creados en el anterior nivel, junto con la información de lugares, puntos de vista y portales de comunicación entre mundos. Si existen rutas predefinidas, estas pueden describirse en esos planos y mapas conectando los correspondientes puntos de vista para la navegación intra-mundo. Por su parte, la conexión entre portales y puntos de vista en mundos distintos describe la navegación inter-mundos.

De forma similar, también se completan los planos y mapas del **espacio real**, pero con las diferencias destacadas ya en el modelo del espacio, como la ausencia de portales que lleven a otros mundos, ya que el espacio real es único.

Todos esos diagramas, planos y mapas describen el orden espacial y temporal de los objetos de la interfaz y de la aplicación, la integración del espacio digital y virtual –p. ej. del mundo 3D con el resto de la interfaz 2D-, y la de estos dos con el real, a través de la descripción espacial de los dispositivos de entrada y salida como **superficies y volúmenes de interacción**. De estas últimas descripciones se destacarán las necesidades de espacio real que exige la interfaz bajo desarrollo, necesidades que deberán ser satisfechas en el momento de poner en marcha y mantener la solución en su ubicación final y que, para evitar que no sea así, deben haber sido previstas en la propuesta de la solución.

#### 6.4.1.2.4 Validación

También como en el anterior nivel, los diseñadores crean prototipos de la interfaz que puedan ser evaluados por expertos en el dominio o sometidos a test con usuarios reales. Pero, a diferencia de ese anterior nivel, los diseñadores reciben ahora la ayuda de los programadores y los creadores de modelos 3D para este propósito, adelantándose en este caso a la propia implementación para corregir posibles errores en el diseño, no sólo a través de la opinión de los expertos o los usuarios, sino atendiendo también a los comentarios técnicos del equipo de desarrollo que llevará a cabo la implementación final.

#### 6.4.1.3 Herramientas ejemplares para guiar al diseñador

El modelo de proceso de la metodología TRES-D, como tal, identifica las actividades que conlleva el desarrollo, en este caso, de una interfaz de usuario 3D, les da un orden y una estructura, y las asigna a los diferentes roles que intervienen en dicho proceso. Además de eso, también proporciona nuevas herramientas o recomienda el uso de otras ya existentes para la realización de dichas actividades. Ejemplo de ello son las diferentes notaciones citadas hasta este momento, y que sirven para documentar diferentes aspectos del trabajo. Sin embargo, la etapa que nos ocupa ahora, el diseño, es eminentemente creativa, y en ella la labor del diseñador es la de tomar una decisión tras otra. Y aunque esa estructura le indica sobre qué tiene que decidir, y esas notaciones le permiten plasmar sus decisiones, esas decisiones las tomará él mismo a partir de una mezcla de experiencia e ingenio.

En el caso de una interfaz convencional, como ya se ha apuntado en otras ocasiones, la gran experiencia que han documentado tantos y tantos autores facilita esta labor. En el caso de las interfaces de usuario 3D, si bien es cierto que se cuenta con una experiencia de muchos años, esta no les ha llevado a alcanzar, como también se ha insistido, la madurez deseada. Ya sea poca o mucha la experiencia, esta debe convertirse en conocimiento útil que ayude al diseñador a tomar sus decisiones, y para ello debe llegar al mismo de la mejor forma posible. Pero disperso en artículos científicos, y con la formalidad propia de estos, no parece el mejor vehículo. Es deseable contar con títulos que reúnan y ordenen ese conocimiento destacando las prácticas más exitosas, libros de los cuales hay pocos, así como herramientas que lo pongan a disposición del diseñador de forma simple y directa. Un ejemplo es la propia definición de sistema de Realidad Virtual, sobre la que uno se puede perder en las muchas explicaciones dadas por diferentes autores, pero para decidir en la práctica qué es y qué no es un sistema así parece mejor recurrir a un árbol de decisión tal como el que se propone en [Marsh, 1998].

Dos autores preocupados por hacer guías accesibles al diseñador son Kaur y Eastgate, de cuyo trabajo en este campo ya se habló en el apartado 2.7.10. En particular, Kaur propuso una herramienta hipertexto para presentar las guías de usabilidad al desarrollador, creando con el lenguaje HTML una primera versión de demostración en la que incluyó las guías correspondientes al diseño de componentes y de la interacción [Kaur, 1998]. En el caso de Eastgate, este autor propuso tres herramientas en tres formatos diferentes: un diagrama de flujo para decidir cómo representar los objetos interactivos, un checklist para la reducción del número de objetos y el control de su nivel de detalle, y una tabla para guiar en el tema de la navegación [Eastgate, 2001]. En lo que se refiere a la metodología TRES-D, cualquiera de estas herramientas tiene cabida en esta etapa de diseño que ahora nos ocupa, guiando a los diseñadores tanto en cuestiones de interacción como de representación de los objetos, y la relación entre ambos aspectos.

Siguiendo esos ejemplos, en esta Tesis se propone sumar dos herramientas más a las anteriores, abordando cuestiones que aquellas no resuelven. Y es que las herramientas de Kaur y Eastgate se dirigen hacia entornos virtuales, y de escritorio, principalmente. Esto significa que se preocupan más por la apariencia visual de los objetos, y no tanto de la interacción con dispositivos no convencionales, a excepción de la tabla que Eastgate dedica a la navegación. Precisamente, la navegación es una de las tareas consideradas universales, pero hay más –recuérdese el apartado 2.6.2-. Para cubrir ese vacío, las dos nuevas herramientas que se proponen aquí tienen por fin guiar al diseñador en la elección de las técnicas de interacción más apropiadas, según el caso, para las tareas de selección –también manipulación- y de entrada simbólica – introducción de texto, para ser más concretos-. En ambos casos, el formato que se consideró más apropiado fue el de un diagrama de flujo o **árbol de decisión**.

La primera de esas herramientas fue publicada en [García, 2005a] y [García, 2005b], y se planteó con el propósito de resumir las conclusiones del estudio que en esos artículos se describe y que se centró en la evaluación de siete **técnicas para la selección y manipulación de objetos** en entornos virtuales inmersivos. En la evaluación tomaron parte 10 participantes, realizando tareas de selección de objetos de diferentes tamaños a diferentes distancias. De los datos obtenidos se realizó un análisis estadístico –cuyos detalles se dan en las citadas publicaciones- que confirmó lo que muchos otros autores ya habían afirmado, y es que ninguna de las técnicas era mejor en todos los casos. Ahora bien, dado un caso, sí puede recomendarse una u otra, y es aquí donde entra en juego la herramienta propuesta, planteando al diseñador una serie de cuestiones hasta dar con la técnica que mejor se adapta a su caso.

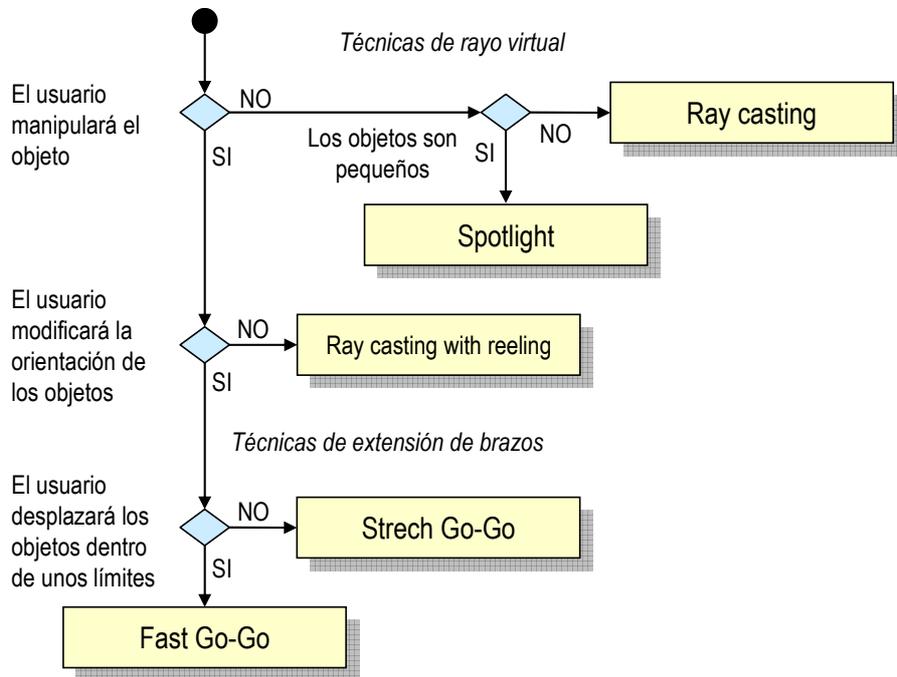
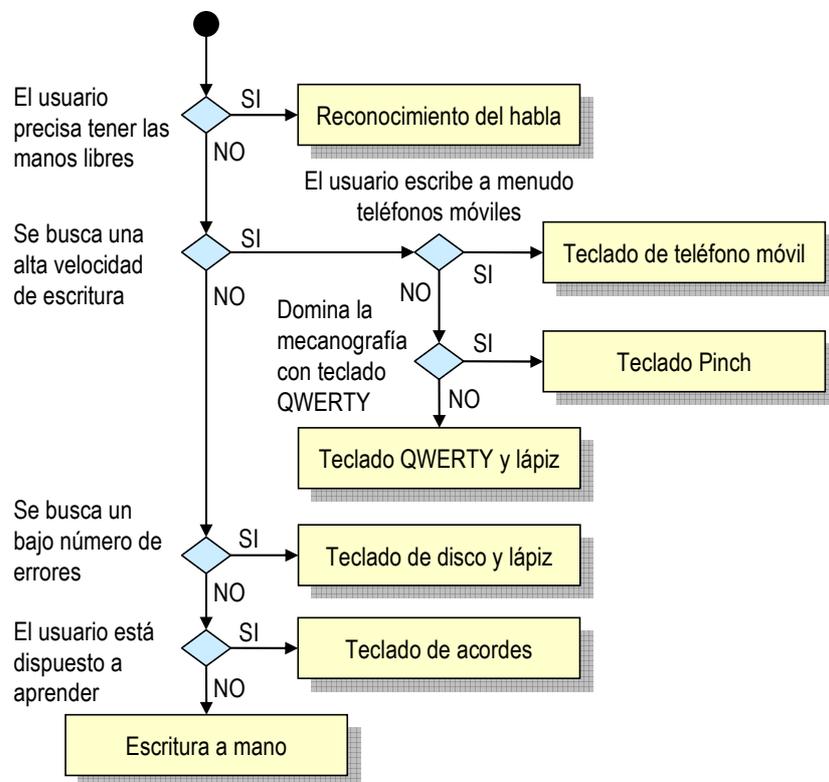


Figura 6.2 Herramienta para guiar en la elección de una técnica de selección y manipulación

Esta primera herramienta se ilustra en la Figura 6.2. La primera decisión a tomar por el diseñador de interfaces de la aplicación será la de si existirá manipulación dentro del sistema. En caso negativo, basándonos en la evaluación realizada, las técnicas de rayo virtual demuestran ser más efectivas que las de extensión de brazos. Lo siguiente será comprobar si hay objetos pequeños, y por tanto difíciles de seleccionar, en el mundo virtual. En caso afirmativo, *Spotlight* se muestra como la técnica más adecuada, debido al mayor volumen de colisión, comparado con *Ray casting*. En el caso de que exista la posibilidad de manipular objetos en el sistema, surge otra cuestión, y es si esa manipulación incluirá cambios en la orientación de los objetos. En caso negativo, una técnica como *Ray casting with reeling* será la adecuada, dada su facilidad para la selección y una manipulación relativamente sencilla. Si la manipulación incluye rotaciones, la elección deberá ser de una técnica de extensión de brazos, y en este punto cabe realizarse otra pregunta: ¿tendrá algún límite la distancia a la que se desplazarán los objetos dentro del entorno virtual que estamos desarrollando? En caso negativo, las técnicas de extensión de brazos basadas en la longitud del brazo real del usuario impedirían que este alcanzara cualquier punto del espacio virtual, por lo que la elección debería ser *Stretch Go-Go*, ya que permite alcanzar cualquier punto del espacio, pese a ser menos intuitiva.

La segunda de las herramientas fue publicada en [González, 2007], y también se planteó con el fin de resumir las conclusiones del estudio que este otro artículo describe, y que en este caso se fijaba en la evaluación de seis **técnicas para la introducción de texto** en entornos virtuales inmersivos. En la evaluación tomaron parte otros 10 participantes, cuya tarea era la de introducir diferentes fragmentos de texto con cada una de esas técnicas. Como en el anterior estudio, también se llevó a cabo un análisis estadístico, de nuevo confirmando que ninguna de las técnicas era la mejor en todos los casos. Los datos del estudio, eso sí, permitían recomendar una u otra técnica según el caso, para lo que el diseñador debe hacerse algunas preguntas.



**Figura 6.3** Herramienta para guiar en la elección de una técnica de introducción de texto

Esta segunda herramienta se ilustra en la Figura 6.3. Lo primero que debe preguntarse el diseñador es si el usuario necesitará tener sus manos libres para otras tareas. Si es así, la técnica que se recomienda aquí es el reconocimiento de voz, que si bien no se incluye en este estudio sí se hacía en el anterior. En caso contrario, cabe preguntarse si se precisa una alta velocidad de escritura. Si es así, el árbol lleva al diseñador a una rama en donde se encuentran las técnicas basadas en teclados QWERTY y de teléfono móvil, fundamentando la decisión

según las habilidades del usuario. La otra rama lleva al diseñador a preguntarse si es preciso un número bajo de errores. Si es así, la técnica recomendada será la del teclado de disco. En caso contrario, y dependiendo de la disposición del usuario a aprender, se recomienda bien el teclado de acordes o bien la escritura a mano.

Bien es cierto que las herramientas aquí propuestas tienen sus limitaciones a la hora de ser utilizadas en el diseño interfaces de usuario 3D en general, primero porque se dirigen a una aplicación concreta de estas, como son los entornos virtuales inmersivos, y segundo porque existen otras técnicas de interacción para las tareas dadas que no son contempladas en ellas. En cualquier caso, no se presentan como herramientas acabadas, sino como entes vivos que pueden evolucionar según vaya creciendo nuestra comprensión de este campo, y a las que se pueden añadir otras muchas para abarcar los vacíos del diseño que aún quedan por rellenar.

#### 6.4.2 Implementación

Con el diseño ya terminado, el proceso de desarrollo entra ahora en la etapa de implementación. En realidad, como ya ocurriera en la fase de diseño, el equipo no ha tenido que esperar a completar todo el trabajo anterior para poner un pie en la etapa siguiente. De hecho, se acaba de ver cómo, en esa misma fase de diseño, programadores y creadores de modelos 3D hacían ya acto de presencia colaborando con los diseñadores para realizar esas pruebas técnicas, necesarias en esa fase anterior, y realizar ya unos primeros prototipos, labor que no deja de ser un avance de la implementación que ahora se aborda en toda su extensión.

También aquí, como en el caso de los niveles de diseño I y II, cabe comentar un poco el porqué del nombre que se ha dado a esta fase. Puede parecer obvio pero, en realidad, tras el estudio de las diferentes metodologías que se recogen en el segundo capítulo, no lo es tanto. Así, si bien en los proyectos de ingeniería del software suele ser bastante común utilizar este término al referirse al trabajo de programación que hace de la aplicación una realidad, en el caso de los mundos virtuales no es así, prefiriéndose términos tales como creación, construcción, modelado o ensamblado. En este último campo, incluso encontramos el término implementación usado no como sinónimo de programación, sino más bien de despliegue –véase p. ej. [Eastgate, 2001]–. Aquí, al hablar de implementación nos estaremos refiriendo tanto al trabajo de programación como al de creación y ensamblado de los modelos 3D, como se verá a continuación. En este sentido, podría haberse optado por un término diferente a cualquiera de los otros, si bien el término implementación no sólo es el más usual en los proyectos software sino que, por extensión, también lo es en los proyectos de interfaces de las

aplicaciones, y la metodología TRES-D es, ante todo, una metodología de desarrollo de interfaces.

Como en la fase de diseño, aquí también se distingue entre el trabajo más orientado hacia las tareas y la interacción como el que se dirige más hacia los objetos y el contenido. Más específicamente, la primera línea se asociará con la labor de programación de las técnicas de interacción y de la función y el comportamiento de los objetos mientras que, en paralelo, la segunda línea se centrará más en el sub-modelo gráfico de los objetos. Esta separación responde más a los dos roles que asumirán el trabajo en esta fase, esto es, programadores y creadores de contenido digital, aunque la separación no es tan clara como en un principio podría pensarse. De hecho, una animación, como comportamiento de objeto, es más bien una tarea de los creadores de contenido, p. ej. animadores de modelos 3D, mientras que ciertos acabados visuales de los objetos suelen ser más bien labor de programadores, p. ej. ingenieros de software de gráficos. En cualquier caso, una vez más son dos líneas relacionadas entre sí, y en ellas unos deberán colaborar con los otros para llevar la implementación a buen puerto.

#### 6.4.2.1 Las tareas

Centrándonos en el trabajo de los **programadores**, estos se ocuparán entonces de escribir el código de la interfaz y de la aplicación, repartándose el trabajo y creando así componentes separados que luego ir integrando entre ellos y el trabajo del resto del equipo.

Tal y como se ha comentado ya, de los objetos programarán su función y comportamiento, contándose entre ellos los procesos invisibles para el usuario, aquellos que se ejecutan en la sombra, como por ejemplo el propio bucle de simulación. Pero, como también se ha dicho antes, el trabajo de los programadores puede extenderse hacia el sub-modelo gráfico, por ejemplo transformando la estructura de los objetos en un grafo de escena –p. ej. como se describe en [Huda, URL2]- o añadiendo efectos especiales a la apariencia de dichos objetos.

En cualquier caso, los programadores harán uso de entornos de desarrollo, librerías de programación y otros toolkits que puedan hacer más fácil su trabajo, y que en su mayoría ya fue previsto junto con el hardware en etapas anteriores. Un ejemplo de ello es la librería VUIToolkit, de la que ya se ha hablado algo y de la que se comentará algo más tras este apartado. Aún y todo, es muy fácil que los programadores no encuentren librerías de apoyo cuando se trata de implementar una interfaz de usuario poco convencional, y en ese caso tendrán que programar los necesarios componentes desde cero.

### 6.4.2.2 Los objetos

En cuanto a los creadores de contenido digital, aquí nos centraremos en el trabajo de los **creadores de modelos 3D**, los cuales darán la forma final a los objetos según la especificación dada por los artistas y diseñadores en la etapa anterior. Como con los programadores, también aquí se reparte el trabajo entre los diferentes creadores de contenido, reparto que ya se ha anticipado en el diseño, y se utilizan un conjunto de herramientas y librerías software, también previsto en etapas anteriores.

El proceso a seguir por estos creadores se basa en la práctica recogida en muchas de las metodologías estudiadas, modelando primero la geometría –p. ej. como una malla de polígonos-, añadiendo la apariencia después –p. ej. colores y texturas-, y ensamblando todo al final. Aquí caben, sin embargo, las dos estrategias que se mencionaban en el trabajo doctoral de Kaur, top-down y bottom-up, esto es, empezar con una estructura básica e ir añadiendo detalle de forma incremental o bien construir modelos por separado e integrarlos una vez terminados.

A esas tres actividades básicas cabe añadir la de reducción del nivel de detalle, y las de importación y exportación, siendo estas últimas necesarias cuando los modelos se toman de colecciones de objetos ya creados o cuando las herramientas de autor utilizadas empleen formatos de fichero diferentes.

Tras el ensamblado, se mejora cada mundo creado con imágenes de fondo, iluminación y sonido. Entonces se añaden las animaciones, interacciones y se pone en conjunto con el resto de la interfaz de usuario, integrándose este trabajo con el de los programadores.

### 6.4.2.3 Validación

El trabajo de unos y otros se coordina para, en unas primeras iteraciones, refinar la implementación, esto es, tanto el código como el nivel de detalle de los objetos, como para conseguir el rendimiento esperado para la aplicación interactiva. Una vez alcanzado ese rendimiento, puede pensarse entonces en mejorar aún más la experiencia del usuario, como se apuntaba en la metodología CLEVR [Seo, 2001]. Por supuesto, los técnicos contarán también con la colaboración de diseñadores, artistas y expertos en el dominio, para asegurar la correcta evolución hacia el producto final. Asimismo, se realizan nuevas pruebas con usuarios que permitan detectar, ya con la interfaz de usuario final, posibles

problemas en los que no se hubieran reparado en la fase de diseño ni hubieran aparecido en ninguno de los prototipos que en ella se hicieron.

Con todo el hardware y software integrado y dando forma a la interfaz de usuario 3D proyectada, se da por concluida su implementación pasando el testigo a la siguiente fase, camino ya de ser entregada al cliente.

#### 6.4.2.4 La librería VUIToolkit como ejemplo

Como se ha comentado ya, el trabajo de los programadores será más sencillo si estos cuentan con librerías de programación o toolkits que permitan establecer una correspondencia directa entre los elementos diseñados y los elementos que estas ofrecen. A lo largo de esta Tesis se ha destacado el problema de la falta de conjuntos estándar para el desarrollo de interfaces de usuario 3D –recuérdese el apartado 2.7-, si bien esto no significa que no se hayan hecho esfuerzos para cambiar el *status quo*. Así, en aquel apartado se citaron toolkits como el desarrollado en el marco del proyecto INQUISITIVE para la plataforma MAVERIK, el que creó G. Seidman llevando a la práctica las ideas de su grupo de trabajo en widgets para VRML, o los componentes para X3D del proyecto CONTIGRA. Todos ellos recopilan un conjunto de técnicas de interacción y/o controles, conceptos ambos relacionados pero no iguales, diferencias que se han reflejado en los modelos de objetos y de elementos de interacción propuestos en el presente trabajo.

Como también se ha adelantado, otro ejemplo de toolkit es la librería VUIToolkit [Molina, 2005a], una colección de controles desarrollada en dos versiones basadas en sendos lenguajes estándares, siendo uno de ellos VRML97 y el otro su tercera y más reciente versión, X3D, ambos dirigidos a la creación de mundos virtuales para la Web 3D. La característica más destacable de esta librería es que transforma los widgets de las interfaces convencionales en su verdadera representación 3D, cambiando los gráficos 2D –con profundidad simulada mediante superposiciones y sombras- por los gráficos 3D –cuyas primitivas tienen profundidad verdadera-, “virtualizando”–o “tresdeificando”, si es que tal palabra existe- dichas interfaces. Así, si el usuario pulsa un botón, por ejemplo, este se desplazará a lo largo de la tercera dimensión como cabría esperar, en lugar de mostrar una pobre secuencia de imágenes que simulan dicho desplazamiento al jugar con las sombras que en ellas se dibujan. El resultado son los controles que aquí se han distinguido como widgets 3D, y que sirven para crear interfaces de usuario 2D virtualizadas, según la clasificación dada para el continuo digital-virtual-real también propuesto. La Figura 6.4 ilustra el uso de esta librería para representar una interfaz convencional en un entorno 3D, virtualizándola sobre la pantalla de un ordenador portátil, también virtual.



Figura 6.4 Ordenador portátil virtual con widgets 3D de la librería VUIToolkit en su pantalla

Pero volviendo a la metodología TRES-D, y a la etapa de implementación que nos ocupa, es muy probable que el desarrollador deba crear su propio toolkit partiendo prácticamente de cero. En este caso, la librería VUIToolkit también puede servir para ilustrar cómo los modelos de objetos y de elementos de la interacción pueden plasmarse en la implementación, que es lo que se aborda en los siguientes párrafos y justifica la inclusión aquí de este apartado.

Cabe entonces empezar explicando que las dos versiones de la librería VUIToolkit comparten la misma arquitectura, basada en parte en los trabajos previos del antiguo grupo de trabajo en Widgets para VRML, así como en el mecanismo de prototipos o PROTO's que puede encontrarse tanto en el lenguaje VRML97 como en X3D y que puede asimilarse al de clase en un lenguaje orientado a objetos. Así, para cada widget 3D final se han creado dos prototipos, uno para VRML97 y otro para X3D. Los detalles de esa arquitectura y de cada widget 3D en particular pueden encontrarse en el Anexo A. Aquí, como se ha dicho, la explicación se centrará en aquellos relacionados con los meta-modelos de objetos y de elementos de la interacción, el lector puede acudir al citado anexo para conocer los demás.

Así, según el **meta-modelo de elementos de la interacción**, los widgets 3D que se incluyen en la librería VUIToolkit son controles, y como tales pueden

participar en una técnica de interacción como objetos que reciben acciones virtuales y las transforman en unidades de información, pero no son técnicas de interacción por sí mismos. En una técnica de interacción se incluyen otros elementos, como el hardware que opera el usuario, el avatar o representación virtual del usuario que, como una marioneta, este maneja a través de dicho hardware, y las acciones virtuales que ese avatar ejecuta en el espacio digital o virtual en respuesta a las acciones reales del propio usuario. Al basarse en los lenguajes VRML97 y X3D, estos otros elementos de la interacción dependerán del visor, usualmente un visor para PC en el que el dispositivo de entrada es un ratón y el avatar un cursor, aunque también los hay para entornos más inmersivos. En cualquier caso, las acciones virtuales se reducen entonces a tocar y arrastrar con el dispositivo apuntador. Para responder a dichas acciones, en la librería VUIToolkit se recurre al mecanismo de sensores de VRML97 y X3D, unos nodos especiales que se asocian con la geometría deseada y hacen que esta sea sensible a tales acciones.

En cuanto al **meta-modelo de objetos**, estos widgets 3D, de nuevo como controles que son, cuentan con una funcionalidad, un comportamiento y otros sub-modelos. La funcionalidad de estos controles la dan los mensajes que pueden recibir, esto es, las acciones virtuales a las que pueden responder, y en este sentido son los mismos sensores –normalmente nodos *TouchSensor*'s- de los que antes se ha hablado los que hacen de interfaz entre el usuario –mejor dicho, su representación virtual- y el control. A ello se suman los parámetros de la interfaz del prototipo que permiten cambiar, en ejecución, las propiedades del widget 3D. El comportamiento viene dado por su lógica interna, que en cada prototipo se materializa en un nodo *Script*, el cual mantiene el conjunto de variables de estado del widget 3D. A este nodo le llegan entonces los diferentes eventos que se producen, ejecuta las funciones que atienden a los mismos, y modifica el estado del widget 3D en consecuencia. Las animaciones también forman parte del comportamiento del widget, controladas por nodos *TimeSensor*'s que marcan el paso de las mismas. Al ser controles gráficos, un sub-modelo que no puede faltar es, cómo no, el gráfico, siendo la estructura del widget 3D su jerarquía de formas, la geometría son las primitivas gráficas como cajas, polígonos y líneas, y la apariencia son los colores y texturas que dan el acabado final a esa geometría. La percepción no tiene correspondencia con ningún elemento de los prototipos pues esa característica es subjetiva, representa la impresión que el usuario tendrá del control. Por último, y aunque no se corresponde con ningún elemento particular del modelo, un conjunto de rutas –sentencias *ROUTE*'s en VRML97 y X3D- conectan los sensores, la lógica y las formas.

### 6.4.3 Despliegue y mantenimiento

Terminada la aplicación con su interfaz de usuario 3D, no ha terminado con ello todo el trabajo pues estos sistemas, por las características de su tecnología, requieren si cabe mayor atención y mantenimiento que una aplicación convencional.

Así, por una parte, debe tenerse en cuenta el espacio que exigirá el hardware de entrada y salida en el lugar de **despliegue**, no sólo por el volumen que puedan ocupar estos dispositivos, sino por el volumen que pueda necesitarse para la operación de los mismos con la interfaz de usuario creada. No es la filosofía de esta metodología dejar nada al azar, y por ello las pertinentes comprobaciones deben haberse hecho con anterioridad a este punto. De hecho, los planos del espacio real que manejan los diseñadores deben reflejar precisamente la disposición de los equipos en este lugar. Lo que se realiza aquí es una tarea de instalación del hardware, llevada a cabo por técnicos de montaje, y un conjunto de pruebas técnicas por parte del equipo de desarrollo para comprobar la correcta ejecución de la aplicación y su interfaz, realizando en su caso ajustes menores. También puede ser este el momento para realizar una última evaluación con usuarios, con el fin de recoger datos finales de la solución creada.

Por otra parte, ese hardware poco convencional puede requerir también un **mantenimiento** más exigente que el de los productos de consumo. De nuevo, esta característica debe considerarse no ahora, sino antes, en el momento de escoger el hardware que integra la solución. Y en esa decisión, como se vio, influye no sólo el tipo de aplicación, sino también otros factores, como las características del usuario. Por ejemplo, de un público general que vaya a hacer un uso ocasional de la aplicación no puede esperarse el mismo trato del equipo que tendría, por el contrario, un profesional que lo utilice como herramienta de trabajo en su día a día. Pero con independencia del trato, en la elección de estos exóticos dispositivos debe hacerse una estimación de cuál será su vida útil, y prever qué hacer ante los posibles fallos que puedan aparecer en su mecanismo o el desgaste que con el uso puedan sufrir.

## 6.5 Resumen

El modelo de proceso de la metodología TRES-D se apoya en las ideas y en el vocabulario que proporcionan los tres meta-modelos presentados en el capítulo anterior, y da forma a un marco de trabajo en donde se unen diferentes prácticas

y herramientas para el desarrollo general de interfaces de usuario 3D, abarcando con todo ello los objetivos marcados para esta metodología.

Los principios que marcan este modelo de proceso son: primero, reducir en lo posible los riesgos que esperan al desarrollador en un campo aún inmaduro; segundo, distinguir entre diseño e implementación, y dentro del diseño distinguir entre un diseño independiente de la implementación y otro dependiente de esta; tercero, orientar el desarrollo tanto a tareas e interacción como a objetos y contenido, esto es, tanto a diseñadores de interfaces de usuario y programadores como a artistas y creadores de contenido digital; por último, involucrar no sólo a unos y otros profesionales, sino también al cliente, a los expertos en el dominio del problema y, por supuesto, al usuario.

El modelo de proceso se divide entonces en dos grandes fases, la primera destinada a comprender el problema y dar una propuesta de solución, y la segunda a desarrollar dicha propuesta hasta su despliegue final y posterior mantenimiento. Cada fase está compuesta, a su vez, de tres etapas cada una. Así, la primera fase, o de estudio previo, se compone de la presentación del problema, el análisis del problema, y la propuesta de solución. La segunda fase, o de estudio detallado, se compone de diseño –diseño I y diseño II-, implementación, y despliegue y mantenimiento. En cada etapa es posible avanzar trabajo de las que le siguen, con el fin de adelantarse a los problemas que puedan surgir. De hecho, en la fase de propuesta de solución se avanza en el trabajo de las etapas de la fase de estudio detallado. Entre ambas fases media la aprobación de esa solución por parte del cliente. Ya en la segunda fase, el diseño se realiza a dos niveles, uno alejado de la implementación –diseño I- y otro pegado a dicha implementación –diseño II-. La elección del software y hardware de la solución separa ambos niveles de diseño. Pero tanto el diseño I y II como la implementación se dividen en dos líneas de trabajo paralelas, una orientada hacia la interacción y otra hacia los objetos, siguiendo los dos modelos propuestos, desplazándose el peso hacia una u otra línea según el desarrollo particular, pero cruzándose entre sí, en especial por su relación con el tercer modelo, el del espacio. La última etapa, de despliegue y mantenimiento, no puede faltar en el desarrollo de este tipo de interfaces muchas veces caracterizadas por su hardware nada convencional.

Como marco de trabajo, además de dividir el proceso en fases y etapas, el modelo también recomienda diferentes herramientas para cada actividad que tiene lugar a lo largo del proceso de desarrollo. Algunas de esas herramientas permiten guiar a los diseñadores en sus decisiones, otras ayudan a plasmar las ideas de estos, y luego están las que sirven para hacer realidad dichos diseños. Respecto a las primeras, a las ya conocidas se han añadido dos más aquí, dos árboles de selección de técnicas de interacción, uno orientado hacia las técnicas de selección y manipulación de objetos, y otro hacia las técnicas de introducción

de texto. Sobre las segundas, se ha recurrido a herramientas ya existentes, algunas como los diagramas UML son bien conocidas por los ingenieros de software, otras como los bocetos, viñetas y planos forman parte de la práctica de creadores y animadores de modelos 3D. Y, de las últimas, ante la falta de conjuntos estándar de controles para interfaces de usuario 3D, se ha presentado la librería VUIToolkit de widgets 3D, y se han dado los detalles de su implementación destacando la relación de los elementos de su arquitectura con los meta-modelos de objetos y de elementos de la interacción.

Con todo, en la metodología TRES-D se ha tratado de aunar soluciones propias a lo mejor de las propuestas estudiadas en un proceso de creación iterativo e incremental que pueda adaptarse a la variable complejidad de los diferentes desarrollos de interfaces de usuario 3D, proceso que se ilustrará con casos prácticos en el capítulo siguiente.



# 7

## Casos de estudio

### 7.1 Introducción

En este capítulo se presentan varios casos de estudio con los que se pretende ilustrar el uso de la metodología TRES-D descrita en el anterior, así como de las contribuciones que esta metodología incluye y que también fueron descritas en aquel y anteriores capítulos. En total, se presentan aquí cuatro casos de estudio, tres de ellos dedicados al propio desarrollo estructurado de interfaces tridimensionales con dicha metodología, y un cuarto y último caso de estudio en el que la atención se centra más en el uso de la librería VUIToolkit, antes introducida pero descrita más en profundidad en el Anexo A.

La metodología TRES-D, en sus diferentes estados de madurez, ha venido aplicándose a diferentes proyectos docentes y de investigación en los que el autor de esta Tesis se ha implicado. No es extraño por ello que los tres primeros casos de estudio correspondan entonces a tres proyectos realizados bajo la supervisión del mismo. Todos ellos tuvieron por objeto la realización de una aplicación cuya interfaz de usuario era del tipo tridimensional, y además coincidían en el empleo de guantes de datos como dispositivos de entrada. Pero dejando a un lado estas similitudes, también había importantes diferencias, no porque los modelos de guantes eran diferentes, sino porque el fin de cada aplicación y su correspondiente interfaz también lo era. Esto último permite también ver estos proyectos como ejemplo de las diferentes y variadas aplicaciones que pueden tener las interfaces tridimensionales, como ya se apuntó en el capítulo dos. En cualquier caso, para la realización de estos proyectos se propuso un enfoque de desarrollo estructurado que finalmente se concretó en la propuesta aquí llamada TRES-D, aportando esta las ideas principales para el desarrollo de dichos proyectos, y estos a su vez una valiosa experiencia para mejorar y dar la forma a la propuesta final de esta Tesis. Estos tres casos de estudio han sido recogidos en [Molina, 2006b] y [Molina, 2006c].

El cuarto caso de estudio que se expone en este capítulo no se centra tanto en ese desarrollo estructurado, sino en la aplicación de otra de las contribuciones de esta Tesis, como es la librería VUIToolkit. Las posibilidades de la Realidad Virtual como herramienta para la visualización y prototipado, se unen en este caso de estudio a las características únicas de la librería VUIToolkit, para representar widgets de interfaces convencionales en un mundo virtual sobre

representaciones virtuales de las plataformas que soportan dichas interfaces. El resultado es un entorno para el prototipado rápido de interfaces de usuario distribuidas o DUI's (*Distributed User Interfaces*) que tiene al lenguaje UsiXML como tecnología de fondo. Este trabajo ha sido recogido en múltiples publicaciones, y puede verse en [Molina, 2006a], [Molina, 2006c] y [Vanderdonckt, 2007].

## 7.2 Desarrollo de interfaces basadas en guantes

Como aplicación práctica de la metodología propuesta, en los apartados siguientes se detalla el desarrollo de tres aplicaciones y sus correspondientes interfaces de usuario 3D, desde la exposición inicial del problema hasta su implementación y evaluación. Además de la metodología, las tres interfaces desarrolladas tienen en común el uso de un guante de datos como dispositivo de entrada, si bien los diferentes requisitos de cada proyecto llevaron a la elección de un modelo diferente en cada caso.

Desde la creación del primer modelo por T. Zimmerman y J. Lanier, el guante de datos ha sido considerado, junto con el visiocasco, uno de los dispositivos más representativos de la Realidad Virtual, y por ende de las interfaces de usuario 3D. Un guante de datos puede ser usado para crear una correspondencia directa entre la mano real del usuario y la mano virtual que se dibuja en el entorno. Sin embargo, la usual falta de realimentación táctil o de fuerzas impide que el usuario pueda tocar o coger los objetos como lo haría en el mundo real. El guante también puede ser usado como dispositivo de control, con el que el usuario transmite órdenes utilizando un lenguaje postural/gestual. El problema entonces es que el usuario debe aprender ese lenguaje, y para esa misma función podrían plantearse otros dispositivos menos intrusivos, como por ejemplo las varitas –en inglés, “wands”- con botones. A pesar de todo ello, los guantes de datos siguen siendo una opción muy a tener en cuenta en la creación de este tipo de interfaces de usuario 3D, y los tres proyectos que siguen también pretenden ser una prueba de ello.

Durante el desarrollo de cada proyecto, el alumno asumía los roles de arquitecto, diseñador, artista, programador y creador de contenido digital. Por su parte, el autor de esta Tesis asumió el rol de cliente, el de experto en el dominio, de usuario de la solución actual e incluso de futuro usuario de la solución que debía proponerse. A ello se sumaba el papel propio que, como docente e investigador, debía asumir para guiar al alumno en la aplicación de la metodología TRES-D. A este respecto, se cuidó que en cada desarrollo se abordaran las dos fases de las que consta dicha metodología, si bien alguna etapa no se llegó a realizar en toda su extensión –principalmente la de despliegue y mantenimiento-, lo cual puede

excusarse por el contexto docente en el que se realizaban. Distinto es que según el proyecto concreto se prestó mayor atención a unas actividades o a otras, bien a aquellas más centradas en las tareas y la interacción o en los objetos y el contenido, pues dos de los tres proyectos conllevaban la construcción de un entorno virtual –y con ello la creación de su contenido- y el tercero trataba la creación de una interfaz de usuario basada en gestos manuales –más preocupado por la interacción que debía soportar-. Los tres implicaban, sin embargo, el desarrollo de un interfaz de usuario 3D, y es esa atención a la diversidad uno de los aspectos cuidados en la metodología TRES-D y que aquí se trata de demostrar.

## 7.2.1 Tristéreo

### 7.2.1.1 Presentación del problema

Como se explicó en el capítulo anterior, la metodología TRES-D comienza con la fase de estudio previo, la cual se divide en tres etapas. En la primera etapa el cliente presenta el problema a una representación del equipo de desarrollo, entre los cuales destaca el arquitecto, quien extrae los requisitos más importantes y estima, en un primer momento, si el problema es practicable. De ser así, el arquitecto debe trazar un plan para las dos siguientes etapas, esto es, el análisis y la propuesta de la solución.

El problema que aquí se presenta se centra en un uso particular de las interfaces de usuario 3D que se nombraron en el capítulo dos: el entretenimiento y la diversión. Y es que una fracción muy importante de los videojuegos que se comercializan hoy en día son en 3D, su crecientes ventas suponen un impulso al mercado del hardware de gráficos 3D, y a su vez los avances en ese hardware impulsan la publicación de nuevos títulos más espectaculares, cerrando un círculo del que también se beneficia el campo de las interfaces de usuario 3D. Sin embargo, los gráficos 3D son tan sólo una parte de la interfaz, un canal más de salida, y a pesar del nombre no dejan de ser una proyección de un mundo 3D sobre un plano 2D.

Con este proyecto se pretendía entonces que el usuario pudiera sumergirse en un videojuego en el que la salida fueran gráficos 3D estéreo –de forma tal que los objetos virtuales invadieran el espacio real del usuario-, y que la entrada del usuario también fuera en tres dimensiones –de modo que el usuario pudiera interactuar con esos objetos en ese mismo espacio-, todo ello con las restricciones de coste propias de un videojuego de consumo.

Concretando los requisitos principales, el videojuego a desarrollar debía aprovechar la proliferación de grandes pantallas panorámicas de TV en los hogares para transformarlas en un centro de ocio de Realidad Virtual, utilizando esos gráficos 3D estéreo para proyectar los objetos virtuales fuera del plano de la pantalla. Por experiencias previas de otros autores –como, por ejemplo, el proyecto Immersadesk-3 [Czernuszenko, 1997]-, se sabe que el uso de estéreo activo –esto es, basado en la multiplexación temporal de las imágenes sobre la pantalla, combinada con unas gafas de obturación a pilas- es problemático en este tipo de pantallas, por lo que debía optarse por la solución más segura, como es el empleo de anaglifos.

El otro requisito principal era que el usuario pudiera interactuar con esos objetos en ese mismo espacio de Realidad Virtual, pero se imponía al proyecto que la entrada del usuario estuviera basada en el uso de un guante de datos que permitiera seguir, en el espacio 3D, los movimientos de la mano del usuario y que fuera, además, asequible para el consumidor.

Los requisitos del proyecto eran en sí arriesgados, pues algo parecido ya se intentó cuando se lanzó al mercado el guante Mattel PowerGlove para la consola NES, y finalmente se dejó de comercializar por falta de títulos. Se asumió entonces ese riesgo por las nuevas oportunidades que ofrece la tecnología actual y se fijó como prototipo de plataforma un PC conectado a una pantalla de plasma Hitachi Platara de 32" y un guante de datos P5 de la compañía Essential Reality, todo ello disponible en el laboratorio.

#### 7.2.1.2 Análisis del problema

Con los requisitos principales dados, extraídos a partir de los objetivos que debe cumplir la solución y los límites en los que debe desarrollarse esta, se fijaron también dos meses de plazo para la presentación de una propuesta de videojuego que aprovechara esa plataforma, para su evaluación previa antes del estudio detallado.

Una parte de esos dos meses se dedicaron a esta segunda etapa, el análisis del problema, en la cual el equipo de desarrollo debe familiarizarse con el problema profundizando en los requisitos y estudiando soluciones previas. En este caso, el trabajo del alumno –aquí asumiendo el rol de analista-, se centró en estudiar intentos pasados de llevar la Realidad Virtual al entretenimiento en casa –tanto hardware como software, esto es, títulos de juegos-, pero principalmente la apuesta de los creadores del guante P5 por cambiar con este dispositivo la forma de jugar a los títulos del momento. Desafortunadamente, no se contó con usuarios que hubieran utilizado alguna de esas soluciones pasadas, pues la

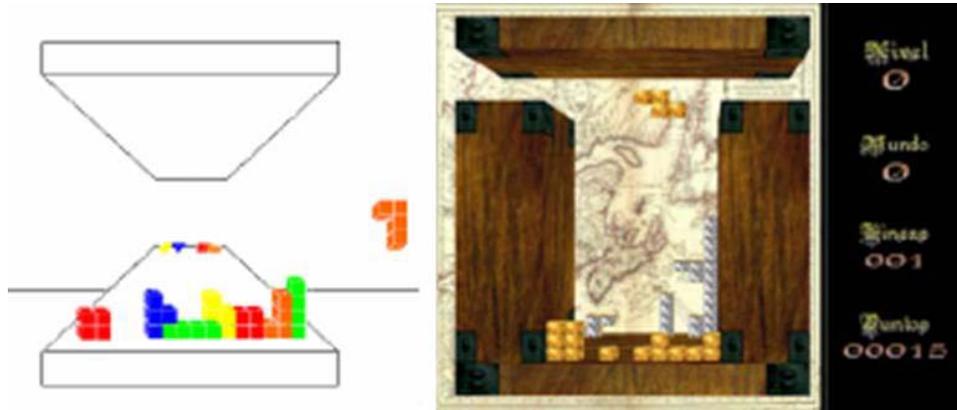
mayoría no llegaban a comercializarse en España, una circunstancia que también compartía el guante P5, más allá de aquellos que lo habían probado en el laboratorio.

### 7.2.1.3 Propuesta de solución

Tras la etapa de análisis, y dentro del tiempo que restaba de los dos meses fijados, la tercera y última etapa del estudio previo debía dar como resultado la propuesta de videojuego a desarrollar. Esta tomó forma a través de sesiones de tormentas de ideas en las que se buscaba el género que más se adecuara a las posibilidades que ofrecían los dispositivos, y también a través de reuniones o focus groups con posibles usuarios para conocer su opinión y revisar con ellos las propuestas.

Así, a la vista de las características probadas de la pantalla de plasma y del guante P5, el trabajo se centró finalmente en dos propuestas concretas, ambas como secuelas de dos videojuegos muy conocidos: Asteroids y Tetris. Aunque el juego Tetris lleva en el mercado muchos años, la propuesta que aquí se hacía incluía notables novedades con respecto al original, y entre ellas destacaban principalmente dos. La primera es que el espacio de juego era tridimensional, con forma de urna, y el juego se desarrollaba en las caras frontal y trasera de la misma, dos partidas simultáneas entre las que el usuario podía alternar girando la urna. La segunda novedad es que las piezas ya no caían directamente desde arriba, sino que aparecían por un lateral y el usuario debía atraparlas, girarlas y dejarlas caer sobre el lugar deseado, mediante gestos de su mano transmitidos a través del guante. Todo ello se concretó en un storyboard, un conjunto de requisitos específicos del propio videojuego –piezas, niveles, etc.-, y diagramas UML de casos de uso para capturar las necesidades de interacción.

Los bocetos realizados para cada propuesta fueron analizados en reuniones en las que participaban desarrolladores y usuarios. En la Figura 7.1, a la izquierda, se muestra uno de los bocetos con los que se ilustró la propuesta basada en el juego Tetris, y a la que se llamó TTristéreo. Esta fue la que contó con mayor entusiasmo por parte de todos, por lo que se optó por su desarrollo. El resultado de ese desarrollo también puede verse en la misma figura, a la derecha, pero para llegar a él tuvo que completarse otra fase de la metodología TRES-D, el estudio detallado, cuya primera etapa es el diseño.



**Figura 7.1** Primer boceto de TTristéreo (izquierda) e interfaz de usuario final (derecha)

#### 7.2.1.4 Diseño

Siguiendo la metodología propuesta, el diseño se aborda a dos niveles, uno abstracto –Diseño I- y otro de presentación –Diseño II-. Así, el diseño comienza en el nivel abstracto, centrándose en la funcionalidad –en términos de operaciones, sus parámetros y condiciones- del sistema y dejando a un lado los detalles de la plataforma sobre la que se ejecutará el videojuego. A este nivel se analizan y se detallan los casos de uso, plasmando todo ello en diagramas de secuencia, de colaboración y de clases. Paralelamente, se crean planos y bocetos que ilustran elementos del juego, como las piezas y la urna, pero cuya apariencia final se deja para el siguiente nivel. Una parte de este diseño abstracto, concretamente los menús del videojuego, es evaluado mediante la creación de un primer prototipo electrónico que se da a probar a un conjunto de usuarios y que, aunque su fin es el de comprobar que la funcionalidad ha sido bien capturada, los comentarios de los usuarios aportan ya información valiosa para el diseño de presentación.

Tras el diseño abstracto, el diseño continúa entonces a ese nivel de presentación –Diseño II-. Una parte de este diseño se centró en la forma que el usuario utilizaría el guante P5 para actuar sobre las piezas del juego. Para ello era preciso detallar qué acciones físicas se correspondían con cada una de las operaciones, esto es, las técnicas de interacción. Esto fue fácil, ya que el guante P5 permitió establecer una correspondencia directa entre las acciones en el mundo real con esas mismas acciones en el mundo virtual (Figura 7.2). Otra parte de este Diseño II se dedicó a la forma final que la urna y las piezas del juego presentarían al propio usuario. Esta parte también fue fácil, ya que estos elementos eran sencillos y su apariencia final no diferiría mucho de los bocetos elaborados en el nivel anterior. Tanto en una parte como en la otra debe

destacarse la importancia del espacio, y ahora en particular el papel que juegan los dispositivos, como nexo de unión entre el espacio real del usuario y el virtual de la urna y las piezas. Aquí, el uso de gráficos 3D estéreo convierten la pantalla de TV en un volumen de interacción que se solapa con el del sistema de tracking del guante. En cualquier caso, las líneas generales de este diseño ya fueron esbozadas en la propuesta de la solución al realizar bocetos del videojuego y las oportunas pruebas técnicas, si bien en esta ocasión se detallan para completar los diagramas del diseño abstracto y añadir otros nuevos.



Figura 7.2 Gestos de coger, rotar y solar una pieza en TTristéreo

El diseño prosigió alternándose entre estos dos niveles, volcándose el peso del desarrollo cada vez más al diseño de presentación, haciendo posible comenzar la implementación del videojuego para ir mostrándose a los posibles usuarios y darles la oportunidad de evaluarlo.

#### 7.2.1.5 Implementación

Con la implementación del videojuego bien avanzada se llevaron a cabo dos tipos de evaluaciones con el fin de conocer la opinión de los usuarios sobre el mismo. La primera de esas evaluaciones tenía por objeto llegar al mayor número posible de usuarios a modo de beta-testing, y para que el no tener guante no fuera un impedimento se preparó una versión del videojuego basada en teclado que fue distribuida a más de treinta personas de diferentes edades.

La segunda evaluación se centraba más en el uso del guante como entrada y las imágenes anaglíficas como salida, y en ella participaron siete usuarios a los que se les permitió hacer uso del prototipo con el guante P5 y la pantalla de plasma. En un principio, los usuarios necesitaron de cierta práctica para manejar correctamente el guante. Una vez dominado el uso del mismo, los usuarios se sienten más cómodos y más entusiasmados con el juego. Sin embargo, al cabo de un tiempo de juego, el usuario comienza a notar cansancio en el hombro y en el resto del brazo, un problema que es común a la interfaz del tercer proyecto, como se verá más adelante. A ese cansancio se sumaba el de la vista por el uso

de las gafas de filtros de colores, lo que en conjunto reduce el tiempo que el usuario desea jugar con el videojuego. Con todo, los usuarios encontraron el prototipo novedoso y atractivo, y afirmaban que con el guante la diversión realmente se multiplica.

## 7.2.2 Virtual Reality Prismaker

### 7.2.2.1 Presentación del problema

Otro de los usos de las interfaces de usuario 3D nombrados al principio de esta Tesis era el de servir de interfaz de Realidad Virtual. Con este segundo proyecto que se describe se perseguía entonces un mundo virtual que recreara un juego de bloques de construcción, comercializado este bajo el nombre de Prismaker<sup>®</sup>. El proyecto toma el testigo de otro anteriormente realizado con Java3D y orientado hacia plataformas PC basadas en teclado y ratón.

Aunque los resultados pedagógicos de aquel primer proyecto fueron buenos, la interacción basada en aquellos dispositivos hacía difícil el manejo de los bloques en el espacio 3D. Sabida la existencia de otros proyectos similares que reproducían el juego Lego<sup>®</sup> en entornos semi-inmersivos, se proponía entonces recrear el juego Prismaker en un sistema de Realidad Virtual inmersiva buscando la máxima naturalidad posible en la interacción.

El primer requisito del proyecto era la reproducción por ordenador de esos bloques, para que el niño pueda tener un número ilimitado de los mismos y el educador infinitas posibilidades para plantear ejercicios de aprendizaje con ellos. Vistas las grandes dificultades que conllevaba el manejo de los bloques con teclado y ratón, el segundo requisito es que el niño pueda manipular con sus propias manos, y de forma directa, esos bloques virtuales. Se considera entonces que la experiencia acumulada en el proyecto antiguo puede facilitar la realización del nuevo, aunque el cambio de paradigma puede dificultar la reutilización, con el riesgo de tener que empezar todo desde el principio. De nuevo, se establecen dos meses para la realización de una propuesta de diseño para este proyecto.

### 7.2.2.2 Análisis del problema

Siguiendo con la metodología TRES-D, la etapa que sigue a la presentación del problema es la del análisis del mismo, de modo que el equipo de desarrollo invierte parte de los dos meses fijados en familiarizarse con el juego Prismaker y

la aplicación Virtual Prismaker, nombre este con el que se conocía al juego de ordenador que resultó del anterior proyecto.

De Prismaker ya se ha dicho que es un juego de construcción basado en bloques. Esos bloques son todos del mismo tamaño y se pueden encontrar en cuatro colores distintos: rojo, azul, verde y blanco. Con estos bloques es posible construir diferentes figuras, para unirlos entre sí se emplean unos finos tubos de plástico blanco que encajan en cualquiera de los 16 agujeros que tiene cada pieza. Además de esos bloques, Prismaker también ofrece otros tipos de piezas, entre las que destaca la tapadera, que puede unirse a cualquier bloque y sobre la que se superpone una pegatina que puede representar cualquier símbolo de interés pedagógico, como por ejemplo símbolos matemáticos para aprender a sumar, restar o multiplicar.

En cuanto al juego Virtual Prismaker, el equipo tuvo acceso al software y al material empleado para su desarrollo. El proyecto, dirigido a los niños en las escuelas, no llegó a implantarse en estas, por lo que no había usuarios que pudieran transmitir sus experiencias en el uso de dicho juego. Sí se contaba, al menos, con algunos de los desarrolladores que entonces llevaron a cabo dicho proyecto.

### 7.2.2.3 Propuesta de solución

Siguiendo la metodología, el diseño comenzó con la generación de diferentes propuestas para el proyecto, plasmadas en bocetos que son discutidos y evaluados por los desarrolladores con la participación también de posibles usuarios.

Conocidas las operaciones que debe poder realizar el niño con los bloques, y plasmadas en diagramas UML de casos de uso, debía decidirse qué hardware era el más apropiado para lograr el requisito de naturalidad. Se analizaron entonces las características de los guantes disponibles en el laboratorio. Ninguno de ellos cuenta con realimentación de fuerzas, lo que impedirá al usuario sentir la forma o el peso de las piezas en sus manos, restando naturalidad. Sin embargo, sí le permitirán realizar de forma natural acciones como la de coger una pieza. Precisamente, la semejanza de ese gesto con un gesto "pinch" llevó finalmente a la inclusión del sistema PinchGloves en la propuesta. Ese gesto serviría también para tomar nuevas piezas en el mundo virtual, para lo que se pensó en situar en el ese mundo reproducciones de algunas cajas de piezas Prismaker como fuentes inagotables de estas piezas, así como una estantería donde dejar las figuras que se crearan con ellas.

Junto a los Pinch Gloves se hace necesario un sistema de localización o “tracking”, que para este proyecto fue el Flock of Birds, de la compañía Ascension. Como visiocasco, se contaba con un modelo Proview XS35, fabricado por Kaiser. Todas estas decisiones no son ajenas a la elección del software para el desarrollo del proyecto. Frente al uso de librerías de bajo nivel del proyecto TTristéreo, en este caso se opta por la librería VR Juggler por el soporte que ofrece a los dispositivos elegidos. También se decide utilizar los lenguajes VRML/X3D para guardar las figuras que sean dejadas en la estantería, de modo que puedan ser vistas en cualquier otra plataforma sin mayor requisito que un browser para la Web 3D. Unas pruebas técnicas iniciales respaldan estas decisiones de diseño.

Todos los requisitos específicos de la propuesta se plasman en un documento que, una vez que cuenta con el correspondiente visto bueno, sirve de punto de partida para la siguiente etapa.

#### 7.2.2.4 Diseño

Como en el proyecto TTristéreo, el diseño en detalle comienza desde el nivel abstracto –Diseño I-, analizando los casos de uso que se convierten en esta ocasión en una lista de operaciones más que una descomposición jerárquica de tareas, y creando un primer diagrama de clases. A este nivel independiente de la plataforma se dibujan los planos del mundo virtual a construir, una habitación de juegos en la que se sitúa una mesa, sobre ella las cajas de piezas, y cerca de ella la estantería.



**Figura 7.3** Gestos “pinch” para coger y soltar en VRPrismaker

Ya en el nivel de presentación –Diseño II-, se aborda la forma concreta en la que el usuario hace uso de los dispositivos para manipular las piezas y formar las figuras, esto es, las técnicas de interacción asociadas a las operaciones. Todas ellas se basaban en dos acciones: coger y soltar. Los guantes PinchGloves permitían, como se ha dicho, una correspondencia natural entre un gesto “pinch”

y la acción de coger. Para soltar, tras pruebas con usuarios, se optó por definir un gesto “pinch” adicional (Figura 7.3). La operación concreta depende de los objetos en contacto. Para ello se hace necesario conocer cuándo el usuario –su avatar- está tocando un objeto y cuándo dos objetos han chocado entre sí, es decir, debe detallarse la gestión de colisiones en el mundo virtual. Esa gestión es más compleja que en el antiguo proyecto, pues en aquel se ajustaban los bloques a una rejilla para facilitar su manipulación, y no se simulaba la gravedad, mientras que ahora para lograr el requisito de naturalidad se quiere que los bloques puedan ser manipulados libremente en el espacio y que caigan al suelo si son soltados.

De nuevo, el diseño continúa alternándose entre estos dos niveles, y puede comenzarse con la implementación del prototipo de la interfaz y su evaluación con usuarios.

#### 7.2.2.5 Implementación

En esta etapa se programa el bucle de simulación, por un lado, y se crean los objetos del mundo virtual, por otro. Respecto a lo primero, la librería VR Juggler proporciona ya dicho bucle, lo que tiene que hacer el programador es completar el código de ciertas funciones con las órdenes que pintarán el mundo en la pantalla o detectarán las colisiones entre objetos. En cuanto a lo segundo, a la geometría de cada objeto se le añade su apariencia final, reutilizando para ello algunas de las texturas ya creadas en el antiguo proyecto.



**Figura 7.4** El entorno virtual de VRPrismaker (izquierda) y usuario (derecha)

Tras varios ciclos de refinamiento de la implementación, se contaba con un prototipo en condiciones para proceder a una evaluación del sistema con

usuarios (Figura 7.4). Esta evaluación fue realizada por un grupo lo más heterogéneo posible de gente, en concreto nueve usuarios, seis hombres y tres mujeres. Estos usuarios rellenaron un cuestionario inicial que desvelaba que algunos tenían experiencia con el uso de ordenadores pero otros no, alguno incluso había usado alguna vez una aplicación de realidad virtual, y había también quien desconocía qué era todo aquello.

El proceso de evaluación se dividía, para cada usuario, en tres etapas. Inicialmente, se permitía al usuario interactuar libremente con la aplicación durante cinco minutos. A continuación, se mostraba al usuario una imagen de una figura y se le pedía realizarla en no más de diez minutos. Después se hacía lo mismo con una nueva figura, y también con otra tercera. Al final del proceso el usuario rellenaba un cuestionario en el que plasmaba sus impresiones acerca del aspecto, la funcionalidad y la usabilidad de la aplicación.

Tanto la apariencia como el comportamiento de los objetos en el mundo virtual fueron valorados en general de forma positiva, mientras que la opinión de los usuarios respecto a la interacción estaba dividida. La mitad de ellos calificaba la interacción como fácil o muy fácil, en tanto que la otra mitad afirmaba lo contrario. Estos últimos se quejaban de que el tamaño de las piezas era pequeño para poder encajarlas con comodidad. En la realidad tampoco es fácil encajar los bloques con los pequeños tubitos que proporciona Prismaker, por lo que desde este punto de vista el grado de naturalidad de la interfaz virtual podría equipararse esa realidad. Otras quejas apuntaban al peso del visiocasco, pero en general todos encontraron los guantes cómodos de usar.

### **7.2.3 Una interfaz de película: "Minority Report"**

#### **7.2.3.1 Presentación del problema**

Las interfaces de usuario 3D parecen ir siempre de la mano de los gráficos 3D. Sin embargo, como ya se indicara al definir aquí qué es una interfaz de usuario 3D, también lo son aquellas que, aunque la salida no sea 3D, sí lo sea la entrada. Ejemplo de ello es la interfaz que podía verse en la película "Minority report" (S. Spielberg, 2002), en la que el protagonista manipulaba imágenes que se proyectaban sobre la superficie plana de una pantalla de cristal, comunicando las órdenes al sistema mediante gestos que realizaba con las manos en el espacio tridimensional. El filme contó con el asesoramiento de Jaron Lanier, y entonces despertó gran interés no sólo entre el público que acudía a ver la película, sino también entre el resto investigadores del campo de las interfaces de usuario – véase por ejemplo la discusión sobre ella en la lista de correo [3D UI, URL],

§10/10/2004 y 13/4/2005-. La pregunta que se planteó entonces fue si una interfaz así era posible con la tecnología actual, y si sería tan útil y usable como aparenta en la película. Para hallar respuestas, se desarrolló el proyecto que se describe a continuación.

El requisito principal de este proyecto no era otro que el crear una aplicación que contara con una interfaz lo más parecida posible a la vista en la película. Dado este requisito, lo primero que se hizo entonces fue realizar un análisis completo de la interfaz tal y como aparece en la película. En este sentido, la película fue para el equipo de desarrollo como un escenario realizado en video, pero con el presupuesto de las producciones de Hollywood.

### 7.2.3.2 Análisis del problema

Tras la exposición de los requisitos, una vez más se fijaron dos meses para la presentación de una propuesta de solución, empezando a consumirse ese tiempo con esta etapa, el análisis del problema. Aquí se prestó especial atención a la interacción y, en particular, a los gestos que realizaban los protagonistas para manipular las imágenes en movimiento que se mostraban en la pantalla. En total se contabilizaron 11 posibles acciones, aunque el gesto asociado variaba de un actor a otro, e incluso para un mismo actor, por lo que el número de gestos observados resultaba ser mayor, 17. El análisis también determinó que en los gestos era importante la flexión y abducción de los dedos, la posición y orientación de las manos, y su movimiento.

### 7.2.3.3 Propuesta de solución

Aunque la forma de interactuar con el sistema venía impuesta por los propios requisitos del proyecto, antes de iniciar su desarrollo resultaba imperativo decidir qué software y, más importante aún, qué hardware haría posible esa interfaz. En la película los actores utilizan dos guantes sin aparentes cables y unos posibles LED's en la yema de los dedos pulgar, índice y corazón, en lo que parece un sistema basado en visión por computador, aunque no se distingue cámara alguna.

Sin embargo, la experiencia previa en el empleo de guantes de datos suponía un menor riesgo para el proyecto, y se evaluaron entonces diferentes modelos para determinar cuál era el más apropiado. Así, se vio que la abducción de los dedos sólo era medida por el Cyberglove, pero este guante no permitía conocer la posición y orientación de la mano como sí lo hacía, por ejemplo, el guante P5. Sin embargo, esa falta era suplida al complementar el guante con un sistema de

tracking como, por ejemplo, el Ascension Flock of Birds que también fue utilizado para el proyecto VRPrismaker.

Además del hardware, se estimó necesario contar con un mecanismo que reconociera los gestos a partir de los datos que proporcionaran el propio guante y el sistema de tracking. Para ello se analizaron también las diferentes opciones. Unos métodos son más apropiados para reconocer posturas, como el Template Matching, y otros para reconocer gestos con movimiento, como los modelos ocultos de Markov. Algunos de los gestos analizados sí incluían movimiento, lo que podía aconsejar el uso esos últimos métodos. Sin embargo, tras un análisis más detallado se concluyó que la mano no cambiaba de postura durante todo el movimiento de un gesto, por lo que se consideró que un método como el Template Matching bastaría. Esta decisión, no obstante, añadió un requisito más, y fue el de tener que desarrollar también un entrenador de gestos.

En cuanto al software, se estimó oportuno hacer uso de las apropiadas librerías de entrada y salida, tanto para leer los datos del guante y el tracker como para mostrar en pantalla las imágenes en movimiento. De nuevo, la realización de unas pruebas técnicas preliminares permitió asegurar que el diseño propuesto era factible.

#### 7.2.3.4 Diseño

En base a las líneas maestras trazadas en la propuesta de solución, se abordó el diseño en detalle empezando de nuevo desde el nivel abstracto –Diseño I-, para luego bajar al nivel de presentación –Diseño II-.

Así, se define en primer lugar la interfaz de forma independiente de la plataforma, transformando los casos de uso en una lista de operaciones, con sus parámetros y condiciones. En este sentido, se hace uso de un diagrama de estados (Figura 7.5) para detallar cuándo las operaciones pueden ser ejecutadas y, por tanto, cuándo los gestos asociados a las mismas tienen sentido para la aplicación.

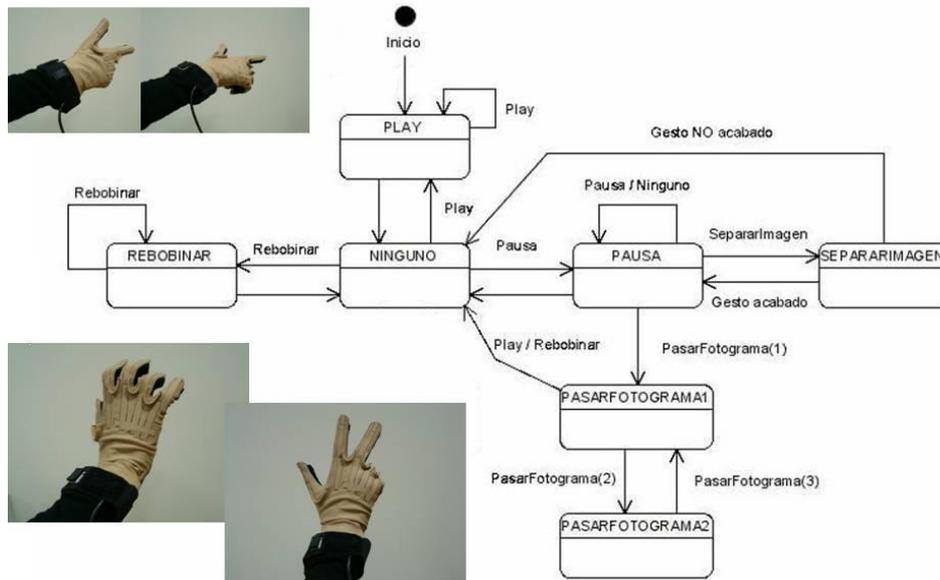


Figura 7.5 Diagrama de estados y gestos asociados a cada operación

En el diseño a nivel de presentación se detallaron las técnicas de interacción asociadas a las operaciones, como por ejemplo la técnica ray casting asociada a los gestos apuntar y seleccionar. Además se extiende el diagrama de clases con otras nuevas relacionadas con la entrada de datos –Cyberglove y Flock of Birds– y la salida gráfica –con un diseño de pantalla inspirado en el de la película–.

En realidad, en esta etapa y la siguiente se desarrollan dos aplicaciones en paralelo, siendo una el entrenador de gestos y la otra la aplicación que persigue reproducir la vista en la película. Así, la primera de ellas guarda en plantillas los gestos entrenados por cada usuario para cada operación, y la segunda interpreta esos gestos a partir de las plantillas utilizando el motor de pattern matching programado.

Conforme avanzaba el diseño fue posible comenzar la implementación de prototipos para poder ir corrigiendo fallos gracias a la opinión de los expertos y de los posibles usuarios.

### 7.2.3.5 Implementación

Con la implementación casi completa, se procedió a realizar una evaluación de la interfaz en la que participaron siete usuarios, tres de ellos hombres y cuatro mujeres. Como en el proyecto VRPrismaker, completaron un cuestionario inicial

que desveló que la gran mayoría de ellos trabajaban diariamente con ordenadores, todos sabían manejar un reproductor de video, 4 habían visto la película "Minority Report", y 3 habían utilizado antes algún dispositivo de Realidad Virtual.

De nuevo, el proceso de evaluación se dividía en otras tres etapas. En la primera, cada usuario dedicaba un tiempo para utilizar el entrenador de gestos y personalizar así las plantillas. Entonces, en una segunda etapa, cada usuario se enfrentaba a la interfaz creada y se le pedía realizar una serie de tareas de búsqueda y manipulación de imágenes, como si fueran protagonistas de la película. Finalmente, el usuario rellenaba un nuevo cuestionario con el que se perseguía conocer sus impresiones acerca del uso del sistema, pero sobre todo sobre la usabilidad y utilidad de la interfaz.

En general, los usuarios consideraron que el guante era cómodo de usar, aunque los cables –apuntaban- les dificultan los movimientos. Los usuarios no valoraron el esfuerzo físico como excesivo, aunque sí opinaban que debería ser menor; de hecho, se observa que los usuarios que más tiempo dedicaron a la interfaz fueron los que más importancia dieron a ese esfuerzo físico. Aún así, todos los usuarios aprobaron la interfaz, e incluso al compararla con la interfaz basada en botones de un reproductor de video, 5 de los usuarios valoraron la interfaz basada en gestos como mejor. Sin embargo, sólo 2 de los usuarios preferiría manipular así todas las aplicaciones de su PC, aunque hasta 4 de ellos cambiaría su mando a distancia por una interfaz como esta.

## **7.3 Prototipado de interfaces de usuario distribuidas**

Este cuarto caso de estudio no pretende ilustrar el desarrollo estructurado basado en la metodología TRES-D, como se ha hecho con los anteriores, sino que se centra en otra de las contribuciones que acompañan a aquella en esta Tesis, la librería VUIToolkit. En concreto, el objeto de este caso de estudio es la aplicación de dicha librería al diseño y prototipado rápido de interfaces de usuario distribuidas.

### **7.3.1 Aproximación al problema**

Las interfaces de usuario distribuidas –o también llamadas DUI's, por sus siglas en inglés- se llaman así por distribuir parte o la totalidad de una interfaz de usuario por distintas plataformas, como el software en un sistema distribuido. Esas plataformas son parte del contexto de uso, en el cual se incluyen además otros elementos que pueden no estar directamente relacionados con el sistema

informático, pero sí con el usuario, como por ejemplo su espacio de trabajo y el mobiliario presente en el mismo.

El contexto de uso, o la evolución del mismo a lo largo del tiempo, puede hacer entonces necesaria la distribución de la interfaz de usuario en partes por distintas plataformas. Esto significa que esos fragmentos de la interfaz de usuario, inicialmente diseñados para una plataforma original, deben ser adaptados a sus destinos en nuevas plataformas.

El diseño de una interfaz de usuario distribuida puede resultar así costoso, por lo que se apostó por una aproximación basada en modelos y la Realidad Virtual. Los modelos describen el entorno físico del usuario y la interfaz que este maneja, y a partir de ellos se crea el entorno virtual. La Realidad Virtual permite dentro de ese mismo entorno virtual la manipulación directa de la interfaz de usuario, con el fin de realizar el ajuste de sus elementos sobre la misma plataforma en la que se muestran, o migrar parte o la totalidad de la misma a otras plataformas, distribuyendo así la interfaz por el espacio.

Para esta aproximación, se asume que el desarrollo descansa sobre el marco de trabajo Cameleon, utilizando UsiXML como lenguaje de descripción de interfaces de usuario. De estos dos nombres propios, el último ya ha sido citado anteriormente en esta Tesis, aunque la descripción de ambos se encuentra en el Anexo dedicado a la librería VUIToolkit. A los modelos que describe UsiXML para cada capa de Cameleon se suma aquí el modelo del contexto de uso, como un conjunto de entidades –cada una con sus propios atributos- que describe: el usuario –edad, habilidades, preferencias, etc.-, la plataforma informática –portátil, de escritorio, de pared, etc.-, y el entorno físico –casa, oficina, etc.-.

El entorno físico se representa aquí como un mundo virtual, utilizando los lenguajes VRML/X3D para su descripción y un browser Web3D para explorarlo. El entorno físico incluye también una representación virtual de las plataformas, usando el PROTO “Screen” de la librería VUIToolkit para describir las superficies de interacción de estas. Sobre ellas se representan entonces las interfaces de usuario distribuidas gracias al resto de prototipos que incluye la misma librería VUIToolkit.

En los siguientes apartados se describe el caso de estudio concreto, con el mundo virtual que representa el entorno de usuario, las plataformas y las aplicaciones que en ellas se ejecutan, terminando con un ejemplo de migración en dicho entorno virtual.

### 7.3.2 El entorno de usuario y las plataformas

El entorno de usuario elegido como caso de estudio es el de un pequeño despacho, en el cual el usuario tiene a su disposición cinco tipos de diferentes plataformas: un PC, dos ordenadores portátiles, un Pocket PC y un proyector portátil. Con el lenguaje UsiXML se describe ese entorno y las plataformas guardando en una especificación XML los detalles que más interesan a los desarrolladores involucrados en el caso de estudio, como puedan ser los ingenieros de software y diseñadores de interfaces.

Partiendo de esa especificación, se crean una serie de PROTO's –hardware, muebles, etc.- con los que componer la escena VRML97/X3D. Cada uno de esos prototipos tiene asociada una representación audiovisual interactiva, la cual no es obra de una generación automática a partir de una especificación, sino el resultado de un proceso estructurado de creación llevado a cabo por un diseñador de mundos virtuales, como el proceso marcado por la metodología TRES-D.



**Figura 7.6** Vista general del entorno mostrando las cinco plataformas

El resultado puede verse en la Figura 7.6, la cual muestra la representación virtual del entorno físico y en él las cinco plataformas que lo componen, las cuales se enumeran en la siguiente tabla al tiempo que se describen las características de las pantallas:

Nº	Plataforma	Tamaño de pantalla	Razón de aspecto	Resolución	Resolución máxima
1	Panasonic PT-LB10SU	¾ d*	4:3	800x600	800x600
2	Toshiba PocketPC e750	3.8"	3:4	240x320	240x320
3	Acer Aspire 2000	15"	16:10	1280x800	1280x800
4	Dell Latitude C840	15"	4:3	1600x1200	1024x768
5	NEC LCD1960NX	19"	5:4	1280x1024	1024x768

Tabla 7.1 Propiedades de pantalla de las diferentes plataformas

### 7.3.3 Interfaces de usuario

Las interfaces de usuario que se incluyen en este caso de estudio corresponden a dos aplicaciones concretas: un reproductor de radio en Internet –*Internet Radio Player*- (Figura 7.7, izquierda), y un cliente de mensajería instantánea –*Instant Messaging Client*- (Figura 7.7, derecha). La primera incluye controles para buscar emisoras de radio, filtrar la búsqueda y seleccionar de una lista la emisora deseada, reproducción, pausa y parada, así como control del volumen. La segunda incluye facilita la introducción de mensajes y el seguimiento de la conversación en línea que se está manteniendo.

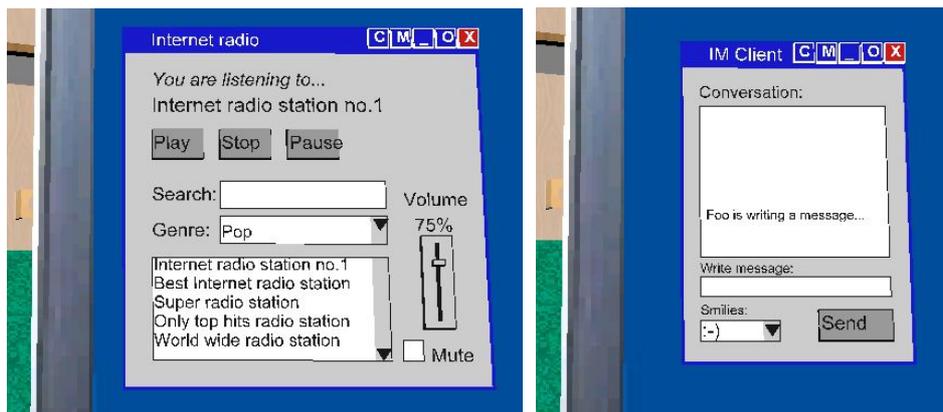


Figura 7.7 Ventanas virtualizadas de las dos aplicaciones de este caso de estudio

Ambas interfaces de usuario son especificadas con el lenguaje UsiXML a través de los diferentes modelos. Para hacer la tarea más sencilla, se cuenta con herramientas visuales que permiten acometer los diferentes modelos. Así IdealXML permite crear el modelo abstracto de la aplicación, y GrafiXML el modelo concreto. La interfaz final es generada en forma de fichero VRML97 o X3D a partir de los modelos creados, y utilizando los prototipos de la librería VUIToolkit, como se ha explicado en anteriores ocasiones.

### 7.3.4 Ejemplo de migración

En este apartado se ilustrará cómo el usuario utiliza el entorno virtual para llevar a cabo la migración de una interfaz de usuario desde una plataforma original hasta otra plataforma destino (Figura 7.8). Para soportar dicha migración, se realizaron ciertos cambios en la librería VUIToolkit. El resultado visible de dichos cambios se aprecia en el prototipo VUIWindow, que incluye un nuevo botón en la barra de título, etiquetado con la letra “M”. Cuando se pulsa dicho botón, el usuario atrapa esa ventana, y a partir de ese momento la ventana seguirá al usuario dondequiera que apunte en la pantalla de esa plataforma, pudiendo incluso arrastrarla a otra plataforma distinta. Si, apuntando a otra plataforma, el usuario hace clic con el ratón, la migración de esa parte de la interfaz es ejecutada, acompañada en su caso de una adaptación a la nueva plataforma.



Figura 7.8 Secuencia de imágenes que muestra la migración desde un portátil al Pocket PC

## 7.4 Resumen

En este capítulo se han descrito varios casos de estudio que llevan a la práctica las propuestas que reúne esta Tesis. En total, cuatro son los casos de estudio que aquí se han desarrollado, tres de ellos ilustran el desarrollo estructurado de interfaces de usuario 3D que propone la metodología TRES-D, y el cuarto ilustra

el uso de la librería VUIToolkit, una de las contribuciones que se han presentado junto a la anterior metodología.

Empezando con los tres primeros, estos casos de estudio no sólo tenían en común el desarrollo de una interfaz de usuario 3D con la metodología TRES-D, también el empleo de guantes de datos como interfaz de entrada. Sin embargo, cada proyecto correspondía a una aplicación diferente, cada una con sus requisitos particulares, lo que motivó la elección de diferentes modelos de guantes. Uno de los proyectos tenía por objeto la creación de un videojuego para PC de sobremesa que permitiera al jugador sumergirse en el mismo utilizando gráficos 3D estéreo y un guante de datos, por lo que el interés se centraba en encontrar una solución económica. El segundo de los proyectos perseguía recrear virtualmente un juego de construcción de bloques en un sistema de Realidad Virtual inmersivo, buscando con ello la máxima naturalidad. Y con el tercero se deseaba reproducir la interfaz de usuario vista en la película “Minority report”, para lo cual se utilizarían todos los medios al alcance.

Los tres proyectos fueron abordados en dos fases, según la metodología TRES-D, con un estudio previo que da como resultado una propuesta de solución en la que se incluyen requisitos específicos de la interfaz de usuario 3D, destacando en este caso el modelo de guante, y un estudio detallado en el cual se completa el diseño y se lleva a cabo la implementación, destacando aquí la separación entre interacción y contenido característica de la metodología TRES-D. Los tres meta-modelos propuestos –de los objetos, de los elementos de interacción y del espacio- están presentes durante todo el desarrollo de esos proyectos, así como el empleo de las herramientas y guías recogidas en la propuesta metodológica para el desarrollo de esta clase de interfaces.

El cuarto y último caso de estudio no se centra en el modelo de proceso como los tres anteriores, sino que, como se ha dicho, ilustra el uso de la librería VUIToolkit. Esta librería, cuya característica principal es la de transformar los widgets de las interfaces planas en su representación 3D verdadera, esto es, en widgets 3D, se presenta en este caso de estudio como herramienta para el diseño y prototipado rápido de interfaces de usuario distribuidas o DUI's. Aquí, a partir de una especificación UsiXML que describe la interfaz de usuario en base a los cuatro modelos del marco de trabajo Cameleon, junto con la descripción del contexto de uso que incluye el entorno físico como una escena VRML97/X3D, se crea una simulación virtual de ese entorno físico que puede verse a través de un browser para la Web 3D. Dicha simulación representa también las plataformas y las interfaces de usuario de las aplicaciones que corren sobre aquéllas, utilizando la librería VUIToolkit para representar dichas interfaces y permitir su manipulación y distribución por las diferentes plataformas, con el propósito de probar diferentes alternativas de diseño.

Con estos cuatro casos de estudio, termina aquí el trabajo recogido en esta Tesis, cuyas principales contribuciones se resumirán en el siguiente y último capítulo, destacando las conclusiones que de todo ello se extraen, así como las futuras líneas de trabajo que se abren a partir del mismo.

## 8

# Conclusiones

### 8.1 Echando la vista atrás...

Cuando este trabajo doctoral empezaba a dar sus primeros pasos, el campo de las interfaces de usuario tridimensionales era un campo prometedor, pero aún inmaduro. Esta situación contrastaba con la que disfrutaban las interfaces de usuario más convencionales, cuya madurez permitía el rápido desarrollo de aplicaciones apoyándose en métodos y herramientas bien conocidas, con una cada vez mayor automatización.

Y no es porque las tridimensionales fueran de reciente creación. De hecho, los inicios de unas y otras pueden datarse allá por los años 60, conviviendo el ratón de Doug Engelbart con la Espada de Damocles de Ivan Sutherland. Sus caminos a partir de ese momento fueron, sin embargo, más bien distintos. Así, el ratón se convierte unos años después en el símbolo del ordenador personal como parte de las interfaces WIMP, la cara visible de la mayoría de las aplicaciones informáticas en el hogar y en la oficina. Por su parte, el visiocasco se convierte en portada de la prometedora Realidad Virtual, y con ello de las interfaces tridimensionales, aunque no ganan la popularidad de las anteriores, limitándose a aplicaciones en mercados mucho más especializados.

Sin embargo, pese a la popularidad alcanzada por las interfaces convencionales, estas no quedan libres de limitaciones, lo cual motiva aún más la búsqueda de una nueva generación de interfaces, las que van Dam llamaba post-WIMP y que para Jacob eran no-WIMP. A lo largo de todos estos años, la interacción que han ofrecido los gestores de ventanas más conocidos no ha variado mucho, más allá de los cambios estéticos que aprovechaban los avances tecnológicos en el terreno del ordenador personal. La sensación que se tiene entonces es que no aprovechan la enorme potencia de cálculo de los ordenadores actuales, ni tampoco todas las posibilidades del ser humano.

Entre los avances tecnológicos, sin duda ha destacado el hardware de gráficos 3D, sumido en una espiral de complicidad con los cada vez más exigentes videojuegos 3D. Cómo aprovechar esa potencia gráfica en una nueva interfaz que sustituyera al tradicional escritorio era una pregunta en la mente de muchos. Si a ello unimos Internet, la respuesta podría ser el proyecto Croquet, en donde el escritorio del usuario es sustituido por un espacio 3D virtual que se une al de

otros usuarios en una red de iguales o peer-to-peer. El propio Alan Kay, pionero de las interfaces WIMP actuales, abanderó ese proyecto. Sobre esa y otras muchas ideas se discute en listas de correo como *www-vrml* o *3dui*. Las discusiones en esta última reflejan el estado de un campo que quiere crecer y madurar, alcanzando hitos como la celebración de la primera conferencia específica sobre interfaces de usuario 3D, o la publicación del libro “3D User Interfaces: Theory and Practice” cuyo título es toda una demostración de intenciones al imitar el título de la famosa obra de Foley *et al.*, referencia por mucho tiempo en el campo de los gráficos por ordenador.

Anterior a la publicación de ese libro, los mismos autores, Bowman *et al.*, firmaban un artículo que servía de introducción al campo de las interfaces de usuario 3D, y que ahora puede verse como una instantánea de ese campo justo cuando comenzaba este trabajo doctoral. Como parte de esa instantánea, se exponían las dificultades que tenía el diseño de las interfaces tridimensionales y, haciendo referencia a los tres pilares en los que según Shneiderman debía apoyarse el éxito en el diseño de la interfaz, faltaban guías, herramientas y expertos. El panorama descrito por esos autores alentó aún más la realización del presente trabajo, recordando que una de las lecciones más importantes aprendidas en la ingeniería del software, según recoge Sommerville, es que las mejoras en los procesos de diseño y de calidad requieren prácticas sistemáticas que involucran métodos bien fundados.

Partiendo de la experiencia propia en metodologías para el desarrollo de interfaces de usuario convencionales, se dirigió entonces una búsqueda de prácticas y propuestas para la creación de interfaces tridimensionales. No fueron pocas las encontradas, y seguramente habrá más que aquí se ignoran, pues en un campo emergente como este es fácil que cada desarrollador haya hecho de su experiencia su propio método. Más allá del número, resulta más interesante fijarse en la diversidad de puntos de vista con los que los diferentes autores se aproximan al problema. Tras el análisis de todas ellas, se observaron no obstante carencias para acometer el desarrollo general de una interfaz de usuario 3D. La respuesta a todo ello es una nueva metodología, a la que se ha llamado TRES-D. Pero la contribución de esta Tesis al campo no es sólo una secuencia ordenada de fases, etapas y actividades. El siguiente apartado resume todas las que este trabajo reúne.

## 8.2 Contribuciones

El marco de trabajo TRES-D consta de un modelo de proceso en el cual se acomodan las diferentes herramientas que puedan servir al desarrollador en su trabajo, como ayudas en forma de árbol de decisión para seleccionar las técnicas

más apropiadas o librerías de widgets y controles que rompen con la necesidad de implementar todo desde cero. Pero no son sólo guías, herramientas o expertos lo que necesita el campo de la interfaces de usuario 3D, pues se ha observado también cierta confusión en los términos que utilizan los autores y los conceptos que con ellos transmiten. Por ello, desde estas líneas también se ha querido poner orden a través de la definición de interfaz tridimensional y sus elementos, su espacio de diseño y sus aplicaciones. Estos y otros esfuerzos se incluyen entre las contribuciones de este trabajo:

- ✓ Se ha realizado una revisión del estado actual de las interfaces de usuario 3D desde una perspectiva abierta y crítica. Una de las principales dificultades con las que se ha topado esta revisión es que, en muchas ocasiones, estas interfaces no se presentan como tales, sino bajo otros nombres, como interfaces de Realidad Virtual o Aumentada, interfaces post-WIMP o no-WIMP, y en general como “las 3D”.
- ✓ Ante tanta diversidad, uno de los primeros pasos que se dieron en este trabajo fue el de dar una definición a estas interfaces de usuario 3D. Con la definición dada, no sólo se entiende como tales aquellas interfaces cuya salida es un espacio tridimensional –principalmente gráficos 3D-, sino también aquellas otras cuya entrada se lleva a cabo en las tres dimensiones del espacio –principalmente gestos-. Junto a esa definición se dio también una lista de los diferentes usos y aplicaciones que se le han dado y se le dan a estas interfaces, no sólo aquellos que han tenido éxito sino también esos otros que no han tenido tanto o han fracasado, ilustrando todo ello con experiencias previas que puedan servir de referencia a futuros diseñadores.
- ✓ Tras esa definición, le tocó el turno al espacio de diseño. Aquí se extendió el conocido continuo realidad-virtualidad de Milgram y Kishino para transformarlo en un nuevo continuo digital-virtual-real en el que no sólo se dibuja ese espacio difuso entre la realidad y los mundos virtuales, sino también entre estos y las interfaces de usuario más convencionales. Además, este nuevo continuo no utiliza la recta para su presentación, sino un plano con sus dos ejes, siendo uno el número de dimensiones y el otro el grado de inmersión. Con este continuo es posible ubicar aplicaciones de las interfaces 3D que, como los escritorios 3D, no parecían tener su hueco en el anterior.
- ✓ El espacio de diseño se ha completado con la identificación y descripción de los diferentes elementos que componen las interfaces de usuario 3D, estos son, el espacio, los objetos, su comportamiento y la interacción. Sobre esta última se profundizó aún más hablando del diálogo, las tareas, técnicas y objetos de interacción, y los dispositivos

físicos. Todo ha servido de base para la propuesta de meta-modelos que acompaña a la metodología TRES-D.

- ✓ En esa misma línea, y ampliando la revisión del estado actual de las interfaces tridimensionales, se ha abordado también la problemática del diseño de cada elemento de la interfaz en particular, y de esta en general. Las dificultades aparecen, por una parte, por el hecho de que la interacción rompe con el diálogo secuencial y por turnos de las interfaces convencionales y, por otra parte, por la falta de un conjunto de controles y técnicas de interacción estándar. De lo primero, se ha visto que distintos autores coinciden en combinar una notación de flujo continuo de datos con otra notación de estados discretos, bien en diagramas separados pero relacionados o bien en la misma representación. De lo segundo, a pesar de los esfuerzos de otros autores aún hay mucho trabajo por hacer, lo cual ha motivado las herramientas de guía y el toolkit que se han incluido en el marco de trabajo TRES-D.
- ✓ Se ha revisado un amplio número de metodologías para el desarrollo de interfaces de usuario 3D, desde diferentes aproximaciones y en cualquiera de sus aplicaciones. En total, estas han sido veintiséis, un primer modelo de tres procesos derivado de las enseñanzas de Shneiderman más veinticinco repartidas en las siguientes siete categorías: una cadena de producción de cine de animación 3D, siete propuestas para crear mundos virtuales para PC, dos métodos con un desarrollo más participativo, cuatro que hacen uso del análisis de tareas, tres más que se basan en la ingeniería del software, cinco métodos para el desarrollo más allá del PC, y cuatro metodologías de desarrollo de interfaces convencionales. Para cada una de ellas se ha aportado una descripción que en muchos casos no resume un único artículo, sino que es el resultado de revisar más de uno y unir a partir de ellos las distintas piezas que componen el puzzle en el que a veces se convierte la propuesta. Cada descripción se ha acompañado de un análisis que resalta las bondades de la metodología, pero también sus defectos, comparándolo con el resto de propuestas. Por si eso fuera poco, y a fin de dibujar una imagen más completa, cada categoría termina con otro análisis en el que se analizan en conjunto las propuestas incluidas en la misma.
- ✓ Una de esas veintiséis metodologías que se describen y analizan es la llamada IDEAS, la cual este autor conocía bien por haber colaborado en la realización de diferentes casos de estudio en el marco del trabajo doctoral de su autora, M. D. Lozano. En base a esta experiencia previa y a los primeros progresos que se realizaban en el campo de las interfaces tridimensionales, se realizó una primera propuesta de metodología de

desarrollo para estas, la cual se bautizó como IDEAS-3D. Esta propuesta trataba de beneficiarse de un desarrollo dirigido por modelos apoyado en una herramienta, IDEAS-CASE, que automatizaba gran parte del proceso, destacando la generación automática de la interfaz final a partir de esos modelos. El éxito fue parcial, ya que si bien se consiguió hacer lo propio con unas sencillas interfaces 3D para escritorio, no era suficiente para abordar el desarrollo de cualquier interfaz 3D en general.

- ✓ Con la investigación más avanzada, se plantea entonces un nuevo marco de trabajo al que se bautiza como TRES-D, que como se indicó en su momento son las siglas de “ThRee dimEnsional uSer interface Development”. Entre los primeros objetivos de este nuevo reto se encontraba el proporcionar un lenguaje que facilitara la comunicación entre los desarrolladores, sin ambigüedades. Así, se han propuesto tres meta-modelos que describen los tres principales elementos de estas interfaces: los objetos, los elementos de la interacción y el espacio. El primero incluye una taxonomía de objetos, y descompone cada uno en función, comportamiento y sub-modelos, entre ellos, el gráfico. El segundo esclarece la relación entre tarea de interacción, técnica de interacción y objetos de interacción, distinguiendo entre widgets, widgets 3D y objetos, todos ellos como controles que participan en una técnica de interacción. Y el tercero distingue entre espacio digital, virtual y real, relaciona los dos primeros a través del puerto de vista, y a su vez esos dos con el tercero a través de los dispositivos físicos como superficies y volúmenes de interacción.
- ✓ Como modelo de proceso se ha propuesto una secuencia de dos grandes fases, la primera el estudio previo y la segunda el estudio detallado. Con ello se ha querido reducir el riesgo que conlleva el desarrollo en un campo aún inmaduro como es el de las interfaces tridimensionales. Además, a lo largo de ambas se involucra a diferentes roles para asegurar no sólo una interfaz funcionalmente correcta, pero también satisfactoria para el futuro usuario. Cada fase consta, a su vez, de tres etapas. El estudio previo empieza con la presentación del problema, y sigue con el análisis del problema y la propuesta de solución. En esta última se estudian diferentes propuestas avanzando en diferentes frentes que corresponden en realidad al estudio detallado, pero con ello se consigue adelantarse a los posibles problemas que puedan derivarse de cada propuesta. La aceptación, por parte de una de esas propuestas marca la separación entre esa fase y la siguiente. El estudio detallado comienza entonces con el diseño, sigue con la implementación y termina con el despliegue y el mantenimiento. El diseño se divide a su vez en dos, uno independiente de la implementación –Diseño I-, en donde se describen las tareas y las operaciones pero también se hacen bocetos del

contenido, y otro –Diseño II- pegado a esa implementación, describiendo las técnicas de interacción y fijando el nivel de detalle del contenido que se reparte entre los creadores de modelos 3D. A nivel de implementación, se distingue la labor de programación de la de creación de esos modelos. Así, tanto en el diseño como en la implementación se aborda el desarrollo en dos líneas de trabajo que se entrecruzan, una dedicada a las tareas y la interacción y otra a los objetos y el contenido. La metodología puede así volcarse sobre una u otra línea, o las dos, según el desarrollo particular, un rasgo que diferencia la propuesta TRES-D de las demás.

- ✓ Como ejemplos de herramientas que se integran en el marco de trabajo propuesto, se han presentado dos árboles de decisión que guían al diseñador, ya en el Diseño II, en la elección de técnicas de interacción. Partiendo de las llamadas tareas de interacción universales, uno de estos árboles ayuda entonces a decidir la técnica más apropiada para la tarea de selección/manipulación de objetos, y el otro para la tarea de introducción de texto, ambas en el contexto de entornos virtuales inmersivos, el segundo en particular para aquellos entornos que sumergen al usuario a través de un visiocasco.
- ✓ Completando la propuesta, se ha incluido también una librería llamada VUIToolkit para facilitar al desarrollador la implementación de interfaces tridimensionales. Esta librería ofrece un conjunto de controles para los que se han creado dos versiones, una con el lenguaje estándar VRML97, y otra con la revisión más reciente de ese lenguaje, llamada X3D. Los controles que ofrece VUIToolkit tienen la particularidad de ser widgets 3D, esto es, representan widgets de interfaces convencionales pero con su verdadera dimensión espacial, lo que se ha llamado interfaz de usuario virtualizada, acercándose más de este modo al modelo mental que el usuario tiene de esos controles, y facilitando la integración de interfaces convencionales en nuevos entornos tridimensionales.
- ✓ Finalmente, se ha puesto todo ello en práctica a través de cuatro casos de estudio particulares, tres de ellos ilustrando el modelo de proceso de la metodología TRES-D y un cuarto demostrando el uso de la librería VUIToolkit. El objeto de los tres primeros casos fue, como no podía ser de otra manera, tres interfaces de usuario tridimensionales, que compartían además la peculiaridad de hacer uso de guantes de datos, pero cada uno en una aplicación diferente que exigía la elección de un modelo diferente. Así, uno era un videojuego para escritorio que sumergía al usuario en el juego con el económico guante P5, otro llevaba un juego de construcción de bloques a un sistema de Realidad

Virtual con visiocasco y guantes PinchGloves, y el tercero reproducía con el guante Cyberglove y un sistema de tracking la conocida interfaz de la película “Minority report”. El cuarto caso de estudio hacía uso de la librería VUIToolkit, apoyándose en el lenguaje UsiXML y el marco de referencia Cameleon, para el diseño y prototipado rápido de interfaces de usuario distribuidas, representando el entorno físico del usuario como una simulación de Realidad Virtual en la que el diseñador puede manipular y distribuir las interfaces virtualizadas con simples movimientos del ratón.

### 8.3 Publicaciones

Gran parte de las contribuciones que resume el anterior apartado han sido presentadas en eventos tales como conferencias y congresos tanto de carácter nacional como internacional. La participación en dichos eventos ha permitido estar en contacto con otros investigadores del campo de las interfaces de usuario, y de las tridimensionales en especial, recogiendo valiosas observaciones sobre el presente trabajo y diseminando el mismo a través de las actas que se han publicado de esos eventos. Aunque han sido muchas, cabe destacar la participación, año tras año desde 2003, en el congreso internacional Interacción organizado por la asociación AIPO, así como la participación en eventos tales como Web3D, IUI y VRST organizados por la asociación ACM. Más concretamente, y relacionando las contribuciones con sus correspondientes publicaciones, estas han sido:

- El continuo digital-virtual-real que aquí se presenta es la revisión del continuo para la virtualización de interfaces de usuario que se recoge en [Molina, 2005a], publicado en las actas del 10th ACM Int. Conf. on 3D Web Technology, Web3D 2005, celebrado en Bangor, Gales, Reino Unido.
- La revisión del conjunto de metodologías propuestas por otros autores para la creación de interfaces de usuario, en particular aquellas que se dirigen hacia las tridimensionales o a las aplicaciones de estas, es resumida en [Molina, 2005b], publicado en las actas del First Int. Workshop on Methods and Tools for developing Virtual Reality Applications, MeTo-VR, dentro del congreso VSMM 2005 celebrado en Gante, Bélgica.
- La metodología IDEAS-3D, anterior al marco de trabajo TRES-D, es presentada en diferentes eventos, en el 10th Workshop on the Design, Specification and Verification of Interactive Systems, DSV-IS 2003,

celebrado en Madeira, Portugal [Molina, 2003a], en el congreso Interacción 2003 celebrado en Vigo [Molina, 2003b], y en el congreso HCI International 2003 celebrado en Creta, Grecia [Molina, 2003c].

- Uno de los meta-modelos aquí propuestos, el de elementos de la interacción, es descrito en detalle en [Molina, 2006b], publicado en las actas del 13th IEEE Mediterranean Electrotechnical Conference, MELECON 2006, celebrado en Benalmádena, Málaga, como contribución a la sesión especial titulada "New interaction paradigms in Virtual Environments".
- El modelo de proceso de la metodología TRES-D es descrito en [Molina, 2005b] y en [Molina, 2006b], antes reseñados, así como en [Molina, 2006d], publicado en las actas del 13th ACM Symposium on Virtual Reality Software and Technology, VRST 2006, celebrado en Limassol, Chipre.
- El árbol de decisión para la elección de técnicas de selección/manipulación de objetos en entornos virtuales inmersivos es recogido en [García, 2005a], en las actas del congreso HCI International 2005 celebrado en Las Vegas, Nevada, EE.UU., y en [García, 2005b], en las actas del congreso Interacción 2005 celebrado en Granada.
- El árbol de decisión para la elección de técnicas de introducción de texto en entornos virtuales basados en visiocasco se describe en [González, 2007], en las actas del congreso Interacción 2007 celebrado en Zaragoza.
- La librería VUIToolkit se presenta por primera vez en [Molina, 2005a], antes reseñada.
- Los tres primeros casos de estudio, aquellos que ilustran la aplicación de la propia metodología TRES-D, han sido recogidos en [Molina, 2006b] y [Molina, 2006d], también antes reseñadas.
- El cuarto caso de estudio, dedicado a la librería VUIToolkit, puede encontrarse en [Molina, 2006a], en las actas del 10th ACM Int. Conf. on Intelligent User Interfaces, IUI 2006, celebrado en Sydney, Australia, en [Molina, 2006c], actas del 6th Int. Conf. on Computer-Aided Design of User Interfaces, CADUI'2006, celebrado en Bucarest, Rumanía, y en [Vanderdonckt, 2007], actas del 1st Int. Workshop on Model Based Software Engineering for Ambient Intelligence Applications.

## 8.4 Proyectos

A lo largo de los años de desarrollo de este trabajo doctoral se ha tenido también la oportunidad de participar en diferentes proyectos a nivel regional y nacional, colaborando en diferentes tareas con otros investigadores de esta y otras Universidades, y cuya financiación ha permitido la asistencia a los eventos citados en el apartado anterior. Estos proyectos han sido:

- **“Metodologías de desarrollo de interfaces de usuario dinámicas”**, proyecto financiado por la Junta de Comunidades de Castilla-La Mancha (ref. PBC-03-003) a lo largo de tres años, de 2003 a 2005, y que se realizó en colaboración con la Universidad Politécnica de Valencia (UPV) y la Universidad Miguel Hernández de Elche.
- **“Entornos virtuales colaborativos aplicados a sistemas de aprendizaje”**, proyecto financiado también por la Junta de Comunidades de Castilla-La Mancha (ref. PAI-06-0093) por otros tres años, de 2006 a 2008, continuando la colaboración con la Universidad Miguel Hernández de Elche.
- **“Sistemas Adaptativos y Colaborativos con soporte WEB (ADACO)”**, proyecto financiado por el Ministerio de Ciencia y Tecnología (ref. TIN2004-08000-C03-01) también por un periodo de tres años, de 2005 a 2007, colaborando esta vez con el grupo GEDES de la Universidad de Granada y el grupo GRIHO de la Universitat de Lleida.

Además, el grupo de investigación se ha integrado en la **red de excelencia SIMILAR** que, financiada por el sexto programa marco de la Comisión Europea, tiene como meta la creación de interfaces persona-ordenador similares a la comunicación persona-persona, centrándose en la investigación en interfaces multimodales. La integración en esta red ha sido posible gracias a las buenas relaciones que se mantienen con el profesor Dr. Jean Vanderdonckt, del Belgian Computer-Human Interaction group (BCHI), gracias también a las cuales ha sido posible realizar una estancia en dicha Universidad, como se detalla a continuación.

## 8.5 Estancias en el extranjero

Una importante fracción del trabajo aquí expuesto es el resultado de una estancia de siete meses en la **Univesité catholique de Louvain (UCL)**, en el IAG –

Louvain School of Management, Louvain-la-Neuve, Bélgica, bajo la supervisión del profesor Dr. Jean Vanderdonckt. El enorme conocimiento que este profesor tiene del campo de las interfaces de usuario fue de gran ayuda para el avance de este trabajo. Allí surgió, por ejemplo, la idea de crear una librería de widgets 3D, y durante la misma se completó una primera implementación de lo que luego se llamaría VUIToolkit. Después, y continuando la colaboración con dicho profesor, llegaría la aplicación de esa librería al diseño y prototipado de interfaces de usuario distribuidas. Todo ello se ha traducido además en varias publicaciones conjuntas, véase [Molina, 2005a], [Molina, 2006a], [Molina, 2006c] y [Vanderdonckt, 2007].

## 8.6 Trabajos futuros

El desarrollo de un marco de trabajo como el que se ha propuesto aquí es una tarea muy ambiciosa, y por definición incompleta, ya que desde un principio no se ha perseguido una solución cerrada sino una que pueda adaptarse a los cambios que vayan surgiendo en el campo de las interfaces de usuario 3D, para crecer con él. A continuación se enumeran algunos de los aspectos en los que cabría trabajar en el futuro:

- En el modelo de objetos se justifica la introducción de sub-modelos para describir el objeto desde diferentes perspectivas, aunque compartiendo comportamiento y función. Sin embargo, sólo se ha detallado un sub-modelo, el gráfico. Sería interesante incluir otros, empezando por ejemplo con sub-modelo háptico. La estimulación del tacto es uno de los mayores retos de la Realidad Virtual, y una de las fronteras que separan a esta de crear mundos indistinguibles del real. Ya lo decía Bertrand Russell, “...es el tacto lo que nos proporciona el sentido de ‘realidad’... toda nuestra concepción de lo que existe fuera está basado en el sentido del tacto”.
- Sería deseable contar con una herramienta CASE que diera soporte al desarrollo con la metodología TRES-D propuesta, siguiendo el ejemplo de la que se desarrolló en su momento para la metodología IDEAS y que entonces también se quiso aprovechar para esa propuesta inicial que se llamó IDEAS-3D. También podría crearse un sitio Web que, sin llegar a ser una herramienta CASE al uso, pueda servir de referencia y guía para el desarrollador que siga la metodología TRES-D, siguiendo el ejemplo de la herramienta Web que creó Kaur. En caso de desarrollar esa Web, podría dotarse con la funcionalidad necesaria para servir de repositorio público de las guías aquí propuestas y cualquier otra ayuda que se desarrolle en el futuro.

- Como marco de trabajo, TRES-D ofrece un modelo de proceso y unas herramientas, pero no logra la automatización que se perseguía en aquella primera propuesta llamada IDEAS-3D. Si se abandonó en aquel momento fue porque los modelos no eran lo suficientemente ricos como para describir cualquier interfaz tridimensional, e incluso con ellos faltaban librerías software que permitieran transformar los modelos en la interfaz final. Es necesario seguir trabajando en los modelos para hacer posible esa automatización, llegando a la generación del propio código de la aplicación basada en esos mismos modelos. El hecho de que la librería VUIToolkit se base en el modelo CUI de UsiXML es toda una declaración de intenciones, pero aún queda trabajo por hacer.
- Sobre los árboles de decisión que se han propuesto como herramientas para la guía en la elección de técnicas de interacción, estos han sido dos y tenían por objeto dos de las llamadas tareas universales, una la selección/manipulación de objetos, y otra la entrada simbólica. Sin embargo, quedan otras tareas universales, como la navegación o el control del sistema. Así, al trabajo por hacer cabe añadir la creación de nuevos árboles para esas otras dos tareas. Pero también cabría replantearse el propio formato en el que se presentan estas guías, y en lugar de utilizar árboles de decisión podría optarse por cambiar a un formato de patrones en los que pueda detallarse también el contexto para que el que se sugiere una u otra técnica.
- De la librería VUIToolkit, decir que aún quedan por añadir elementos del modelo CUI de UsiXML, siendo el más llamativo el objeto Box. Este se utiliza para distribuir los widgets sobre la superficie del contenedor en base a un conjunto de relaciones espaciales, y su incorporación en la librería permitiría prescindir de aquellos campos con los que se especifica ahora la posición absoluta de cada elemento. Además, podría plantearse llevar esta librería más allá de los lenguajes VRML97/X3D y sus visores, creando por ejemplo una versión que pueda integrarse con la librería VR Juggler, tan utilizada en el desarrollo de entornos virtuales.
- El interés por la librería VR Juggler antes citada deriva, en particular, del uso que se está haciendo de ella en el laboratorio del grupo de investigación LoUISE. En particular, se ha empleado y se sigue usando para la creación de entornos virtuales colaborativos o CVE's (por sus siglas en inglés, esto es, Collaborative Virtual Environments). Para este tipo de aplicaciones también cabe utilizar la metodología TRES-D, pues no en vano se ha resaltado el avatar del usuario dentro del modelo de

objetos y de los elementos de la interacción, todo un guiño hacia los mundos habitados. En la metodología SENDA ya se entendió que para definir estos avatares se precisaba una nueva herramienta, y por ello sus autores introdujeron los conceptos de uso. Sin embargo, más allá del uso de un mundo tridimensional como medio para la conversación entre avatares, la colaboración en entornos tridimensionales plantea importantes retos, toda una nueva línea de investigación.

Con todo, el campo de las interfaces de usuario 3D ofrece muchas oportunidades para la investigación. La atención aquí se ha centrado en el propio desarrollo de estas interfaces, y aunque la principal contribución de este trabajo es una metodología para dicho desarrollo, también se ha querido aprovechar el mismo para experimentar diferentes usos de esas interfaces, como muestran los casos de uso expuestos. No puede negarse que con ellos también se ha querido contribuir a la búsqueda de esa aplicación definitiva a la que tantos aluden.

El mercado de consumo, y en particular el de los videojuegos, ha estado en el punto de mira de al menos uno de esos casos de estudio, como vehículo para lograr la popularización de estas interfaces. Pero frente a los intentos de llevar a ese exigente mercado la tecnología de visiocascos y gráficos 3D estéreo, y la de los guantes de datos y el reconocimiento de gestos manuales, ha sido otra tecnología, la de los acelerómetros que se incluyen en tantos headtrackers, la que implantada en un mando de juego ha logrado triunfar en ese mercado de la mano de Nintendo y su famoso Wiimote. Su popularización y las inquietudes de muchos hacen que cada día se encuentre un nuevo uso a esta tecnología de las interfaces tridimensionales.

Y es que no hay duda de los beneficios que tiene seguir una buena metodología para el desarrollo de la interfaz, pero tampoco la hay de que, al final, la imaginación y el ingenio dependen del propio desarrollador, y el veredicto, justo o no, le corresponde al usuario.

# 9

## Conclusions

### 9.1 Summary

Three-dimensional user interfaces have not reached the same state of maturity that can be observed in desktop interfaces, where well-funded methods and practices allow a rapid development, highly automated. Starting from past experiences in conventional user interfaces, this doctoral dissertation then focused on existing approaches to the design and creation of three-dimensional interfaces. Many proposals were found, and many more may have been ignored but, beyond the number, attention must be paid to the different points of view from where each author approaches to the given problem. Following a careful review of all of them, it was concluded that a new proposal was needed. That proposal was named TRES-D, which stands for “ThRee dimEnsional uSer interface Development”. However, this work is not simply a sequence of phases, stages and activities.

The following list sums up the contributions of this doctoral dissertation. Many of them have been presented in national and international scientific events, being published in their proceedings. It is worth mentioning the participation, every year since 2003, in *Interacción*, an international conference organized by AIPO, and the participation in other events such as *Web3D*, *IUI* and *VRST*, which are organized by ACM. All these publications are also cited in the next list, matching them with their related contribution:

- ✓ Many terms and definitions are related to 3D, so the first step to take was to define what a 3D user interface is. Together with the proposed definition, a list of past and current applications was given, some being successful, others not.
- ✓ The well-known reality-virtuality continuum by Milgram and Kishino has been transformed into a new digital-virtual-real continuum. A first version of this design space is found in [Molina, 2005a], introduced to explain the virtualization of user interfaces. This new continuum makes room for other applications of 3D user interfaces, such as 3D desktops.

- ✓ The different elements that compose a 3D user interface have been identified and described, being these the objects, their behaviour and interaction, and the space. Later, this work served as a base for the definition of the three meta-models included in the TRES-D proposal.
- ✓ The problems that the design of each element, and the whole interface, encompasses have been studied. On the one hand, interaction does not follow the turn-taking dialogue typically found in conventional interfaces. On the other hand, there is neither standard set of interaction techniques nor controls. These findings encouraged the proposal of design tools and toolkits in the TRES-D framework.
- ✓ Twenty-six existing methodologies have been described and analyzed, grouping them in seven different categories according to some similarities found in these approaches. Most of them are aimed at the development of 3D user interface applications, such as virtual worlds. This large review is summarized in [Molina, 2005b].
- ✓ Based on the experience gained through applying the IDEAS methodology to the development of conventional interfaces, a first proposal, previous to TRES-D, was presented as an extension to that methodology, named IDEAS-3D. This is presented in [Molina, 2003a], [Molina, 2003b] and [Molina, 2003c].
- ✓ Leaving aside IDEAS-3D, and heading to the TRES-D framework, three new meta-models were then introduced to better understand 3D user interfaces and facilitate their design and implementation. One meta-model tackles objects and their classification, a second one defines interaction elements and their relationships, and the last one is devoted to space. From them, the interaction meta-model is described in detail in [Molina, 2006b].
- ✓ The TRES-D process is found in [Molina, 2005b], [Molina, 2006b], and [Molina, 2006d]. This process is characterized, first, by dividing the development into two main phases. At the end of the first phase –the previous study–, a proposal is presented, and only if it is accepted, then the second phase follows –the detailed study–. This second phase includes design and implementation, where user interface designers and programmers tackle tasks and interaction and, in parallel, artists and digital content creators focus on objects and content. Design is also divided in two horizontal abstraction levels, not only to differentiate between interaction tasks and techniques as in other approaches, but also between artists' impressions and object specs given to content creators.

- ✓ As part of the framework, two guidance tools have been presented, being these two decision trees that help designers selecting interaction techniques. One of them guides towards the appropriate object selection/manipulation technique in immersive environments, and it is described in [García, 2005a] and [García, 2005b]. The other one guides towards the right text input technique in HMD-based environments, and it is described in [González, 2007].
- ✓ In addition to design time tools, a toolkit for programmers has also been included, to help them translating flat interfaces to three-dimensional environments. Its name is VUIToolkit, a set of 3D widgets meant for virtualized user interfaces, which was first introduced in [Molina, 2005a].
- ✓ Finally, the proposed framework has been brought to practice through four case studies. Three of them demonstrated the process of the TRES-D methodology and has been detailed in [Molina, 2006b] y [Molina, 2006d]. The fourth one showed the application of VUIToolkit in the design and prototyping of distributed user interfaces, as seen in [Molina, 2006a], [Molina, 2006c] and [Vanderdonckt, 2007].

At the same time this doctoral work was carried out, it has been able to participate in regional and national projects funded by Junta de Comunidades de Castilla-La Mancha (PBC-03-003 and PAI-06-0093) and the Ministerio de Ciencia y Tecnología (TIN2004-08000-C03-01), tightly related to methodologies and virtual environments.

Part of this work has been done during a stay at the Université catholique de Louvain (UCL), Belgium, as a member of the Belgian Computer-Human Interaction (BCHI) group, and being supervised by professor Dr. Jean Vanderdonckt. This collaboration resulted in several publications co-authored with this professor. Thanks to him, our research group became fellow member of the SIMILAR Network of Excellence, funded by the 6<sup>th</sup> framework program of the European Commission, where the main aim is creating human-machine interfaces SIMILAR to human-human communication.

All in all, the proposed framework is not presented as a finished work but as a living being that must continuously evolve in order to accommodate the advances in the field of 3D user interfaces. Some ideas for further work are:

- The concept of sub-models is presented in the object meta-model so that each object can be described from different points of view, but sharing the same behaviour and function. However, only the graphics one has been given, others as the haptics one should be added.

- Supporting the proposed methodology with a CASE tool is desirable, as in IDEAS. In the simplest case, a Web site could guide developers in their work, as Kaur proposed, but even in this case the site could be enhanced with services to store and retrieve guides and tools, as a public repository.
- TRES-D presents a process model and recommends tools for each activity, but it does not achieve the automation that was aimed when the first proposal, IDEAS-3D, was presented. Further work on the proposed meta-models must be done in order to address that goal. The fact that VUIToolkit is based on the CUI model of UsiXML is one step in that direction, but there is still work to be done.
- New decision trees could be produced for other universal tasks, such as navigation and system control. Furthermore, it should be discussed the way these tools are presented, as a pattern format could be more useful to detail when each recommendation applies.
- As for the VUIToolkit, there are still CUI classes that can be added, such as the CUI Box. VUIToolkit could also be ported to other development environments beyond Web3D languages, such as VR Juggler, a widely-used programming library for VR applications.
- VR Juggler is also used at our lab, currently to develop Collaborative Virtual Environments or CVEs. For this kind of environments, the proposed methodology also applies, with avatars being considered an important class in the object meta-model. In habited virtual worlds, the authors of SENDA realized that specific tools were needed, and so they proposed the Use Concept. However, beyond conversational worlds, collaboration puts forward important challenges that open a whole new line of research.

# Anexo A

## La librería VUIToolkit

### A.1 Introducción

Existen muchas razones para dar soporte a las interfaces de usuario 2D en entornos tridimensionales. Algunas de esas razones ya se expusieron en el segundo capítulo, concretamente en el apartado 2.4.1, en donde se citaba la manipulación indirecta y el legado de aplicaciones 2D. Esas y otras razones pueden encontrarse en [Molina, 2005a]. Pero no es este anexo el lugar para discutir esas razones, se da por hecho que las interfaces de usuario 3D deben soportar muy bien la interacción en dos dimensiones, y lo que se pretende con la librería que aquí se describe es facilitar esa integración.

A la hora de sumergir una interfaz plana en las tres dimensiones del espacio, esta suele mantener sus dos dimensiones, dibujándose sobre un polígono como una textura, esto es, una imagen con la que se trata de dar mayor detalle a la geometría que representa el polígono, pero que en esta ocasión representa los botones, las cajas de texto y otros widgets que conforman dicha interfaz. Este es el concepto de “application texture”, sobre el que declaraba su interés T. Parisi en los foros que el Web3D Consortium aloja en su sitio Web [Web3D, URL], y que precisa que las aplicaciones puedan volcar su salida gráfica sobre un bitmap que, a su vez, pueda ser usado como textura por el motor de gráficos 3D. Aunque sencillo de explicar, llevarlo a la práctica exigía un cambio en la infraestructura software, ya que hasta hace poco los gráficos 3D vivían en un contexto diferente al de esos gráficos 2D –véase los comentarios de McDaniel en [WWW-VRML, URL], §3/7/2003-. El cambio no se ha hecho palpable hasta la llegada del nuevo sistema operativo Windows Vista de Microsoft, aunque ya en la versión Windows 2000 se unificaron los bitmaps GDI y las texturas DirectX –tal y como explica Dantzich en [3D UI, URL], §5/10/1999-.

Sin embargo, la anterior solución olvida que muchas de esas interfaces 2D exhiben, en realidad, dos dimensiones y media (2.5D) pues, como también se explicó en el segundo capítulo, transmiten claves de profundidad cuando pintan sombras o superponen unos elementos sobre otros –véase apartado 2.4.2-. Al sumergir entonces dichas interfaces en un entorno tridimensional, esas claves pueden dejar de ser simuladas para darles su verdadera dimensión. Es lo que en esta Tesis se ha denominado “interfaz de usuario virtualizada” o VUI

(*Virtualized User Interface*). Por ejemplo, cuando se muestra un botón, las sombras que se dibujan sobre él le otorgan un relieve aparente, y con esas mismas sombras se juega para darle la apariencia de estar hundido cuando es seleccionado. Sin embargo, en un entorno tridimensional, ese relieve podría ser verdadero, no simulado, de modo tal que al pulsar o soltar el botón, este se desplazara realmente en la tercera dimensión, sustituyendo el engaño al que se recurre en los gráficos 2D por una verdadera representación del botón a través de los gráficos 3D. De esta forma, el comportamiento del widget, transformado ahora en un widget 3D, se ajusta mucho mejor al modelo mental que el usuario tiene del mismo.

Entre la idea de la “aplicación como textura” y la de “interfaz de usuario virtualizada”, cabe aquí recordar a N. Trevett en una de las charlas invitadas al simposio Web3D celebrado en el año 2000, destacando en aquella ocasión la cuestión de la transición de un tipo a otro de interfaz de usuario y señalando que, para hacer las interfaces de usuario útiles para todos los usuarios, era necesario transformar a las 3D las interfaces planas procedentes del tradicional escritorio 2D. De no ser así, al lanzar una aplicación con una interfaz convencional en un escritorio 3D, por ejemplo, todo vuelve a las dos dimensiones del plano tan pronto como la aplicación toma el control, perdiéndose la experiencia 3D en la transición. Es por ello que en este trabajo se apuesta por la virtualización, migrando de forma progresiva las interfaces de usuario 2D a los entornos 3D pero explotando en todo momento las características propias de estos últimos.

El resultado es la librería VUIToolkit, una librería o “toolkit” de widgets 3D de la que se han creado dos versiones, una con el lenguaje VRML97 [VRML Spec, URL] y otra con el lenguaje X3D [X3D Spec, URL], en ambos casos como un conjunto de prototipos. Los lenguajes VRML97 y X3D fueron elegidos por ser dos estándares abiertos, el primero es la versión del lenguaje VRML más popular y mejor adoptada por los diferentes visores –en inglés, “browsers” o “plug-ins”-, la segunda es la última versión de ese lenguaje en ser estandarizada, y cuyo soporte en esos mismos visores y otros nuevos aún está siendo depurado y completado, lo que representó un mayor reto a la hora de crear la librería VUIToolkit para dicho lenguaje. Tanto con el uno como con el otro, la arquitectura software que se siguió para crear los prototipos que forman parte del toolkit se basó en parte en el trabajo y los desarrollos del antiguo grupo de trabajo en widgets para VRML [VRML WWG, URL], y la característica original del toolkit es la de transformar los widgets 2D planos en su verdadera representación 3D.

Pero en lugar de tener que crear diferentes versiones de una misma interfaz de usuario para su representación en dos o en tres dimensiones, cada interfaz es especificada una única vez en el lenguaje UsiXML –el cual se citó ya en el apartado 2.7.9 pero que aquí se presentará en mayor detalle- y a partir de ella es

representada, en su caso, como una interfaz de usuario final en 3D gracias a la librería VUIToolkit. Para ello, el desarrollo de esta librería partió de las clases descritas en el modelo de interfaz de usuario concreta –independiente de cualquier toolkit- del lenguaje UsiXML. Así, cada prototipo de la librería cuenta con una lista de parámetros en la que se incluyeron los atributos de la clase UsiXML a la que correspondía, añadiéndose después otros nuevos necesarios para transformar el objeto de interfaz concreto en un objeto de interfaz final representado en un entorno 3D. De esta forma, la interfaz de usuario especificada en el lenguaje UsiXML puede ser transformada en una interfaz para mundos VRML97 o X3D de forma automática, haciendo uso del conjunto de prototipos de la librería VUIToolkit.

Esta librería fue descrita por primera vez en [Molina, 2005a], y aplicaciones de la misma se publicaron en [Molina, 2006a] y [Molina, 2006c]. En lo que sigue, se abordará el marco de referencia Cameleon, el lenguaje UsiXML y el modelo para la interfaz de usuario concreta o nivel CUI del marco de referencia. Después se discutirá cómo hacer corresponder ese modelo con los nodos, rutas y otros mecanismos de los lenguajes VRML97 y X3D. Entonces se profundizará en el diseño de la librería, repasando el trabajo previo de otros autores en este campo para dar con la arquitectura más idónea. Finalmente, se dan los detalles de cada uno de los prototipos que forman esta librería.

## A.2 El marco de referencia Cameleon y el lenguaje UsiXML

Cameleon [Calvary, 2003] es un marco de referencia para el desarrollo de interfaces de usuario a través de múltiples caminos –en inglés, “multi-path”- y que define los pasos para el desarrollo de interfaces de usuario para aplicaciones interactivas en múltiples contextos de uso. En el caso más simple, ilustrado en la Figura A.1, estructura los procesos de desarrollo para dos contextos de uso en cuatro pasos, en cada uno de los cuales es posible manipular cualquier aspecto de interés como un modelo o como la propia representación de la interfaz de usuario:

1. **Interfaz de usuario final** (FUI, *Final User Interface*): es la interfaz de usuario en operación, esto es, cualquier interfaz de usuario que corra en un sistema informático particular, ya sea como código nativo –p. ej. tras la compilación en un entorno de desarrollo interactivo- o de forma interpretada –p. ej. a través de un navegador Web-.
2. **Interfaz de usuario concreta** (CUI, *Concrete User Interface*): sintetiza una interfaz de usuario abstracta para un contexto de uso dado en un conjunto de objetos de interacción concretos (CIOs, *Concrete*

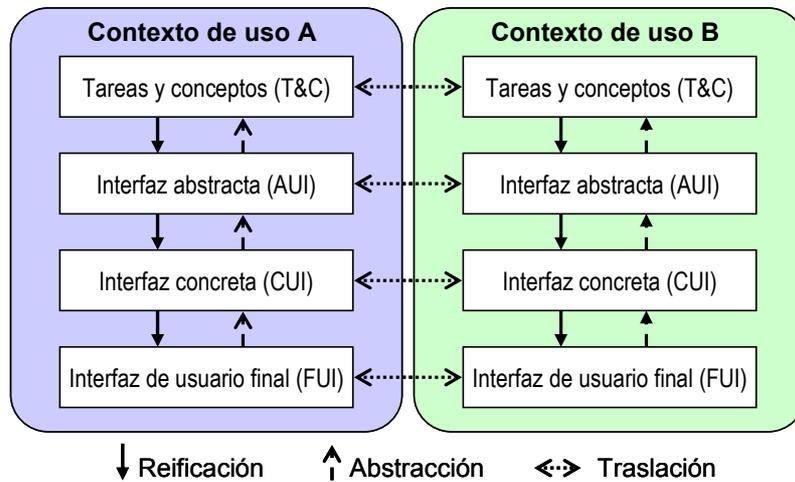


Figura A.1 El marco de trabajo Cameleon

*Interaction Objects*) [Vanderdonckt, 1993] con el fin de definir la disposición de los widgets y la navegación por la interfaz. Esta capa abstrae una FUI en una definición de interfaz de usuario que es independiente de cualquier plataforma informática. Aunque una CUI hace explícito el “look and feel” final de una FUI, se trata aún de un prototipo –en inglés, “mock-up”- que sólo puede correr en un determinado entorno. Una CUI puede considerarse también como una reificación de una AUI de la capa inmediatamente superior.

3. **Interfaz de usuario abstracta (AUI, *Abstract User Interface*):** define contenedores y componentes individuales abstractos [Limbourg, 2004] por medio de la agrupación de tareas, al tiempo que define un esquema de navegación entre los contenedores y selecciona componentes individuales abstractos (AIOs, *Abstract Interaction Objects*) [Vanderdonckt, 1993] para cada concepto de modo que sean independientes de cualquier modalidad. Una AUI abstrae una CUI en una definición de interfaz de usuario que es independiente de cualquier modalidad de interacción –p. ej. interacción basada en gráficos, interacción basada en la síntesis y reconocimiento del habla, interacción basada en cámaras de video, etc.-. Una AUI puede también considerarse como una expresión canónica de la representación de los conceptos y tareas del dominio de un modo que es independiente de cualquier modalidad de interacción.
4. **Tareas y conceptos (T&C, *Task and Concepts*):** describe las diferentes tareas que deben ser llevadas a cabo y los conceptos del dominio que

requieren dichas tareas para ser ejecutadas. Estos objetos se consideran instancias de clases que representan los conceptos.

Este marco de trabajo exhibe tres tipos de transformación: (1) Abstracción –y, respectivamente, Reificación (2)- es el proceso que da como resultado entidades que son más abstractas –respectivamente, concretas- que las entidades que se toman como entrada de este proceso –abstracción es, de este modo, lo opuesto a reificación-; y (3) Traslación es un proceso que da como resultado entidades para un contexto de uso particular a partir de otras entidades de un paso de desarrollo igual pero dirigido a un contexto de uso diferente. En lo que respecta al diseñador, el desarrollo de interfaces de usuario de múltiples caminos hace referencia a un método de ingeniería de interfaces de usuario que le permite: (1) comenzar una actividad de desarrollo en cualquier punto de entrada del marco de trabajo; y (2) contar con el apoyo necesario para la realización de todos los tipos de transformaciones básicas y sus diferentes combinaciones –véase Figura A.1-.

Para llevar este marco de trabajo a la práctica, se precisa un lenguaje que facilite la expresión y la manipulación –p. ej. creación, modificación y borrado- de los modelos en cada uno de los pasos del desarrollo y para cada contexto de uso. Con este fin se introdujo y se definió el lenguaje UsiXML (*USer Interface eXtensible Markup Language*) [UsiXML, URL] –véase también [Limbourg, 2004]-. Este lenguaje basado en XML consta de alrededor de 150 conceptos diferentes con los que es posible expresar los modelos a diferentes niveles de abstracción, y realizar las transformaciones de abstracción, reificación y translación ilustradas en la anterior figura.

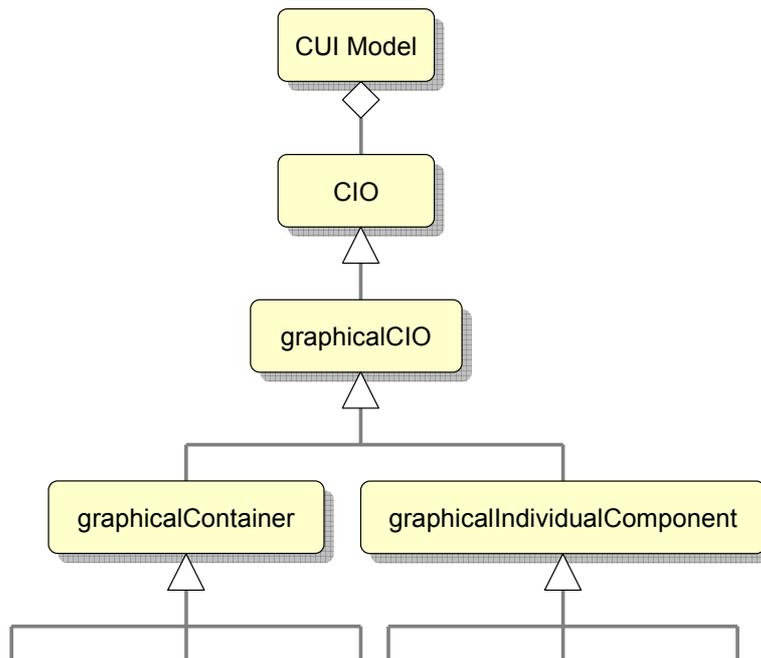
En lo que resta se asumirá que la interfaz de usuario está descrita principalmente a nivel CUI, ya que este nivel es independiente de cualquier plataforma informática y, en particular, de cualquier toolkit. A partir entonces de las mismas especificaciones de una CUI, es posible obtener una interfaz gráfica de usuario final en paralelo a la virtualización de esa misma interfaz. Es por ello que el siguiente apartado se centra en el nivel CUI.

### A.3 La interfaz de usuario concreta en UsiXML

En la Figura A.1, el último nivel de abstracción antes de obtener una interfaz de usuario final es el CUI. Una descripción de una CUI puede obtenerse a través de un proceso de ingeniería que aborde, de forma sucesiva, los niveles T&C y AUI, o bien de forma directa. Una CUI es tomada como una descripción que no hace referencia a ninguna plataforma informática particular, ni a ningún toolkit de esas plataformas. Por esta razón, el modelo CUI consta de una descomposición jerárquica de CIO's. Un objeto de interacción concreto o CIO se define como

cualquier entidad de una interfaz de usuario que los usuarios pueden percibir –tal como un fragmento de texto, una imagen o una animación- y/o que pueden manipular –tal como un botón, una lista o una casilla de validación- [Vanderdonckt, 1993][Limbourg, 2004]. Un CIO viene caracterizado por varios atributos, como estos que siguen, aunque no se limitan sólo a ellos: *id*, *name*, *icon*, *content*, *defaultContent*, *defaultValue*.

Dado que un CIO es independiente de cualquier plataforma informática, se desconoce a este nivel cuál será el toolkit que se usará para la interfaz de usuario final, pero no la modalidad de interacción. De este modo, cada CIO puede ser derivado en sub-CIO's dependiendo de la modalidad elegida: *graphicalCIO* para interfaces gráficas de usuario, *auditoryCIO* para interfaces por voz, *3dCIO* para interfaces de usuario tridimensionales, etc. Aquí se prestará atención a *graphicalCIO* dado que representa a las entidades básicas de las interfaces 2D y, ahora también, a la virtualización de estas, los widgets 3D. Cada *graphicalCIO* hereda entonces las propiedades de la clase *CIO* padre, y cuenta con sus particulares atributos, como por ejemplo: *isVisible*, *isEnabled*, *fgColor* y *bgColor* para indicar los colores de primer plano y de fondo, etc.



**Figura A.2** Descomposición de un modelo de interfaz concreta en CIO's

De nuevo, cada *graphicalCIO* es sub-tipado en una de las dos posibles categorías –véase Figura A.2-: por una parte, *graphicalContainer* para todos

aquellos widgets que pueden contener a otros, como por ejemplo *page*, *frame*, *window*, *dialog box*, *box*, *table* y sus derivados correspondientes; y por otra parte, *graphicalIndividualComponent* para todos los demás widgets que usualmente se encuentran en dichos contenedores. Un *graphicalIndividualComponent* no puede descomponerse en otros CIO's. El lenguaje UsiXML soporta toda una serie de widgets definidos como *graphicalIndividualComponent*, como por ejemplo: *textComponent*, *videoComponent*, *imageComponent*, *imageZone*, *radioButton*, *toggleButton*, *icon*, *checkbox*, *item*, *comboBox*, *button*, *tree*, *menu*, *menuItem*, *drawingCanvas*, *colorPicker*, *hourPicker*, *datePicker*, *filePicker*, *progressionBar*, *slider*, y *cursor*.

Gracias al citado mecanismo de herencia, todo elemento final del modelo CUI hereda las propiedades de su padre y ancestros dependiendo de la categoría a la que pertenezcan. Las propiedades que se han incluido en cada clase de UsiXML lo han sido porque pertenecen a la intersección de los conjuntos de propiedades de entidades similares en los principales toolkits y administradores de ventanas, tales como Windows GDI, Java AWT y Swing, y HTML. Sólo se han tomado entonces aquellas propiedades de interés que son comunes a los diferentes conjuntos estudiados. De esta forma, un CIO puede especificarse independientemente del hecho de que sea luego representado como Java, HTML o, incluso, como se verá a continuación, VRML97/X3D. Esta cualidad es la que a menudo se cita como **independencia de plataforma**.

#### A.4 Correspondencia entre UsiXML y VRML97/X3D

La independencia de plataforma de la que se ha hablado en el anterior apartado plantea, sin embargo, el reto de lograr una correspondencia correcta y apropiada entre la especificación independiente de la plataforma (el modelo CUI) y una particular para una plataforma (el FUI). Los lenguajes VRML97 y X3D ofrecen un amplio abanico de facilidades para representar mundos virtuales para la Web 3D, pero desgraciadamente ninguno de estos estándares incluyen primitivas que puedan relacionarse directamente con los CIO's definidos en el nivel concreto de UsiXML. La solución que se propuso pasaba entonces por implementar, a partir de las facilidades de esos lenguajes VRML97 y X3D, un repertorio de widgets 3D que permitiese establecer la correspondencia uno-a-uno con los CIO's de ese nivel concreto. El resultado es la librería VUIToolkit, en sus dos versiones, una para cada lenguaje. Gracias a ella, es posible generar nuevas interfaces finales para mundos virtuales VRML97 o X3D a partir de especificaciones de interfaces de usuario ya descritas con UsiXML y que, hasta el momento, sólo habían sido procesadas para obtener interfaces finales en entornos bidimensionales.

Aún y todo, ninguna de las dos versiones del toolkit cubre todo aquello que se especifica en UsiXML, y al mismo tiempo se han tenido que introducir nuevos conceptos que no se contemplaban en el propio UsiXML. Y es que, a la hora de tender puentes entre el nivel CUI de UsiXML y los mundos virtuales para la Web 3D, pueden darse los distintos casos que a continuación se describen, y que se han tratado de resumir en la Tabla A.1:

1. **Correspondencia directa** entre un CIO de la CUI y una primitiva VRML97 o X3D. Esta correspondencia puede ser uno-a-uno (biyectiva) o uno-a-muchos (composición de objetos). Como se ha apuntado antes, no es posible establecer una correspondencia uno-a-uno dado que estos lenguajes para la Web 3D definen elementos básicos tales como formas, sensores, rutas y un lenguaje de script que deben combinarse para crear elementos interactivos, como son los widgets 3D. El nuevo estándar X3D no cambia esta situación, a pesar de introducir nuevos nodos de geometría 2D que facilitan la tarea de dibujar interfaces 2D en un mundo 3D. Por esta razón, los diferentes elementos que forman el toolkit tuvieron que ser implementados prácticamente de cero, ensamblando formas para darles el aspecto deseado y añadiéndoles los correspondientes comportamientos a través de sensores, rutas y scripts.
2. **Nueva correspondencia** entre un CIO y su equivalente en VRML97 o X3D. Los lenguajes VRML97 o X3D no proporcionan ningún elemento que pueda hacerse corresponder con un CIO. En este caso, es necesario llenar este vacío introduciendo un nuevo widget en el mundo 3D a través de su apropiada implementación. Así se ha hecho con los CIO's que se han empleado como punto de partida para el toolkit, para cada uno de los cuales se ha incluido un nuevo widget 3D en dicho toolkit que puede usarse para representar al CIO en la interfaz final. Sin embargo, esta correspondencia no es completa, pues algunos de los atributos definidos en el nivel CUI de UsiXML no se emplean finalmente en el mundo 3D, al tiempo que se han añadido otros atributos por ser necesarios para la correcta descripción de los widgets en el mundo 3D, como por ejemplo aquellas propiedades que tienen por fin especificar la posición y las dimensiones del widget.
3. **Correspondencia imposible**. A pesar del toolkit implementado y de los diferentes widgets incluidos en él, aún quedan otros CIO's cuyo equivalente en VRML97 o X3D no ha sido implementado aún o es poco practicable o imposible su implementación. Un ejemplo de ello es el CIO de tipo box definido en el nivel CUI de UsiXML, utilizado para especificar la disposición o "layout" de los widgets como una descomposición jerárquica de cajas verticales y horizontales, y

siguiendo un conjunto de relaciones lógicas. Dado que el presente toolkit no incluye dicho CIO, en su lugar debe llevarse a cabo una transformación que tome la interfaz concreta como entrada y calcule las correspondientes coordenadas y dimensiones de cada widget para poder mostrar la interfaz final.

Nivel CUI de UsiXML	VRML97 y X3D	
CIO individual $c$	$\exists$ widget 3D $w / f(c) = w$	Correspondencia directa
	$\exists w_1, w_2 \dots w_n / f(c) = \sum w_i$	Nueva correspondencia
	$\nexists$ widget 3D $w / f(c) = w$	Correspondencia imposible

**Tabla A.1** Correspondencias posibles entre el nivel CUI de UsiXML y VRML97/X3D

Otro aspecto a tener en cuenta a la hora de relacionar una interfaz concreta y su representación final en VRML97 o X3D es que en la definición de la interfaz gráfica de usuario se asume que la plataforma final contará con una pantalla de presentación, en la cual se dibujarán los diferentes elementos, y con la que el usuario podrá interactuar por medio del teclado y el ratón. Al llevar esta idea a un mundo virtual, se hace necesario también definir cuáles son las correspondencias con la pantalla, el teclado y el ratón.

Con respecto a la pantalla, el toolkit creado incluye un prototipo llamado “Screen”, cuya función en dicho toolkit es doble. Por una parte, este prototipo hace posible la especificación de un rectángulo que represente la típica pantalla, también rectangular, que puede encontrarse en casi cualquier sistema informático, detallando sus dimensiones –especificando su anchura y altura en metros, o bien la longitud de su diagonal en pulgadas y su razón de aspecto- y su resolución en pixels. Este prototipo acepta además un bitmap como imagen de fondo, y cuenta con un sensor que envía eventos de salida conteniendo las coordenadas del pixel al que el usuario señala con su dispositivo apuntador. La segunda función del prototipo “Screen” es la de servir como contenedor de elementos de una típica GUI, elementos que pueden ser cualquiera de los incluidos en el toolkit. Para ello, el prototipo cuenta también con un campo llamado *children*. Este recuerda a los campos de mismo nombre que pueden encontrarse en otros nodos VRML97 o X3D, como por ejemplo *Group* o *Transform*. De hecho, el mecanismo usado en esos lenguajes para agrupar nodos bajo un mismo padre por medio de un campo *MNode*, como es el caso del campo *children*, es el mismo que se emplea en el propio toolkit para establecer las relaciones jerárquicas entre los diferentes *graphicalCIO*'s.

En lo que respecta al teclado y al ratón, en las especificaciones de los lenguajes VRML97 y X3D se asume que el usuario interactúa con el mundo virtual usando un dispositivo apuntador, y muchos de los sensores están dirigidos hacia la

interacción usuario-mundo basada en dicho dispositivo. De esta forma, el ratón podría hacerse corresponder con el dispositivo apuntador que el usuario necesita para explorar mundos VRML97 o X3D, y de hecho en la mayoría de las ocasiones ese dispositivo suele ser el ratón del propio PC en el que corre el visor de mundos virtuales para la Web 3D. Y, de igual forma, el teclado bien podría ser el del mismo PC, y en este caso lo único que haría falta es contar con un sensor o conjunto de sensores que respondieran a la pulsación de las teclas. Esto es posible con el lenguaje X3D puesto que introduce este tipo de sensores, como por ejemplo *StringSensor*. Sin embargo, resulta imposible de llevar a VRML97 sin recurrir a extensiones definidas fuera del estándar, como por ejemplo el *KbdSensor* que implementa e interpreta el visor Cortona de la compañía ParallelGraphics. Aún y todo, quedaría una alternativa al uso de extensiones como esa, y es la introducción en el mundo de un teclado virtual que pudiera ser manejado con el anterior dispositivo apuntador, esto es, pulsando teclas virtuales con el ratón.

## A.5 Diseño e implementación de la librería VUIToolkit

A la hora de crear la librería VUIToolkit, se decidió abordar de forma paralela dos versiones de la misma, una para el lenguaje VRML97 [VRML Spec, URL], y otra para la tercera y más reciente versión de dicho lenguaje, conocida no ya como VRML sino como X3D (*eXtensible 3D graphics*) [X3D Spec, URL]. El uso de VRML97 nos permite aprovecharnos de un buen número de visores bien conocidos desde hace años –como por ejemplo el visor Cortona antes mencionado–, y que a lo largo de este tiempo han ido madurando ofreciendo mayor robustez y fiabilidad –siguiendo con Cortona, este andaba ya por su versión 5.1 en el momento de escribir estas líneas–. Por otra parte, la versión para X3D nos permite hacer uso de las nuevas características que se incorporan en esta tercera versión, como por ejemplo los nodos de geometría 2D, los sensores para teclado y la posibilidad de especificar en nodos *Script* campos que, al mismo tiempo, pueden recibir y generar eventos, características todas ellas no presentes en VRML97 y cuyo uso requeriría depender de extensiones propietarias. Salvando esas diferencias, la arquitectura del repertorio de widgets 3D que forman la librería VUIToolkit es idéntica en ambas versiones, tanto para VRML97 como para X3D, y está basada en el mecanismo de prototipado presente en ambos lenguajes. Los siguientes apartados abordan esa arquitectura y la implementación de los widgets 3D en detalle.

### A.5.1 Diseño de la arquitectura

La idea de widgets en entornos 3D ha provocado muchos e interesantes debates en listas de correo como en la lista pública dedicada al lenguaje VRML – [WWW-VRML, URL], ahora [VRML Email, URL]-, siendo uno de los temas tratados cuál debería ser el conjunto de widgets estándar para este tipo de entornos. El interés en este tema era tal que incluso llegó a crearse un grupo de trabajo sobre widgets en el marco de lenguaje VRML, el llamado VRML Widget Working Group [VRML WWG, URL]. Dos eran las arquitecturas que este grupo de trabajo planteaba para los widgets, una por capas según las ideas de P. Isaacs y S. Becker, y otra basada en componentes según la idea de G. Seidman.

La primera de esas arquitecturas se divide en tres capas, estando la primera de estas dedicada al control del tiempo y a la interacción puntero-superficie en contacto, la segunda a la interpretación y control de esa interacción (determina el “feel” de un widget”), y la tercera a la geometría (el “look” de un widget). La segunda arquitectura identifica cinco componentes básicos, que son los sensores, el pegamento interno, la geometría, el pegamento externo y las interfaces de usuario de Realidad Virtual (análogos a las GUI’s). En base a esas arquitecturas se propusieron un buen número de diferentes widgets, de los cuales cabe destacar el conjunto del propio G. Seidman, bastante completo. Sin embargo, todo el trabajo que se realizó entonces parece que no fructificó, el grupo cesó su actividad y el lenguaje VRML siguió sin contar con un conjunto estándar de widgets.

Posteriormente, con el desarrollo de la tercera versión del lenguaje, X3D, aparecieron nuevos proyectos que también abordaban el problema de los widgets en esos entornos 3D. Así, otro de los proyectos que se han realizado en este campo y que merece la pena destacar es el conjunto de componentes para entornos 3D del proyecto [CONTIGRA, URL]. Este proyecto no tiene como objetivo único la estandarización de un repertorio de componentes, sino también de metáforas y técnicas de interacción, estructurados todos ellos en forma de jerarquía. Cada componente se especifica utilizando un lenguaje basado en XML, lo cual proporciona al proyecto todas las virtudes de XML, y se aprovecha del lenguaje X3D para describir la apariencia y geometría final de cada uno de esos componentes. Entre esos componentes encontramos algunos widgets clásicos en aplicaciones 2D y 3D, como el botón o el conmutador –en inglés, “toggle-button”-, pero también otros componentes específicos para entornos 3D, como por ejemplo el menú de anillo o “ring menu”. La ventaja de este repertorio es que **separa geometría y comportamiento**, siguiendo ideas similares a las utilizadas por el grupo de trabajo de widgets para VRML, y

permite de esta forma que el autor escoja la forma que más se ajusta a su proyecto. La desventaja es que muchos de los componentes de la jerarquía no han sido aún desarrollados o no están disponibles al público, lo que claramente limita sus posibilidades para erigirse como un estándar en el lenguaje X3D, y no está clara la forma en que una aplicación tradicional 2D podría ser trasladada a un entorno 3D utilizando ese conjunto de componentes, ya que algunos de los widgets utilizados en aplicaciones 2D no cuentan con una correspondencia directa en esa jerarquía.

Volviendo entonces a la librería VUIToolkit, para el diseño de su arquitectura se tuvieron presentes las ideas de los anteriores trabajos, llevándolas a la práctica a través del mecanismo de creación de prototipos o PROTO's, un mecanismo que puede encontrarse tanto en el lenguaje VRML97 como en X3D y que puede asemejarse al de "clase" que ofrecen los lenguajes de programación orientados a objetos. De esta manera, las dos versiones de la librería comparten la misma arquitectura, y para cada widget 3D final se han creado dos prototipos, uno para VRML97 y otro para X3D, compuesto cada uno de: una interfaz basada en un conjunto de campos –tanto para establecer valores de inicio como para enviar y recoger eventos–, formas que representan la apariencia y la geometría del widget –configurable a través de la interfaz del prototipo–, los sensores que hacen posible que el usuario pueda interactuar con la geometría, una lógica interna que define el comportamiento de cada widget según su estado, y un conjunto de rutas que conectan formas, sensores y lógica.

### A.5.2 Implementación de los widgets 3D

Como se ha comentado en el anterior apartado, cada widget 3D de la librería VUIToolkit ha sido implementado como un PROTO, siendo esta la palabra reservada en los lenguajes VRML97 y X3D que da comienzo a la definición de un prototipo. Para crear la interfaz de cada uno de los prototipos, se ha partido de los atributos que poseen cada uno de los correspondientes elementos en el nivel concreto de UsiXML, algunos de ellos heredados de las clases *CIO*, *graphicalCIO* y *graphicalIndividualCIO*.

Al añadir un atributo de un CIO a un prototipo VRML97 o X3D es necesario ajustar el tipo de datos de cada atributo a los que lista la especificación de esos lenguajes. Por ejemplo, un atributo *boolean* de un CIO puede hacerse corresponder de forma directa con un campo *SFBool* en VRML97 o X3D. Sin embargo, aunque el color de un elemento se especifica en UsiXML como un atributo de tipo *string*, el tipo de datos para dicho atributo en el toolkit debería ser *SFColor*, que es la forma más apropiada de describir tal propiedad en VRML97 y X3D. El código que se reproduce en la Figura A.3, ha sido copiado

de la versión para VRML97 del prototipo *VUIToggleButton*. Dicho código muestra claramente una sección que contiene las correspondencias con la clase *CIO*, después otra con las propiedades de la clase *graphicalCIO*, le sigue otra relativa a *graphicalIndividualComponent*, y por último los atributos propios de *toggleButton*.

```

PROTO VUIToggleButton [
# USIXML: cio
# ...
# USIXML: graphicalCio
# ...
# USIXML: graphicalIndividualComponent
# ...
# USIXML: toggleButton
field          SFBool    defaultState FALSE
# VRML97 GUI Toolkit fields
field          SFInt32   top      0
field          SFInt32   left     0
field          SFInt32   width    75
field          SFInt32   height   25
exposedField  MFNode    label    [ ]

eventOut      SFTIME    touchTime
eventIn       SFBool    set_state
eventOut      SFBool    state_changed

eventIn       SFInt32   set_top
eventIn       SFInt32   set_left
eventIn       SFInt32   set_width
eventIn       SFInt32   set_height
eventOut      SFInt32   top_changed
eventOut      SFInt32   left_changed
eventOut      SFInt32   width_changed
eventOut      SFInt32   height_changed

```

**Figura A.3** Definición de *toggleButton* en VRML97

Con el fin de centrar la discusión en los campos más interesantes, algunas de esas secciones han sido recortadas. En particular, no se muestra ninguno de los campos que se derivan de las clases *CIO*, *graphicalCIO* y *graphicalIndividualCIO*. En cambio, sí se muestra, por ejemplo, el atributo *defaultState* que se define en la clase *toggleButton* como una propiedad de tipo *boolean* y que se ha incluido en el prototipo como un campo de tipo *SFBool*.

No todos los atributos que se definen en el nivel concreto de UsiXML tienen un significado en el toolkit, al tiempo que se han añadido otros atributos en el prototipo como parámetros relacionados con la plataforma final, que en este caso

son mundos virtuales creados con los lenguajes VRML97 y X3D. Por ejemplo, en el nivel concreto de UsiXML no hay atributos que definan la posición o las dimensiones de un widget usando unidades tales como el pixel o el metro, debido a que los widgets son dispuestos en un contenedor usando un conjunto de relaciones lógicas. En cambio, el toolkit que aquí se presenta para dibujar la interfaz final en un mundo virtual sí precisa que cada widget 3D cuente con su posición y dimensiones expresadas en pixels, y por esta razón la interfaz del prototipo *VUIToggleButton* cuenta con cuatro nuevos campos: *top*, *left*, *width* y *height*.

Otro de los aspectos que se han cuidado en este toolkit es que fuera posible modificar en tiempo de ejecución las propiedades de los elementos. Los campos de tipo *field* permiten especificar un valor inicial, pero no permiten modificar su valor en ejecución.

Para conseguir esto último, la interfaz de cada prototipo en la versión para VRML97 cuenta, para cada parámetro, con otro campo *eventIn* –sumideros de eventos en las rutas- con el mismo nombre pero con prefijo *set\_*, así como otro campo *eventOut* –fuentes de eventos- también con el mismo nombre pero esta vez con sufijo *\_changed*. La lógica interna de cada prototipo atiende los eventos de entrada y efectúa las modificaciones pertinentes, generando eventos de salida que permiten comunicar al resto del sistema los cambios producidos en su estado.

Esta multiplicación de parámetros no sería necesaria si la especificación VRML97 permitiera el uso en los nodos *Script*, la lógica del prototipo, de parámetros de tipo *exposedField*, que encapsulan los tres anteriores tipos en uno –*field*, *eventIn* y *eventOut*–.

Afortunadamente, la especificación X3D sí lo permite, aunque el tipo en cuestión tiene un nombre diferente, *inputOutput*, pero en cualquier caso puede usarse en nodos *Script* y conectarse a través de sentencias *IS* con el correspondiente parámetro en la interfaz del prototipo. Aprovechando esta característica, la versión para X3D de cada prototipo cuenta en su interfaz un número de parámetros menor, pero conserva la misma funcionalidad que su equivalente en VRML97, como puede comprobarse comparando la anterior interfaz de *VUIToggleButton* con la que se muestra en la Figura A.4.

```

<ProtoDeclare name="VUIToggleButton">
<ProtoInterface>
<!-- USIXML: cio -->
<!-- . . . -->
<!-- USIXML: graphicalCio -->
<!-- . . . -->
<!-- USIXML: graphicalIndividualComponent -->
<!-- . . . -->
<!-- USIXML: toggleButton -->
<field accessType="initializeOnly"
name="defaultState"
type="SFBool" value="false"/>

<!-- X3D GUI Toolkit fields -->
<field accessType="inputOutput" name="top"
type="SFInt32" value="0"/>
<field accessType="inputOutput" name="left"
type="SFInt32" value="0"/>
<field accessType="inputOutput" name="width"
type="SFInt32" value="75"/>
<field accessType="inputOutput" name="height"
type="SFInt32" value="25"/>
  <field accessType="inputOutput" name="state"
type="SFBool"/>
  <field accessType="inputOutput" name="label"
type="MFNode"/>
<field accessType="outputOnly"
name="touchTime" type="SFTime"/>

```

**Figura A.4** Definición de *toggleButton* en X3D

Siguiendo con el ejemplo dado de *VUIToggleButton*, en su interfaz se incluyen también otros campos que hacen posible conectar este widget 3D con el resto de la interfaz y la aplicación, como por ejemplo *touchTime*, el cual es usado para generar un evento de salida cada vez que el botón es pulsado, y el campo *state*, el cual hace posible cambiar el estado del botón con un evento de entrada y comunicar el cambio en el estado del botón con otro evento de salida —en la versión para VRML97 se cuentan dos campos más, *set\_state* y *state\_changed*—. Aparte, en la comparación también pueden observarse las diferencias entre la notación clásica de VRML97 y la nueva notación de X3D basada en el lenguaje XML.

Al código mostrado en ambas figuras le sigue, en la implementación del prototipo, un grafo de escena que incluye las formas que dan al widget 3D su apariencia y geometría. En un entorno 2D, el widget sería dibujado usando primitivas tales como puntos, líneas, texto e imágenes. En VRML97 la forma

debe ser modelada usando los nodos que ofrece el lenguaje, tales como polilíneas, polígonos, texto y texturas. En X3D el abanico de primitivas es mayor, al incluir nuevos nodos de geometría 2D. En ambos lenguajes para la Web 3D, la geometría del widget 3D es especificada como una jerarquía de transformaciones cuyas hojas son las formas que dan al widget 3D el aspecto que tiene una vez mostrado en pantalla.

Otro elemento de cada widget 3D es el sensor o conjunto de sensores que se incluyen en el prototipo con el fin de que el usuario pueda interactuar con las formas mostradas. Volviendo al ejemplo del prototipo *VUIToggleButton*, este incluye un nodo *TouchSensor* asociado a la geometría del botón, de modo que cuando el usuario ejecuta la acción de tocar dicha geometría con su dispositivo apuntador, el sensor detecta la acción y genera un evento de salida que contiene el instante de tiempo en el que el botón fue tocado. Además, este prototipo incluye dos nodos *TimeSensor* que lanzan dos animaciones distintas, una para el caso en el botón es pulsado y otra para el caso en el que el botón es liberado. En ambas animaciones, la geometría del botón cambia moviéndose a lo largo de la tercera dimensión, tal y como se esperaría en el mundo real, en lugar de recurrir a trucos visuales como ocurre en los entornos 2D.

El evento generado por el *TouchSensor* es dirigido hacia un nodo *Script* que representa la lógica interna del widget 3D. Este nodo *Script* tiene un conjunto de campos que se hacen corresponder, por medio de sentencias *IS*, con algunos de los campos definidos como interfaz del prototipo, de modo que el *Script* recibe los valores iniciales y dispone el widget 3D acorde a los mismos, cambiando después la geometría y el comportamiento basándose según el estado del widget 3D en cada momento. Cada evento externo, como por ejemplo una acción de usuario, es entonces dirigido hacia este nodo *Script*, ejecutando la correspondiente función que interpreta el evento y cambia el estado del widget 3D, generando eventos de salida que comunican los cambios a otras partes de la interfaz o de la aplicación.

La última parte del cualquiera de los prototipos que forman este toolkit es un conjunto de rutas, siendo este el mecanismo descrito en las especificaciones VRML97 y X3D para dirigir eventos desde los nodos que los generan hasta los nodos que los reciben. En particular, las rutas que se incluyen en cada prototipo tienen el papel de conectar las formas y los sensores con la lógica interna del widget 3D, de modo que el nodo *Script* pueda dirigir el comportamiento del widget 3D, recibiendo eventos de entrada y generando eventos de salida hacia el grafo de escena del prototipo.

Hasta el momento, los prototipos implementados son: *VUIWindow*, *VUIButton*, *VUIToggleButton*, *VUICheckBox*, *VUITextComponent*, *VUIImageComponent*, *VUISlider*, *VUICursor* y *VUIComboBox*. Las Figuras A.5 y A.6. muestran una

interfaz de usuario final en la que se puede observar, al menos, una instancia de cada uno de ellos, y que corresponde a un reproductor de radio a través de Internet –para más detalles, véase [Molina, 2005a]-.

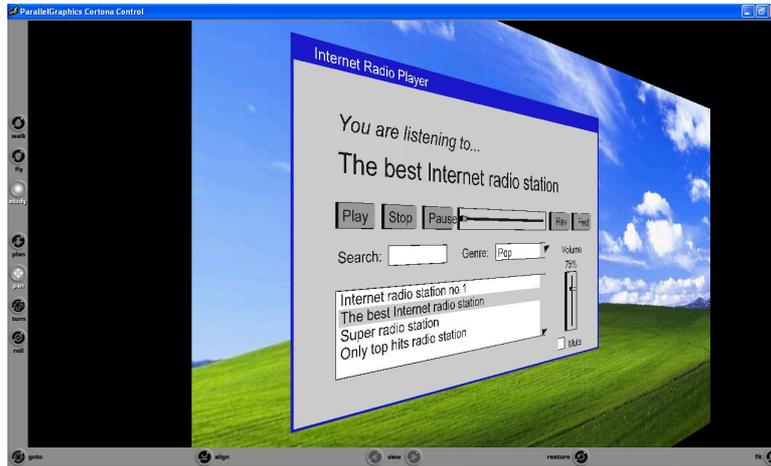


Figura A.5 Widgets 3D en una interfaz de usuario final vista a través de Cortona

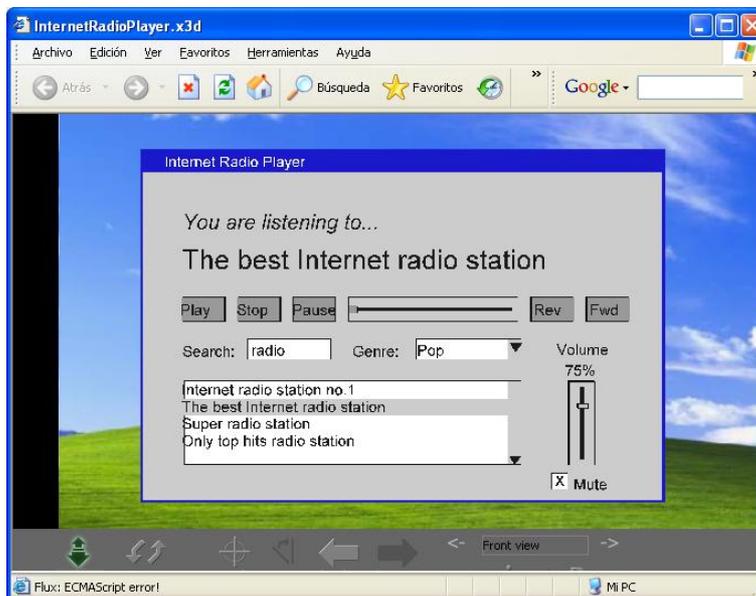


Figura A.6 Vista de frente de los widgets 3D, esta vez a través del visor Flux

En los siguientes párrafos se describen los pormenores de cada uno de los elementos que componen el toolkit, cada uno de los cuales ha sido realizado tomando de base el correspondiente elemento del nivel concreto de UsiXML.

### A.5.2.1 VUIWindow

Este elemento contenedor se corresponde con el elemento *window* del nivel CUI de UsiXML, y como es de imaginar representa ese elemento básico presente en cualquier entorno bidimensional actual. Para la creación del interfaz del prototipo *VUIWindow* se ha partido entonces de los atributos del propio elemento *window*, el cual, según la jerarquía descrita en la especificación de UsiXML, hereda parte de los mismos de la definición de los elementos *CIO*, *graphicalCIO* y *graphicalContainer*. Por ejemplo, dado que *window* hereda de la clase *graphicalCIO* atributos que permiten especificar si la ventana es o no visible (*isVisible*), si responde o no a la interacción (*isEnabled*), el grado de transparencia de su geometría (*transparencyRate*) y la apariencia del elemento (*fgColor*, *bgColor*, *borderWidth*, *borderColor*), esos mismos atributos también se han incluido en el prototipo *VUIWindow*.

Las dimensiones de la ventana son especificadas a través de los atributos *width* y *height* pertenecientes a la clase *graphicalContainer*, y su posición inicial en la pantalla viene dada según el valor de los atributos *windowTopMargin* y *windowLeftMargin* pertenecientes a la clase *window*. A efectos de este toolkit, la unidad de medida utilizada en esos cuatro atributos es el pixel.

Además, el prototipo *VUIWindow* también añade atributos propios de este toolkit. En concreto, cuenta con tres nuevos atributos para controlar las características de la barra de título de la ventana que no estaban en la especificación de UsiXML. También incluye un atributo *children* que, como se ha explicado anteriormente, sirve para añadir los widgets 3D que contendrá la ventana, y que es complementado con los campos *addChildren* y *removeChildren* con los que cuentan los nodos que agrupan otros nodos –los llamados “grouping nodes” en las especificaciones de VRML97 y X3D–.

En cuanto al “look and feel” que ofrece el elemento *VUIWindow*, cabe apuntar que dicho elemento cuenta con una barra de título que no sólo contiene el propio título de la ventana sino que, como cabría esperar, el usuario podrá arrastrar con el ratón para mover la ventana dentro de los límites de la pantalla que la contiene. También cuenta con unos bordes que, si la ventana es redimensionable (atributo *isResizable* tomado de la especificación de *window*) el usuario podrá arrastrar para modificar las dimensiones de la ventana. En el interior de la ventana se ubican los widgets 3D, pero separados una cierta distancia para que puedan distinguirse claramente de la ventana, y no se produzcan efectos indeseados al concurrir en un mismo plano la geometría de la ventana y de los widgets que contiene.

### A.5.2.2 VUIButton, VUIToggleButton y VUICheckBox

Estos tres elementos han sido creados a partir de las clases *button*, *toggleButton* y *checkBox*, respectivamente, descritas todas ellas en el nivel concreto de UsiXML como clases que heredan de *CIO*, *graphicalCIO* y *graphicalIndividualComponent*. Al igual que antes, de la clase *graphicalCIO* heredan atributos como *isVisible*, *isEnabled*, *transparencyRate*, *fgColor*, *bgColor*, *borderWidth* y *borderColor*.

En cuanto a la apariencia, *VUIButton* y *VUIToggleButton* son similares entre sí, la cara de ambos botones posee un borde configurable en grosor y color, y ambos cuentan con caras laterales que le dan forma no de rectángulo sino de caja, proporcionando esa tercera dimensión que en los sistemas bidimensionales sólo es simulada mediante trucos basados en sombras. Uno y otro cuentan con un sensor de pulsación que arranca una animación que modifica el espesor del botón en respuesta a la interacción del usuario, haciéndole ver que el botón ha sido presionado.

Por su parte, la apariencia de *VUICheckBox* es la que cabría esperar de este elemento, pues su geometría dibuja una casilla que, en función de su valor inicial y de las acciones del usuario, es marcada o no con una “X”.

Además de los atributos especificados en el nivel concreto de UsiXML, estos tres elementos añaden cuatro nuevos atributos para especificar su posición y dimensiones (*top*, *left*, *width* y *height*). Además, un quinto atributo de nombre *label* y tipo *MFNode* hace posible asignar a cualquiera de estos widgets 3D otro elemento de interfaz que haga la vez de etiqueta de botón o de etiqueta de caja, ya sea un fragmento de texto o una imagen, esto es, un *VUITextComponent* o un *VUIImageComponent*, o incluso la combinación de ambos.

En cuanto al comportamiento de estos elementos, empezaremos con *VUIButton*, el cual responde a la acción de pulsar del usuario generando un evento de salida *touchTime*, que es precisamente el que genera el sensor de pulsación que contiene este elemento. La diferencia con *VUIToggleButton* es que *VUIButton* vuelve a su estado inicial tras cada pulsación.

En cambio, *VUIToggleButton* y *VUICheckBox* presentan un comportamiento similar, pues poseen dos posibles estados –pulsado y no pulsado, marcado y no marcado, respectivamente– y cambian de estado con cada nueva acción del usuario. Por ello, la interfaz de sus prototipos es similar, contando con un atributo *defaultState* que toman de las clases *toggleButton* y *checkBox*,

respectivamente. Cada vez que el usuario realiza la acción de pulsar sobre uno de esos dos tipos de elementos, se genera un evento de salida *state\_changed*.

### A.5.2.3 VUITextComponent y VUIImageComponent

Sobre estos elementos podría discutirse si son o no widgets 3D en el caso de que como tal entendiéramos un elemento interactivo de la interfaz, pero lo cierto es que estos elementos no sólo nos permiten mostrar texto e imágenes estáticas. Estos elementos han sido creados partiendo de la especificación de las clases *textComponent* e *imageComponent* que, al igual que los elementos *CIO* del apartado anterior, heredan atributos de *CIO*, *graphicalCIO* y *graphicalIndividualComponent*. Aunque son dos clases distintas, ambas añaden los atributos *hyperLinkTarget* y *defaultHyperLink* al conjunto que heredan de las otras tres clases. Esos atributos permiten convertir el texto o la imagen en la parte visible de un hiperenlace, convirtiéndolos entonces en verdaderos elementos interactivos de la interfaz.

Empezando entonces a describir el prototipo *VUITextComponent*, cabe decir que incorpora todos los atributos de la clase *textComponent*, pero muchos de ellos no son realmente utilizados, ya que el texto es representado mediante los nodos *Text* y *FontStyle* de las especificaciones de VRML97 y X3D. Esta implementación queda entonces limitada por las características que ofrecen esos dos nodos. Por ejemplo, *FontStyle* permite dar formato negrita o cursiva, pero el tachado o el subrayado no está soportado –si bien podría realizarse dibujando la línea apropiada-. Además, los campos de *FontStyle* sólo permiten especificar valores iniciales, por lo que cualquier modificación posterior de los mismos sólo es posible si se reemplaza el nodo –quizás utilizando llamadas como *createVrmlFromString*-. Más importante aún es que el texto creado con esos nodos se compone de muchos y diminutos triángulos que pueden saturar el sistema de generación de gráficos 3D de los equipos más modestos. Además, en el caso de la versión para X3D, esta implementación obliga el uso del perfil “Immersive”.

Además de representar etiquetas y enlaces, *VUITextComponent* también puede ser utilizado como campo de edición de texto, sin más que asignar el valor *true* al atributo *isEditable*, especificado en la clase *textComponent*. En el caso de la versión VRML97, la introducción de texto ha sido implementada utilizando una extensión de la compañía ParallelGraphics, el nodo *KdbSensor*, dado que la especificación de VRML97 no proporciona los medios para ello. En el caso de la versión X3D, la implementación se resuelve felizmente con el uso del nodo *StringSensor*, contemplado en la propia especificación. En cualquiera de los dos casos, uno de los problemas que se planteó fue que dichos sensores sólo

estuvieran activos cuando el elemento *VUITextComponent* tuviera el foco, lo cual se resolvió añadiendo un *TouchSensor* de tal forma que, sólo cuando el usuario sitúa el puntero sobre el elemento, los sensores de teclado recogen entonces la entrada por teclado.

En cuanto al elemento *VUIImageComponent*, decir que el URL de la imagen se indica como valor del atributo *imageUrl*, un nuevo atributo añadido al prototipo de este elemento para ese propósito.

#### A.5.2.4 VUISlider y VUICursor

Implementados a partir de la especificación de las clases *slider* y *cursor*, estos prototipos se instancian y combinan para definir el clásico elemento de selección basado en cursor deslizante. Al igual que los elementos CIO del apartado anterior, heredan atributos de *CIO*, *graphicalCIO* y *graphicalIndividualComponent* que nos permiten especificar el estado y apariencia de estos elementos.

Para combinar un elemento con otro, se ha añadido un atributo llamado *cursor* en la interfaz de *VUISlider*, un campo de tipo *MFNode* cuyo propósito es el de anidar otro nodo, más específicamente una instancia de un prototipo *VUICursor*.

Los atributos *minValue* y *maxValue* incluidos en el prototipo *VUISlider* y tomados de la clase *slider* nos permiten indicar los valores mínimo y máximo que será posible seleccionar con este elemento. Sin embargo, el valor inicial se indica mediante el atributo *defaultPosition* especificado en la clase *cursor* e incluido en el prototipo *VUICursor*. Esto planteaba un problema, dado que la transformación de traslación apropiada para el elemento *VUICursor* particular se calcula en la lógica del prototipo *VUISlider*, el cual debía entonces ser capaz de leer el valor del campo *defaultPosition* de *VUICursor*. El problema se solucionó permitiendo el acceso directo a ese campo desde el nodo *Script* del prototipo *VUISlider*, indicando un valor true en el campo *directOutput* del correspondiente nodo *Script*.

Por último, además de poder especificar las dimensiones y posición de este par de elementos, el prototipo *VUISlider* cuenta con un atributo llamado *orientation* que, tomado de la especificación de *slider*, nos permite indicar si la orientación del elemento es horizontal o vertical, característica que también ha sido implementada.

#### A.5.2.5 VUIComboBox

Al igual que los elementos anteriores, *VUIComboBox* también ha sido implementado a partir de la especificación de un elemento presente en el nivel concreto de USIXML. En particular, la clase que se ha tomado como base es *comboBox*, que también hereda atributos de *CIO*, *graphicalCIO* y *graphicalIndividualComponent*. Podría decirse que es el elemento cuya implementación ha sido más costosa, por cuanto combina dos componentes típicos de una interfaz gráfica de usuario, como son la lista de elementos y la lista desplegable. De hecho, el atributo *isDropDown* del prototipo *VUIComboBox*, tomado de la clase *comboBox*, es el que nos permite elegir entre una y otra presentación de los elementos.

En cuanto a las propiedades que se incluyen en el prototipo pero que no estaban presentes en la especificación del correspondiente CIO en UsiXML, cabe apuntar que, además de aquellos que nos permiten indicar la posición y dimensiones del componente (*top*, *left*, *width* y *height*) también se incluyen los atributos *selectedItem* e *Items*, con tipos *SFString* y *MFString*, respectivamente, y que sirven para enumerar los elementos de la lista e indicar cuál de ellos es el seleccionado por defecto.

## Referencias

- [3D UI, URL] 3D UI list archives. URL: <http://people.cs.vt.edu/~bowman/3dui.org/list.html>
- [3D WM, URL] 3D Workspace Manager. URL: <http://www.3dwm.org>  
Última fecha conocida en línea: febrero 2004.
- [3DNA, URL] 3DNA Desktop. URL: <http://www.3dna.net/>
- [AIPO, URL] AIPO. Asociación Interacción Persona-Ordenador. URL: <http://griho.udl.es:8080/aipo/>
- [Alexander, 1977] C. Alexander, S. Ishikawa y M. Silverstein. A Pattern Language: Towns, Buildings, Construction. Oxford University Press. 1977. Resumido en URL: <http://downloadlode.org/Etext/Patterns/>
- [Assumpcao, 1991] J.M. Assumpcao Jr. Proposal for a New Generation of User Interfaces. Marzo 1991. URL: <http://www.lsi.usp.br/~jecel/jpaper8.html>
- [Barrilleaux, 2001] J. Barrilleaux. 3D User Interfaces with Java3D. Manning Publications, 2001.
- [Bardon, 2001] D. Bardon, D. Berry, C. Bjerke y D. Roberts. Crafting the Compelling User Experience Using a Methodical Software-Engineering Approach to Model Users and Design. Tutorial. 2001. URL: [http://www-3.ibm.com/ibm/easy/eou\\_ext.nsf/publish/1650](http://www-3.ibm.com/ibm/easy/eou_ext.nsf/publish/1650)
- [Boehm, 1988] B. Boehm. A Spiral Model of Software Development and Enhancement. IEEE Computer, mayo 1988, Vol. 21, No. 5, pp. 61-72.
- [Bowman, 1999] D.A. Bowman y L.F. Hodges. Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments. Journal of Visual Languages and Computing, Vol. 10, No. 1, 1999, pp. 37-53.
- [Bowman, 2001] D.A. Bowman, E. Kruijff, J.J. LaViola Jr. e I. Poupyrev. An Introduction to 3-D User Interface Design. Presence, Vol. 10, No. 1, febrero 2001, pp. 96-108.
- [Bowman, 2002] D.A. Bowman, J.L. Gabbard y D. Hix. A Survey of Usability Evaluation in Virtual Environments: Classification and Comparison of Methods. Presence: Teleoperators and

- Virtual Environments, Vol. 11, No. 4, 2002, pp. 435-455.
- [Bowman, 2004] D.A. Bowman, E. Kruijff, J.J. LaViola Jr. E I. Poupyrev. 3D User Interfaces: Theory and Practice. Addison-Wesley, 2004.
- [Bowman, 2006] D.A. Bowman, C.A. Wingrave, J. Lucas, A. Ray, N.F. Polys, Q. Li, Y. Haciahmetoglu, J. Kim, S. Kim, R. Boehringer y T. Ni. New Directions in 3D User Interfaces. The International Journal of Virtual Reality, Vol. 5, No. 2, 2006, pp. 3-14.
- [Burdea, 1996] G. Burdea y P. Coiffet. Tecnologías de la Realidad Virtual. Paidós, 1996.
- [Boyd, 1999] D. Boyd y L. Sastry. Development of the INQUISITIVE Interaction Toolkit – Concept and Realisation. User Centered Design and Implementation of Virtual Environments (UCDIVE) Workshop, York, Reino Unido, septiembre 1999.
- [Calvary, 2003] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon y J. Vanderdonckt. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers, Vol. 15, No. 3, 2003, pp. 289-308.
- [Carlsson, 1993] C. Carlsson y O. Hagsan. DIVE – A Platform for Multi-User Virtual Environments. Computers & Graphics, Vol. 17, No. 6, noviembre-diciembre 1993, pp. 663-669.
- [Carroll, 1985] J.M. Carroll y M.B. Rosson. Usability Specifications as a Tool in Iterative Development. En: H.R. Hartson (ed.), Advances in Human-Computer Interaction 1, 1985, pp. 1-28.
- [Celentano, 2001a] A. Celentano y F. Pittarello. Class of Experience: A High Level Approach to Support Content Experts for the International Workshop on Authoring of 3d Environments. Structured Design of Virtual Environments and 3d-Components. Web3D Conference, Paderborn, Alemania, febrero 2001.
- [Celentano, 2001b] A. Celentano y F. Pittarello. A Content Centered Methodology for Authoring 3d Interactive Worlds for Cultural Heritage. International Cultural Heritage Informatics Meeting (ICHIM), Milán, Italia, septiembre 2001, Vol. 2, pp. 315-324.
- [ClockWise, URL] ClockWise Technologies Ltd. Win3D. URL: <http://>

- [www.clockwise3d.com](http://www.clockwise3d.com) Última fecha conocida en línea: agosto 2006.
- [Cognetics, URL] Cognetics Corp. The LUCID Framework. URL: <http://www.cognetics.com/lucid>
- [Conner, 1992] D.B. Conner, S.S. Snibbe, K.P. Herndon, D.C. Robbins, R.C. Zeleznik y A. van Dam. Three-Dimensional Widgets. Proc. of the ACM Symposium on Interactive 3D Graphics (I3D), Special Issue of Computer Graphics, Vol. 25, No. 2, marzo 1992, pp. 183-188.
- [CONTIGRA, URL] Project CONTIGRA (COmponent-orieNted Three-dimensional Interactive GRaphical Applications). URL: <http://www-mmt.inf.tu-dresden.de/english/projekte/CONTIGRA/>
- [Coutaz, 2003] J. Coutaz, Ch. Lachenal y S. Dupuy-Chessa. Ontology for Multi-surface Interaction. Proc. of 9<sup>th</sup> IFIP TC 13 Int. Conf. on Human-Computer Interaction (INTERACT'2003), Zurich, Suiza, septiembre 2003, pp. 447-454.
- [Croquet, URL] The Croquet Project. URL: <http://croquetproject.org/>
- [Cuppens, 2004] E. Cuppens, C. Raymaekers y K. Coninx. VRXML: A User Interface Description Language for Virtual Environments. Workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages. Advanced Visual Interfaces, Gallipoli, Italia, mayo 2004.
- [Czernuszenko, 1997] M. Czernuszenko, D. Pape, D. Sandin, T. DeFanti, G.L. Dawe y M.D. Brown. The ImmersaDesk and Infinity Wall projection-based virtual reality displays. Computer Graphics, Vol. 31, No. 2, 1997, pp. 46-49.
- [Dachselt, 2002] R. Dachselt, M. Hinz y K. Meißner. CONTIGRA: An XML-Based Architecture for Component-Oriented 3D Applications. Proc. of ACM Web3D 2002 Symposium, Tempe, Arizona, EE.UU., febrero 2002, pp. 155-163.
- [Daly, 2002] L. Daly, D. Brutzman y A. Hudson. Introducing X3D. Tutorial. SIGGRAPH, San Antonio, Tejas, EE.UU., julio 2002. URL: <http://www.realism.com/x3d/presentations/s2002/>
- [D'Cruz, 2003] M. D'Cruz, A.W. Stedmon, J. Wilson y P. Moderny. Building Virtual Environments using the Virtual Environment Development Structure: A Case of Study. En: D. Harris, V. Duffy, M. Smith y C. Stephanidis (eds.),

- Human – Centred Computing, Cognitive, Social and Ergonomic Aspects, Vol. 3 de las actas de 10<sup>th</sup> HCI International Conference, Creta, Grecia, junio 2003, Lawrence Erlbaum Associates, pp. 1396-1400.
- [Eastgate, 2001] R. Eastgate. The Structured Development of Virtual Environments: Enhancing Functionality and Interactivity. Tesis doctoral. Universidad de York. 2001.
- [EON, URL] Eon Reality Inc. URL: <http://www.eonreality.com/>
- [Fei, URL1] Y. Fei. About Infinite-3D. URL: <http://www.infinite-3d.com/about.html>
- [Fei, URL2] Y. Fei. Cube Features. URL: <http://www.infinite-3d.com/cube.html>
- [Fencott, 1999] C. Fencott. Towards a Design Methodology for Virtual Environments. User Centered Design and Implementation of Virtual Environments (UCDIVE) Workshop, York, Reino Unido, septiembre 1999.
- [Fencott, 2001] C. Fencott y J. Isdale. Design Issues for Virtual Environments. International Workshop on Structured Design of Virtual Environments and 3D-Components. Web3D Conference, Paderborn, Alemania, febrero 2001.
- [Fencott, 2005a] C. Fencott. A Methodology of Design for Virtual Environments. En: M.I. Sánchez (ed.), Developing Future Interactive Systems, Idea Group, 2005, capítulo 3.
- [Fencott, 2005b] C. Fencott. A Methodology of Design for Virtual Environments. First Int. Workshop on Methods and Tools for developing Virtual Reality Applications (MeTo-VR). 11<sup>th</sup> Int. Conf. on Virtual Systems and Multimedia (VSMM), Gante, Bélgica, octubre 2005. URL: <http://wise.vub.ac.be/metovr/proceedings.html>
- [Figuroa, 2002] P. Figuroa, M. Green y H.J. Hoover. InTml: A Description Language for VR Applications. Proc. of ACM Web3D 2002 Symposium, Tempe, Arizona, EE.UU., febrero 2002, pp. 53-58.
- [Foley, 1996] J.D. Foley, A. van Dam, S.K. Feiner y J.F. Hughes. Computer Graphics: Principles and Practice. Second Edition in C. Addison-Wesley, 1996.
- [Forsberg, 1997] A.S. Forsberg, J.J. LaViola, L. Markosian y R.C. Zeleznik. Seamless Interaction in Virtual Reality. IEEE Computer

- Graphics and Applications, Vol. 17, No. 6, noviembre-diciembre 1997, pp. 6-9.
- [Furness, 2001] T.A. Furness III. Toward Tightly Coupled Human Interfaces. En: R. Earnshaw *et al* (eds.), *Frontiers in Human-Centred Computing, Online Communities and Virtual Environments*, Springer-Verlag, 2001, pp. 80-98.
- [Gabbard, 1997] J.L. Gabbard. A Taxonomy of Usability Characteristics in Virtual Environments. MS Thesis. Virginia Polytechnic Institute and State University, Blacksburg, Virginia, EE.UU., 1997.
- [Gabbard, 1999] J.L. Gabbard, D. Hix y J.E. Swan II. User-Centered Design and Evaluation of Virtual Environments. *IEEE Computer Graphics and Applications*, Vol. 19, No. 6, noviembre 1999, pp. 51-59.
- [García, 2001] F.L.S. García, F. Camargo, G. Tissiani. Metodología para criação de ambientes virtuais tridimensionais. 15º Simpósio Nacional de Geometria Descritiva e Desenho Técnico, IV Int. Conf. on Graphics Engineering for Arts and Design, Sao Paulo, Brasil, noviembre 2000.
- [García, 2005a] A.S. García, J.P. Molina y P. González. Exemplar VE design guidance tool for selection and manipulation interaction techniques. Proc. of 11th International Conference on Human-Computer Interaction, HCI International 2005, Las Vegas, Nevada, EE.UU., julio 2005. CD-ROM.
- [García, 2005b] A.S. García, J.P. Molina y P. González. Aproximación a la evaluación de interfaces de Realidad Virtual. VI Congreso de Interacción Persona-Ordenador, Interacción'05, organizado dentro del primer Congreso Español de Informática, CEDI'2005, Granada, septiembre 2005.
- [González, 2007] G. González, J.P. Molina, A.S. García, D. Martínez y P. González. Evaluación de técnicas para la introducción de texto en Entornos Virtuales Inmersivos. VIII Congreso de Interacción Persona-Ordenador, Interacción'05, organizado dentro del segundo Congreso Español de Informática, CEDI'2007, Zaragoza, septiembre 2007, pp. 313-321.
- [Granollers, 2002] T. Granollers, J. Lorés, G. Raimat, E. Junyent y E. Tartera. Vilars, un Nuevo Modelo de Diálogo Aplicando Realidad Aumentada. Actas del III Congreso Interacción, Leganés, Madrid, mayo 2002, pp. 276-279.

- 
- [Hay, URL] B. Hay. Creating an Interactive 3D Product Using VRML. Tutorial. URL: <http://www.virtualrealms.com.au/vrml/tute01/tutorial.htm>
- [Herndon, 1994] K.P. Herndon, A. van Dam y M. Gleicher. The Challenges of 3D Interaction: a CHI '94 workshop. ACM SIGCHI Bulletin, Vol. 26, No. 4, octubre 1994, pp. 36-46.
- [Huda, URL1] S.Z. Huda. Analysis of RGB Computer Room and its Representation as UML Diagrams. URL: <http://www.public.asu.edu/~zakiul/vrml/week14/week14.htm>  
Última fecha conocida en línea: enero 2003.
- [Huda, URL2] S.Z. Huda. Week 15 report on Project 6. URL: <http://www.public.asu.edu/~zakiul/vrml/week15/week15.htm>  
Última fecha conocida en línea: enero 2003.
- [IBM, URL] IBM. RealPlaces Design Guide. URL: [http://www-3.ibm.com/ibm/easy/eou\\_ext.nsf/Publish/580](http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/580) Última fecha conocida en línea: julio 2006.
- [INQUISITIVE, URL] The INQUISITIVE Project (INcreasing Quality of User Interfaces for Strategic Interactive Tasks In Virtual Environments). URL: <http://www.cs.york.ac.uk/hci/inquisitive/>
- [Jacob, 1996] R.J.K. Jacob. A Visual Language for Non-WIMP User Interfaces. Proc. of IEEE Symposium on Visual Languages, Boulder, Colorado, EE.UU., septiembre 1996, pp.231-238.
- [Jacob, 1999] R.J.K. Jacob, L. Deligiannidis y S. Morrison. A Software Model and Specification Language for Non-WIMP User Interfaces. ACM Transactions on Computer-Human Interaction, Vol. 6, No. 1, marzo 1999, pp. 1-46.
- [Kaur, 1998] K. Kaur. Designing virtual environments for usability. Tesis doctoral. Centre for HCI Design, City University, Londres, Reino Unido, Junio 1998.
- [Kim, 1998] G.J. Kim, K.C. Kang, H. Kim y J. Lee. Software Engineering of Virtual Worlds. Proc. of ACM Symposium on Virtual Reality Software and Technology (VRST), Taipei, Taiwan, noviembre 1998, pp. 131-139.
- [Larimer, 2003] D. Larimer y D.A. Bowman. VEWL: A Framework for Building a Windowing Interface in a Virtual Environment. Proc. of INTERACT, Zurich, Suiza, septiembre 2003, pp. 809-812.

- [Leavitt, 2001] N. Leavitt. 3D Technology: Ready for the PC? IEEE Computer Magazine, Vol. 34, No. 11, noviembre 2001, pp. 17-20.
- [Leftwich, 1993] J. Leftwich. InfoSpace: A Conceptual Method for Interacting with Information in a Three-Dimensional Virtual Environment. 3<sup>rd</sup> Int. Conf. on Cyberspace, Austin, Tejas, EE.UU., mayo 1993.
- [Letelier, 1998] P. Letelier, I. Ramos, P. Sánchez y O. Pastor. OASIS version 3.0: A Formal Approach for Object Oriented Conceptual Modelling. Universidad Politécnica de Valencia. 1998.
- [Limbourg, 2004] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon y V. López. UsiXML: a Language Supporting Multi-Path Development of User Interfaces. Proc. of 9<sup>th</sup> IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI), en conjunto con 11<sup>th</sup> Int. Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS), Hamburgo, Alemania, julio 2004.
- [Lozano, 2001] M. Lozano. A Methodological Approach for the Specification and Development of Object Oriented User Interfaces. Tesis doctoral. Universidad Politécnica de Valencia. 2001.
- [Lozano, 2002a] M.D. Lozano, P. González, F. Montero, J.P. Molina e I. Ramos. Integrating Usability within the User Interface Development Process of Web Applications. Proc. of 2nd Int. Workshop on Web Oriented Software Technology (IWWOST '02), Málaga, junio 2002, pp. 134-147.
- [Lozano, 2002b] M.D. Lozano, P. González, F. Montero, J.P. Molina e I. Ramos. A Graphical User Interface Development Tool. Proc. of the 16th British HCI Conference, Volume 2, Londres, septiembre 2002, pp. 62-65.
- [Marsh, 1998] T. Marsh, P. Wright, S. Smith, y D. Duke. A Shared Framework of Virtual Reality. Proc. of 5th UK-VRSIG, Exeter, UK, 1998.
- [McIntosh, 2000] P.G. McIntosh. Knowledge Acquisition Using VRML and the Unified Modelling Language: the Case of the RGB Computer Room. Knowledge-Based Systems, Vol. 13, No. 1, febrero 2000, pp. 21-26.
- [Méndez, 2001] G. Méndez, M.I. Sánchez y A. de Antonio. Hacia una

Metodología de Desarrollo para la Construcción de Entornos Virtuales. Actas de las VI Jornadas de Ingeniería del Software y Bases de Datos (JISBD), Almagro, Ciudad Real, noviembre 2001, pp. 327-342.

- [Microsoft, URL] Microsoft Research. The TaskGallery. URL: <http://research.microsoft.com/adapt/TaskGallery/>
- [Milgram, 1994] P. Milgram y F. Kishino. A Taxonomy of Mixed-Reality Visual Displays. IEICE Transactions on Information and Systems, Special issue on Networked Reality, Vol. E77-D, No. 12, diciembre 1994, pp.1321-1329.
- [Mine, 1995] M. Mine. Virtual Environment Interaction Techniques. Technical Report TR95-018, Universidad de Carolina del Norte, Chapel Hill, EE.UU., 1995.
- [Molina, 2002] J.P. Molina, P. González y M.D. Lozano. A Unified Envisioning of Future Interfaces. Proc. of the Second IASTED Int. Conf. Visualization, Imaging and Image Processing (VIIP 2002), Málaga, Spain, septiembre 2002, ACTA Press, pp. 185-190.
- [Molina, 2003a] J.P. Molina, P. González, M.D. Lozano, F. Montero y V. López. Bridging the gap: developing 2D and 3D user interfaces with the IDEAS methodology. En: J. Jorge *et al.* (eds.), DSV-IS: Issues in Designing New-generation Interactive Systems, Proc. of the Tenth Workshop on the Design, Specification and Verification of Interactive Systems, Springer-Verlag Lecture Notes on Computer Science, 2003, pp. 303-315.
- [Molina, 2003b] J.P. Molina, P. González y M.D. Lozano. Desarrollo de interfaces de usuario tridimensionales: un enfoque metodológico basado en IDEAS. Actas del Cuarto Congreso Internacional Interacción Persona-Ordenador, Interacción 2003, Vigo, junio 2003, Reprogalicia Edicións. CD-ROM.
- [Molina, 2003c] J.P. Molina, P. González y M.D. Lozano. Developing 3D UIs using the IDEAS Tool: A case of study. En: J. Jacko and C. Stephanidis (eds.), Human - Computer Interaction, Theory and Practice, Proc. of 10th Int. Conf. on Human-Computer Interaction, HCI International 2003, Creta, Grecia, junio 2003, Part I, Volume 1, pp. 1193-1197.
- [Molina, 2004] J.P. Molina y P. González. Model-based design and new user interfaces: current practices and opportunities. First Int. Workshop on Making model-based user interface design

- practical: usable and open methods and tools, dentro de IUI / CADUI, Funchal, Madeira, Portugal, enero 2004. CEUR-Workshop Proceedings, Vol. 103. URL: <http://CEUR-WS.org/Vol-103>
- [Molina, 2005a]** J.P. Molina, J. Vanderdonckt, F. Montero y P. González. Towards Virtualization of User Interfaces based on UsiXML. Proc. of ACM Web3D 2005 Symposium, 10th Int. Conf. on 3D Web Technology, Bangor, Reino Unido, marzo-abril 2005, ACM, pp. 169-179.
- [Molina, 2005b]** J.P. Molina, A.S. García, V. López & P. González. Developing VR applications: the TRES-D methodology. First Int. Workshop on Methods and Tools for developing Virtual Reality Applications (MeTo-VR). 11<sup>th</sup> Int. Conf. on Virtual Systems and Multimedia (VSMM), Gante, Bélgica, octubre 2005. URL: <http://wise.vub.ac.be/metovr/proceedings.html>
- [Molina, 2006a]** J.P. Molina, J. Vanderdonckt & P. González. Direct manipulation of User Interfaces for Migration. Proc. of 10th ACM Int. Conf. on Intelligent User Interfaces (IUI'2006), Sydney, Australia, enero-febrero 2006, pp. 140-147.
- [Molina, 2006b]** J.P. Molina, A.S. García, D. Martínez, F.J. Manjavacas, V. Blasco & P. González. An Interaction Model for the TRES-D Framework. Proc. of 13th IEEE Mediterranean Electrotechnical Conference (MELECON 2006), special session "New interaction paradigms in Virtual Environments", Benalmádena, Málaga, mayo 2006. CD-ROM.
- [Molina, 2006c]** J.P. Molina, J. Vanderdonckt, P. Gonzalez, A.F. Caballero, & M.D. Lozano. Rapid Prototyping of Distributed User Interfaces. En: G. Calvary *at al.* (eds.), Computer-Aided Design of User Interfaces V, Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2006, Springer-Verlag Information Systems Series, 2006, pp. 151-166.
- [Molina, 2006d]** J.P. Molina, A.S. García, D. Martínez, F.J. Manjavacas, V. Blasco, V. López & P. González. The development of glove-based interfaces with the TRES-D methodology. Proc. of 13th ACM Symposium on Virtual Reality Software and Technology (VRST), Limassol, Chipre, noviembre 2006. pp. 216-219.

- 
- [Murray, 1999] N. Murray y T. Fernando. A Task Manager for Virtual Environments. User Centered Design and Implementation of Virtual Environments (UCDIVE) Workshop, York, Reino Unido, septiembre 1999.
- [National, 1997] National Research Council. More Than Screen Deep: Toward Every-Citizen Interfaces to the Nation's Information Infrastructure. National Academy Press, 1997.
- [Neale, 2001] H. Neale y S. Nichols. Designing and Developing Virtual Environments: Methods and Applications. Visualization and Virtual Environments Community Club (VVECC) Workshop: Design of Virtual Environments, Oxfordshire, Reino Unido, enero 2001.
- [Nielsen, 1998] J. Nielsen. 3D is Better than 2D. Jakob Nielsen's Alertbox for November 15, 1998. URL: <http://www.useit.com/alertbox/981115.html>
- [Nooface, URL] Nooface: In Search of the Post-PC Interface. URL: <http://nooface.com/>
- [Norman, 1986] D.A. Norman. Cognitive Engineering. En: D.A. Norman y S.D. Draper (eds.), User Centered System Design, Lawrence Erlbaum Associates, 1986, pp. 31-61.
- [Parés, 2001] N. Parés y R. Parés. Interaction-Driven Virtual Reality Application Design, A Particular Case: El Ball del Fanales or Lightpools. Presence, Vol. 10, No. 2, abril 2001, pp. 236-245.
- [Paternò, 1999] F. Paternò. Model-based Design and Evaluation of Interactive Applications. Springer, 1999.
- [Pereira, 2000] A.T.C. Pereira, I.B. Rebelo y G. Tissiani. Design de Interfaces para Ambientes Virtuais: Como obter Usabilidade em 3D. Actas de la IV Sociedad Iberoamericana de Gráfica Digital, SIGraDI'2000, Rio de Janeiro, Brasil, septiembre 2000, pp. 25-28.
- [Pierce, 1999] J.S. Pierce, M. Conway, M. van Dantzich y G. Robertson. Toolspaces and Glances: Storing, Accessing, and Retrieving Objects in 3D Desktop Applications. Proc. of ACM Symposium on Interactive 3D Graphics (I3D), Atlanta, Georgia, EE.UU., abril 1999, pp. 163-168.
- [Pittarello, 2001] F. Pittarello y A. Celentano. Interaction locus: a multimodal approach for the structuring of virtual spaces. Proc. of HCITaly Symposium, Florencia, Italia, septiembre 2001.

- [Polys, 2005] N.F. Polys, R. Hetherington, D. Brutzman y D. Gracacin. Engineering Virtual Environments with X3D. Tutorial. ACM Web3D 2005 Symposium, 10th Int. Conf. on 3D Web Technology, Bangor, Reino Unido, marzo-abril 2005. URL: [http://people.cs.vt.edu/~bowman/3dui.org/web3d2005/X3D\\_IT\\_Tutorial/](http://people.cs.vt.edu/~bowman/3dui.org/web3d2005/X3D_IT_Tutorial/)
- [Poupyrev, 1996] I. Poupyrev, M. Billinghurst, S. Weghorst y T. Ichikawa. The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR. Proc. of ACM Symposium on User Interface Software and Technology (UIST), Seattle, Washington, EE.UU., 1996, pp. 79-80.
- [Poupyrev, 1998] I. Poupyrev, S. Weghorst, M. Billinghurst y T. Ichikawa. Egocentric Object Manipulation in Virtual Environments: Empirical Evaluation of Interaction Techniques. Computer Graphics Forum, Vol. 17, No. 3, septiembre 1998, pp. 41-52.
- [Preece, 1994] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland y T. Carey. Human-Computer Interaction. Addison-Wesley, 1994.
- [Reddy, 2005] M. Reddy. 3D Graphics and the Film Production Pipeline. Conferencia invitada, ACM Web3D 2005 Symposium, 10th Int. Conf. on 3D Web Technology, marzo-abril 2005, Bangor, Reino Unido.
- [Roberts, 1999] D. Roberts. RealPlaces, 3D Interfaces for Office Applications. Artículo invitado, Workshop on Tools for Working With Guidelines, TFWWG, Biarritz, Francia, octubre 2000.
- [Robertson, 1991] G.G. Robertson, J.D. Mackinlay y S.K. Card. Cone Trees: Animated 3D Visualizations of hierarchical information. Proc. of SIGCHI'91, Nueva Orleans, Luisiana, EE.UU., abril-mayo 1991, pp. 189-194.
- [Robertson, 1993] G.G. Robertson, S.K. Card y J. Mackinlay. Information Visualization Using 3D Interactive Animation. Communications of the ACM, Vol. 36, No. 4, abril 1993, pp. 57-71.
- [Rosson, 2002] M.B. Rosson y J. Carroll. Usability Engineering: Scenario Based Development of Human-Computer Interaction. Morgan Kaufmann, 2002.
- [Sánchez, 2001] M.I. Sánchez y A. de Antonio. Design Tasks in Virtual

- 
- Environments Development. 13<sup>th</sup> Int. Conf. on Software Engineering and Knowledge Engineering (SEKE), Buenos Aires, Argentina, junio 2001.
- [Sánchez, 2003]** M.I. Sánchez, A. de Amescua, J.J. Cuadrado y A. de Antonio. Software Engineering and HCI Techniques Joined to Develop Virtual Environments. Int. Conf. on Software Engineering (ICSE), Portland, Oregón, EE.UU., mayo 2003.
- [Sánchez, 2005a]** M.I. Sánchez, A. de Antonio y G. Méndez. Desarrollo de entornos virtuales para Web. En: M.P. Díaz (ed.), Ingeniería de la Web y Patrones de Diseño, Pearson Educación, 2005, pp. 163-200.
- [Sánchez, 2005b]** M.I. Sánchez, A. de Antonio y A. de Amescua. SENDA: A Whole Process to Develop Virtual Environments. En: M.I. Sánchez (ed.), Developing Future Interactive Systems, Idea Group, 2005, capítulo 4.
- [Sastry, 2000]** L. Sastry y D. Boyd. Interactions in Virtual Environments. ERCIM News (Special theme: Web Technologies), No.41, abril 2000, pp. 49-50.
- [Sastry, 2001]** L. Sastry, M. Wilson y D. Boyd. Interacting in Task-Oriented Virtual Worlds. Visualization and Virtual Environments Community Club (VVECC) Workshop: Design of Virtual Environments, Oxfordshire, Reino Unido, enero 2001.
- [Scali, 2003]** S. Scali, M. Wright y A.M. Shillito. 3D Modelling is not for WIMPs. En: J. Jacko y C. Stephanidis (eds.), Human - Computer Interaction, Theory and Practice (Part I), Vol. 1 de las actas de 10<sup>th</sup> HCI Int. Conf., Creta, Grecia, junio 2003, Lawrence Erlbaum Associates, pp. 701-705.
- [Seo, 2001]** J. Seo, D.N. Kim, y G.J. Kim. VR Object Reuse through Component Combination. International Workshop on Structured Design of Virtual Environments and 3D-Components. Web3D Conference, Paderborn, Alemania, febrero 2001.
- [Skov, 2001]** M. Skov J. Stage. Using Software Engineering Approaches to Model Dynamics in Interactive Software Systems. En: L. Qvirtrup (ed.), Virtual Interaction: Interaction in Virtual Inhabited 3D Worlds, Springer-Verlag, 2001.
- [Shneiderman, 1998]** B. Shneiderman. Designing the User Interface: Strategies for Effective Human-Computer Interaction. Pearson

- Education, 1998.
- [Shneiderman, 2001] B. Shneiderman. Supporting Creativity with Advanced Information-Abundant User Interfaces. En: R. Earnshaw *et al* (eds.), *Frontiers in Human-Centred Computing, Online Communities and Virtual Environments*, Springer-Verlag, 2001, pp. 469-480.
- [Shneiderman, 2002] B. Shneiderman. 3D or Not 3D: When and Why Does it Work? Conferencia invitada. ACM Web3D Symposium, febrero 2002, Phoenix, Arizona, EE.UU. URL: [http://www.cs.umd.edu/hcil/pubs/presentations/Web3D-4\\_files/frame.htm](http://www.cs.umd.edu/hcil/pubs/presentations/Web3D-4_files/frame.htm)
- [Smith, 1999] S. Smith, D. Duke y M. Massink. The Hybrid World of Virtual Environments. *Computer Graphics Forum*, Vol. 18, No. 3, septiembre 1999, pp. 287-307.
- [Smith, 2001] S. Smith y D. Duke. Design Support for Virtual Environments. Visualization and Virtual Environments Community Club (VVECC) Workshop: Design of Virtual Environments, Oxfordshire, Reino Unido, enero 2001.
- [Sommerville, 1999] I. Sommerville. *Software Engineering*, 5th edition. Addison-Wesley, 1999.
- [SphereXP, URL] The SphereXP, a 3D desktop replacement for Microsoft Windows XP. URL: <http://www.spheresite.com/>
- [Stuart, 2001] R. Stuart. *The Design of Virtual Environments*. Barricade Books, 2001.
- [Sun, URL] Sun Microsystems. Project Looking Glass. URL: [http://www.sun.com/software/looking\\_glass/index.xml](http://www.sun.com/software/looking_glass/index.xml)
- [Sutcliffe, 1994] A. Sutcliffe y P. Faraday. Designing Presentation in Multimedia Interfaces. Proc. of CHI-94: Human Factors in Computing Systems, Boston, Massachusetts, EE.UU., abril 1994, pp. 92-98.
- [Sutcliffe, 2003] A. Sutcliffe. *Multimedia and Virtual Reality: Designing Multisensory User Interfaces*. Lawrence Erlbaum Associates, 2003.
- [Tanriverdi, 2001] V. Tanriverdi y R.J.K. Jacob. VRID: A Design Model and Methodology for Developing Virtual Reality Interfaces. Proc. of ACM Symposium on Virtual Reality Software and Technology (VRST), noviembre 2001, Baniff, Alberta, Canada, pp. 175-182.

- 
- [Tebutt, URL] D. Tebutt. Desktop Evolution. URL: [http://www-3.ibm.com/ibm/easy/eou\\_ext.nsf/publish/582](http://www-3.ibm.com/ibm/easy/eou_ext.nsf/publish/582)
- [Trevisan, 2004] D.G. Trevisan, J. Vanderdonckt y B Macq. Conceptualising mixed spaces of interaction for designing continuous interaction. *Virtual Reality Journal*, Vol. 8, No. 2, marzo 2004, pp. 83-95.
- [UsiXML, URL] UsiXML.org, home of the User Interface eXtensible Markup Language. URL: <http://www.usixml.org/>
- [van Dam, 1997] A. van Dam. Post-WIMP User Interfaces. *Communications of the ACM*, Vol. 40, No. 2, febrero 1997, pp. 63-67.
- [Vanderdonckt, 1993] J. Vanderdonckt y F. Bodart. Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. *Proc. of ACM Conf. on Human Aspects in Computing Systems (InterCHI)*, abril 1993, Amsterdam, Holanda, pp. 51-60.
- [Vanderdonckt, 2007] J. Vanderdonckt, H. Mendonça y J.P. Molina. Distributed User Interfaces in Ambient Environment. *Proc. of 1<sup>st</sup> Int. Workshop on Model Driven Software Engineering for Ambient Intelligence Applications, MDSE4AmI'07*, Springer-Verlag Lecture Notes in Computer Science, 2007, pp. 44-52.
- [VRML Email, URL] VRML Email list Archives. Desde marzo de 2004. URL: [http://www.web3d.org/x3d/publiclists/vrml\\_list\\_archives/](http://www.web3d.org/x3d/publiclists/vrml_list_archives/)
- [VRML Spec, URL] VRML Specification. URL: <http://www.web3d.org/x3d/specifications/#vrml97>
- [VRML WWG, URL] VRML Widgets Working Group Website. URL: <http://zing.ncsl.nist.gov/~gseidman/vrml/wwg/>
- [Web3D, URL] Web3D Consortium. URL: <http://www.web3d.org/>
- [Willans, 2000] J.S. Willans, M.D. Harrison y S.P. Smith. Implementing Virtual Environment Object Behaviour from a Specification. *Int. Workshop on Structured Design of Virtual Environments and 3D-Components*. Web3D Conference, Paderborn, Alemania, febrero 2001.
- [Wilson, 2002] J.R. Wilson, R. Eastgate y M. D'Cruz. Structured Development of Virtual Environments. En: J. Jacko (ed.), *Handbook of Virtual Environments: Design, Implementation, and Applications*. Lawrence Erlbaum Associates, 2002.
- [Wingrave, 2005] C.A. Wingrave. *The Future Virtual Reality Melting Pot*.

- 
- En: M.I. Sánchez (ed.), Developing Future Interactive Systems, Idea Group, 2005, capítulo 2.
- [WWW-VRML, URL] The WWW-VRML Hypermail Archives. De mayo de 1995 a diciembre de 2003. URL: <http://web.archive.org/web/20031206100600/www.web3d.org/www-vrml/hypermail/>
- [X3D Spec, URL] X3D Specifications, Encodings and Language Bindings. URL: <http://www.web3d.org/x3d/specifications/#x3d-spec>