Universidad de Valladolid

Escuela Técnica Superior de Ingeniería Informática

Departamento de Informática

Tesis doctoral:

Interacción Multimodal con Espacios Virtuales

Presentada por Héctor Olmedo Rodríguez para optar al grado de doctor por la Universidad de Valladolid

Dirigida por:

Dr. David Escudero Mancebo

Dr. Valentín Cardeñoso Payo

A mi madre. Por darme la vida y dedicarme la suya.

El mundo es un escenario,
y todos los hombres y mujeres
son meros actores.

William Shakespeare

Agradecimientos

Las primeras palabras de agradecimiento quiero dedicarlas a mis directores de tesis David Escudero Mancebo y Valentín Cardeñoso Payo sin los que este trabajo hubiera sido imposible, en todos los aspectos. También a la familia de David: Alejandra, Julia y Carmen que han sufrido reuniones de trabajo, ensayos para ponencias y llamadas a horas intempestivas.

Al resto de miembros del grupo ECA-SIMM, César González-Ferreras y Arturo González Escribano que me ayudaron mucho sobre todo en los inicios de este trabajo y con los que he compartido orgulloso, la autoría de varias publicaciones.

A quienes impartieron los cursos de doctorado a los que asistí cuando comencé esta aventura: Mercedes Martínez González, Carlos Enrique Cuesta Quintero, César Llamas Bello, Pablo de la Fuente Redondo, Jesús Vegas Hernández y especialmente a Javier Finat por haberme motivado tanto.

A los organizadores y asistentes de las diferentes ediciones de los congresos INTERACCIÓN y JOREVIR.

A los antiguos alumnos de la Escuela Técnica Superior de Ingeniería Informática y de la Escuela Universitaria Politécnica: Enrique Dacosta, José Ramón Pinto Montanillo, Gonzalo Gabino Diestro, Ignacio Fernandez-Divar Escacho, Alejandro Alcántara Zarzuela, Samuel García Blanco, Jose Antonio Cabrerizo Fernández, Alberto Sanz Prieto, Francisco Javier Delgado del Hoyo, Gustavo Río Briones, José Ignacio de Uribe Ladrón de Cegama, Raúl Santos Tejido, Yuri Torres De La Sierra e Iván Ramos Muñoz. Todos ellos han contribuido con sus Trabajos y Proyectos de Fin de Carrera en esta tesis.

A VERBIO por dejar su software disponible en la web y a David Font por contestar a mis cuestiones sin tener soporte contratado.

A mis compañer@s en la Junta de Castilla y León donde trabajé cuando comencé con todo esto. Especialmente a Isabel Lasheras, Eduardo Acedo y Myriam Grijalba por ser mis apoyos en el día a día durante varios años y a Antonio Cadenas Prieto por su apoyo "logístico".

A mis compañer@s de Hewlett-Packard, Hewlett-Packard Customer Delivery Services y otros partners durante mi trabajo en Osakidetza y especialmente a Joana García Ramos por prestarme sus apuntes de "TIM" y por compartir las penas y glorias diarias durante todo ese tiempo.

A mis compañer@s de la Universidad del País Vasco: Josu Doncel, Javier Mikel Olaso, Silvia Nieto, Mireia Díez Sánchez, Inari Badi y Natanael Ayllón Rozas por haberme ayudado en la fase de evaluación de usuarios y haberme apoyado durante el tiempo que compartimos. A Mireia agradecerle especialmente permitirme acceder a los recursos bibliográficos de la UPV-EHU y de la Brno University of Technology.

A mis compañeros Jorge Usabiaga, Iker Silvano, Naiara Espejo, Inés Martín, Carlos Varela y Gorka Unanue por compartir sus penas y alegrías conmigo durante esos casi ocho meses. A Gorka agradecerle que modelara la última versión del museo y a Inés agradecerle su apoyo y mantener la amistad.

A Amaya Elu Anacabe e Idoia Tolosa Beristain por confiar en mí para colaborar en HIRUDART.

A mis compañer@s de la Universidad de Deusto: Eva Rodríguez, Iñaki Prado, Roberto Montero, Mercedes Madrazo, Leire Uriagereka y especialmente a Ana Marcos.

A mis amigos Mikel Villán, Alberto Cuevas, Ander Gandarias, Karmel Zuazo, Iñaki Zuazo, Asier Jainaga, Gonzalo Uncilla por estar ahí tantas veces. A Iñaki agradecerle especialmente permitirme acceder a los recursos bibliográficos de la UPV-EHU.

A Virginia López Alonso por escucharme y apoyarme.

A Nathalia Alviz por animarme a finalizar este trabajo, por permitirme acceder a los recursos bibliográficos de la UPV-EHU y por compartir su tiempo en Euskadi conmigo.

A Jorge Augusto porque reencontrarme con él después de tantos años me ha motivado mucho.

A Gonzalo, María Jesús y Pablo porque son mi segunda familia.

A Juan Manuel Pascual Gaspar, por su compañerismo, por su amistad, por su compresión, por su apoyo, por ayudarme a empezar con Java, con Tex, etc.

A mis abuelos Gabriela y Jose Andrés. A mi padre Teófilo. A mi hermana Minerva por permitir que le robe sus gominolas. A mi madre Aurori.

A aquell@s que me he encontrado en mi vida y que cuando imprimo estas líneas no soy capaz de recordar pero que quizás deban aparecer en estas páginas bien por haberme apoyado o bien por todo lo contrario...

A los fabricantes siguientes por haber diseñado los programas y dispositivos utilizados en la redacción y desarrollo de esta tesis: DELL, APPLE, MICROSOFT, DROPBOX, ALTOVA, ADOBE, CORTONA, MIKTEX Project, APACHE Foundation, etc.

Por último, agradecer a Ennio Morricone, John Williams, Joe Hisaishi, Bingen Mendizabal, Alberto Iglesias, Danny Elfman, Pino Donnagio, John Barry, Michael Nyman y tantos otros el haber compuesto sus obras que me han acompañado durante las largas horas de redacción de este trabajo, haciéndolo más soportable y permitiéndome soñar con el día en que finalmente fuese publicado.

Resumen

Esta tesis aborda el problema de definir un lenguaje de marcas para modelar escenas, comportamiento e interacción basándonos en una filosofía de integración de componentes de aplicaciones de interacción multimodal en entornos gráficos 3D en base a la metáfora de película cinematográfica interactiva. Se reutilizan lenguajes de marcas ya definidos para describir gráficos, interacción gráfica e interacción vocal. Con la definición de este lenguaje se propone un marco común para desarrollar aplicaciones que permitan al usuario una interacción multimodal con escenarios 3D. Se han definido las bases de una arquitectura que ejecuta este tipo de aplicaciones respetando el marco común propuesto. Para ello, tras ubicar la problemática en el estado del arte relacionado, se presenta el lenguaje y la arquitectura intentando dar respuesta a la necesidad encontrada. Finalmente se muestra con la evaluación de un caso de uso desarrollado y ejecutado en el marco propuesto, los objetivos alcanzados y los que aun están en desarrollo.

Palabras clave: Interacción Vocal, Interacción Gráfica, Interacción Persona Ordenador, Multimodalidad, Sistemas de Diálogo, Avatar, Entornos Virtuales, 3D, Realidad Virtual, Aplicaciones de Internet Ricas, Comportamiento

Abstract

Based on a philosophy of integrating components from multimodal interaction applications with 3D graphical environments, reusing already defined markup language for describing graphics, graphical and vocal interactions based on the interactive movie metaphor, a markup language for modeling scenes, behavior and interaction is searched to define. With the definition of this language, we hope to have a common framework for developing applications that allow multimodal interaction at 3D stages. Thus we have defined the basis of an architecture that allows us to integrate the components of such multimodal interaction applications at 3D virtual environments.

Keywords: Vocal Interaction, Graphical Interaction, Human-Computer Interaction, Multimodality, Dialogue systems, Avatar, Virtual Environments, 3D Virtual Reality, Rich Internet Applications, Behaviour

Índice general

Lista de Figuras

Lista de Tablas

Listados de código fuente

1 Introducción

1.1 La interacción multimodal

Según la R.A.E. se define interacción como la "acción que se ejerce recíprocamente entre dos o más objetos, agentes, fuerzas, funciones, etc.". En el campo de estudio de la interacción persona ordenador, los objetos que interactúan son dos: el usuario y el sistema. La acción estará compuesta de las órdenes que el usuario envíe al sistema y de los mensajes que el sistema devuelva al usuario. En un entorno multimodal las acciones tendrán una naturaleza visual o sonora según sea la interacción gráfica o vocal respectivamente.

En los sistemas basados en escritorio la interacción del usuario con el sistema es fundamentalmente una interacción gráfica. El canal de entrada al sistema está basado en teclado y ratón mientras que el canal de salida del sistema es el monitor primordialmente.

En los sistemas de atención telefónica la interacción del usuario con el sistema es una interacción vocal o interacción basada en diálogos. El canal de entrada al sistema es el micrófono del teléfono y el canal de salida del sistema es el auricular.

Si se tuviese un sistema que permitiera introducir datos a través de teclado y ratón pero también a través de un micrófono y a su vez mostrara la salida bien por el monitor, bien por los altavoces del sistema o ambas a la vez (presentación multimedia), se diría que dicho sistema permite una interacción multimodal.

Esto que parece algo sencillo y que podría parecer una simple evolución de cada una de las interacciones presentadas, tiene una complejidad que radica en la propia naturaleza de cada una de las modalidades de interacción. Y es que si se ve de una manera básica, centrándose en la metáfora de los sistemas presentados: un escritorio no habla y un teléfono no muestra imágenes (refiriéndose al teléfono tal y como lo inventó Antonio Meucci y más tarde patentó Alexander Graham Bell).

Esta complejidad ha supuesto que la mayoría de los sistemas que intentan fusionar distintos tipos de interacción sean meros prototipos o sistemas muy focalizados a un objetivo concreto, como es el caso de los sistemas GPS que actualmente se integran en los salpicaderos de los automóviles.

La fusión multimodal permite acceder a ciertos sistemas a usuarios que en un principio no podrían acceder a éstos debido a su incapacidad de interactuar por el canal definido. Así, personas con discapacidad visual acceden a sistemas que en un principio sólo ofrecían una interacción gráfica y personas con discapacidad auditiva acceden a sistemas que sólo estaban pensados para interactuar con ellos desde una interacción basada en diálogos. Aumenta así el número de usuarios a los que pueden ir dirigidos los sistemas que permiten interacción multimodal sobre el número de usuarios a los que van dirigidos los sistemas que permiten interacción de un solo modo. Pero además de esta mejora en la accesibilidad, la multimodalidad puede aumentar las capacidades de todos los usuarios incrementando su productividad. La sustitución de la interacción gráfica por una interacción vocal libera las manos del usuario en ciertos momentos permitiéndole realizar otras tareas. Asimismo, la

interacción gráfica puede complementar la descripción vocal de un itinerario presentado gráficamente por los sistemas GPS referidos en el párrafo anterior. Además, las posibilidades que se abren en el mundo del entretenimiento son infinitas.

A lo largo de este capítulo de introducción se presenta la interacción multimodal en entornos virtuales que es la base de este trabajo. Se explicitará la motivación de esta tesis que no es otra que la necesidad de establecer un marco común para desarrollar aplicaciones que permitan interacción multimodal con entornos virtuales. Este marco común se verá que se basa en un lenguaje de especificación único y en una arquitectura soporte. Se presentarán los objetivos planteados con este trabajo de tesis para enfrentarlos con los resultados conseguidos. Finalmente se introducirá la estructura de la memoria de tesis.

1.2 La interacción multimodal en entornos virtuales

En referencia al concepto de espacios virtuales del título de esta tesis, éste se ha elegido para englobar imágenes bidimensionales o viñetas con movimiento, entornos virtuales tridimensionales y realidad virtual clásica. El trabajo se centrará en un principio en los entornos virtuales tridimensionales es decir, escenas que transcurren en mundos tridimensionales visualizados en una pantalla convencional con las que el usuario va a poder interactuar vía teclado, ratón y micrófono.

Existen varias definiciones para la realidad virtual : "Realidad Sintética", "Mundos Virtuales o Ficticios", "Ciberespacio", "Ambientes Virtuales" (Virtual Environments) y más particularmente "Presencia" pero se considera más apropiada la definición que indica que la realidad virtual consiste en simulaciones tridimensionales interactivas que reproducen ambientes y situaciones reales (Parra Márquez, García Alvarado & Santelices Malfanti, 2001).

Se dice que una aplicación de realidad virtual ha de cumplir con ciertas condiciones:

- Simulación, en el sentido de la capacidad de ser un sistema que permita la representación de una realidad;
- Interacción, para controlar el sistema o mundo representado (en nuestro caso ratón, teclado y micrófono);
- Percepción, permitiendo "engañar" a los sentidos a través de elementos externos (en nuestro caso monitor y altavoces).

Es por todo esto que la realidad virtual y los espacios virtuales en general, ofrecen el escenario ideal para la fusión multimodal y es en la vertiente interactiva de los espacios virtuales donde se encuentra la necesidad de integrar modos de interacción. En concreto la interacción vocal y la interacción gráfica. Si se consigue esta fusión de una manera natural, se hará que la realidad simulada en el sistema se perciba más claramente.

Así, la interacción vocal puede ser una componente clave de la interacción multimodal ya que la interacción gráfica ha sido explotada de manera principal por todos los sistemas, desde los basados en escritorio hasta los basados en realidad virtual ya que es la más obvia. Aportar interacción vocal puede ser clave en la mejora de las cualidades de los sistemas basados en espacios virtuales, pudiendo de esta forma eliminar los elementos que permiten y obligan a esa interacción gráfica (teclado y ratón) en algunos casos; complementando la información gráfica (con órdenes vocales) y mejorando el feedback gráfico o realimentación gráfica (con complementos acústicos) en otros.

La realización de aplicaciones relacionadas con mundos virtuales que presenten una interfaz multimodal es un desafío tecnológico de indudable utilidad práctica. El desafío estriba en la integración de dos mundos cercanos en la vida real pero distantes aún en el mundo de las nuevas tecnologías.

Aplicaciones de este tipo permitirían modificar un entorno gráfico simplemente pronunciando las órdenes y el usuario viendo los resultados y escuchando la respuesta del sistema se percataría del cumplimiento de lo requerido. Asimismo, podría seguir interactuando gráficamente (teclado y ratón) con el mundo 3D en aquellas tareas en las que se encuentre cómodo trabajando con el mundo virtual.

Sin embargo, las aplicaciones 3D con interfaz vocal escasean. Esto está motivado por las limitaciones tecnológicas que radican en la propia naturaleza de cada tipo de interacción. Cada modo de interacción se basa en lo que llamamos metáforas que son aproximaciones de elementos de la realidad en los que nos basamos al definir la interacción con los sistemas para hacer que éstos sean más familiares al usuario. Las metáforas de interacción vocal poco van a tener que ver con las metáforas de interacción gráfica como se verá y ésto ha provocado que cada una se desarrolle de una manera divergente.

En próximas revisiones se podrá extrapolar dicha interacción a dispositivos hápticos combinados con una presentación gráfica realizada a través de dispositivos montados en la cabeza o Head Mounted Displays. Se conseguirán así sistemas más inmersivos. De esta manera se abarcará desde la interacción multimodal con imágenes bidimensionales hasta entornos virtuales tridimensionales y todo lo planteado podrá ser extrapolado a la interacción multimodal en realidad virtual.

1.3 Hacia un marco de integración

El objetivo es la estandarización de un método de desarrollo para poder unir el mundo gráfico y el vocal, maximizando la capacidad de reutilización de código y datos. De esta forma se crean aplicaciones donde se reciben eventos desde la entrada gráfica y la entrada vocal, y que tendrán asociadas acciones de complejidad arbitraria. Dichas acciones se ejecutan sobre elementos del mundo virtual, en las cuales pueden intervenir uno o varios de estos elementos virtuales que a su vez, pueden interactuar o no entre ellos. Además, tomarán decisiones en tiempo real en función del estado de cada uno. Para ello se establece un marco basado en un lenguaje de especificación común y una arquitectura soporte.

1.3.1. Un lenguaje de especificación común

Los ámbitos de la realidad virtual y de los sistemas de diálogo se caracterizan por una relativa disponibilidad de prototipos realizados en laboratorios de investigación (Gustafson, Bell, Beskow, Boye, Carlson, Edlund, Granström, House & Wirn, 2000) y de algún sistema comercial que por lo general, han descuidado la necesidad de ajustarse a algún estándar de desarrollo o de especificación. El estándar en sistemas de diálogo es VoiceXML (VoiceXML Forum, 2010) mientras que en realidad virtual hay un estándar de definición de escenas X3D (Brutzman & Daly, 2007) evolucionado de VRML (Hartman & Wernecke, 1996). La disponibilidad de estos estándares ha supuesto un marco de referencia para que los desarrolladores adapten sus sistemas, con las consiguientes aportaciones en cuanto a facilidad de uso en lo que se refiere a la definición de escenarios 3D, diálogos y portabilidad de módulos reutilizables. De la misma forma, disponer de un estándar para especificar los dos ámbitos podría suponer un impulso al desarrollo de aplicaciones que integren interacción multimodal y mundos 3D. Se podrá especificar así escenas 3D, comportamiento, interacción gráfica, interacción vocal y la integración de ambos modos de interacción. Dado que existen lenguajes para especificar cada uno de estos elementos por separado, tiene sentido que se reutilicen elementos de cada uno de ellos. Por ello se revisarán los lenguajes existentes actualmente para especificar las necesidades definidas.

1.3.2. Una arquitectura soporte

A pesar de que el desarrollo de la interacción gráfica está consolidado, para desarrollar una interfaz multimodal se necesitan resolver requisitos de diseño y técnicos tales como el reconocimiento del habla, la comprensión del lenguaje y basarse en una metáfora de interacción vocal (Dahl, 2004). Reconocimiento del habla y comprensión del lenguaje se han resuelto definiendo lenguajes de especificación de interacción vocal (VoiceXML Forum, 2010). Sin embargo, el problema de la metáfora de interacción vocal sigue sin resolverse y menos aún de cara a una integración con las

metáforas de interacción gráficas. La aproximación general a la fusión multimodal hace necesaria la definición de una arquitectura soporte reutilizable para construir nuevas aplicaciones multimodales. La arquitectura soporte definida debería ser capaz de poner en marcha una aplicación que permita interacción multimodal con mundos virtuales. Esta arquitectura software ha de ser multiplataforma y deberá basarse a su vez en los estándares existentes actualmente para definir arquitecturas que permitan interacción multimodal como (Barnett, Dahl, Kliche, Tumuluri, Yudkowsky, Bodell, Porter, Raggett, Raman & Wahbe, 2008). Para ello, se hará una recopilación de propuestas existentes a nivel de arquitectura que serán la base de la que se proponga.

1.4 Objetivos de la tesis

Considerando que las tres componentes de la interacción multimodal para entornos 3D son:

- *la especificación 3D* que básicamente consiste en modelar objetos del entorno virtual que pueden ser estáticos y, o dinámicos.
- *la especificación de la interacción gráfica* basada en teclado y ratón como la conocemos hasta ahora.
- *la especificación de la interacción vocal* donde se establece el diálogo o los diálogos entre el usuario y el sistema.

A pesar de que hay estándares para cada una de ellas por separado, no existen propuestas de estandarización que engloben estas tres componentes y aquí radica la justificación de esta tesis. El objetivo principal es definir un marco de representación que será el lenguaje de especificación y una arquitectura para interpretar los ficheros escritos en este lenguaje. Todo ello deberá ser evaluado para así demostrar los resultados esperados. En resumen, los objetivos serán:

1. Definir un lenguaje de especificación de espacios virtuales con interacción multimodal.
2. Definir una arquitectura software que permita interpretar los programas escritos en dicho lenguaje.
3. Desarrollar y explotar aplicaciones sobre la arquitectura soporte definida para así ilustrar las capacidades del lenguaje de especificación único y de la arquitectura soporte para implementarlas.
4. Evaluar el desarrollo, implementación y explotación de los sistemas especificados.

1.5 Resultados conseguidos

A continuación se presentan las contribuciones más relevantes realizadas en el marco de la presente tesis:

Revisiones de la literatura. Se revisan en detalle y desde un punto de vista crítico los trabajos más relevantes de la literatura intentando dar una visión global de lo existente y dejando al descubierto aquellas necesidades o ámbitos de estudio menos trabajados de cara a que sean los puntos a cubrir por la investigación realizada. Se aborda de una manera ordenada y sencilla pero que permite a la vez al lector poder indagar más siguiendo las referencias de las fuentes utlizadas (ver capítulo 2).

Lenguaje de especificación. Se presenta un lenguaje de especificación que permite declarar escenas, diálogos y comportamientos tal y como se ha deducido que era necesario a partir de encontrar las carencias oportunas en el estado del arte. Para ello se han utilizado técnicas de representación propias de la naturaleza del lenguaje definido de una manera didáctica, sencilla y auto explicativa (ver capítulo 3).

Cobertura de metáforas de interacción y cooperación entre modalidades. Se muestra la capacidad del lenguaje de especificación propuesto para especificar las distintas metáforas de interacción gráfica y vocal así como la cooperación entre ambas (ver capítulo 4). Estas últimas han sido establecidas a partir de la revisión realizada en el estado del arte que ha marcado el ámbito a cubrir por el lenguaje de especificación definido. Se muestran ejemplos de implementación basados en un caso real que se detallan y explican remarcando los aspectos del lenguaje que implementan la problemática específica de cada uno pero cubriendo su implementación global.

Arquitectura. Se ha diseñado una arquitectura software que permite implementar aplicaciones multimodales para interacción con espacios virtuales intentando seguir los estándares que existen para desarrollar aplicaciones multimodales (ver sección 5.2). Esta arquitectura interpreta los ficheros escritos en el lenguaje de especificación propuesto y construye las aplicaciones citadas. De una forma modular se ha representado cada parte de la arquitectura y se han detallado cada uno de sus elementos y subelementos relacionándolos con los estándares mencionados.

Aplicaciones. Se implementa la arquitectura diseñada desarrollando pequeñas aplicaciones multimodales que interaccionan con un mundo virtual usando distintas tecnologías de manera que se demuestra la independencia de plataforma de la arquitectura propuesta (ver sección 5.3). Siguiendo la estructura modular utilizada para describir la arquitecura, se han descrito las implementaciones de cada una de las aproximaciones tecnológicas desarrolladas dejando clara la relación entre dichas implementaciones, la arquitectura global y los estándares en los que se ha basado esta tesis.

Participación en congresos y publicaciones. La propuesta de lenguaje y arquitectura para especificar y desarrollar aplicaciones multimodales que interactúan con espacios virtuales ha sido difundida en congresos internacionales y publicaciones. En las primeras publicaciones (Olmedo-Rodríguez, Escudero-Mancebo, González-Escribano, González-Ferreras & Cardeñoso-Payo, 2006) se presenta el lenguaje de especificación para seguidamente introducir la arquitectura software de una manera teórica (Olmedo-Rodríguez, Escudero-Mancebo, González-Escribano, González-Ferreras & Cardeñoso-Payo, 2007b) y (Olmedo-Rodríguez, Escudero, González, González & Cardeñoso, 2007a). Tras revisar las posibles tecnologías con las que se podría implementar la arquitectura (Olmedo-Rodríguez, Cardeñoso-Payo & Escudero-Mancebo, 2008a) y (Olmedo-Rodríguez, Escudero-Mancebo & Cardeñoso-Payo, 2008b), se demuestra que ésta sigue la filosofía de los principales grupos de estandarización (Olmedo-Rodríguez, Escudero-Mancebo, Cardeñoso-Payo, González-Ferreras & González-Escribano, 2009c). Finalmente se han ido describiendo los métodos de evaluación de la propuesta (Escudero-Mancebo, Olmedo-Rodríguez & Cardeñoso-Payo, 2008), (Olmedo-Rodríguez, Escudero-Mancebo & Cardeñoso-Payo, 2009a), (Olmedo-Rodríguez, Escudero-Mancebo & Cardeñoso-Payo, 2009b) y (Olmedo-Rodríguez, Sanz Prieto, Escudero-Mancebo & Cardeñoso-Payo, 2010). En las últimas publicaciones (Olmedo-Rodríguez & Augusto, 2012), (Olmedo-Rodríguez & Augusto, 2013) y (Olmedo-Rodríguez, 2013) se toma contacto con tecnologías de realidad aumentada y realidad mixta para extender la propuesta a dichos entornos como una línea de investigación para el trabajo futuro de esta tesis. En la tabla C.1 del apéndice C se relacionan las publicaciones en orden cronológico.

1.6 Estructura de la memoria

En el capítulo 2 se presenta el estado del arte de todas las tecnologías relacionadas con el ámbito de la tesis, introduciendo los lenguajes de marcado, las metáforas de interacción, las arquitecturas para multimodalidad, los sistemas de diálogo, las tecnologías RIA para 3D, las aplicaciones multimodales y la evaluación de aplicaciones multimodales. Todo esto nos servirá para tener una visión más amplia y global del campo de estudio en el que se sitúa el trabajo de tesis.

A continuación, en el capítulo 3 se presenta el lenguaje de especificación común a utilizar para definir interacción gráfica y vocal, escenas y comportamiento. Tras dejar claros sus principios de diseño y metas se detallarán sus conceptos y elementos utilizando metodologías propias de la naturaleza de este lenguaje.

Seguidamente en el capítulo 4 se demostrarán las capacidades de este lenguaje para especificar las distintas metáforas de interacción gráfica y vocal así como la cooperación entre ambas modalidades presentando ejemplos de implementación de casos concretos implementados con el lenguaje de especificación común.

Tras ello, en el capítulo 5 se verán los requisitos de la arquitectura soporte que implemente las aplicaciones descritas y se establecerán los aspectos arquitectónicos destacando que nuestra propuesta no construye una aplicación desde cero si no que reutiliza tecnologías que cumplen los requisitos definidos basándose en estándares de arquitecturas para multimodalidad.

Finalmente el capítulo 6 se centrará en la evaluación de una aplicación desarrollada siguiendo la propuesta de arquitectura soporte y lenguaje de especificación común en base a las metodologías de evaluación presentadas. Servirá para investigar sobre las preferencias de los usuarios ante las diferentes modalidades en base a su experiencia de interacción.

En el capítulo 7 se presentarán las conclusiones oportunas en base al esquema de evaluación propuesto y se finalizará con un resumen global a modo de revisión del trabajo de tesis.

2
Estado del arte

Para ubicar este trabajo y presentar los elementos implicados, tras introducir algunas aplicaciones multimodales de interacción con espacios 3D que nos harán comprender la problemática a solucionar, se verán las posibilidades que existen para especificar interacción vocal, escenas y comportamiento con lenguajes de marcado. Puesto que es necesario implementar metáforas de interacción con estos lenguajes, se enumerarán las metáforas correspondientes a la interacción gráfica, las correspondientes a la interacción vocal, así como las diferentes formas de cooperación entre ambas. Se hace necesaria una arquitectura software capaz de construir aplicaciones multimodales en base a los guiones expresados con los lenguajes de marcas, por ello se revisarán las propuestas existentes actualmente para diseñar sistemas multimodales. Estos sistemas permitirán interacción vocal, por lo que es obligado conocer las herramientas para implementar sistemas de diálogo. Además, estos sistemas representarán escenas 3D sobre entornos web. Por ello, se introducirán las tecnologías 3D RIA. Finalmente para evaluar las aplicaciones desarrolladas, se describirán las posibilidades que se ofrecen para evaluar sistemas multimodales.

2.1 Aplicaciones multimodales 3D

Sin ánimo de ser exhaustivos repasaremos los intentos de la comunidad científica para desarrollar aplicaciones multimodales que interactúan con espacios 3D ya que a nivel comercial existen pocos o nulos ejemplos.

Siguiendo un orden temporal empezaremos repasando el proyecto "Talk and Draw" de Boeing que desarrolló una estación de trabajo AWACS (Salisbury, Hendrickson, Lammers, Fu & Moody, 1990) para dirigir operaciones militares. Permitía a través del ratón seleccionar objetos gráficos que se combinaban con comandos vocales. En este sistema se mantenía un contexto a partir de la entrada gráfica que complementaba el reconocimiento vocal.

El proyecto llamado The DIVERSE System (Mcglashan & Axling, 1996b) integraba reconocimiento de voz que interaccionaba con un agente que permitía al usuario seleccionar y manipular objetos en un mundo virtual modelado en un sistema DIVE (Distributed Interactive Virtual Environment) (Avatare, Frécon, Hagsand, Jää-Aro, Simsarian & Ståhl, 1997). Su evolución supuso un campo de testeo de las diferentes metáforas de interacción vocal propuestas por los autores (Mcglashan & Axling, 1996a).

CUBRICON (Billinghurst, 1998) es un interfaz similar al de "Talk and Draw" y desarrollado para dirigir y controlar fuerzas aéreas. En este caso la relación entre discurso, texto y gesto se formalizó a través de gramáticas ATN modificadas (Kaplan, 1975).

El sistema QuickSet (Oviatt & Cohen, 2000) propone una arquitectura multimodal para discurso y gesto basado en agentes y colaborativo que se ejecuta en ordenadores personales de diferentes

tamaños. Se desarrolló junto a aplicaciones basadas en mapas, medicina, simulación militar, entrenamiento, visualización virtual en 3D de terrenos y gestión de desastres. Permitía al usuario crear y posicionar entidades en un mapa virtual a través de discurso, escritura caligráfica y/o manipulación directa. Las entradas de las distintas modalidades se almacenan en estructuras atributo-valor similares a XML.

AdApt (Gustafson, Bell, Beskow, Boye, Carlson, Edlund, Granström, House & Wirn, 2000) se desarrolló para aplicaciones inmobiliarias permitiendo hacer click en un mapa interactivo los inmuebles de nuestro interés y preguntar a un agente los datos que nos interesen: precio, características, etc.

La arquitectura de Ulysse (Nugues, 2000) también se basa en DIVE (Avatare, Frécon, Hagsand, Jää-Aro, Simsarian & Ståhl, 1997) y realiza una representación del usuario en el mundo virtual permitiendo la interacción con un agente conversacional.

La comunidad virtual RIF (van Ballegooij & Eliéns, 2001) es un entorno multiusuario donde varios usuarios interactúan con un mundo virtual desarrollado en VRML a través de queries escritas a modo de chatterbots.

El sistema Gandalf (Traum & Rickel, 2001) se basa en agentes humanoides conversacionales que cohabitan en un espacio con el usuario permitiendo interacción gestual y seguimiento de los ojos del usuario para que éste consulte datos sobre el Sistema Solar.

La arquitectura para interacción multimodal "Flip the monitor" (Kaiser, Olwal, McGee, Benko, Corradini, Li, Cohen & Feiner, 2003) integra reconocimiento de voz, interacción gestual y seguimiento de ojos para interactuar con un agente 3D.

Soportadas por la arquitectura ICARE (Bouchet & Nigay, 2004) se desarrollaron tres aplicaciones: MEMO permite al usuario anotar localizaciones físicas con notas digitales que tienen una localización física y que pueden ser leídas y eliminadas por otros usuarios. Para ello soporta cinco modalidades de entrada activas y pasivas. MID (Multimodal IDentification) soporta tres modalidades equivalentes que permiten al usuario identificarse (discurso, una secuencia de botones pulsados utilizando el ratón y una clave de acceso introducida vía teclado). FACET (Bouchet, 2004) es un simulador de vuelo en tiempo real basado en un avión militar para estudiar futuras técnicas de interacción a implementar en el avión real.

El proyecto NICE (Corradini, Mehta, ole Bernsen & Charfuelan, 2005) desarrolla cuentos interactivos donde se puede interactuar de manera gestual y vocal. Se implementa con módulos especializados que se comunican vía TCP/IP utilizando estructuras XML.

La arquitectura OpenInterface (Serrano, Nigay, Lawson, Ramsay, Murray-Smith & Denef, 2008) permitió desarrollar el juego 3D para teléfonos móviles "Funny Rabbit" y un navegador de mapas multimodal "Multimodal Map Navigation" que permitía señalar en su pantalla táctil "Diamond touch" un lugar e indicar de manera vocal que se hiciera zoom del lugar señalado.

Dos aplicaciones tuvieron a la arquitectura SAIBA (Kipp, Heloir, Schröder & Gebhard, 2010) como soporte: RealActor (Cerekovic, Pejsa & Pandzic, 2009) y Elckerlyc (van Welbergen, Reidsma, Ruttkay & Zwiers, 2010) que generaban comportamiento para agentes conversacionales y comportamiento verbal y no verbal multimodal para humanoides virtuales, respectivamente.

Varias aplicaciones se han desarrollado con MINT Framework (Feuerstack, 2013) como plataforma soporte, desde GestureUI (Mauro Dos Santos Anjo & Feuerstack, 2012) que es un reconocedor automático de lengua de signos, hasta un navegador multimodal que reconoce gestos y permite apuntar con el dedo el destino en un mapa (Feuerstack & Pizzolato, 2011), pasando por una tienda online de muebles que permite colocar los muebles en una realidad aumentada (Feuerstack, Oliveira & Araujo, 2011).

Viendo la heterogeneidad de las tecnologías utilizadas y ante la imposibilidad de reutilización y colaboración para desarrollar nuevas propuestas se deduce que es necesaria la estandarización del diseño de sistemas multimodales y la necesidad de diseñar guiones que establezcan las diferentes situaciones a tratar. Para escribir estos guiones es necesario un lenguaje como marco aglutinador.

Para definir qué aplicaciones de realidad virtual pueden aprovechar mejor la multimodalidad,

se presenta una clasificación en cuatro grandes grupos de las aplicaciones de realidad virtual en la actualidad (Vince, 2004) y que se muestran en la tabla 2.1.

Tabla 2.1: Clasificación de las aplicaciones de la VR en la actualidad

GRUPOS	APLICACIONES DE LA REALIDAD VIRTUAL
Sistemas de producción industrial	Visualización de conceptos de ingeniería Entrenamiento del personal Evaluación de aspectos ergonómicos Servicio usando un prototipo virtual Visualización de armas virtuales Exploración de estrategias de servicio Simulación de la interacción de ensamblajes Simulación de la dinámica de estructuras articuladas Análisis de stress Gestión de desarrollo de producto distribuido Simulación de procesos de fabricación Ingeniería de procesos en grandes proyectos Simulación de máquinas y presión Ingeniería concurrente Ergonomía Diseño de puentes Prototipos virtuales Ingeniería visual Visualización espacial
Simuladores de entrenamiento	Medicina (Modelado detallado del cuerpo, Cirugía mínimamente intrusiva, Terapia virtual) Enseñanza, aprendizaje Simuladores de vuelos civiles Simuladores de vuelos militares Simulaciones estratégicas Simulación de conducción de trenes Simulación de vehículos
Entretenimiento y herencia cultural	Juegos de ordenador Juegos recreativos Experiencias de parques temáticos y Museos
Centros de RV	Diseño de interior Desarrollo urbano Diseño aeroportuario Análisis del movimiento humano

Como conclusión, vemos que todas intentan dar solución a un mismo problema: la multimodalidad. Cada una con tecnología propia siendo imposible la reutilización y colaboración para desarrollar nuevas propuestas. Pero en todas se diseñan guiones que establezcan las diferentes situaciones a tratar. Es necesaria la estandarización del diseño de estos guiones, de ahí nuestra propuesta de un lenguaje de especificación que consideramos ha de basarse en lenguajes de marcado. Por ello, presentamos a continuación los lenguajes de marcado existentes.

2.2 Lenguajes de marcado

Existen numerosos lenguajes de marcado para especificar interacción vocal, otros tantos para especificar escenas bien sean en un plano bidimensional o en un espacio 3D. También existen lenguajes de marcado para especificar el comportamiento de los elementos que aparecen en las escenas. Se presentarán ejemplos de todos ellos junto a los lenguajes de marcado denominados "híbridos" (Pihkala, Honkala & Vuorimaa, 2002) ya que se basan en varios lenguajes de marcado básicos o dedicados es decir, se basan en lenguajes de marcado para especificar interacción vocal, para especificar escenas y/o para especificar comportamiento.

2.2.1. Lenguajes de marcado para especificar interacción vocal

Algunos de los principales lenguajes de especificación de interacción vocal utilizados por los sistemas de diálogo actuales son los siguientes:

VoiceXML Voice eXtensible Markup Language o VXML (VoiceXML Forum, 2010) es un lenguaje XML desarrollado por el VoiceXML Forum, un consorcio formado por más de 325 compañías fundado en 1999 por AT&T, IBM, Lucent y Motorola, y cuya primera especificación fue publicada en el año 2000 (Boyer, Linda et al., 2000). El W3C la elevó a categoría de "Recomendación W3C" en marzo de 2004 (McGlashan, Scott et al., 2004). Es un lenguaje XML para crear diálogos de audio formados por habla sintética, audio digitalizado, reconocimiento de habla y tonos DTMF (Dual Tone Multi-Frequency), grabación del habla, telefonía y conversaciones de iniciativa mixta. Su misión es la de facilitar el desarrollo de aplicaciones vocales de iniciativa mixta para ámbito telefónico. Este lenguaje usa un enfoque declarativo para la construcción de aplicaciones, las cuales son creadas mediante la combinación de etiquetas VoiceXML en uno o más documentos. Según sus autores, su principal ventaja es la de traer las ventajas del desarrollo basado en web a las aplicaciones vocales interactivas. La recomendación de VoiceXML 2.0 se engloba dentro de una plataforma mayor conocida como W3C Speech Interface Framework (Larson, 2010), cuyo grupo responsable es el W3C VBWG (W3C Voice Browser Working Group (W3C, 2010c)). En resumen, es un formato estándar para la especificación de la interacción persona ordenador. Sirve para desarrollar las aplicaciones como se hace habitualmente con el formato HTML. Los ficheros VoiceXML son interpretados por los voice browsers. Existen diferentes entornos para trabajar con VoiceXML como Tellme de Microsoft (Microsoft, 2011b). Existe también una extensión en el WebSphere Studio para realizar aplicaciones Java basadas en VoiceXML (IBM, 2012).

SALT Speech Application Language Tags (Microsoft, 2003) es un lenguaje de especificación basado en XML formado por un pequeño conjunto de etiquetas que pueden ser usadas embebidas en otros lenguajes de marcado ya existentes como HTML, XHTML o XML que actúan como lenguajes contenedores y cuya finalidad es la de proporcionar un medio con el cual poder escribir aplicaciones vocales tanto para sistemas basados únicamente en voz (aplicaciones de telefonía) como para la creación de aplicaciones web multimodales. SALT ha sido desarrollado por el SALT Forum, un consorcio de empresas fundado en 2001 por Cisco, Comverse, Intel, Microsoft, Philips y SpeechWorks, al cual se han añadido posteriormente más de 60 miembros. En Julio de 2002 este foro envió la primera versión del lenguaje SALT como propuesta de

lenguaje multimodal al grupo de interacción multimodal W3C para su consideración. El principal objetivo de SALT es el de la creación de un estándar libre de derechos de autor, e independiente de la plataforma para la creación de aplicaciones multimodales. Compitió con VoiceXML como apuesta de Microsoft, utiliza SAPI o Speech API para la comunicación entre módulos. El principal inconveniente es que sólo es soportado en entornos de Microsoft. En 2004, VoiceXML se convirtió en el estándar recomendado, así que Microsoft tuvo que hacer que sus sistemas (Speech Server 2006) también pudieran trabajar con VoiceXML y en 2007 Microsoft desarrolló Tellme (Microsoft, 2011a) uno de los editores más potentes de VoiceXML. Finalmente en ese mismo año el consorcio SALT Forum dejó de apoyar el lenguaje SALT ya que Microsoft se volcó de lleno en el desarrollo de productos basados en VoiceXML.

SRGS Speech Recognition Grammar Specification (Hunt & McGlashan, 2010) es una especificación de representación de gramáticas. Las gramáticas están pensadas para ser usadas por reconocedores de habla así como por otros procesadores de gramáticas de forma que los desarrolladores puedan especificar las palabras y patrones de palabras a ser escuchadas por un reconocedor de habla basado por ejemplo en VoiceXML. La especificación de las gramáticas puede hacerse de dos formas, aunque ambas son semánticamente equivalentes:

- mediante ABNF (Crocker, 2010) (Augmented BNF), una forma de representación textual no XML similar a las tradicionales gramáticas BNF (Garshol, 2010) y derivadas (como lo es por ejemplo Java Speech Grammar Format, de la que hablaremos más adelante).
- mediante elementos XML, para poder realizar transformaciones automáticas.

SSML Speech Synthesis Markup Language (Burnett, Walker & Hunt, 2010) es una recomendación W3C que forma parte de un conjunto mayor de especificaciones para la creación de navegadores vocales desarrollada por el W3C. Ha sido diseñada para proporcionar un lenguaje XML que asista la generación de habla sintética en la web y otras aplicaciones. Su función principal es la de proporcionar a los autores de contenido sintetizable una forma estándar de controlar aspectos del habla de salida tales como la pronunciación, volumen, velocidad, tono, etc. a través de diferentes plataformas con capacidades de síntesis, estando por tanto dirigida a la producción de contenido sintetizado de calidad. Al igual que el anterior puede usarse junto a VoiceXML.

X+V XHTML + Voice Profile (Axelsson, Jonny et al., 2010) es un lenguaje XML orientado al desarrollo de páginas web multimodales. X+V está formado por un subconjunto de VoiceXML combinado con XHTML (especificación XML de HTML). X+V fue desarrollado inicialmente por las empresas IBM, Motorola y Opera, que lo enviaron al W3C para su revisión en Diciembre de 2001. X+V está basado en su totalidad en estándares, de forma que combina la interacción vocal a través de un lenguaje ampliamente utilizado como es VoiceXML junto a la presentación visual de la página web que es llevada a cabo mediante XHTML. La comunicación entre ambos lenguajes se realiza mediante el estándar de eventos XML formando estos tres lenguajes la piedra angular sobre la que se sustenta X+V. Dado que VoiceXML lo permite, con X+V pueden crearse tanto aplicaciones de habla básicas de diálogos dirigidos, como interacciones más complejas de diálogos de iniciativa mixta. Una de las ventajas de X+V es que no introduce ningún nuevo lenguaje de programación para incorporar voz a una página web, sino que reutiliza lenguajes ampliamente conocidos y documentados y que han sido probados con éxito en multitud de situaciones reales.

AIML Artificial Intelligence Markup Language (ALICE, 2011a) es un dialecto XML para la creación de agentes de software de lenguaje natural (AITOOLS, 2011). Creado por Richard Wallace, su última versión data de 2005. Usado fundamentalmente para la realización de Chatterbots (Paillard, 2011), se utiliza en el proyecto Alicebot de la fundación ALICE (ALICE, 2011b). Se basa en la ley de Zipf (Wallace, 2011). Los estados de la ley de Zipf dan un corpus

del lenguaje natural. La frecuencia de cada palabra es inversamente proporcional al rango en la tabla de frecuencias, así las palabras más frecuentes aparecerán en primer lugar en la tabla. Existen corpus, en los que se han realizado tests y en muy pocos casos ha sido necesario revisar más allá de la mitad de la tabla de frecuencias para encontrar la palabra.

JSGF Java Speech Grammar Format (Hunt, 2010) es un formato para la especificación de gramáticas para aplicaciones de diálogo desarrollado por Sun y existe un proyecto que ha desarrollado su propio editor (Holzapfel, 2011). Es un formato muy bueno para gramáticas sencillas y pequeñas.

CCXML Call Control XML (Baggia & Scott, 2010) es el lenguaje de marcado estándar W3C para controlar cómo las llamadas de teléfono son asignadas, respondidas, transferidas, conferenciadas, y más. CCXML trabaja mano a mano con VoiceXML (Amyot & Simoes, 2006) para proveer una solución basada en XML 100 % estándar para cualquier aplicación de telefonía. Es un lenguaje de máquina de estados basado en eventos diseñado para soportar características de control de llamadas en aplicaciones de voz (específicamente incluyendo VoiceXML pero no limitado a ello).

2.2.2. Lenguajes de marcado para especificar escenas

Los entornos virtuales tridimensionales son representaciones visuales generadas por ordenador que permiten al usuario una interacción que normalmente se realiza a través de interfaces gráficas, interacción gráfica. Para su especificación se utilizan lenguajes de programación tales como Java o C++ pero también navegadores que interpretan lenguajes de marcado similares al HTML usado para realizar páginas web y que son presentados a continuación.

VRML Virtual Reality Markup Language (Hartman & Wernecke, 1996) es un formato de fichero web para describir formas 3D y mundos interactivos. Los ficheros de texto VRML utilizan una extensión .wrl y cualquier servidor web puede hospedarlos. Para visualizar mundos VRML se necesita un navegador VRML que puede estar integrado en un navegador web. Para crear mundos VRML basta cualquier editor de textos o una aplicación de creación de mundos. Permiten a los autores controlar formas y un cierto grado de animación e interacción. Para un control mayor, los autores pueden escribir pequeños programas en Java o Javascript utilizando el API External Authoring Interface, EAI (Phelps, 2010). Su uso es cada vez menos frecuente.

X3D Extensible 3D (Web3D Consortium, 2010) es el sucesor de VRML. Se basa en XML y como su predecesor, a través de los documentos que siguen este formato se pueden definir las características gráficas de una escena así como de los elementos que interactúan en ésta. La especificación de X3D se divide en componentes, que son conjuntos de objetos y servicios con funcionalidad relacionada. Realmente, esto puede incluir nuevos nodos, una nueva apariencia u otras características. Algunos componentes que define el estándar son iluminación, scripting, geometrías 2D y 3D, sonido, etc. Por otra parte, un profile es un conjunto definido de componentes, que permite en un navegador dar el nivel de servicio requerido para un determinado fichero de X3D, sin tener que implementar la especificación de forma completa. Se permite realizar nuevos nodos/componentes/profiles. Al igual que VRML tiene el API EAI, existe el API SAI (Scene Access Interface) (Hudson, 2010) para X3D. Su utilización está en crecimiento.

3DML 3D Markup Language (Figueroa, Green & Hoover, 2001) es un formato para crear sitios web tridimensionales. Los ficheros están escritos en un formato similar a XML y pueden visualizarse con un plugin del navegador web o con un navegador 3DML independiente llamado Flatland Rover (Powers, 2005). Un mundo 3DML se llama "Spot". En éste pueden insertarse "blocks", que pueden ordenarse en "levels" teniendo el mismo tamaño cada uno. Se puede navegar entre los spots usando el ratón o los cursores. 3DML es una aplicación XML basada en un DTD. Existe un fichero DTD para todo el conjunto de elementos que define los tres elementos

principales: application, package, o platform class. Otros ficheros definen paquetes específicos de filtros, dispositivos, aplicaciones o plataformas. Su uso es puramente testimonial.

MPEG4 ISO/IEC 14496-1, también conocido como MPEG-4 Systems, (The Moving Picture Experts Group, 2013) provee la tecnología para representar y proveer de manera interactiva y síncrona contenidos audiovisuales compuestos de varios objetos que incluyen audio, video, gráficos vectoriales 2D/3D, etc. Especifica un modelo de terminal para gestión de tiempo y buffer, un framework para la identificación, descripción y dependencias lógicas de streams elementales, una estructura de paquetes portadores de sincronización de información y una representación multiplexada de streams elementales e independientes en un único stream (M4Mux).

2.2.3. Lenguajes de marcado para especificar comportamiento

Las limitaciones de VRML y X3D para especificar el comportamiento de los elementos integrantes de la escena han llevado a definir lenguajes de especificación para comportamiento. Se muestran dos de ellos a continuación.

Behavior3D El comportamiento de los elementos que se presentan en una escena es algo que no se ha representado de una manera amplia en ninguno de los lenguajes de definición de entornos virtuales tridimensionales, bien sea VRML o X3D el lenguaje utilizado, por ello en el proyecto CONTIGRA (Dachselt, Hinz & Meiner, 2002) se definió un lenguaje basado en XML que modela el comportamiento de los elementos que forman parte de las escenas 3D. A pesar de que X3D considera el comportamiento de los nodos que componen una escena, este comportamiento es necesario que se extienda añadiendo scripts a los elementos que especifiquen comportamientos o definiendo prototipos de comportamiento para los nodos. Behavior3D (Dachselt & Rukzio, 2003) es un XMLSchema automáticamente generado que integra todos los comportamientos disponibles dando un repertorio de definiciones de comportamiento. Comprende un concepto genérico de orientación a objetos uniendo nodos predefinidos, scripts, el concepto de prototipo de X3D y el concepto de clase de VRML++ (Diehl, 1997). Con Behavior3D se desarrolló un nuevo concepto de comportamiento declarativo para proveer definiciones de comportamiento ricas y expresivas utilizadas en los gráficos de comportamiento. Los nodos de comportamiento (tanto los realizados por defecto como los propios del usuario) pueden ser descritos en el nivel de desarrollo del nodo usando el lenguaje XMLSchema Behavior3DNode. Las instancias de documento basadas en esta gramática forman nuevas definiciones de nodos de comportamiento. Una vez estos nodos se han definido, las descripciones son utilizadas para generar automáticamente una gramática llamada Behavior3D que contiene el repertorio de definiciones de comportamiento disponibles. Esta gramática es empleada en el nivel de uso de nodos para construir gráficos de comportamiento.

VHML Virtual Human Markup Language (Marriott, Beard, Stallo & Huynh, 2001) permite definir el comportamiento de avatares de una manera sencilla y visual a través de etiquetado del texto que debe reproducir el personaje, con un lenguaje de marcado basado en XML y que está orientado a la personificación de avatares. Es capaz de englobar todos los aspectos de la interacción persona ordenador a través de personajes virtuales: animación facial, animación corporal, producción texto-habla y representación emocional. Permite definir un gestor de diálogo de interacción y controlar componentes hypermedia y multimedia. Es un estándar de especificación pública. Necesita un motor para interpretar cada etiqueta VHML y ejecutarlas sobre el avatar.

2.2.4. Lenguajes de marcado "híbridos"

Los lenguajes de especificación híbridos integran varios de los lenguajes de especificación vistos hasta ahora y que podrían denominarse básicos o dedicados. Entre los lenguajes de marcas híbridos

podemos citar los siguientes:

MIML Multimodal Interaction Markup Language (Araki & Tachibana, 2006) permite integrar y evaluar información de discurso, gesto y el contexto de una aplicación dada usando una aproximación de parseo o procesamiento sintáctico/semántico. Se basa en XML para su implementación aunque parte de una especificación basada en unos diagramas de transición de estados denominados temporal Augmented Transition Network (tATN) (Latoschik, 2002) que también son XML, e interacciona con agentes controlados por distintos lenguajes.

MPML y MPML-VR Multimodal Presentation Markup Language (Okazaki, Saeyor, Dohi & Ishizuka, 2005) es un lenguaje de marcas o lenguaje de especificación diseñado para presentaciones multimodales. MPML-VR (Ishizuka Laboratory, 2010) es una extensión de MPML que usa VRML 2.0 para poder presentar espacios tridimensionales a través de un agente antropomórfico o avatar de aspecto humano. Originalmente MPML utilizaba un agente para animar presentaciones de tipo POWERPOINT pero la necesidad de dar más realismo a las presentaciones hizo plantearse la posibilidad de presentar espacios tridimensionales en lugar de simples fotografías en dos dimensiones. Se basa en el control de los estados de una escena es decir, controlar las transiciones de los estados de una escena basándose en elementos ”evento” tales como un click de ratón o un comando de voz o ”speech input”.

2.2.5. Otros lenguajes de marcado

Además de los lenguajes de marcado vistos, existen otros que también buscan definir especificaciones para aplicaciones concretas o diseñados para un objetivo definido.

EMMA Extensible MultiModal Annotation markup language (Johnston, 2010) es un lenguaje para especificación multimodal diseñado explícitamente para la especificación y tratamiento de la multimodalidad. Concretamente es un lenguaje diseñado para la especificación de entrada a través de discurso, entrada a través de teclado o escritura caligráfica, entrada gestual y entrada gráfica. Fue desarrollado por el grupo de interacción multimodal del W3C (MMIWG) (W3C, 2010b) y permite transportar los modos de entrada a un gestor de interacciones. Existe un mecanismo de unificación o discriminación entre las distintas modalidades de entrada para por ejemplo resolver casos de entradas contradictorias, o selección entre valores de distintas modalidades en base a determinadas reglas o prioridades. Actualmente se está discutiendo por parte del MMIWG la forma de mapear la información suministrada por EMMA con la información mantenida por VoiceXML 2.1.

EmotionML El Emotion Markup Language (Burkhardt & Schröder, 2010) fue concebido como un lenguaje ”plug-in” apropiado para tres áreas diferentes: (1) anotación manual de material incluyendo emocionalidad, tal como anotación de videos, grabaciones de discurso, de caras, de textos, etc.; (2) reconocimiento automático de emociones desde sensores, incluyendo sensores psicológicos, grabaciones de discurso, expresiones faciales, etc., así como desde combinaciones multimodales de sensores; y (3) generación de respuestas de sistema relativas a la emoción, que pueden incluir razonamiento acerca de implicaciones emocionales de eventos, prosodia emocional en discurso sintético, expresiones faciales y gestos de agentes corpóreos o robots, la elección de música y colores de iluminación en una habitación, etc. Como cualquier formato estándar, el primer y principal objetivo de un EmotionML es doble: permitir a un componente tecnológico representar y procesar datos, y permitir interoperabilidad entre diferentes componentes tecnológicos procesando datos. Ejemplos concretos de tecnología existente que podrían aplicar EmotionML son: minería de opiniones/análisis de sentimientos en Web 2.0, para seguir automáticamente la actitud del cliente referente a un producto a lo largo de blogs; monitorización afectiva, tal como aplicaciones de asistencia a ancianos, detección de miedo para propósitos de vigilancia, o usar sensores adaptables para testear la satisfacción del cliente; diseño de personajes y control para juegos y mundos virtuales; robots sociales,

tales como robots guía atrayendo visitantes; síntesis de habla expresiva, generando discurso sintético con diferentes emociones, tales como felicidad o tristeza, amigable o arrepentido; reconocimiento de emociones (por ejemplo, para localizar clientes enfadados en sistemas de diálogo); apoyo para personas con discapacidades, tales como programas educacionales para gente con autismo.

SCXML State Chart XML o State Machine Notation for Control Abstraction (Barnett, Jim et al., 2010) es un lenguaje de marcado basado en XML que provee una máquina de estados genérica basada en un entorno de ejecución según los diagramas de estados de Harel (Harel, 1987) usados en UML (Object Management Group, 2012). Es capaz de describir complejas máquinas de estados. Por ejemplo, es posible describir notaciones tales como subestados, estados paralelos, sincronización, o concurrencia. Combina conceptos de CCXML (Baggia & Scott, 2010). La especificación define una máquina de estados y una sintaxis de gestión de eventos junto a un conjunto estandarizado de elementos de control de llamadas. Se recomienda su uso en el estándar de la arquitectura MMI (Barnett, Dahl, Kliche, Tumuluri, Yudkowsky, Bodell, Porter, Raggett, Raman & Wahbe, 2008).

SVG Scalable Vector Graphics (W3C, 2010a) es un lenguaje para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL), en XML. SVG se convirtió en una recomendación del W3C en septiembre de 2001, por lo que ya ha sido incluido de forma nativa en el navegador web del W3C Amaya (Quint & Carcone, 2010). La versión 1.5 de Mozilla Firefox soporta gráficos hechos con SVG y desde su versión 8, también el navegador Opera ha implementado SVG 1.1 Tiny en su núcleo. Otros navegadores web, necesitan un conector o plug-in, para lo que se puede utilizar el Visualizador SVG de Adobe (Adobe, 2010d). SVG rivaliza con Adobe Flash (Adobe, 2010b) pero a diferencia de éste último es un estándar abierto. El SVG permite tres tipos de objetos gráficos: formas gráficas vectoriales (p.e. caminos consistentes en rectas y curvas, y áreas limitadas por ellos); imágenes de mapa de bits/digitales y texto. Los objetos gráficos pueden ser agrupados, transformados y compuestos en objetos previamente renderizados y pueden recibir un estilo común. El texto puede estar en cualquier espacio de nombres XML admitido por la aplicación, lo que mejora la posibilidad de búsqueda y la accesibilidad de los gráficos SVG. El juego de características incluye las transformaciones anidadas, los clipping paths, las máscaras alfa, los filtros de efectos, las plantillas de objetos y la extensibilidad. El dibujado de los SVG puede ser dinámico e interactivo. Una tendencia actual es construir sitios web dinámicos que se comportan como aplicaciones de escritorio, usando AJAX.

SMIL Synchronized Multimedia Integration Language (lenguaje de integración multimedia sincronizada) (Michel, 2010a) es un estándar del World Wide Web Consortium (W3C) para presentaciones multimedia. El lenguaje SMIL permite integrar audio, video, imágenes, texto o cualquier otro contenido multimedia. La recomendación SMIL está a cargo del grupo de trabajo Synchronized Multimedia Activity (Actividad sobre sincronización multimedia, bajo el acrónimo SYMM (Michel, 2010b)) del World Wide Web Consortium. Los objetivos de esta recomendación son:

- Definir un lenguaje basado en XML que permita a los autores crear presentaciones multimedia. Mediante SMIL, un autor puede describir el comportamiento temporal de su presentación multimedia, asociar hiperenlaces a contenido multimedia y describir la disposición de la presentación en la pantalla.
- Facilitar la reutilización de la sintaxis y semántica de SMIL en otros lenguajes basados en XML, en particular aquellos que requieren representar sincronización y temporización.

Como estándar abierto del W3C, SMIL se puede usar libremente y consiste en un conjunto de etiquetas XML que describen fuentes de contenido como: imagen estática (JPEG, PNG, etc.), audio (MP3, WAV, etc.), vídeo (MPG, AVI, etc.), texto plano, flujos de texto (SUB, RT,

etc.) y animaciones (SVG, VML, etc.). Un reproductor apropiado puede leer e interpretar un fichero SMIL y reproducir las acciones que en él se describen. La utilidad más obvia de SMIL es la creación de presentaciones multimedia o transparencias. Sin embargo, no se descartan otras utilidades como subtitulado de películas, apoyo a disminuidos sensoriales, karaoke, noticiarios teletipo, etc. SMIL también se está adoptando como medio de armonizar los formatos de mensajería multimedia en telefonía móvil (MMS).

Tabla 2.2: Revisión de los lenguajes de marcado presentados

Lenguaje	Diálogo	Escena	Comportamiento	Proyectos / Características
VoiceXML	Si	-	-	Estándar de voz
SALT	Si	-	-	Obsoleto
SRGS	Si	-	-	Gramáticas
SSML	Si	-	-	W3C
X+V	Si	-	-	IBM, Opera
AIML	Si	-	-	A.L.I.C.E.
JSGF	Si	-	-	Java
CCXML	Si	-	-	W3C
VRML	-	3D	-	primer estándar VR
X3D	-	3D	-	evolución de VRML
3DML	-	3D	-	Flatland Rover
Behavior3D	-	-	Si	CONTIGRA
VHML	Si	2D	Si	InterFace / MetaFace / MentorSystem
MIML	Si	-	Si	tATN, gesto (híbrido)
MPML/MPML-VR	Si	2D/3D	-	Agentes / Telefonía móvil (híbrido)
EMMA	Si	-	-	W3C, gesto, escritura caligráfica
SCXML	-	-	Si	máquina de estados
EmotionML	-	-	Si	anotación manual de datos, emoción
SMIL	-	2D	-	W3C
SVG	-	2D	-	W3C
XMMVR	Si	2D/3D	Si	-

2.2.6. Resumen y conclusiones

La tabla 2.2 muestra un resumen de los lenguajes de marcado introducidos. Se puede ver que al igual que existen lenguajes de marcado híbridos que permiten especificar diálogo, escena o comportamiento de manera única, otros permiten especificar una combinación de más de una de estas modalidades e incluso otras como son el gesto y/o la escritura caligráfica o anotación manual de datos. Puesto que ya existen lenguajes de especificación básicos para definir parte de lo que se desea (X3D y VRML para escenas 3D y VoiceXML para diálogos), optar por definir un lenguaje de especificación ”híbrido” utilizando lenguajes de especificación ”básicos” facilitará mucho los cosas y permitirá reutilizar módulos ya definidos. Por ello, una condición del lenguaje buscado es que sea ”híbrido”.

2.3 Metáforas de interacción

Las metáforas de interacción permiten definir relaciones entre el usuario y el sistema comparando casos concretos de interacción persona ordenador con situaciones de la vida real (Szabó,

1995). Metáforas de interacción populares son la metáfora de escritorio (desktop) (Erickson, 1995) o la metáfora de sitio web (website) (CERN, 2013). Puesto que es de interés definir interacción gráfica e interacción vocal, se presentarán las diferentes metáforas definidas para cada una de estas modalidades así como las posibilidades de cooperación entre ambas.

2.3.1. Interacción gráfica

La interacción gráfica basada en teclado y ratón como se conoce hasta ahora, siempre gira en torno al modelo de eventos y en base a espacios de acción. Claros ejemplos de la interacción 2D basada en eventos son las X Windows (Young, 1994) donde cada widget 2D tiene un comportamiento predefinido en base a eventos provocados por clicks de ratón o por pulsar determinadas teclas. Al igual que X Windows se basa en una metáfora de escritorio, en el proyecto CONTIGRA (Dachselt, Hinz & Meiner, 2002) se definen los espacios de acción o action spaces (Dachselt, 2000) que son aproximaciones metafóricas para estructurar los interfaces de usuario tridimensionales. Esto no es más que asemejar los elementos con los que va a interactuar el usuario en el espacio virtual 3D con elementos reales, de manera que el usuario asumirá de una manera natural la interacción con el mundo virtual 3D. Se definen dos metáforas fundamentales, cinco metáforas estructurales y cinco metáforas navegacionales.

Las metáforas fundamentales son básicamente dos:

- **Metáfora de teatro (mt)**: El usuario tiene una posición estática y un punto de vista donde el mundo cambia a su alrededor.
- **Metáfora de locomoción (ml)**: El usuario tiene una posición dinámica y será trasladado a través de una estructura.

Las metáforas estructurales se basan en localizaciones físicas:

- **Habitaciones (mt+ml)**: Pueden considerarse como partes de ambas metáforas fundamentales y se aplican en áreas como entrenamiento virtual, educación o comercio electrónico.
- **Escenarios giratorios (mt)**: Son ejemplos característicos de la metáfora de teatro, ya que el punto de vista del usuario se mantiene constante.
- **Edificios con plantas, pasillos, niveles (ml)**: Son ejemplos típicos de la metáfora de locomoción.
- **Estaciones espaciales, moléculas, redes de burbujas (ml)**: Pertenecen a la metáfora de locomoción. Permiten más "libertad geométrica" que las anteriores. Esferas, habitaciones, burbujas o células tridimensionales son colocadas de manera no lineal creando estructuras moleculares, estaciones espaciales o redes conectadas por tubos por ejemplo.
- **Metáforas urbanas (ml)**: También son ejemplos de la metáfora de locomoción.

Las metáforas navegacionales se basan en diferentes modos de desplazamiento. Las más conocidas son:

- **Elevador / rampa hidráulica**: Tal y como sus nombres referencian, debido al movimiento lineal, soportan tareas secuenciales donde se produce un desplazamiento mayoritariamente vertical.
- **Vehículo en vía / tren**: Similar a la anterior pero el desplazamiento sería horizontal. Son posibles dos direcciones con esta metáfora secuencial en la mayoría de los casos: hacia la izquierda y/o hacia la derecha.
- **Deslizamiento / cable ferrocarril**: Es una metáfora secuencial de un único sentido evolucionada de la anterior, donde sólo se permite una secuencia estricta de action spaces visitados.
- **Silla voladora / alfombra**: Pertenece al grupo de metáforas paralelas ya que el usuario puede volar a más de uno o dos action spaces.

- **Tele-Transporte / "Beam me up"**: A pesar de la desorientación, es una metáfora apropiada para estructuras completamente libres y permite la elección de varios espacios destino.

Como ya se ha indicado y se intuye de la descripción de estas dos metáforas fundamentales, cinco metáforas estructurales y cinco metáforas navegacionales, estas tres clasificaciones no son excluyentes y esto podemos verlo en la tabla 2.3 donde el número de signos "+" denota mejor integración de pares de metáforas mientras que pares de metáforas con el signo "-" están peor o menos integradas.

Tabla 2.3: Combinación de metáforas estructurales y navegacionales (Dachselt, 2000)

	Elevator	Rail Vehicle	Slide	Flying Chair	Tele-Portage
Rooms (mt, ml)	++	+	+	+	+
Buildings (ml)	+++	++	+	-	+
Space Station (ml)	++	++	-	+	++
Molecules (ml)	+	-	+++	+	+
Urban Metaphors (ml)	-	+++	++	+++	++

Además de las metáforas presentadas, el transporte de objetos se utiliza como metáfora de interacción en algunos mundos virtuales donde algunas herramientas o elementos de interfaz pueden ser utilizadas en varios action spaces o incluso deberían estar disponibles en cualquier lugar. Usando la metáfora de locomoción, estos elementos deberían ser llevados a través de habitaciones de manera miniaturizada (como iconos 3D). Usando la metáfora de teatro, deberían ser parte de la estructura visual estática (por ejemplo, un portal). Podríamos tomar como ejemplo un escenario giratorio, donde una "pared de herramientas" puede ser accedida por ambos (o incluso más) lados. Hasta ahora la especificación de la interacción gráfica se hacía dentro de la especificación de los espacios 3D aunque también se han definido especificaciones del comportamiento de los objetos de los mundos 3D ante diferentes eventos de interacción pero la insuficiencia de éstas ha propiciado la creación de lenguajes para especificación de comportamiento.

Para ilustrar esta integración supongamos que nuestro mundo representa un edificio donde nos movemos de un piso a otro usando un ascensor, esto se ve y se comprende claramente ya que es algo basado en la realidad, por lo que decimos que existe una buena integración entre la metáfora estructural de "edificio" y la metáfora navegacional de "ascensor" (+++) según se muestra en la tabla. Sin embargo, si queremos representar una ciudad en la que nos desplazamos de un lugar a otro en un ascensor, estaremos basándonos en una fantasía más propia de Tim Burton (IMDB, 2005) que en la realidad, por lo que diremos que la integración entre la metáfora estructural de "ciudad" y la metáfora navegacional de "ascensor" es la peor, como se nos deja claro en la tabla (-).

2.3.2. Interacción vocal

En un sistema de diálogo siempre existe una interacción o diálogo entre el usuario y el sistema que se transporta a través de un canal. Este diálogo tiene una aplicación y se produce en un escenario determinado. Puesto que el diálogo usuario-sistema es bidireccional, se ha de establecer una política de turnos o intervenciones en la que ha de existir una iniciativa por parte del usuario, por parte del sistema o mixta es decir, que tanto uno como otro pueden iniciar el diálogo. A través de este diálogo básicamente se intercambia información por lo que se tendrá que definir cómo se representa esa información (de manera textual, visual, hablada, mecánica...) y se tendrá que definir un protocolo de intercambio de ésta es decir, su orden y codificación. Para especificar todo esto existen teorías basadas en psicología evolutiva (modelo de contrario, "dónde", "con quién")

y basadas en los "actos del habla" (plano conceptual, "cómo", "para qué"). Asimismo, se tendrá que tener claro el dominio del problema en el que habrá que definir acciones, entidades, mensajes e interrelación entre todos. Para el caso que se trata (interacción vocal con mundos virtuales) las acciones las solicitará el usuario para que se ejecuten por o contra los elementos definidos en el mundo virtual. Por lo que los mensajes serán las peticiones y las respuestas entre el usuario y el mundo virtual. Las interrelaciones se podrán dar entre el usuario y los elementos del mundo virtual además de entre los propios elementos del mundo virtual. Esta interacción vocal seguirá un modelo de flujo o guión que será la base para definir nuestra propuesta de lenguaje de especificación. Para ello se definirán varios niveles:

- Nivel común: donde se genera o desencadena una acción.
- Nivel lógico: donde la acción interactúa con las entidades del dominio.
- Nivel físico: donde se producen los cambios de estados provocados por la acción y la interacción de ésta con las entidades del dominio.

Es necesario indicar además, que los interfaces de discurso comparten dos características (Oviatt & Cohen, 2000):

- **Restricted Language o lenguaje restringido**: El sistema no es capaz de entender lenguaje natural general, sólo lenguaje apropiado para una tarea particular. Los usuarios han de utilizar un vocabulario y gramática limitados si quieren tener una interacción exitosa con el sistema.
- **Feedback o realimentación**: El sistema informa al usuario de sus propios estados y metas para hacer el comportamiento del sistema lo más transparente posible. El sistema puede confirmar que la información se ha comprendido o dar una explicación de porqué no se ha comprendido la intervención del usuario, por ejemplo porque habló muy despacio.

Son posibles cuatro metáforas de interacción vocal (Mcglashan & Axling, 1996a):

- **Proxy o Delegado**: El usuario puede tomar control de varios agentes (cambiar de agente, selección en la acción) en el mundo virtual e interactuar con el mundo virtual a través de ellos, por ejemplo: pintor, ¡pinta la casa de rojo! Provee una estrecha correlación entre las capacidades funcionales y comunicativas de los agentes. Por ejemplo: varios cursores como pintor, transportista, etc. Varias instancias del mismo cursor. El avatar es un tipo de delegación a través del cual se dan acciones secuencializadas ya que hasta que no acabe o le interrumpan, no podrá hacer más acciones. Uno o varios agentes pueden ser iguales aunque con ventanas 2D esto no ocurre.
- **Divinity o Divinidad**: El usuario actúa como un dios y controla el mundo directamente, por ejemplo: que la casa sea roja! (sin foco). Sería la metáfora vocal más sencilla.
- **Telekinesis**: Los objetos y agentes en el mundo virtual pueden ser interlocutores de diálogo del usuario, por ejemplo: casa, ¡píntate de rojo!.
- **Interface Agent o Agente Interfaz**: El usuario se comunica con un agente, separado del mundo virtual, que ejecuta sus comandos. Puede ser problemático porque no hay indicación al usuario de qué comandos pueden ser entendidos y el reconocimiento de discurso tiene que ser capaz de procesar cualquier comando. Un ejemplo es la forma de enviar comandos de voz al navegador Opera donde es preciso comenzar todos los comandos con ”Opera” (Opera, 2010).

2.3.3. Cooperación entre modalidades

Existen cinco tipos básicos de cooperación entre modalidades según (Martin, 1998):

- **Transferencia**: Parte de la información producida por una modalidad es usada por otra modalidad. Habilita el paralelismo de procesamiento en dos modalidades y de ahí una más rápida interacción persona ordenador. Así, puede intervenir por diferentes razones entre dos modalidades de entrada, entre dos modalidades de salida o entre una modalidad de entrada y una modalidad de salida. Usada por ejemplo en interfaces hypermedia cuando un click de ratón provoca la muestra de una imagen, puede orientarse a varios objetivos:
  - Traslación: En aplicaciones de recuperación de información, el usuario puede expresar una solicitud en una modalidad (discurso) y obtener una información relevante en otra modalidad (video). La información de salida puede no ser sólo recuperada si no también producida desde cero, así diversos sistemas generan una descripción gráfica de la escena desde una descripción lingüística o una descripción multimodal.
  - Mejora de reconocimiento: La detección de clicks de ratón puede ser transferida a la modalidad vocal con el objetivo de facilitar el reconocimiento de palabras predecibles (aquí, eso...).
  - Habilitar una interacción más rápida: Cuando parte de la frase pronunciada no ha sido reconocida, puede ser editada con el teclado. Así el usuario no tiene que teclear o pronunciar la frase completa de nuevo.
  - Salida coordinada de lenguaje natural y gráficos: Las dos modalidades trabajan concurrentemente para producir una salida basada en instrucciones recibidas de un gestor multimodal. Si es necesario, cuando una de las modalidades no puede producir una información dada, la información on-line puede ser transferida de esta modalidad a la otra. Como ejemplo, la modalidad gráfica puede ser instada por el gestor a producir una etiqueta textual y puede resultar que no sea posible porque esconde parte de los gráficos. Esta información es enviada a la modalidad de lenguaje natural que es capaz de adaptar dinámicamente su salida para insertar la nueva información textual.
- **Equivalencia**: Ambos modos podrían tratar la misma información. Es decir, una información puede ser procesada por ambas modalidades alternativamente. Así, equivalencia significa alternativa y está claro que tienen que considerarse diferencias entre cada modalidad, cognitivas o técnicas. Tres objetivos buscados:
  - Mejora el reconocimiento de comandos: Así, el usuario de un editor gráfico puede especificar un comando a través del discurso o seleccionando un botón con un bolígrafo. En este caso, la equivalencia permite al usuario seleccionar un comando con un bolígrafo cuando el reconocedor de habla no está trabajando de manera precisa a causa de ruido.
  - Permite adaptación al usuario por personalización: al usuario se le permite seleccionar las modalidades preferidas. De esta forma, la formación de modelos mentales precisos de un sistema multimodal parece dependiente de la implementación de las opciones sobre las que el usuario tiene control.
  - Permite una interacción más rápida: permite al sistema o al usuario seleccionar la modalidad más rápida.
- **Especialización**: Un determinado tipo de información es siempre procesada por la misma modalidad. No es siempre tan absoluta y puede ser definida más precisamente: debería distinguirse especialización relativa a datos, especialización relativa a modalidad y especialización absoluta. Por ejemplo, en varios sistemas existentes, los sonidos están de algún modo especializados en notificación de errores (comandos prohibidos son señalados con un beep). Es una especialización relativa a la modalidad si los sonidos no son usados para transmitir cualquier otro tipo de conocimiento. Es una especialización relativa a datos si los errores sólo producen sonidos y no gráficos ni texto. Cuando hay una relación uno a uno entre un conjunto de información y una modalidad gestionando este conjunto, hablaremos de especialización absoluta.

Esta especialización puede ayudar al usuario a interpretar los eventos producidos por el ordenador (uniéndolos al conocimiento global contextual). Esto significa que la elección de una modalidad dada añade información semántica y de ahí ayuda al proceso de interpretación.

La especialización puede también mejorar el reconocimiento. En el ejemplo de un sistema de información turística, el usuario puede siempre proveer el nombre de ciudades usando el teclado. Esta especialización permite un procesamiento más fácil (y de ahí un mejor reconocimiento) en otras modalidades. Mejora la precisión del reconocedor de habla ya que el espacio de búsqueda es menor. Esto también permite una interacción más rápida ya que decrementa la duración de la integración y el proceso de selección de modalidad.

Cuando una modalidad es especializada, debería respetar la especificidad de esta modalidad incluyendo la información que es buena para ser representada. Por ejemplo, en referencia a la interpretación, el gesto de designación apunta a seleccionar un área específica y el canal verbal provee un marco para la interpretación de la referencia: información categórica, restricciones del número de objetos seleccionados.

La especialización intuitiva de una modalidad puede ir en contra de sus especificaciones técnicas. En un experimento de mago de Oz (Siroux, Guyomard, Multon & Rémondeau, 1997) tratando con una aplicación turística, a pesar de la baja tasa de reconocimiento de los nombres de ciudades, los usuarios no utilizaron la pantalla táctil para seleccionar una ciudad pero usaron la voz en su lugar.

- **Redundancia**: La misma información es procesada por ambas modalidades. Por ejemplo, si el usuario teclea ”quit” en el teclado y pronuncia ”quit”, esta redundancia puede ser usada por el sistema para evitar un diálogo de confirmación y además favorece una más rápida interacción.

  En referencia a la intuición, la redundancia ha sido observada en un estudio basado en mago de Oz (Siroux, Guyomard, Multon & Rémondeau, 1997) en el que a veces el usuario seleccionó una ciudad por voz y tocando en la pantalla táctil.

  En referencia al aprendizaje del uso de interfaces, se ha observado que una salida multimodal redundante que contemple visualización de un texto y restitución vocal del mismo texto permitió un más rápido conocimiento del interfaz gráfico. Un asunto importante aquí es conocer si el canal visual debería llevar exactamente el mismo mensaje que el canal auditivo (reforzamiento palabra por palabra) o uno más corto (reforzamiento por activación). El tipo de reforzamiento elegido por el sistema y la información a ser transmitida parece tener consecuencias de compatibilidad cognitiva de respuestas habladas o manuales del usuario. La redundancia entre texto visual y vocal con reforzamiento palabra por palabra se testeó con descripciones lingüísticas de los objetos que el usuario manipulaba y la acción que realizaba. A pesar de que el discurso coaccionaba a los sujetos a leer las descripciones tecleadas, los sujetos hicieron más errores y fueron más lentos que con la sola salida visual del texto.

- **Complementariedad**: Diferentes partes de información son procesadas por cada modalidad pero tienen que ser combinadas. Permite una interacción más rápida ya que las dos modalidades pueden ser usadas simultáneamente y transportan mensajes más cortos que son así mejor reconocidos que los mensajes largos.

  Puede mejorar la interpretación, así una salida gráfica es suficiente para un experto pero debe ser completada con una salida textual para usuarios noveles. En este tipo de colaboración es importante el criterio usado para unir datos de diferentes modalidades, las aproximaciones más clásicas se basan en que sean coincidentes en el tiempo, secuenciales temporalmente o unidos espacialmente.

  En referencia a la intuición, se han mostrado dos comportamientos: el secuencial que es más extraño y el sinergético. En el comportamiento secuencial, el usuario podría por ejemplo pronunciar ”¿cuáles son los campamentos en...?” y después seleccionar una ciudad con la

pantalla táctil. En el comportamiento sinergético, el usuario podría pronunciar "¿Hay algún campamento aquí?" y seleccionar una ciudad con la pantalla táctil mientras pronuncia "aquí".

En referencia a la salida del ordenador, se ha observado que la unión espacial de información relativa aumenta la capacidad del usuario de enlaces causales y cognitivos. Así, cuando había que recuperar informaciones complementarias de diferentes medios, el comportamiento del usuario tendió hacia la búsqueda secuencial evitando uso sinergético de varias modalidades (Hare & Ryan, 1995).

Modalidades cooperando por complementariedad pueden especializarse en diferentes tipos de información. En el ejemplo de un editor gráfico, el nombre de un objeto puede ser siempre especificado con discurso mientras su posición es especificada con el ratón.

Pero modalidades cooperando por complementariedad pueden ser también equivalentes para diferentes tipos de información. Por ejemplo, el usuario puede seleccionar un objeto con el ratón y su posición vocalmente ("en la esquina superior derecha").

De cualquier forma, el uso complementario de modalidades especializadas aporta las ventajas de la especialización: el reconocimiento del habla es mejorado ya que el vocabulario y la sintaxis es más simple que una descripción lingüística completa como "pon el cuadro rojo que está a la izquierda sobre el rectángulo verde". Como un ejemplo, el uso de la complementariedad natural del audio vocal y las imágenes de los movimientos de los labios mejora el reconocimiento vocal.

Estos tipos de cooperación (excluyendo la transferencia) pueden ser comparados a través de dos dimensiones, fusión e información transmitida:

- Equivalencia y especialización excluyen fusión,
- Redundancia y complementariedad requieren fusión,
- Equivalencia y redundancia requieren transmisión de la misma información,
- Especialización y complementariedad requieren transmisión de diferente información.

2.4 Arquitecturas para multimodalidad

Varias han sido las arquitecturas software definidas para soportar sistemas multimodales. Pero existe un estándar que ha sentado las bases de toda arquitectura software que soporte aplicaciones multimodales. A continuación, tras revisar los objetivos de algunas arquitecturas que permitieron desarrollar aplicaciones con interfaces multimodales, presentaremos en detalle la arquitectura MMI. Cabe resaltar los diferentes objetivos de cada una de estas arquitecturas a la hora de definir una aplicación multimodal.

2.4.1. Arquitectura ICARE

La arquitectura ICARE (Bouchet, 2004) se utilizó para implementar entre otras las aplicaciones MEMO, MID y FACET revisadas en 2.1. ICARE significa Interaction-CARE (Complementarity Assignment Redundancy Equivalence). Es una aproximación que se basa en dos tipos de componentes software: (1) componentes elementales que incluyen componentes de dispositivo y componentes de lenguaje de interacción que permiten desarrollar modalidades y (2) componentes de composición que definen usos combinados de modalidades. Reutilizar y ensamblar los componentes permite el rápido desarrollo de interfaces multimodales. Se eligió la tecnología JavaBeans para desarrollar los componentes software de ICARE por ser multiplataforma y proveer muchos mecanismos para manipulación gráfica que se pueden reutilizar. Las propiedades de los componente ICARE son atributos de clase que se pueden acceder o modificar. La comunicación entre componentes software de ICARE se basa en el modelo de eventos Java, los eventos son mensajes que se envían los componentes y

que transportan datos a ser procesados ya que cada componente implementa una cola de eventos que procesa un hilo. El último componente de la cadena de componentes ICARE debe comunicarse con el controlador de diálogo.

2.4.2. Arquitectura SAIBA

SAIBA (Kipp, Heloir, Schröder & Gebhard, 2010) es una arquitectura en la que se basaban las aplicaciones RealActor (Cerekovic, Pejsa & Pandzic, 2009) y Elckerlyc (van Welbergen, Reidsma, Ruttkay & Zwiers, 2010) introducidas en 2.1. El framework SAIBA sugiere tres módulos principales para el proceso de producción de comportamiento: (1) planificación de objetivo para determinar el contenido general del mensaje en términos de objetivos comunicativos abstractos; (2) planificación de comportamiento para decidir la elección de palabras, gestos, expresiones faciales, etc. en términos de modalidades pero no partes corporales; y (3) realización para representar estos comportamientos con el cuerpo y la voz de un agente concreto. El procesamiento comienza con un documento BML que describe el texto hablado, gestos, movimientos de cabeza, etc. y en base a éste se genera discurso y animación del personaje. Comportamientos reactivos como seguimiento de gestos o ajustes continuos a la emoción actual (cara, postura) se envían directamente al módulo de presentación enviando planificación de objetivo y de comportamiento. Los componentes del sistema están implementados en lo alto del API SEMAINE, un framework opensource para la integración de componentes distribuidos multiplataforma para sistemas interactivos en tiempo real. Los componentes se comunican a través de formatos de representación estándar, incluyendo BML, BMLS y EmotionML. Para la animación de caracteres 3D en tiempo real utiliza el software libre EMBR (Embodied Agents Realizer) que permite control detallado de la animación a través de un lenguaje de especificación basado en poses llamado EMBRScript. EMBR permite gesto, expresiones faciales, cambio de poses, rubor, control de mirada y comportamientos autónomos como respiración y pestañeo. Para la síntesis de discurso se utilizan los componentes de generación de discurso del sistema SEMAINE. Para demostrar el modelado de comportamiento reactivo, se implementó un emparejamiento directo de estados emocionales con comportamiento como expresiones faciales, orientación de la cabeza, respiración y rubor. Se utiliza un componente para simulación de afectividad que continuamente produce EmotionML para expresar la emoción actual y estado del agente. Esto está asociado a comportamientos que son enviados directamente al componente de presentación como documentos EMBRScript. Por ejemplo, el estado de ánimo afecta al rubor y a la respiración con su componente de excitación. Se definieron mapeos para eventos de emoción (alegría, ira, gusto...) para disparar comportamientos (expresiones faciales) y cambios psicológicos (respirar, rubor) basados en la literatura. La intensidad de la emoción determina la duración y la intensidad visible (por ejemplo nivel de rubor, extensión de sonrisa, nivel de elevación de las cejas) de un estado emocional.

2.4.3. Arquitectura OpenInterface

Con la arquitectura OpenInterface (Serrano, Nigay, Lawson, Ramsay, Murray-Smith & Denef, 2008) se han desarrollado proyectos tales como el Multimodal Map Navigation o "Funny Rabbit" presentados en 2.1. El OpenInterface framework (OI) permite desarrollar interfaces multimodales y se compone del OI Kernel y del OpenInterface Interaction Development Environment (OIDE). El primero es una plataforma de ejecución basada en componentes y el segundo es una herramienta gráfica construida sobre la plataforma de ejecución. OI Kernel gestiona la creación y la conexión entre componentes cargando e interpretando dinámicamente los descriptores de aplicación que pueden ser creados gráficamente con el OIDE. Éste entorno gráfico permite manipulación y ensamblado de componentes para definir una interacción multimodal. Existe un repositorio de componentes que incluye drivers de dispositivos, técnicas de interacción, facilidades de fusión multimodal, servicios de desarrollo y combinaciones definidas por el usuario. Es decir, la estrategia de los creadores de esta arquitectura es poder crear nuevos interfaces multimodales en base a los componentes almacenados

en su repositorio. El kernel OI y el OIDE permiten el ensamblado dinámico de componentes: los componentes pueden ser lanzados individualmente. Cada componente tiene un botón que permite a un diseñador iniciar el componente. Los componentes ya disponibles incluyen drivers de dispositivos, técnicas de interacción, facilidades de fusión multimodal, servicios de desarrollo y combinaciones definidas por el desarrollador. OIDE soporta descripciones, consultas y acceso en un conjunto extensible de esquemas de descripción. El modelo define un conjunto de componentes genéricos a medida. Los componentes genéricos definen operaciones genéricas reutilizables. Los componentes a medida son componentes implementados en modo ad-hoc para una técnica de interacción específica o aplicación interactiva. Esta aproximación mixta adoptada demuestra la ventaja expresiva del framework OI. El modelo incluye tres nivel principales para componentes: dispositivos, cadenas de transformación y cadenas. Tareas y componentes del dispositivo pueden ser genéricos o a medida. Por ejemplo componentes de una tarea genéricos están basados en las tareas interactivas de Foley donde los componentes genéricos de dispositivo se basan en taxonomías de dispositivo. El nivel de cadena de transformación define dos tipos de componentes: componentes de transformación y componentes de composición.

2.4.4. MINT Framework

La arquitectura conocida como MINT Framework (Feuerstack, 2013) implementa una plataforma que permite diseñar y ejecutar aplicaciones web multimodales. Estas aplicaciones permiten al usuario alternar dispositivos y modalidades e incluso combinarlos bajo demanda. La idea básica es ofrecer diseño basado en modelos de interactuadores personalizados que son autoejecutables y que pueden ser distribuidos y sincronizados a través de varios dispositivos y modalidades. A diferencia del diseño de interfaces de usuario basado en modelos que genera interfaces aislados para diferentes plataformas por un modelado de interacción estructurado de abstracto a concreto, diseña interactuadores que, una vez han sido diseñados, son ensamblados por constructores de interfaces de usuario para formar interfaces multimodales. Por tanto, adapta la clásica construcción de interfaces de usuario en lugar de requerir al desarrollador aprender nuevos lenguajes y procesos para crear interfaces.

2.4.5. El estándar MMI

La arquitectura MMI (Barnett, Dahl, Kliche, Tumuluri, Yudkowsky, Bodell, Porter, Raggett, Raman & Wahbe, 2008) es una propuesta del grupo MMI Working Group (W3C, 2010b) del consorcio W3C (W3C, 2011) cuyo objetivo es establecer los requisitos que ha de cumplir toda arquitectura que soporte aplicaciones multimodales. Para ello han establecido un marco o framework compuesto por los elementos que se muestran en la figura 2.1 o constituents y que describimos a continuación:

Interaction Manager Coordina las diferentes modalidades, sería el controlador del paradigma MVC (Reenskaug, 1979). Todos los eventos del ciclo de vida que son generados por los Modality Components deben ser enviados al Interaction Manager. A semejanza de una muñeca rusa, los Modality Components pueden contener sus propios Interaction Managers para manejar sus eventos internos. Sin embargo, estos Interaction Managers no son visibles a alto nivel para el Runtime Framework o el Interaction Manager. Si el Interaction Manager no contiene un controlador explícito para un evento, debe respetar cualquier comportamiento por defecto que se establezca para el evento. Si no hay comportamiento por defecto, el Interaction Manager debe ignorar el evento. Así, el controlador por defecto del Interaction Manager para todos los eventos es ignorarlos.

Normalmente existirán etiquetas específicas asociadas con el Interaction Manager indicando cómo responder a eventos. Estas etiquetas contendrán además gran parte de la lógica de interacción de una aplicación. Lenguajes de marcado existentes ya revisados tales como SMIL,

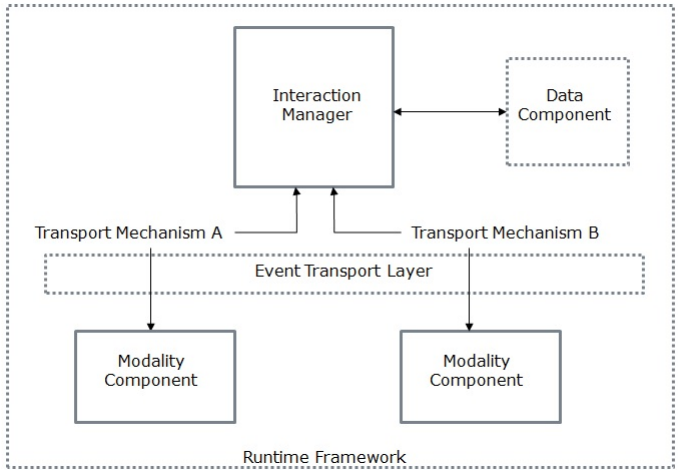

Figura 2.1: Run-Time Architecture Diagram

CCXML, SCXML o ECMAScript pueden ser utilizados para etiquetado del Interaction Manager como una alternativa a definir lenguajes de propósito especial dirigidos específicamente a aplicaciones multimodales. El Interaction Manager cumple múltiples funciones. Por ejemplo, es responsable de la sincronización de datos y foco, etc. entre distintos Modality Components así como el flujo de alto nivel de aplicación que es independiente de los Modality Components. También mantiene el modelo de datos a alto nivel de la aplicación y puede manejar la comunicación con entidades externas y sistemas back-end. Lógicamente estas funciones podrían separarse en constituents o componentes diferenciados y sus implementaciones podrían introducir una estructura interna al Interaction Manager. Sin embargo, para los propósitos del estándar, se dejan las funciones incrustadas en un sólo componente Interaction Manager monolítico. Lenguajes de máquinas de estados como SCXML (Barnett, Jim et al., 2010) son una buena elección para la autoría de tal componente multifunción ya que a través de dichos lenguajes se pueden especificar las máquinas de estados. De esta forma es posible definir una máquina de estados a alto nivel representando el flujo de la aplicación general. En un nivel inferior, se pueden usar máquinas de estados anidadas que manejan la sincronización de la modalidad cruzada en cada fase del flujo de alto nivel.

Data Component Es el proveedor del modelo de datos común, representa el modelo en el paradigma MVC (Reenskaug, 1979) y es responsable de almacenar datos a nivel de aplicación. El Interaction Manager es un cliente del Data Component y es capaz de accederlo y actualizarlo como parte de su lógica de control de flujo, pero los Modality Components no tienen acceso directo a él. Puesto que los Modality Components son cajas negras, pueden tener sus propios Data Components internos y pueden interactuar directamente con servidores back-end. Sin embargo, el único modo en el que los Modality Components pueden compartir datos entre ellos y mantener la consistencia es vía el Interaction Manager. Es por tanto una buena práctica de diseño de aplicación dividir los datos en dos clases lógicas: datos privados que son sólo de interés para un modality component dado y datos públicos que son de interés para el Interaction Manager o para más de un Modality Component. Los datos privados pueden ser gestionados como el Modality Component considere, pero toda modificación de datos públicos, incluyendo el envío a servidores back-end, debería confiarse al Interaction Manager.

Esta especificación no define un interfaz entre el Data Component y el Interaction Manager. Esto lleva a tratar el Data Component como parte del Interaction Manager. (Esto significa que el lenguaje de acceso a datos será cualquiera que el Interaction Manager provea.) El Data

Component es mostrado con una línea punteada en el diagrama de la figura 2.1 porque es distinto lógicamente y podría situarse en un componente separado.

Modality Components Los Modality Components como su nombre indica son responsables de controlar las diferentes entradas y salidas de las modalidades en el dispositivo. Son por tanto responsables de manejar toda la interacción con el o los usuarios. Se comunican con el Interaction Manager vía eventos asíncronos. Los constituents (Interaction Manager y Modality Components) deben ser capaces de enviar eventos y manejar eventos que se les envían asíncronamente. No se requiere que los constituents utilicen estos eventos internamente ya que la implementación de un constituent dado es una caja negra para el resto del sistema. En general, se espera que los constituents envíen eventos automáticamente y controlados por el etiquetado. La mayoría de los eventos definidos aquí vienen en pares petición-respuesta. Esto es que por un lado (el Interaction Manager o un Modality Component) envía una petición y el otro devuelve una respuesta. Las excepciones son los eventos ExtensionNotification, StatusRequest y StatusResponse que pueden ser enviados por cualquier parte. En cada caso se especifica qué parte envía la petición y qué parte devuelve la respuesta. Si la parte equivocada envía una petición o respuesta o si la petición o respuesta es enviada en condiciones erróneas (por ejemplo, respuesta sin una petición previa) el comportamiento de la parte que lo recibe es indefinido. La parte emisora puede enviar la petición porque depende de su lógica interna decidir si quiere invocar el comportamiento que la petición dispararía. La parte receptora debe enviar la respuesta porque es obligatorio enviar la respuesta si se recibe una petición.

Cualquier definición de las responsabilidades de los Modality Components estará por encima de su dominio y será específica de la aplicación. No se define en particular un conjunto de modalidades estándar o eventos que deberían generar o manejar. Los proveedores de plataformas tienen permitido definir nuevos Modality Components y tienen permitido situar en un único component funcionalidad que podría lógicamente parecer pertenecer a dos o más modalidades diferentes. Así una plataforma podría proveer un Modality Component de escritura manual y discurso que podría aceptar voz y escritura simultáneamente. Tales components combinados permiten un emparejamiento mucho más unido entre las dos modalidades que los interfaces separados definidos anteriormente. Además, los Modality Components pueden ser utilizados para realizar funciones de procesado general no asociadas directamente con ningún interfaz de modalidad específico, por ejemplo, control de flujo de diálogo o procesado de lenguaje natural.

En la mayoría de los casos, existirán etiquetas específicas en la aplicación correspondiente a una modalidad dada, especificando cómo la interacción con el usuario debería ser llevada a cabo. Sin embargo, no se requiere esto y específicamente se permite un Modality Component sin marcado cuyo comportamiento esté codificado en su software.

Runtime Framework Provee la estructura básica y controla la comunicación entre los demás elementos o constituents. Es un término para cubrir toda la infraestructura de servicios necesarios para la ejecución exitosa de una aplicación multimodal. Esto incluye inicializar componentes, manejar comunicación, registro, etc. En la versión actual de la especificación se dejan estas funciones pendientes de ser definidas en un modo específico de la plataforma pero se define específicamente una capa de transporte de eventos o Event Transport Layer que maneja comunicaciones entre los componentes y es responsable de repartir eventos entre el Interaction Manager y los Modality Components. Existen múltiples mecanismos de transporte (protocolos) que pueden utilizarse para implementarla y se pueden usar diferentes mecanismos para comunicar con distintos Modality Components con el Interaction Manager. Se establecen los siguientes requisitos en todos los mecanismos de transporte:

- Los eventos deben ser repartidos de manera fiable. En particular, el mecanismo de reparto de eventos debe reportar un error si un evento no puede ser enviado, por ejemplo si el punto de destino no está disponible.

- Los eventos deben ser enviados al destino en el orden en que la fuente los generó. No existe garantía en el orden de los eventos generados por distintas fuentes. Por ejemplo, si el Modality Component M1 genera los eventos E1 y E2 en ese orden, mientras el Modality Component M2 genera E3 y después E4, se requiere que E1 sea enviado al Runtime Framework antes que E2 y que E3 sea enviado antes que E4, pero no hay garantía en el orden de E1 o E2 contra E3 o E4.

No se especifican actualmente mecanismos de reparto o salvaguardas de seguridad interna a utilizar por los Modality Components y el Interaction Manager. Sin embargo, cualquier sistema de seguridad tendrá que cumplir al menos los requisitos siguientes: Autenticación, integridad, autorización, privacidad y no repudio. No se especifican requisitos de seguridad interna de un Modality Component o del Runtime Framework. Tampoco se especifica cómo se establecen las conexiones con los medios ya que el foco principal de la especificación es el flujo de los datos de control. La distribución de medios no fluye típicamente a través del Interaction Manager sin embargo, todos los datos de control lógicamente enviados entre modality components deben fluir a través del Interaction Manager.

Estas propuestas han de servir como base de los requisitos de la arquitectura del sistema multimodal a definir. Todas las demás propuestas son pequeños prototipos de laboratorio que se introducen en el apartado 2.1.

2.5 Sistemas de diálogo

Los sistemas de diálogo son programas informáticos que reciben como entrada frases en lenguaje natural y generan como salida frases orales también en respuesta a la entrada generando así un diálogo (López Cózar, 2011). El propósito de la mayoría de las aplicaciones diseñadas hasta ahora siguiendo esta tecnología se reduce a la consulta o realización de determinadas acciones en un contexto muy reducido como consulta de horarios de tren, avión o autobús, guía de restaurantes de una ciudad, etc. Los sistemas de diálogo intentan simular el diálogo humano a base de la comprensión de la entrada y la generación de la respuesta (GRAH, 2010).

Una aproximación que no se considera un sistema de diálogo son los Chatterbots (Paillard, 2011) que simulan el diálogo pero sólo en apariencia, comparando las entradas mediante reglas y escogiendo la salida de una lista de frases predefinidas. (Ejemplos de Chatterbot en (van Lun, 2011), Eliza del MIT en 1960, Elbot 2008, Alice superbot, Anna de Ikea o Irene de Renfe).

Al estar desarrolladas para un contexto específico resulta difícil reutilizar las aplicaciones para un contexto diferente. Desde hace algún tiempo se están realizando investigaciones y generando herramientas que facilitan la labor de realizar estas aplicaciones sin partir desde cero. Se desea también que estos sistemas sean multilingües y adaptativos ante situaciones desconocidas en el momento del desarrollo.

2.5.1. Visión general de un sistema de diálogo

La complejidad de la elaboración del sistema de diálogo disminuye al fragmentarlo en módulos, cada uno encargado de realizar una tarea específica. Existen varias arquitecturas para definir sistemas de diálogo, pero todas ellas disponen de los módulos básicos siguientes (López Cózar, 2011) que se muestran en la figura 2.2:

- Reconocimiento automático del habla (RAH). El módulo encargado de traducir la voz del usuario en forma de señal acústica en palabras (palabras que existen en la gramática definida para el sistema o para el estado de diálogo correspondiente).
- Comprensión del habla (CH). El módulo que a partir de una secuencia de palabras obtiene el significado semántico de las mismas.

- Gestor de diálogo (GD). Decide qué realizar en base a una secuencia de palabras con significado para el sistema. Las acciones pueden ser realizar una pregunta o dar información al usuario, actualizar la base de datos, abrir una puerta...
- Generador de respuestas (GR). Obtiene del GD la acción a realizar cuando ésta trata de comunicar algo al usuario y construye una frase en lenguaje natural.
- Síntesis de texto a voz (STV). A partir de una frase en lenguaje natural genera una señal acústica en forma de voz humana.

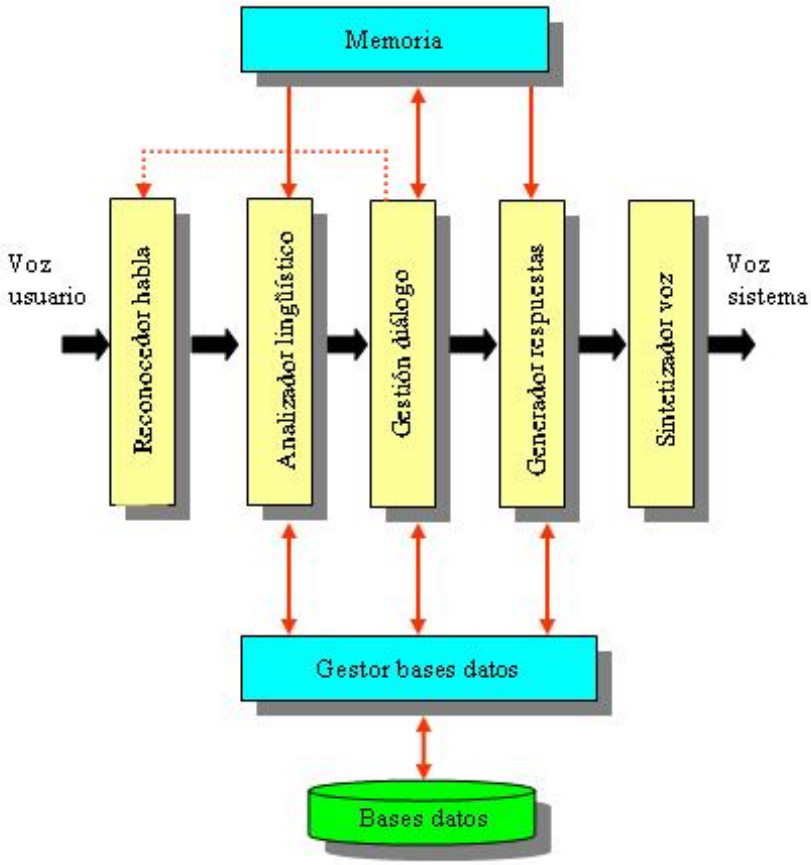

Figura 2.2: Visión general de la arquitectura modular de los sistemas de diálogo (López Cózar, 2011)

Pueden emplearse más módulos aparte de éstos como el de base de datos, que contendría la información que necesita la aplicación y el de memoria que mantendría el histórico de la conversación. Pero depende del uso del sistema de diálogo que se usen éstos u otros distintos, eso sí los cinco anteriores estarán presentes siempre. El modo de agrupar y conectar dichos módulos da las distintas arquitecturas que existen actualmente (Turunen, 2004).

Monolíticos Todo reside en una misma máquina. No existe separación física ni lógica entre los módulos, un módulo entra en ejecución tras finalizar el anterior y cuando finaliza da paso al siguiente módulo. Son los sistemas más básicos y sencillos, también los menos reutilizables y más sensibles a fallos de sistema. Se usan sobre todo en programas de aprendizaje e investigación para pequeños sistemas poco potentes.

Con gestor central Los módulos pueden estar separados físicamente, cada módulo en una máquina distinta, o estar físicamente en la misma máquina, pero en todo caso deben estar separados lógicamente. Para conseguir la separación lógica se desarrolla cada módulo por separado como si fueran a ejecutarse en máquinas distintas pero de manera configurable para que se pueda poner la misma máquina a todos ellos. Estos sistemas se caracterizan por delegar en el módulo del gestor de diálogo todo el control de la aplicación, las llamadas a los módulos y la recogida y paso de datos. Sigue la estructura cliente/servidor con topología de estrella. Esta arquitectura complica aún más el diseño del módulo del gestor y lo hace menos reutilizable por lo que hoy en día se desaconseja su uso.

Con arquitectura Galaxy La arquitectura Galaxy (Seneff, Hurley, Lau, Pao, Schmid & Zue, 1998) es desarrollada por el MIT (Massachusetts Institute of Technology en 1994), y usada también por la DARPA (Defense Advanced Research Projects Agency). La filosofía es la de cliente/servidor usando una topología de estrella. Sitúa independientemente cada módulo y los conecta mediante un Hub. El Hub recibe la información que da un módulo en forma de mensaje, resuelve a qué módulo está destinada dicha información en base a unas reglas y se la envía. El formato del mensaje es bastante sencillo y está basado en frames. La semántica de estos frames no está especificada en ningún sitio, pero básicamente es una colección de estructuras atributo-valor. Galaxy provee un diseño de arquitectura muy general, válido para multitud de aplicaciones y no impone seguir un tipo de desarrollo u otro. Por otro lado no provee ninguna información sobre cómo gestionar los servicios y hacer el diálogo de manera adaptativa.

Con arquitectura OAA OAA (Open Agent Architecture) es la arquitectura que permite construir aplicaciones multiagente (Kuppevelt, Dybkjær, Dybkjær, Bernsen, Wilks, Webb, Setzer & Catizone, 2005). Los componentes o agentes son independientes y se comunican entre ellos pasándose mensajes. Esta arquitectura no se diseñó pensando en sistemas de diálogo, como se hizo con Galaxy, su propósito es más general, pero válido para estos sistemas también.

2.5.2. Gestión del diálogo

El gestor del diálogo es el encargado de realizar la interacción con el usuario. Debe llevar el flujo del diálogo lo más natural y eficientemente posible, manteniendo un histórico de la conversación, comunicándose con la aplicación externa y decidiendo la respuesta que dar al usuario. El gestor de diálogo debe respaldar a los módulos de reconocimiento y comprensión. La tecnología actual hace que el reconocimiento no sea 100 % fiable luego esa tasa de errores debe ser gestionada por el gestor intentando minimizar sus efectos y corregirlos si es el caso. El gestor debe responder fluidamente, el usuario no esperará eternamente una respuesta y su satisfacción será mayor cuanto más rápido obtenga lo que desea. Ambos requisitos son imprescindibles para disponer de un sistema aceptable. Las tareas del gestor entonces son:

- Obtener datos del usuario. Crear preguntas si es necesario.
- Confirmar datos del usuario. Si no es probable el acierto pide confirmación del dato de usuario.
- Corregir errores de diálogo. Ante un error en el diálogo trata de solventarlo.
- Devolver información al usuario. Devuelve la información de una consulta.
- Generar expectativas. Crea una mini gramática de lo que se espera que diga en el siguiente estado de diálogo lo que ayuda al reconocedor a mejorar su tasa de acierto y su rapidez.

El tiempo de respuesta del gestor es muy importante. Según (Carrión, 2008) el usuario considera que la interacción con la máquina es en tiempo real si la respuesta sucede antes de 0,1 segundos. Si tarda menos que 1 segundo el usuario ya nota retraso pero su pensamiento sobre el diálogo no

se ve interrumpido. Antes de 10 segundos el usuario está prestando atención al sistema. Pasado ese tiempo es necesario darle algún tipo de respuesta, un mensaje o la reproducción de alguna música.

Existen tres opciones para que el gestor y el usuario manejen el diálogo (González-Ferreras, 2009):

- Control del diálogo dirigido por el sistema. El sistema tiene claro el objetivo a cumplir y realiza preguntas al usuario para resolver las incógnitas. El orden de las preguntas esta predefinido por el sistema. Estos sistemas también se denominan sistemas de diálogo guiados.
- Control del diálogo dirigido por el usuario. El usuario es el que toma la iniciativa dando datos al sistema. Este tipo de control requiere que el usuario tenga conocimientos sobre cómo interactuar con el sistema. Estos sistemas también se denominan sistemas de diálogo cooperativos.
- Iniciativa mixta. Es una mezcla de las dos anteriores opciones. Normalmente el sistema tendrá la iniciativa, pero el usuario en un momento dado puede interrumpirle y tomar él la iniciativa. Otra variante es dejar la iniciativa al usuario, y cuando ocurra un error o una situación que el usuario no sepa resolver (diálogo repetido) tome el control el sistema. Estos sistemas también se denominan sistemas de diálogo adaptativos y son los más complejos.

Y para que el gestor realice el control de diálogo existen las siguientes estrategias (Griol Barres, 2008) (McTear, 2002):

Sistemas basados en estados finitos En un sistema basado en estados finitos, el diálogo se representa en un diagrama de estados donde los nodos representan estados del diálogo y los enlaces o transiciones entre estados los cambios que hacen que el estado de diálogo cambie. Las ventajas que tienen este tipo de estrategia son muchas, es un modelo muy usado en muchas aplicaciones y es bien conocido. Se puede definir una mini-gramática para cada estado que puedan usar módulos como el reconocedor, con lo que la tasa de reconocimiento será muy alta. Para flujos de diálogo que se repiten en distintas situaciones se pueden diseñar subdiálogos, esto es muy útil para el control de errores y confirmaciones. El usuario está muy restringido, ya que desde un estado se puede pasar a otro solo con los movimientos que estén definidos en las transiciones, con lo que es difícil alcanzar un diálogo natural y flexible. Para cambiar el flujo de diálogo sería necesario cambiar el grafo de estados.

Sistemas basados en frames En un sistema basado en frames el usuario es preguntado por el sistema para rellenar los huecos de una plantilla para realizar un propósito determinado, estos huecos serían las variables de un frame. El flujo de diálogo es más natural que en la estrategia de estados finitos, ya que el usuario puede dar los datos en el orden que quiera e incluso dar varios en una misma frase.

Sistemas basados en planes Para sistemas algo más complejos es más viable utilizar una estrategia de diseño basada en planes. La idea es que los usuarios se comuniquen alcanzando objetivos. Se modela el diálogo como una cooperación entre el usuario y el sistema para alcanzar objetivos comunes. Cada frase no se considera como una cadena de texto, sino como un acto de diálogo en el que el usuario comunica sus intenciones. Este enfoque está relacionado con la teoría de "estado de información". Un estado de información de un diálogo representa la información necesaria para distinguirlo del resto de diálogos. La información consiste en las intervenciones acumuladas del usuario y las acciones previas. Las tareas del gestor son actualizar el estado y seleccionar la siguiente acción basándose en las acciones del usuario observadas.

Sistemas basados en agentes Estos sistemas modelan el diálogo como una comunicación entre agentes. Se emplean técnicas de inteligencia artificial para gestionar la interacción. Estos sistemas intentan que la iniciativa del dialogo sea mixta, así el usuario puede cambiar de tema o dar más información sin que el sistema se lo tenga que pedir. Tienen similitud con

los sistemas basados en planes pero introduciendo el término de agente. El flujo de diálogo es mucho más natural, sin embargo su desarrollo es mucho más complejo. El término de agente no es el mismo que el agente de la arquitectura OAA, pero siguen la misma idea de independencia y adaptabilidad.

Sistemas de diálogo basados en reglas o en métodos estocásticos Estos dos tipos de estrategia se pueden combinar con las anteriores, se utilizan para comprender el modelo lingüístico. La primera solución adoptada fue la comprensión mediante reglas o axiomas, esto generaba dificultades de diseño y el cambio de una tarea implica la realización de un nuevo modelo de reglas. Los basados en métodos estocásticos, los que introducen la probabilidad y el azar en el funcionamiento del sistema, no tienen estos inconvenientes pero tienen la dificultad de la adquisición de un corpus. Dicho corpus contiene ejemplos de diálogos que podrían realizarse con el sistema. Un tipo de modelaje estocástico es el uso de POMPD (Partially Observable Markov Decission Processes) que puede utilizarse para definir un sistema de diálogo. Ver (Williams & Young, 2007) para más detalles. Otro tipo son los sistemas de diálogo como red bayesiana que se basan en el trabajo de Thomas Bayes en el campo de la probabilidad.

2.5.3. Diseño de un sistema de diálogo

Realizar el diseño de un sistema de diálogo es una tarea muy compleja. Existen varios proyectos de investigación, EAGLES (Bernsen, Dybkjær & Dybkjær, 2000) y DISC (Gibbon, Moore & Winski, 1997) en donde se dan recomendaciones y proponen mejores prácticas para el desarrollo de sistemas de diálogo. En cualquier caso, se use el ciclo de vida que se use, a la hora de diseñar un sistema de diálogo hay que tener presente la forma en la que los usuarios interaccionaran con él, es decir, cómo hablaran con el sistema. Se puede actuar de distintas formas:

- Diseño por inspiración. El sistema se construye analizando la funcionalidad que tiene que dar el sistema y se basa en la inspiración de los desarrolladores. El inconveniente de este diseño es la posibilidad de existencia de situaciones de diálogos no previstas por los desarrolladores.
- Diseño por observación. Se basa en el estudio de cómo interactúan los usuarios para resolver las tareas que resolverá el sistema. Es decir se recopila y analiza un corpus de diálogo humano-humano. La recopilación del corpus es una tarea muy costosa, además hay que tener en cuenta que las interacciones humano-humano no son exactamente iguales que las interacciones humano-computador.
- Diseño por simulación. Se emplea la técnica de mago de Oz que consiste en simular que el sistema de diálogo está ya construido, entrenar a un mago, una persona que se hace pasar por el sistema y hacer que los usuarios crean que están hablando con la máquina. La adquisición de corpus por simulación es más fiable que la de por observación.
- "System in the loop". Es una técnica que permite simular el funcionamiento del sistema al completo, simulando únicamente uno o algunos de los módulos. Esta estrategia se realiza en fases tempranas del desarrollo donde algunos módulos no están operativos para poder simular el funcionamiento de todo el sistema.

Hay que tener en cuenta también el diseño de las estrategias de confirmaciones. Las confirmaciones pueden ser implícitas o explícitas. Las confirmaciones explícitas generan más confianza en la entrada al gestor de diálogo, pero se pierde naturalidad en el flujo del diálogo y se requieren muchas más intervenciones del usuario, lo que puede dar la sensación de sistema con poca capacidad de respuesta. En las confirmaciones, hay que tener en cuenta las medidas de confianza si se diseña un sistema de diálogo basado en métodos estocásticos.

2.5.4. Uso del estándar VoiceXML

La construcción de sistemas de diálogo es un proceso complejo como hemos visto. Esto ha motivado que en los últimos años se haya producido un gran esfuerzo de estandarización para simplificar la tarea de diseño y construcción de sistemas de diálogo. El estándar más aceptado, es sin duda VoiceXML que se basa en el lenguaje de marcado del mismo nombre ya descrito (Voice eXtensible Markup Language). El objetivo de este estándar fue facilitar la creación de sistemas comerciales que pudieran ser usados en entornos de explotación reales. Para ello, el estándar permite independizar la aplicación vocal de la plataforma tecnológica empleada, lo que permite que las aplicaciones sean portables entre plataformas de diferentes fabricantes. La aparición de la versión 1.0 de VoiceXML en el año 2000 convirtió el diseño de interfaces vocales en una tarea de producción, al eliminar la necesidad de un diseño a bajo nivel que requería un gran dominio de las tecnologías de síntesis y reconocimiento de habla. VoiceXML facilita el desarrollo de aplicaciones vocales siguiendo un modelo declarativo, al separar el modelado del diálogo, que se realiza mediante un lenguaje de marcas, de la implementación del mismo, llevada a cabo por una plataforma de interpretación externa. El modo de funcionamiento es muy similar al de la web, lo que permite emplear las técnicas de modelado y de desarrollo de aplicaciones web. Además, la existencia de entornos de desarrollo integrados facilita la producción industrial de aplicaciones vocales.

No se desea diseñar un sistema de diálogo completo. Se quiere integrar tecnología existente para implementar interacción vocal en un sistema que permita interacción multimodal. Por ello, se apostará por utilizar herramientas que existen actualmente para desarrollar aplicaciones que integren sistemas de diálogo. Para cada una de ellas se ha tenido en cuenta la disponibilidad de un módulo de reconocimiento de voz, un módulo de síntesis de voz y la disponibilidad de una plataforma de voz compatible con VoiceXML.

2.6 3D en las Rich Internet Applications

Las Rich Internet Applications (RIA) se ejecutan desde un navegador como si fuesen una página web sin precisar de una instalación de software cliente en cada máquina en la que vayan a ser utilizadas. Debido a la popularización de este tipo de desarrollos por sus claras ventajas sobre la típica arquitectura cliente-servidor, éstas han llegado también al desarrollo de aplicaciones que permitan la interacción con mundos de realidad virtual. Los contenidos basados en mundos de realidad virtual en línea son cada vez más demandados en los sitios de Internet. El ancho de banda y la capacidad de procesamiento de gráficos son ahora lo suficientemente avanzados para reproducir experiencias tridimensionales basadas en web, incluyendo aplicaciones para entretenimiento, entrenamiento, simulación, educación, defensa, medicina, arquitectura y comercialización.

El desarrollo de RIA que incluyan 3D es cada vez más interesante con la aparición de sistemas de desarrollo en manos de grandes corporaciones y también de nuevas tecnologías de software libre. Se introducen algunos de estos sistemas haciendo una comparativa de dichas alternativas desde diferentes puntos de vista.

2.6.1. Herramientas para incluir 3D en RIA

Las RIA añaden a la experiencia del usuario en herramientas y funciones de escritorio, lo mejor de la multimedia (voz, vídeo, etc.) y la flexibilidad de presentación y despliegue que ofrecen las páginas web. Son la nueva generación de aplicaciones y es una tendencia ya impuesta. Utilizan una arquitectura cliente-servidor asíncrona, segura y escalable, junto con una interfaz de usuario web. Hacen del uso de la aplicación algo muy sencillo, ofrecen mejoras en la conectividad y despliegue instantáneo de la aplicación, agilizando su acceso. Las posibilidades principales para desarrollar RIA actualmente son entre otras: Adobe Flash, Microsoft Silverlight, Ajax, Java y Adobe Director.

Adobe Flash Es una herramienta de autor desarrollada inicialmente por Macromedia que posteriormente fue adquirida por Adobe. Tiene forma de estudio que trabaja sobre un "escenario" y sobre "fotogramas" destinado a la producción de animación. Utiliza gráficos vectoriales e imágenes ráster, sonido, código de programa, flujo de vídeo y audio bidireccional. En sentido estricto, Flash es el entorno y Flash Player es el programa de máquina virtual utilizado para ejecutar los archivos generados con Flash. Además de Adobe Flash, tenemos Adobe Flex (Adobe, 2010f) que sería el servidor de aplicaciones de esta tecnología. Es capaz de generar contenidos dinámicamente y está basado en el lenguaje MXML (Adobe, 2010g) que describe interfaces de usuario, crea modelos de datos y tiene acceso a los recursos del servidor, todo ello en una estructura de etiquetas. Aunque existe desde versiones anteriores, en las últimas versiones de Adobe Flash, ActionScript es un lenguaje con características de orientación a objetos pero es un lenguaje de guiones prácticamente exclusivo para el lado cliente, debiendo recurrirse a otras soluciones para la generación dinámica e implementación de reglas de negocio en el servidor. El formato de archivo Flash binario y no documentado adecuadamente en sus primeras versiones ha representado siempre un obstáculo para el diseño de sitios accesibles. En versiones anteriores de Flash no existe 3D (Hague & Jackson, 2003). En la última versión cuenta con las capacidades 3D inherentes, herramientas de dibujo vectorial y comandos de programación. Es capaz de mostrar formas vectoriales, calcular expresiones, traslación de las cámaras, etc. Existe la necesidad de utilizar programas externos para crear vistas panorámicas. Por otro lado, no se ofrece funcionalidad de integración de diálogos pero se podría definir un objeto que hiciera de interfaz de un sistema de diálogo desarrollando comportamientos en ActionScript para ello. Adquirir de Adobe un SDK que permitiera trabajar sobre el formato binario de archivo Adobe Flash sería también otra opción para desarrollar un módulo que permita integrar diálogo.

Microsoft Silverlight Es un producto que nace a partir de un nuevo conjunto de servicios llamados de manera genérica Windows Presentation Foundation (WPF) implementados sobre la versión 3.0 de la plataforma Microsoft .NET. Inicialmente denominado Avalon, Microsoft ha creado un plug-in para navegadores que permite ejecutar en su interior aplicaciones con interfaces WPF, así como proveer las herramientas necesarias para desarrollarlas. Las aplicaciones Microsoft Silverlight pueden ser creadas tanto de forma dinámica en el servidor como a través de productos de diseño específicos (Microsoft Visual Studio 2008 o Microsoft Expression Blend) basados en una línea de tiempo a la que van agregándose elementos y efectos visuales escritos en códigos como Visual Basic o C#. Con independencia de la opción elegida, el lenguaje de programación a utilizar puede ser siempre el mismo y el resultado, la información que transmite al navegador, estará en formato XAML (eXtensible Application Markup Language), no en binario. Son aplicaciones que permiten la reproducción de vídeos, gráficos vectoriales, animaciones y otros elementos. Tiene capacidades y sintaxis propias para implementar "3D real" en lenguaje XAML pero se implementa una interacción "2D sobre 3D". En un alto nivel, la interacción con "2D sobre 3D" se consigue realmente interactuando con una versión escondida en 3D de ese contenido en 2D. El 2D se posiciona de tal manera que el punto en el 3D sobre el que se posiciona el ratón está exactamente en el mismo punto en el que está el ratón sobre la versión 2D oculta. Entonces, cuando el usuario hace click, interacciona exactamente con la misma localización (Microsoft, 2010). Existen proyectos de integración de reconocimiento y síntesis de voz en este tipo de aplicaciones basados en Speech Recognition Grammar Specification (SRGS) (Hunt & McGlashan, 2010). También se podría desarrollar un módulo en XAML que permita integrarlo con un sistema de diálogo.

Ajax A diferencia de los anteriores, no es una herramienta comercial. Ajax (Asynchronous JavaScript and XML) es una tecnología ligera y flexible, una filosofía, no es un conjunto de aplicaciones ni un lenguaje de programación concreto. Ha llegado a ser el método preferido para desarrollar aplicaciones web sofisticadas. Pone a disposición de los desarrolladores en Javascript la programación cliente-servidor vía HTML dinámico, obteniendo aplicaciones ri-

cas hospedadas en un navegador web. Para solventar la limitación en capacidad de presentar contenido dinámico de los navegadores actuales, en particular 3D de alto rendimiento en tiempo real, han surgido propuestas como Ajax3D (Parisi, 2010) que combina el poder de X3D (Web3D Consortium, 2010), que es el estándar para 3D en tiempo real en la web, con la facilidad de uso y ubicuidad de Ajax empleando el X3D Scene Access Interface (SAI) (Hudson, 2010) para controlar mundos 3D vía Javascript. Con solo añadir un plugin para X3D a los navegadores web actuales, se puede llevar el poder gráfico de la tecnología de videojuegos a la experiencia web diaria. Una aplicación Ajax3D es un programa hospedado en un navegador web que utiliza SAI para acceder a la escena 3D en tiempo real y utiliza XMLHttpRequest (Aubourg, Julian et al., 2010) para almacenar y recuperar datos de la aplicación 3D y el DOM para manipular el contenido de páginas web en respuesta a cambios en la escena 3D. Aun existen pocas opciones que integren reconocimiento de voz basadas en Ajax

Java Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90, muy extendido y multiplataforma. A pesar de que Sun ha lanzado JavaFx como respuesta a las herramientas comerciales para desarrollar RIA, se puede considerar la tecnología Java como una RIA más puesto que las aplicaciones desarrolladas en Java pueden distribuirse en forma de archivos fácilmente descargables desde servidores. Esto se puede conseguir desarrollando nuestra aplicación Java como un applet a incluir en un navegador de Internet. Las posibilidades que da Java para trabajar con 3D son: "Java+VRML" (EAI), "Java+X3D" (SAI) y Java3D (Bufill, 2003), aunque éste último puede combinar las anteriores. Las dos primeras son fruto de la evolución de los lenguajes de marcas para especificar escenas 3D en los que se basan pero su filosofía es la misma. Además del lenguaje de marcas para definición de escenas 3D, hacen uso de un API Java para manipular las escenas definidas. En el caso de "Java+VRML", el API se denomina External Authoring Interface (EAI) (Brutzman, 2010) y manipula nodos VRML (Hartman & Wernecke, 1996). "Java+X3D" utiliza el API Scene Authoring Interface (SAI) que manipula nodos X3D.

Sin embargo, Java3D es un API en el que se define interacción, comportamiento y otros elementos de la escena que en las dos primeras son definidos en los lenguajes de marcas. Las tres opciones proveen soporte para construir entornos virtuales a gran escala gracias a una descripción dinámica de escenas 3D, comportamiento y animación de cámara.

Se pueden usar herramientas externas para definir elementos 3D y tras ello exportarlos a VRML o X3D e importarlos en desarrollos realizados bien con las APIs EAI o SAI o bien con el API Java3D.

Adobe Director El producto Adobe Director (MacGillivray & Head, 2005) (antes Macromedia Director) es posiblemente la herramienta de autor más extendida en el mercado de la edición de CDs interactivos. Desde la versión 8.5 incluye la opción 3D. Adobe Director se ha adaptado a la supremacía de Internet frente a los CDs interactivos gracias a la disponibilidad del plugin Adobe Director Shockwave que permite visualizar las películas interactivas a través de Internet. Si en lo que se refiere al desarrollo de películas multimedia Flash ha cobrado ventaja frente a Adobe Director por la mayor simplicidad y por ser Flash una herramienta pensada específicamente para la web, el 3D es posiblemente el principal argumento que ha hecho que Adobe Director sobreviva. Adobe Director incluye una serie de herramientas accesibles desde el lenguaje de programación Lingo específico. Permite acceder a componentes 3D (cámaras, luces, modelos, texturas...) como elementos de un lenguaje orientado al objeto completo. Además, Adobe Director añade funciones para tratamiento de ficheros XML y funciones de accesibilidad incluyendo sistemas de conversión texto-voz.

Tabla 2.4: Comparativa de RIA (Noda & Helwig, 2005) (Charte Ojeda, 2008)

	Adobe Flash	Microsoft Silverlight	Ajax	Java	Adobe Director
Riqueza Gráfica	Muy rica	Rica	Media (Misma que HTML)	Rica	Muy rica
Contenedor	Ligero	Ligero	Muy ligero (en el navegador)	Pesado	Ligero
Descarga de Aplicación	Lento	Rápido	Rápido	Lento	Lento
Soporte Audio/Video	Excelente	OK	Pobre (a menos que use ActiveX)	OK	Excelente
Disponibilidad	Ms. Windows, Linux, Mac OS	Ms. Windows, Linux, Mac OS	Ms. Windows, Mac OS	Ms. Windows, Mac OS	Ms. Windows, Mac OS
Consistencia en Diferentes Entornos Informáticos	Muy consistente sobre Ms. Windows y Mac OS X	Relativamente consistente fuera de Ms. IE	Varía	Relativamente consistente	Muy consistente en Ms. Windows y Mac OS X
Requisitos del Servidor	Si (Flex u Open Laszlo (OpenLaszlo, 2010))	Microsoft .NET	Ninguno o mínimos (TIBCO General Interface	Ninguno o mínimos (Nexaweb (Nexaweb, 2010), Java Web Start)	Ninguno
Plug-in/Requisitos de Ejecución en el Cliente	Flash (Player)	Microsoft Silverlight	Plug-in para X3D	Java Runtime (JRE) y Plug-in para VRML/X3D	Shockwave (Player)
Lenguaje de programación	ActionScript	C#, Visual Basic	JavaScript	Java, JavaScript	Lingo, JavaScript
Desafío de Desarrollo	Relativamente fácil con las herramientas Adobe Flex u Open Laszlo	Relativamente fácil con las herramientas Ms. Visual Studio o Ms. Expression Blend	Muy complejo sin herramientas tales como TIBCO	Relativamente fácil con la herramienta Nexaweb	Relativamente fácil con la herramienta Director
Conocimientos de Desarrollo	XML, DOM, JavaScript	-	CSS, XML, XSLT, DOM, ActiveX, X3D	XML, VRML/X3D, Java3D	-
Aspectos de seguridad	Ficheros Flash (binarios comprimidos) generados	Ficheros XAML generados	Códigos JavaScript abiertos al público	Ficheros binarios Class/Jar comprimidos generados	Ficheros shockwave (binarios comprimidos) generados
Coste de Licencia	SI	SI	NO	NO	SI

Tabla 2.5: Capacidades 3D (Adobe, 2010e)

	Adobe Flash	Microsoft Silverlight	Ajax	Java	Adobe Director
3D	"3D emulado" "3D real" en la última versión	"3D real" (XAML)	"3D real" (X3D)	"3D real" (VRML/ X3D/ Java3D)	"3D real" (W3D)
Modelos 3D, luces y cámaras	Implementado con ActionScript	-	Capacidades propias de X3D	Capacidades propias de Java3D, VRML o X3D	Completo
Texturas	Importadas de otras herramientas	Capacidades propias de XAML	Capacidades propias de X3D	Capacidades propias de Java3D, VRML o X3D	Completo
Animaciones	A través de fotogramas clave e interpolaciones	Programables	Programables	Programables	Incluidas en los modelos y programables
Grupos	-	-	Capacidades propias de X3D	Capacidades propias de Java3D, VRML o X3D	Definición Jerárquica del mundo
Cinemática inversa	-	-	-	-	Incluye bones y un Xtra Havok
Dinámica y partículas	-	-	-	-	Con el Xtra Havok Physics
Interacción gráfica	2D OK, 3D en última versión	"2D sobre 3D"	OK	OK	OK
Interacción vocal	Herramientas de micrófono	SGRS	-	VoiceXML	Sintetizador de voz incluido
Dispositivos de interacción complejos	NO	-	A través de X3D	A través de VRML/X3D	Soporta joysticks
Visualización estereoscópica	Spatialview (Spatialview, 2010)	-	A través de X3D	A través de VRML/X3D	-
Capacidades de pantalla completa para dispositivos HMD	Soporta flv	-	A través de X3D	A través de VRML/X3D	Soporta flv, DVD-video, wmv, rm, mov, avi
Audio	Mp3, aif, wav	-	A través de X3D	A través de VRML/X3D	Mp3, aif, wav, swa, au, wma, ra
Proyectos	Papervision3D (Papervision3D, 2010)	Away3D (Away3D, 2010)	CodePlex (Salas, 2010)	AJAX3D	XMMVR

2.6.2. Comparativa y conclusiones

Tras presentar cada una de las tecnologías, resumimos sus características en la tabla 2.4 donde se presenta una comparativa a modo de resumen de las características técnicas de las tecnologías presentadas evaluando "Riqueza gráfica" de las aplicaciones generadas, "Contenedor" donde se ejecutará la aplicación, tiempo de "Descarga de Aplicación", "Soporte Audio/Video", "Disponibilidad", "Consistencia en Diferentes Entornos Informáticos", "Requisitos del Servidor" de la aplicación, "Plug-in/Requisitos de Ejecución en el Cliente", "Lenguaje de programación" necesarios para desarrollar la aplicación, "Desafío de Desarrollo" entendido por el grado de dificultad, "Conocimientos de Desarrollo" además de los lenguajes de programación, "Aspectos de seguridad" de las aplicaciones desarrolladas y "Coste de Licencia" a adquirir para su utilización.

Asimismo, en la tabla 2.5 se presentan las capacidades 3D de cada tecnología analizada, incluyendo capacidades de interacción y otras propias de definición de mundos tridimensionales. Así se evalúa la capacidad de definir y representar elementos "3D" y el formato de archivo utilizado, las posibilidades de definir "Modelos 3D, luces y cámaras", capacidad de incluir "Texturas", la forma de

representar "Animaciones", si se considera la definición de "Grupos", representación de "Cinemática inversa", posibilidad de representar "Dinámica y partículas", posibilidades de interacción gráfica e interacción vocal, posibilidad de hacer uso de "Dispositivos de interacción complejos", conseguir "Visualización estereoscópica", "Capacidades de pantalla completa para dispositivos HMD ", "Audio" y finalmente "Proyectos" representativos del uso de cada tecnología.

Se puede concluir que cada tecnología presentada ofrece ventajas e inconvenientes según el criterio básico a considerar pero debemos tener en cuenta que la riqueza de todas reside en la posibilidad de combinarlas y de integrarlas con otras tecnologías. De esta forma se abre un gran futuro para el desarrollo de RIA para interacción con realidad virtual. Así, gracias a la universalización de redes con mayor ancho de banda, al aumento en capacidad de procesamiento de gráficos de los equipos interconectados y al cada día mayor número de posibilidades de combinar modos de interacción, permite llegar a un número prácticamente ilimitado de usuarios.

2.7 Evaluación de sistemas multimodales

Se presentará la evaluación de los sistemas multimodales como una evolución de la evaluación de los sistemas de diálogo, repasando metodologías y recomendaciones, cuestiones de usabilidad y evaluaciones objetiva y subjetiva de ésta. Finalmente se introducirá una propuesta de evaluación de aplicaciones basadas en entornos virtuales.

2.7.1. Evaluación de los sistemas de diálogo

La evaluación de los sistemas de diálogo es crucial para medir su rendimiento y la aceptación por parte de los usuarios. Además, permite analizar la calidad de los diálogos que se establecen entre el sistema y el usuario. Existen varias metodologías y recomendaciones para evaluar sistemas de diálogo que veremos a continuación, la mayoría se centran en la usabilidad y para ello se diseñan cuestionarios de evaluación subjetiva junto a evaluaciones con usuarios reales o simulados. El esquema más ampliamente utilizado para la evaluación de sistemas de diálogo es PARADISE (Walker, Litman, Kamm & Abella, 1997) cuya evolución PROMISE (Beringer, Kartal, Louka, Schiel, Tuerk & Trk, 2002) es una buena propuesta para evaluar sistemas multimodales .

2.7.1.1. Metodologías y recomendaciones

En el proyecto EAGLES (Dybkjær, Bernsen & Minker, 2004) se señala la importancia del entorno de experimentación, y se distingue entre las evaluaciones llevadas a cabo en el laboratorio frente a las realizadas en entornos reales. El propósito de la evaluación es el que determina la elección del entorno concreto. Por otro lado, se distingue entre las pruebas de caja blanca y las pruebas de caja negra. En el primer caso, la evaluación mide el rendimiento de uno o varios componentes del sistema de diálogo. En el segundo caso, la evaluación considera el rendimiento global del sistema de diálogo, sin analizar el comportamiento de los componentes internos del mismo. En el proyecto DISC (Dybkjær, Bernsen & Minker, 2004) se describe en detalle la terminología a emplear y las plantillas que deben rellenarse para cada propiedad del sistema que se desea evaluar. Otros proyectos son EVALDA (Dybkjær, Bernsen & Minker, 2004) que plantea pruebas de caja negra durante campañas de recogida de datos en largos periodos de evaluación y ATIS (Dybkjær, Bernsen & Minker, 2004) que realiza una evaluación objetiva de la respuesta del sistema de diálogo a la entrada del usuario basándose en anotaciones de las interacciones.

PARADISE (PARAdigm for Dialogue System Evaluation) consiste en un esquema general para la evaluación de sistemas de diálogo (Walker, Litman, Kamm & Abella, 1997). Este esquema es capaz de proporcionar una medida del rendimiento global de un conjunto de diálogos o subdiálogos y de especificar la contribución relativa de diferentes factores al rendimiento global. Permite además comparar diferentes sistemas que trabajan sobre diferentes tareas mediante una normalización sobre

la complejidad de la tarea. Calcula una función de rendimiento que predice la satisfacción del usuario a partir del éxito de la tarea y de los costes del diálogo. La formulación tiene en cuenta la complejidad de la tarea y permite comparar agentes incluso aunque éstos realicen tareas distintas. También es posible medir el rendimiento para subdiálogos de la misma forma que para diálogos completos. Fue utilizado en el proyecto Communicator, financiado por DARPA, que llevó a cabo la evaluación de sistemas de diálogo a gran escala. En el proyecto participaron nueve centros de investigación y todos utilizaron la arquitectura Galaxy. La tarea fue la misma para todos: planificación de viajes, incluyendo reserva de avión, reserva de hotel y alquiler de coche. La forma de realizar la evaluación fue igual para todos, permitiendo de esta manera la comparación de los resultados de los diferentes grupos. PARADISE utiliza métodos de teoría de decisión (Keeney & Raiffa, 1976) para combinar medidas de rendimiento dispares: satisfacción de usuario, éxito de la tarea y costo del diálogo en una sola función de evaluación. Esto requiere la especificación de objetivos y un conjunto de medidas para operar con los objetivos.

Puesto que los sistemas de diálogo son unimodales, al añadir una nueva modalidad a éstos, bien sea reconocimiento de escritura, reconocimiento de cara, etc. la evaluación realizada con PARADISE no sirve. Por lo que se planteó la necesidad de evaluar estos nuevos sistemas multimodales evolucionados. Una propuesta para realizar esta evaluación es PROMISE donde partiendo de la función de rendimiento de PARADISE se propone una nueva función de rendimiento calculando el éxito de la tarea en base a que se complete ésta o no y teniendo en cuenta la cooperación del usuario.

2.7.1.2. Usabilidad de sistemas de diálogo

El objetivo de la evaluación es obtener la información necesaria para poder mejorar la experiencia de los usuarios al interactuar con un sistema de diálogo. Es decir, el objetivo es conseguir sistemas con una mayor usabilidad. El estándar ISO 9241 (UsabilityNet, 2012) define la usabilidad como: grado en el que un producto puede ser usado por usuarios específicos para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso específico. La efectividad hace referencia a la precisión y completitud con la que los usuarios consiguen sus objetivos mientras que la eficiencia es el coste de obtener esos objetivos. Finalmente la satisfacción está relacionada con el confort y la aceptación de los usuarios.

Para evaluar la usabilidad de un sistema se realizan pruebas con usuarios reales. Como resultado de dichas pruebas, se obtienen una serie de medidas objetivas y subjetivas. Las medidas objetivas nos sirven para medir la efectividad y eficiencia del sistema, mientras que las medidas subjetivas nos sirven para medir la satisfacción de los usuarios.

En el caso de los sistemas de diálogo, la evaluación objetiva está bastante bien establecida y se han propuesto numerosas medidas. Entre todas las medidas propuestas, podemos citar entre las más utilizadas las siguientes: tiempo para completar la tarea, tasa de éxito de la tarea, tasa de reconocimiento de habla, tasa de comprensión de habla, porcentaje de turnos de corrección, número de peticiones de ayuda, número de interrupciones del usuario, número de turnos por tarea, tiempo por turno.

En cuanto a las medidas subjetivas, éstas no se pueden observar directamente y se obtienen preguntando a los usuarios después de que hayan usado el sistema. Suelen medirse mediante cuestionarios o entrevistas. Lo más utilizado son los cuestionarios, en los que hay una serie de afirmaciones sobre la percepción de los usuarios acerca del sistema y los usuarios tienen que indicar su grado de conformidad con cada afirmación. Para que los resultados obtenidos sean significativos y reflejen las verdaderas actitudes de los usuarios, es preciso que el cuestionario se diseñe cuidadosamente, se documente y se valide.

2.7.1.3. Cuestionarios para evaluación subjetiva

El desarrollo y validación de un cuestionario para la evaluación subjetiva de sistemas de diálogo es un proceso costoso. El objetivo de SASSI (Subjective Assessment of Speech System Interfaces) (Hone & Graham, 2000) (Möller, Smeele, Boland & Krebber, 2007) es conseguir un cuestionario que sea fiable y válido para medir la experiencia del usuario utilizando una aplicación hablada. Inicialmente se propuso un cuestionario de 50 afirmaciones, que fue utilizado para evaluar ocho sistemas distintos. Para analizar los resultados se emplearon técnicas estadísticas, lo que resultó en la eliminación de varias preguntas, pasando a tener el cuestionario 34 preguntas. A continuación, mediante análisis de componentes principales se identificaron seis factores, donde cada uno de ellos mide un aspecto de la percepción de los usuarios sobre el sistema: corrección en la respuesta del sistema, afabilidad, demanda cognitiva, molestia, habitabilidad, rapidez. Se entiende lo que mide cada factor por su nombre, con la excepción de la habitabilidad, que hace referencia a si el usuario sabe lo que debe hacer y lo que el sistema está haciendo. Puede entenderse también como la adecuación entre el modelo conceptual del usuario acerca del sistema de diálogo como agente conversacional y el propio sistema de diálogo. Puede adaptarse para la evaluación de sistemas multimodales.

2.7.2. Evaluación de aplicaciones basadas en entornos virtuales

La evaluación de aplicaciones basadas en entornos virtuales es todavía un campo de investigación poco desarrollado. Existen propuestas tales como (Freitas, Rebolledo-Mendez, Liarokapis, Magoulas & Poulovassilis, 2009) que propone un marco de cuatro dimensiones para evaluar el desarrollo de aplicaciones basadas en mundos virtuales específicas para aprendizaje, pero no inciden en el proceso de desarrollo de la aplicación a nivel de las tecnologías utilizadas. También existen propuestas de evaluación heurística de aplicaciones basadas en realidad virtual como por ejemplo (Sutcliffe & Gault, 2004) que a su vez siguen las pautas de la evaluación heurística clásica, en este caso (Nielsen, 1995). VRUSE (Kalawsky, 1999) es un cuestionario para sistemas de realidad virtual dirigido a sistemas inmersivos. Es largo y complejo. Pero encontramos una aproximación sistemática inspirada en metodologías de ingeniería de usabilidad que propone usar cuatro fases (Gabbard, Hix & Swan, 1999) (Hix, Swan, Gabbard, McGee, Durbin & King, 1999) y que enumeramos:

1. Análisis de tareas de usuario que es el proceso mediante el cual se identifica una completa descripción de tareas, subtareas y métodos requeridos para usar un sistema, así como otros recursos necesarios para que el usuario o usuarios y el sistema ejecuten tareas cooperativamente.

2. Evaluación basada en pautas expertas (evaluación heurística o inspección de usabilidad) que apunta a identificar problemas potenciales de usabilidad comparando un diseño de interacción (existente o evolucionado) con pautas de diseño de usabilidad establecidas.

3. Evaluación formativa centrada en usuario que es un método de evaluación empírico y observacional que asegura la usabilidad de sistemas interactivos. Incluye usuarios temprana y continuamente durante el desarrollo de la interacción de usuario.

4. Evaluación comparativa global que es una valoración empírica de un diseño de interacción en comparación con otros diseños de interacción maduros para ejecutar las mismas tareas de usuario.

En el capítulo 6 explicaremos en detalle estas cuatro fases y cómo las aplicaremos en nuestra propuesta.

2.7.3. Otros métodos de evaluación de aplicaciones

A continuación se revisan varias propuestas en base a (Khnel, 2012).

SUMI (Kirakowski & Corbett, 1993) o Software Usability Measurement Inventory no se recomienda para evaluación de sistemas multimodales.

SUS (Brooke, 1996) o System Usability Scale es una escala de usabilidad confiable y de bajo costo que puede utilizarse para evaluación global de usabilidad de sistemas cubriendo al menos en parte aspectos como aprendizaje, efectividad, eficiencia, estética, personalidad del sistema y apariencia del sistema.

USE (Lund, 2001) (Usefulness, Satisfaction, and Ease of Use - Usabilidad, Satisfacción y Facilidad de Uso) propone un cuestionario que puede servir para realizar una evaluación subjetiva de sistemas multimodales así como cualquier otro sistema donde se quiera conocer sus capacidades de usabilidad. Se describirá en detalle en el capítulo 6 donde lo usaremos para la evaluación subjetiva de una aplicación desarrollada siguiendo nuestra propuesta.

AttrakDiff (Hassenzahl, 2008) cubre al menos en parte aspectos como aprendizaje, efectividad, eficiencia, estética, personalidad del sistema y apariencia del sistema.

SUXES (Turunen, Hakulinen, Melto, Heimonen, Laivo & Hella, 2009) es un método de evaluación para recoger métricas con experimentos de usuario. Captura expectativas y experiencias de usuario haciendo posible analizar el estado de la aplicación y sus métodos de interacción y comparar resultados. Es un procedimiento completo, comienza con una introducción a la evaluación y un cuestionario de experiencia. Esto es seguido por una introducción a la aplicación y la estimación de expectativas de los usuarios. Luego, se ejecuta el experimento de usuario y la experiencia del usuario se evalua. Los cuestionarios se basan en un conjunto de nueve principios relativos a la velocidad, agradabilidad, claridad, uso libre de error, robustez, curva de aprendizaje, naturalidad, utilidad y uso futuro. Los parámetros de interacción no se analizan. El método se dirige a la cuestión de si el grado de expectativas es alcanzado por la experiencia real con el sistema. Los autores dicen que el método es eficiente y particularmente apropiado para el desarrollo iterativo.

2.8 Resumen

En este capítulo hemos revisado las diferentes posibilidades para construir una propuesta que responda a nuestra hipótesis de partida. Tras revisar las aplicaciones multimodales 3D existentes, se han introducido lenguajes de marcado que permiten especificar diálogo, escena o comportamiento de manera única (dedicados) y otros que permiten especificar una combinación de más de una de estas modalidades e incluso otras como son el gesto y/o la escritura caligráfica o anotación manual de datos (híbridos). Observando que todos estos lenguajes declarativos se basan en marcas es decir, son aplicaciones XML, la primera condición para el lenguaje que hemos de definir es que se base en XML. Puesto que ya existen lenguajes de especificación básicos para definir parte de lo que queremos (X3D y VRML para escenas 3D y VoiceXML para diálogos), optar por definir un lenguaje de especificación "híbrido" utilizando lenguajes de especificación "básicos" facilitará mucho los cosas y nos permitirá reutilizar módulos ya definidos. Por ello, la segunda condición de nuestro lenguaje es que sea "híbrido". Tras estudiar los diferentes tipos de lenguajes de marcado "básicos" (para especificar interacción vocal o diálogo, imágenes 2D, escenas 3D y comportamiento), "híbridos" y "dedicados", deducimos que no existe ninguno que se ajuste a nuestra necesidad: especificar espacios virtuales con interacción multimodal (gráfica y vocal). Así la tercera condición de nuestro lenguaje y la más importante es que permita especificar diálogo, escenas y comportamiento.

Vistas todas las posibilidades y las características propias de los modos de interacción y su naturaleza, podemos ya intuir cuáles serán los problemas a resolver: definir un sistema multimodal capaz de implementar cada una de las metáforas de interacción gráfica y vocal así como todas las posibles cooperaciones entre ambas e integrarlas en aplicaciones basadas en espacios virtuales. La propuesta de arquitectura del grupo MMI del W3C ha de servir como base de los requisitos

de la arquitectura del sistema multimodal a definir ya que todas las demás propuestas realizadas anteriormente son pequeños prototipos de laboratorio que se han introducido en el apartado 2.1 y que han sido solamente intentos de resolver problemas concretos pero sin ser capaces de dar una respuesta general. Conocidas las arquitecturas de los sistemas de diálogo existentes y vista la necesidad de utilizar VoiceXML para resolver nuestro problema, concluimos que el único requisito de la arquitectura del sistema de diálogo a integrar es que incluya un navegador VoiceXML. Revisadas cada una de las tecnologías RIA, sus características y capacidades 3D, se puede concluir que cada tecnología presentada ofrece ventajas e inconvenientes según el criterio básico a considerar pero debemos tener en cuenta que la riqueza de todas reside en la posibilidad de combinarlas y de integrarlas con otras tecnologías. Por ello, la elección de una tecnología RIA para integrarla en nuestro sistema se basará además en la experiencia que tengamos respecto a su utilización.

Finalmente se ha visto que la evaluación de sistemas multimodales puede ser vista como una evolución de la evaluación de sistemas unimodales es decir, evaluación de sistemas de diálogo. Esto es debido a que muchas características se comparten en ambos. También se ha observado que en el caso concreto que nos ocupa, los sistemas basados en entornos de realidad virtual, se han desarrollado pocas pero algunas pautas para la evaluación de este tipo de sistemas, estando la evaluación subjetiva más desarrollada que la evaluación objetiva.

En el capítulo 3 detallaremos nuestra propuesta presentando el lenguaje a utilizar para definir interacción gráfica y vocal, escenas y comportamiento. Seguidamente en el capítulo 4 demostraremos las capacidades de este lenguaje para especificar las distintas metáforas de interacción. Tras ello, en el capítulo 5 veremos los requisitos de la arquitectura que implemente las aplicaciones descritas. Finalmente en el capítulo 6 nos centraremos en la evaluación de una aplicación basada en nuestra propuesta para en el capítulo 7 presentar las conclusiones oportunas.

3
El lenguaje XMMVR

En este apartado se presenta una propuesta que combina interacción gráfica con mundos 3D y sistemas de diálogo planteando una plataforma multimodal de desarrollo de mundos virtuales 3D basada en diálogos especificada por un lenguaje de marcas basado en XML y que hemos denominado XMMVR: eXtensible markup language for MultiModal interaction with Virtual Reality worlds. Junto al lenguaje XMMVR se presenta un marco de referencia que pretende ser una plataforma de implementación de mundos 3D con integración de diálogos. La solución que se aporta respeta los estándares disponibles para interacción gráfica con mundos 3D y sistemas de diálogo sirviendo de vínculo entre ambos mundos, reutilizando el trabajo anteriormente definido por el grupo ECA-SIMM y dando una coherencia argumental a la definición de escenas 3D con interacción hablada.

3.1 Principios de diseño y metas de XMMVR

El objetivo general es disponer de un marco y un lenguaje para modelar aplicaciones de interacción persona ordenador multimodales (interacción gráfica e interacción vocal). La construcción de una aplicación concreta utilizando nuestro marco de trabajo consiste en especificar un mundo virtual, las secuencias de diálogo, las acciones que se generan y su relación con los elementos del mundo. Las secuencias de diálogo se especifican empleando VoiceXML y los mundos virtuales utilizando VRML, X3D, W3D (Adobe Director Shockwave) u otro formato de archivo para especificación de imágenes 3D. Para describir el comportamiento del mundo debemos especificar la estructura de componentes que forman el gestor del mundo y la correspondencia entre los diálogos con el usuario y las acciones de alto nivel incluidas en el gestor del mundo. Todas estas especificaciones se expresarán en un documento estructurado mediante XML conforme al DTD de XMMVR. Esto servirá para desarrollar una aplicación "embebida" en un navegador web que permita a un usuario controlar un mundo virtual 3D a través de la voz vía micrófono del ordenador donde se ejecute gracias a un gestor de dialogo implementado en VoiceXML y que permita, a su vez interactuar con dicho mundo virtual a través del teclado y ratón del ordenador donde se está ejecutando.

Para ello nos hemos apoyado en la metáfora de película cinematográfica (Bouzá, 1997) utilizada en herramientas de autor multimedia como Adobe Director (MacGillivray & Head, 2005). Antes del rodaje de toda película que se precie, se debe realizar una ardua labor para conseguir el reparto más adecuado y las localizaciones idóneas para el rodaje. La película se compone de escenas descritas por el guionista, que se distinguen entre sí porque en cada escena nueva entra un personaje. Para cada escena se tendrán en cuenta efectos de dramatización y la incorporación adecuada de recursos técnicos. Toda aplicación basada en mundos 3D debe seguir los mismos principios y por eso el CAST o reparto de actores ACTOR que intervienen en nuestros mundos 3D y la SEQUENCE o secuencia de escenas SCENE en las que transcurren las acciones de nuestros mundos 3D son la base de todo. Al final el resultado será una aplicación donde el usuario además de ser un espectador va a poder

interactuar con los mundos 3D de la aplicación de manera multimodal (gráfica y/o vocal).

Pero también queremos facilitar el trabajo al creador de aplicaciones basadas en mundos 3D. Así nuestro objetivo es ofrecer al desarrollador de aplicaciones la posibilidad de declarar mediante un script cómo será la aplicación. Para ello, en nuestra metáfora, el desarrollador asume el papel de guionista de cine. Cuenta con unos actores y con unos escenarios que a priori ha definido. Determina la secuencia de escenas y los actores que intervienen en éstas. Establece los diálogos y la interacción posible con el usuario. Todo esto lo define usando el lenguaje XMMVR propuesto. Así el desarrollador se abstrae de cuestiones de implementación dependientes de la plataforma sobre la que se ejecute la aplicación centrándose en lo verdaderamente importante y que da valor a la aplicación con independencia de la tecnología subyacente.

En definitiva, el objetivo es dotar al desarrollador de un lenguaje de especificación que aporte abstracción, modularidad, encapsulamiento, aislamiento, polimorfismo y herencia. La abstracción (Halbert & O'Brien, 1987) permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. Es clave en el proceso de análisis y diseño ya que mediante ella podemos modelar el problema que se quiere atacar. La modularidad (Moon, 1986) es la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos tienen identidad propia pero tienen conexiones con otros módulos. El encapsulamiento (Harrison & Ossher, 1993) supone reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema. El aislamiento (Szyperski, 2002) protege a las propiedades de un elemento contra su modificación por quien no tenga derecho a acceder a ellas. Esto asegura que otros elementos no pueden cambiar su estado interno de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas. Con el polimorfismo (Booch, Maksimchuk, Engle, Young, Conallen & Houston, 2007) comportamientos diferentes, asociados a elementos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al elemento que se esté usando. Con la herencia (Budd, 2001) los elementos no están aislados, sino que se relacionan entre sí, formando una jerarquía de clasificación. Así heredan las propiedades y el comportamiento de sus "ancestros". Decimos que organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los elementos ser definidos y creados como tipos especializados de elementos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo.

En las siguientes secciones se definen elementos del lenguaje para en la última sección defender la cobertura de las propiedades arriba mencionadas y teniendo en cuenta que en el capítulo 4 se presentarán ejemplos del lenguaje que reforzarán nuestra propuesta.

3.2 Conceptos y elementos de XMMVR

El eXtensible markup language for MultiModal interaction with Virtual Reality worlds o XMMVR es una propuesta de definición de un lenguaje de marcas para definir escena, comportamiento e interacción en el que considera cada mundo o película interactiva como un elemento XMMVR teniendo como base la metáfora de película cinematográfica. Se podría decir que es un lenguaje de marcas híbrido porque la idea es utilizar otros lenguajes tales como VoiceXML o X+V para interacción vocal y VRML, X3D, W3D u otro formato de archivo para especificación de imágenes 3D (descripción de escena y actores) que quedarían embebidos en éste. De esta manera, el procesamiento de los ficheros XML válidos para el DTD de XMMVR permite enlazar con los programas y ficheros necesarios para hacer funcionar el mundo especificado gracias a un mapping o mapeo de eventos, condiciones y acciones. Un sistema basado en XMMVR va a ser dirigido por eventos, por ello habrá que definir una mínima lista de eventos y no va a haber línea de tiempo (timeline based paradigm) (Bulterman & Hardman, 2005). El elemento XMMVR es el elemento básico y representa cada mundo o película interactiva. Está formado por el reparto de actores, elemento CAST, la secuencia de escenas que

marcan el transcurrir del mundo o elemento SEQUENCE y un contexto, elemento CONTEXT como se puede ver en la figura 3.1 y se define en el listado 3.1.

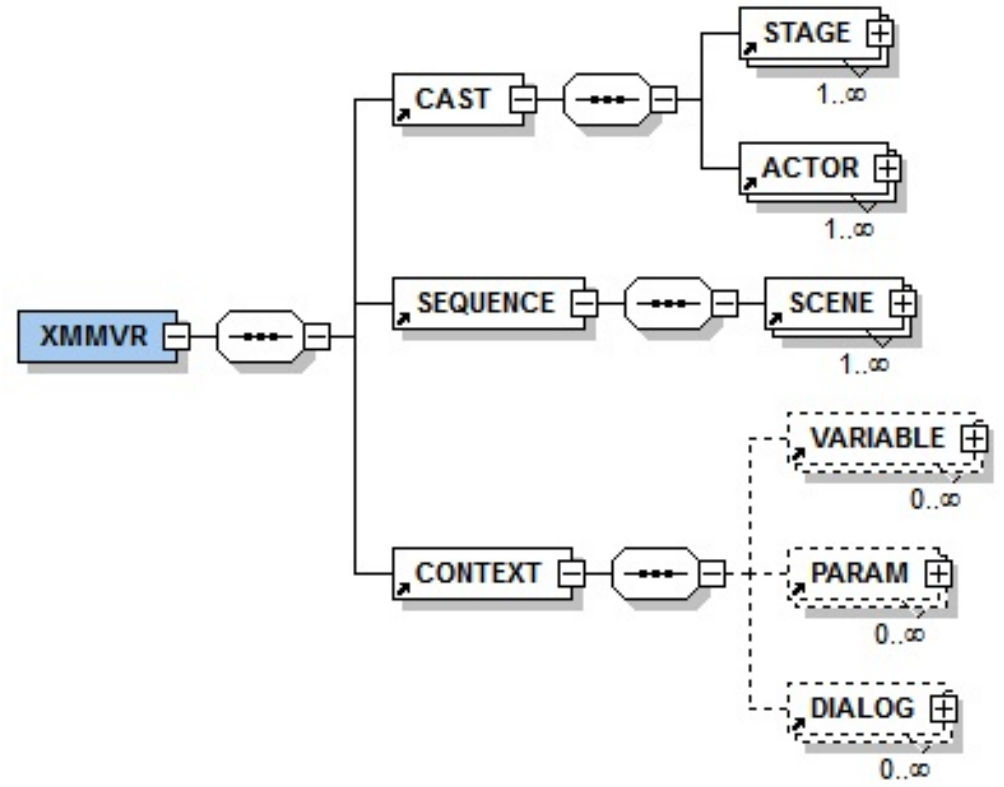

Figura 3.1: XMLSchema XMMVR: Elemento XMMVR

Listado 3.1: DTD XMMVR: Elemento XMMVR

```
1 <!ELEMENT XMMVR (CAST, SEQUENCE, CONTEXT)>
```

3.2.1. Elemento cast

El elemento CAST o reparto será el conjunto de actores ACTOR y de escenarios STAGE que intervendrán en el mundo o película XMMVR es decir, cada uno de los elementos que tienen una apariencia gráfica especificada por un fichero VRML, X3D o W3D y un comportamiento que permite una interacción con el usuario. Si bien siempre existirá al menos un actor en el mundo y un escenario, al usuario lo consideraremos como un espectador sin presencia en el mundo pero que interactúa con los actores de éste bien de forma vocal y, o gráfica. No obstante si incluyésemos un avatar del usuario en la escena, éste sería tratado como un actor más. Se puede ver el XMLSchema del elemento CAST en la figura 3.2 y su DTD en el listado 3.2.

Listado 3.2: DTD XMMVR: Elemento CAST

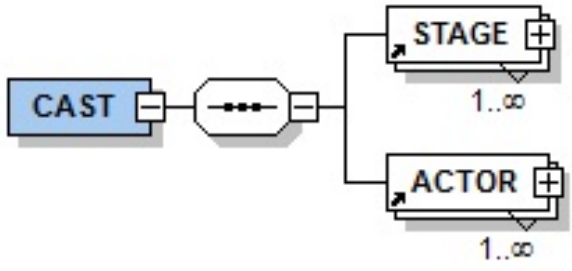

FIGURA 3.2: XMLSchema XMMVR: Elemento CAST

```
<!ELEMENT CAST (STAGE+, ACTOR+ )>
```

3.2.1.1. Elemento stage

El elemento STAGE representa cada uno de los posibles escenarios donde transcurrirá nuestra aplicación. El atributo STAGEID sería su identificador y el atributo FILE3D el fichero descriptivo en el formato correspondiente (VRML, X3D, W3D, etc.). Se puede ver el XMLSchema del elemento STAGE en la figura 3.3 y su DTD en el listado 3.3.

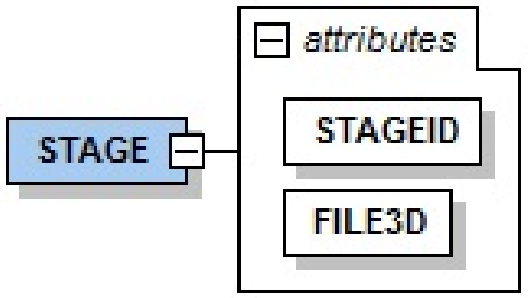

FIGURA 3.3: XMLSchema XMMVR: Elemento STAGE

LISTADO 3.3: DTD XMMVR: Elemento STAGE

```
<!ELEMENT STAGE EMPTY>
<!ATTLIST STAGE
          STAGEID    ID #REQUIRED
          FILE3D CDATA #REQUIRED>
```

3.2.1.2. Elemento actor

Cada uno de los elementos con apariencia que forman parte de la aplicación o mundo virtual. Un ACTOR es todo elemento que puede formar parte del mundo definido y que tendrá o no

comportamientos propios. El atributo ACTORID es el identificador del actor, FILE3D el fichero descriptivo en el formato correspondiente (VRML, X3D, W3D, etc.). El parámetro PARAMDECL puede usarse para definir propiedades que corresponden con un tipo de datos ”int”, ”real”, ”string”, ”vector” o ”color” indicado en la etiqueta TYPE, junto con un nombre y un valor indicados en las etiquetas NAME y VALUE respectivamente. El atributo METHODDECL se refiere a métodos o procedimientos asociados al actor y que podrán ser llamados por la plataforma de ejecución según el nombre indicado en su etiqueta NAME. Se puede ver el XMLSchema del elemento ACTOR en la figura 3.4 y su DTD en el listado 3.4.

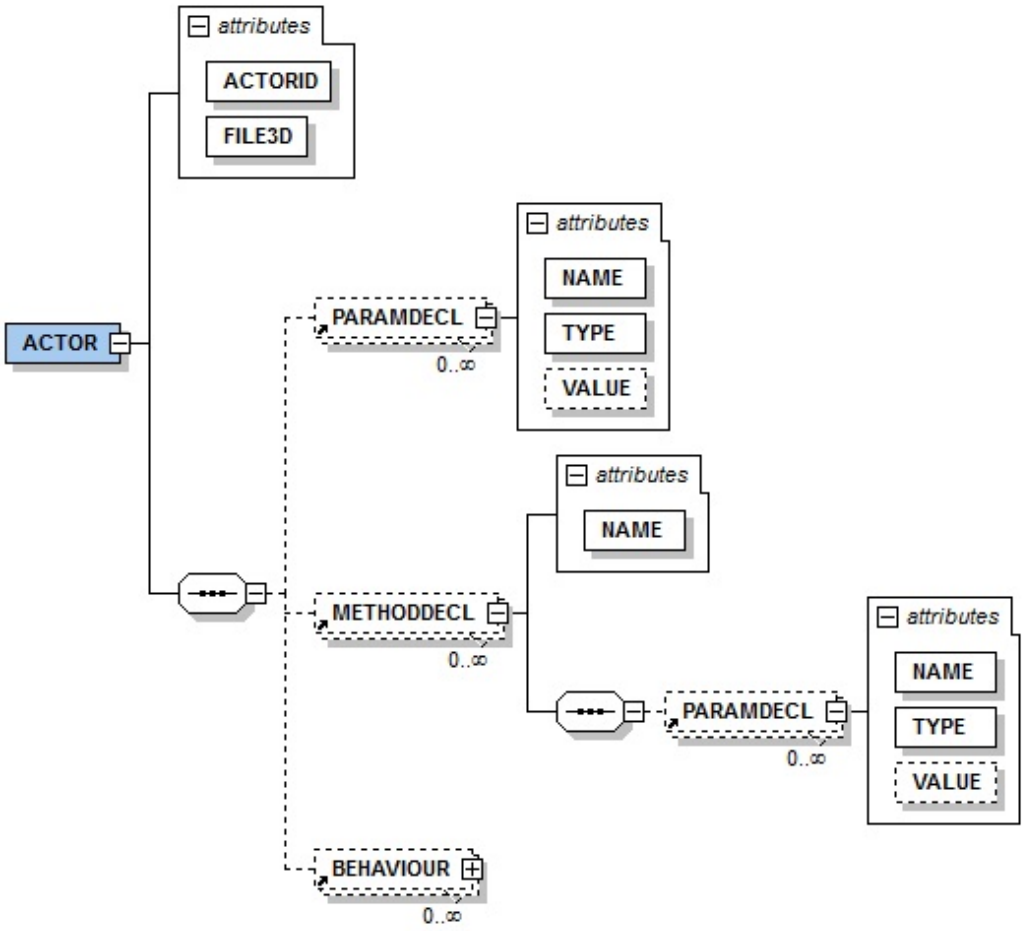

Figura 3.4: XMLSchema XMMVR: Elemento ACTOR

Listado 3.4: DTD XMMVR: Elemento ACTOR

```
<!ELEMENT ACTOR (PARAMDECL*, METHODDECL*,BEHAVIOUR*)>
<!ATTLIST ACTOR
          ACTORID ID #REQUIRED
          FILE3D CDATA #REQUIRED>
<!ELEMENT METHODDECL (PARAMDECL*)>
<!ATTLIST METHODDECL
          NAME CDATA #REQUIRED>
<!ELEMENT PARAMDECL EMPTY>
<!ATTLIST PARAMDECL
          NAME CDATA #REQUIRED
          TYPE (int|real|string|vector|color) #REQUIRED
          VALUE CDATA #IMPLIED>
```

3.2.2. Elemento sequence

Además del reparto, se ha de especificar la secuencia SEQUENCE de escenas SCENE que serán los escenarios donde ocurre la aplicación o mundo y que se presentarán por defecto en el orden en el que se escribieron. En todo mundo XMMVR se considera que ocurre al menos una escena SCENE. Se puede ver el XMLSchema del elemento SEQUENCE en la figura 3.5 y su DTD en el listado 3.5.

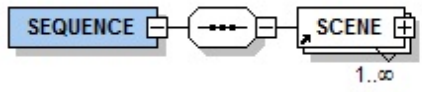

Figura 3.5: XMLSchema XMMVR: Elemento SEQUENCE

Listado 3.5: DTD XMMVR: Elemento SEQUENCE

```
<!ELEMENT SEQUENCE (SCENE+)>
```

3.2.2.1. Elemento scene

Cada escena SCENE ha de tener un escenario STAGE de los indicados en el CAST (apartado 3.2.1). Además habrá uno o varios elementos ACTORINSTANCE. El atributo STAGEID referencia al identificador del escenario, BEHAVIOUR es el comportamiento asociado a la escena y podrán existir ninguno, uno o varios. ACTORINSTANCE son cada una de las ocurrencias de un determinado actor del cast definidos en la escena y existirán ninguno, uno o varios. Se puede ver el XMLSchema del elemento SCENE en la figura 3.6 y su DTD en el listado 3.6.

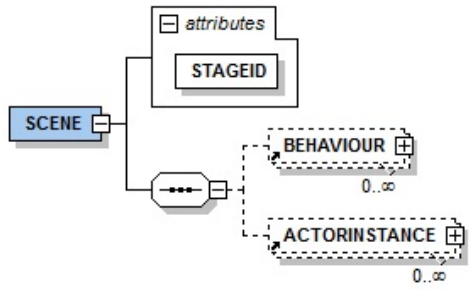

Figura 3.6: XMLSchema XMMVR: Elemento SCENE

Listado 3.6: DTD XMMVR: Elemento SCENE

```
<!ELEMENT SCENE (BEHAVIOUR*, ACTORINSTANCE*)>
<!ATTLIST SCENE
          STAGEID    IDREF #REQUIRED>
```

3.2.2.1.1. Elemento actorinstance Un ACTORINSTANCE sería un modelo de actor con propiedades y comportamientos comunes a los actores que lo implementan. Es similar al concepto de herencia en las clases de orientación a objetos ya que podríamos decir que un actor de un tipo ACTORINSTANCE hereda las propiedades y comportamientos del ACTOR y añade los suyos propios. El atributo ACTORINSTANCEID es el identificador del ACTORINSTANCE y ACTORID el identificador del ACTOR asociado en el CAST (apartado 3.2.1), PARAM se utiliza para definir parámetros propios del ACTORINSTANCE pudiendo haber ninguno, uno o varios que podrán tener un nombre y un valor etiquetados como NAME y VALUE respectivamente. Existirá o no al menos un elemento BEHAVIOUR que representa el comportamiento asociado al ACTORINSTANCE aunque pudieran existir más. Se puede ver el XMLSchema del elemento ACTORINSTANCE en la figura 3.7 y su DTD en el listado 3.7.

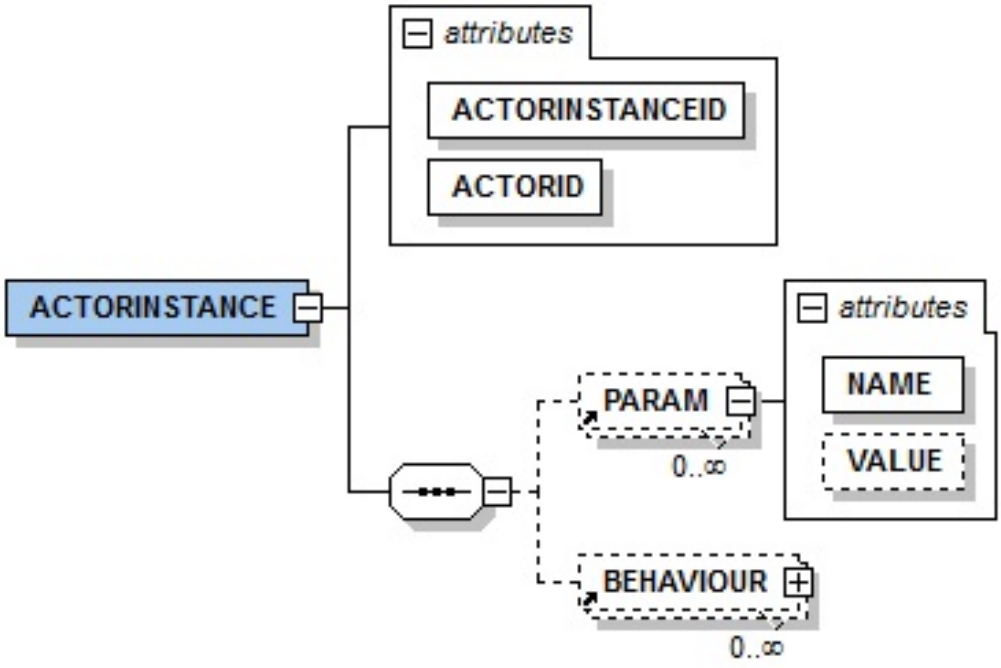

Figura 3.7: XMLSchema XMMVR: Elemento ACTORINSTANCE

Listado 3.7: DTD XMMVR: Elemento ACTORINSTANCE

```
<!ELEMENT ACTORINSTANCE (PARAM*, BEHAVIOUR*)>
<!ATTLIST ACTORINSTANCE
          ACTORINSTANCEID ID #REQUIRED
          ACTORID   IDREF #REQUIRED>
```

3.2.3. Elemento behaviour

Cada elemento BEHAVIOUR representará el comportamiento de cada ACTORINSTANCE o del escenario SCENE es decir, las acciones que realiza ante un determinado evento y en una determinada condición que puede existir o no. Se definirá como una pareja de evento y lista de acciones que puede tener cada ACTORINSTANCE o escena SCENE. Así BEHAVIOURID es el identificador del comportamiento que tendrá un evento asociado "event" y ACTIONBLOQ es el bloque de acciones a ejecutarse ante dicho evento cuando se cumple la condición COND asociada y definida dentro del ACTIONBLOQ. Se puede ver el XMLSchema del elemento behaviour en la figura 3.8 y su DTD en el listado 3.8.

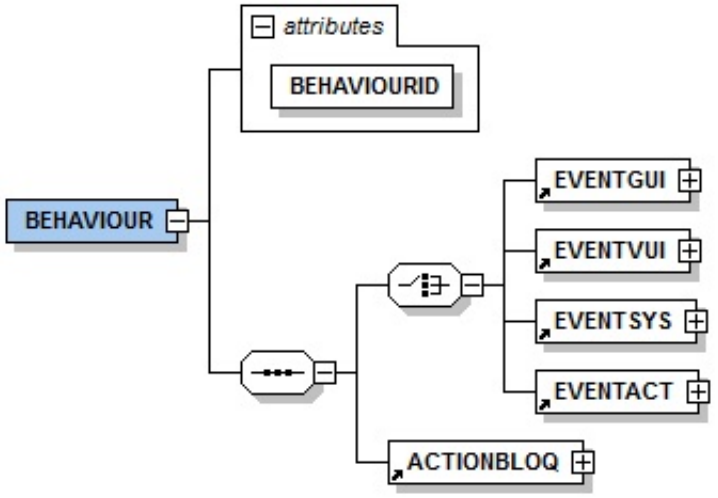

Figura 3.8: XMLSchema XMMVR: Elemento behaviour

Listado 3.8: DTD XMMVR: Elemento behaviour

```
1 <!ELEMENT BEHAVIOUR ((EVENTGUI|EVENTVUI|EVENTSYS|EVENTACT), ACTIONBLOQ)>
  <!ATTLIST BEHAVIOUR
            BEHAVIOURID CDATA #REQUIRED>
```

3.2.3.1. Elemento event

El elemento event puede ser EVENTGUI si es gráfico y tendrá un atributo "TYPE" que indicará que corresponde a un click de ratón "mouseDown", a que una tecla de cursor se ha pulsado ("arrowDOWN", "arrowUP", "arrowLEFT", "arrowRIGHT"), a que se ha producido una colisión entre actores "collision" o a que se acaba de entrar en escena "onEnter". Será un evento EVENTVUI si es vocal, pudiendo ser el lanzamiento de un diálogo "launchedDialog" o el fin de un diálogo "endedDialog". Cuando es un evento del sistema será EVENTSYS que podrá ser un inicio de la escena "onEnterScene" o una salida de la escena "onQuitScene". Si se ha generado un evento por uno de los actores en la escena, será un EVENTACT que tendrá como atributo el identificador del ACTORINSTANCE implicado o ACTORINSTANCEID.

Se puede ver el XMLSchema del elemento EVENT en la figura 3.9 y su DTD en el listado 3.9.

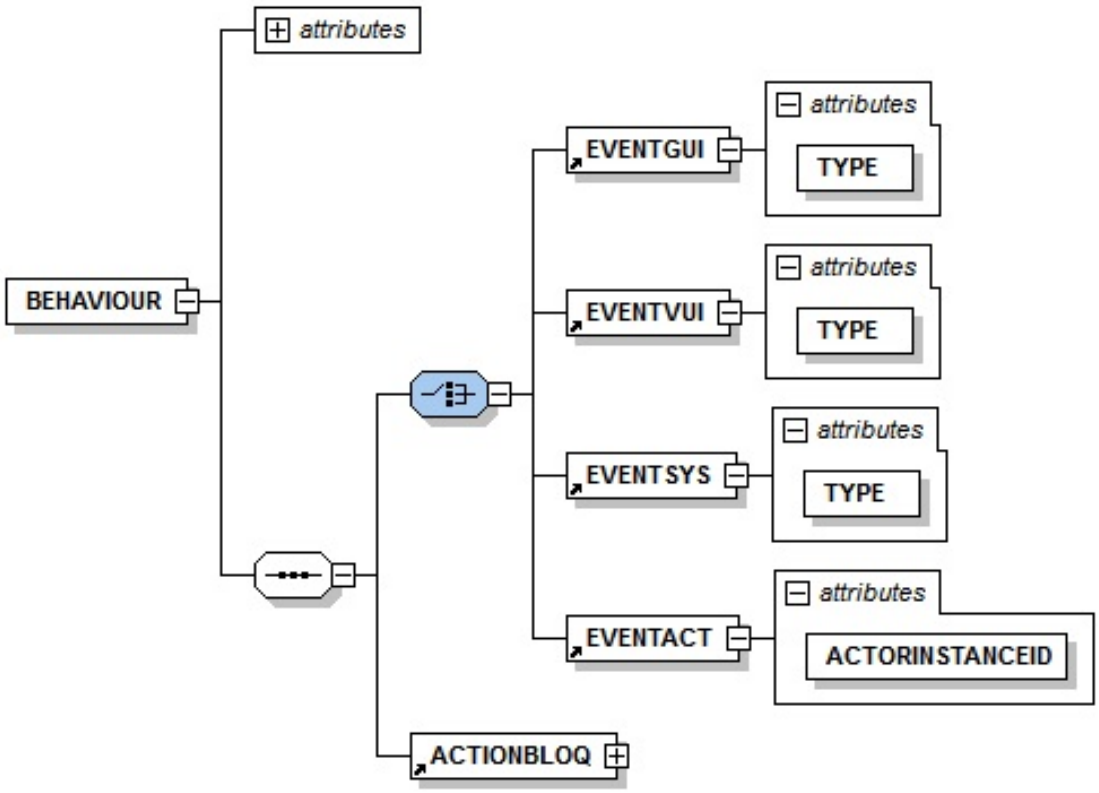

Figura 3.9: XMLSchema XMMVR: Elemento EVENT

Listado 3.9: DTD XMMVR: Elemento EVENT

```
<!ELEMENT EVENTGUI EMPTY>
<!ATTLIST EVENTGUI
    TYPE (mouseDown|arrowUP|arrowDOWN|arrowLEFT|arrowRIGHT|collision|onEnter) #REQUIRED>
<!ELEMENT EVENTVUI EMPTY>
<!ATTLIST EVENTVUI
    TYPE (launchedDialog|endedDialog) #REQUIRED>
<!ELEMENT EVENTSYS EMPTY>
<!ATTLIST EVENTSYS
    TYPE (onEnterScene|onQuitScene) #REQUIRED
    TIMERID CDATA #IMPLIED>
<!ELEMENT EVENTACT EMPTY>
<!ATTLIST EVENTACT
    ACTORINSTANCEID IDREF #REQUIRED>
```

3.2.3.2. Elemento actionbloq

El elemento ACTIONBLOQ tiene al menos un elemento ACTION y un elemento COND que especifica la condición que se ha de cumplir al producirse el evento EVENT asociado para que se ejecuten las acciones del ACTIONBLOQ. Se puede ver el XMLSchema del elemento ACTIONBLOQ en la figura 3.10 y su DTD en el listado 3.10.

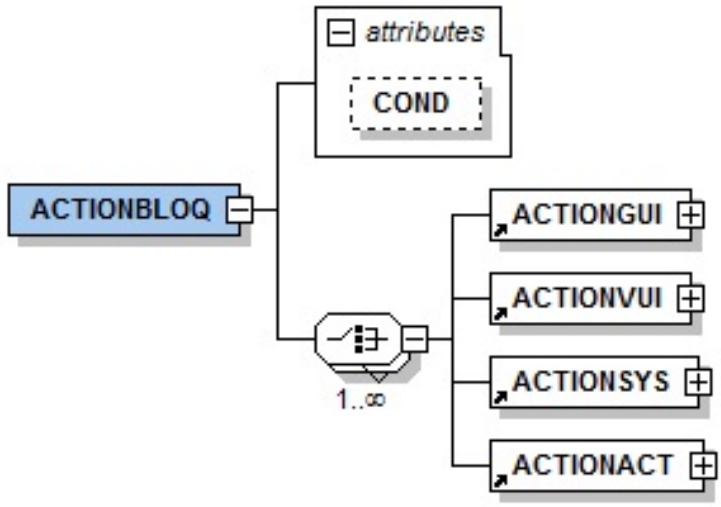

Figura 3.10: XMLSchema XMMVR: Elemento ACTIONBLOQ

Listado 3.10: DTD XMMVR: Elemento ACTIONBLOQ

```
<!ELEMENT ACTIONBLOQ (ACTIONGUI|ACTIONVUI|ACTIONSYS|ACTIONACT)+>
<!ATTLIST ACTIONBLOQ
  COND CDATA #IMPLIED>
```

3.2.3.2.1. Elemento action El elemento action puede ser ACTIONGUI, ACTIONVUI, ACTIONSYS o ACTIONACT dependiendo de la naturaleza de la acción es decir, según sea una acción de tipo gráfico, vocal, de sistema o de interacción entre actores, respectivamente. Cuando la acción sea de tipo ACTIONGUI podrá tener o no, uno o varios parámetros PARAM, un atributo TYPE que podrá ser de tipo ”callback”, un atributo DESTINATION y un atributo MESSAGE. Si la acción es de tipo ACTIONVUI podrá tener o no, uno o varios parámetros PARAM, un atributo TYPE que podrá ser de tipo ”launchDialog” o de tipo ”stopDialog” (lanzar diálogo o parar diálogo respectivamente) y el identificador del diálogo asociado DIALOGID que se relacionará con un fichero de diálogo según se indique en el elemento CONTEXT. En el caso de acción de tipo ACTIONSYS podrá tener o no, uno o varios elementos ASSIGN que son pares NAME, VALUE si el atributo TYPE tiene el valor ”assign”. Además, según el atributo TYPE podrán definirse otros atributos que son: DESTINATION, TIME, TIMERID asociado al atributo TYPE con valor ”setTimer”; PASSING y ROUTE asociado al atributo TYPE con valor ”routing”; DESTINATION asociado al atributo ”goToScene”. Finalmente cuando la acción es de tipo ACTIONACT podrá tener o no, uno o varios parámetros PARAM (NAME, VALUE), el ACTORINSTANCEID asociado a la instancia de actor con la que va a interactuar y el METHOD correspondiente que es el nombre del método a ejecutar sobre dicha instancia de actor. Se puede ver el XMLSchema del elemento ACTION en las figuras 3.11 y 3.12 y su DTD en el listado 3.11.

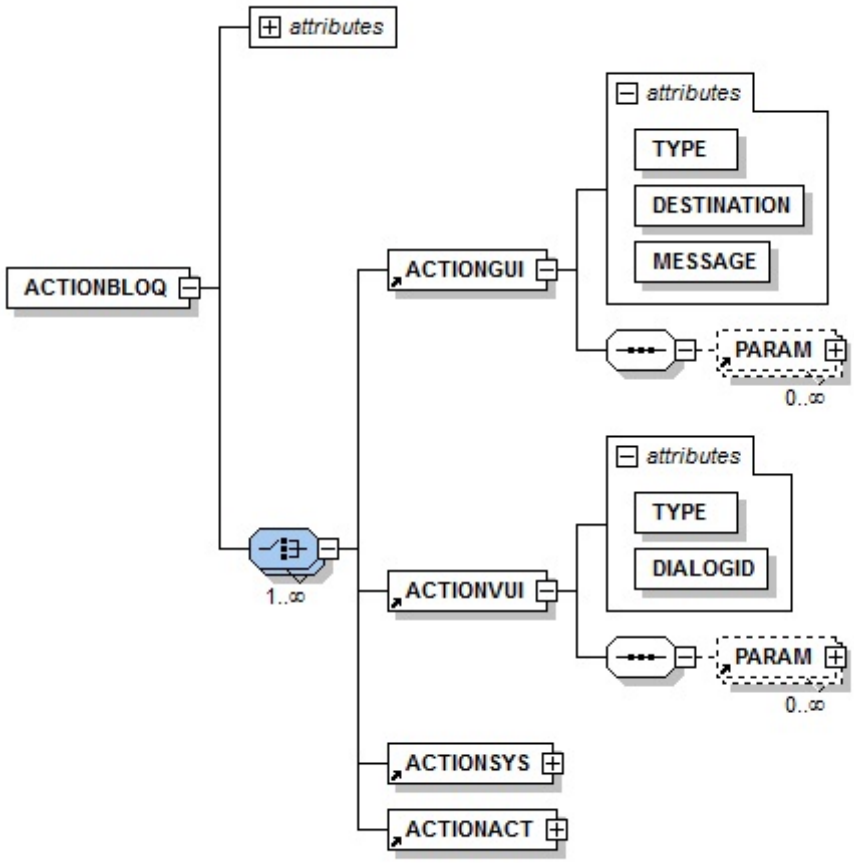

Figura 3.11: XMLSchema XMMVR: Elemento ACTION (parte 1)

Listado 3.11: DTD XMMVR: Elemento ACTION

```
<!ELEMENT ACTIONGUI (PARAM*)>
<!ATTLIST ACTIONGUI
            TYPE (callback) #REQUIRED
            DESTINATION CDATA #REQUIRED
            MESSAGE CDATA #REQUIRED>
<!ELEMENT ACTIONVUI (PARAM*)>
<!ATTLIST ACTIONVUI
   TYPE (launchDialog| stopDialog) #REQUIRED
   DIALOGID CDATA #REQUIRED>
<!ELEMENT ACTIONSYS (ASSIGN*)>
<!ATTLIST ACTIONSYS
   TYPE (assign|goToScene|quitScene|openWindow|trigger|routing|setTimer) #REQUIRED
   DESTINATION CDATA #IMPLIED
   TIME CDATA #IMPLIED
   TIMERID CDATA #IMPLIED
   PASSING CDATA #IMPLIED
   ROUTE CDATA #IMPLIED>
<!ELEMENT ASSIGN EMPTY>
<!ATTLIST ASSIGN
          NAME CDATA #REQUIRED
          VALUE CDATA #IMPLIED>
<!ELEMENT ACTIONACT (PARAM*)>
<!ATTLIST ACTIONACT
  ACTORINSTANCEID CDATA #IMPLIED
  METHOD CDATA #IMPLIED>
```

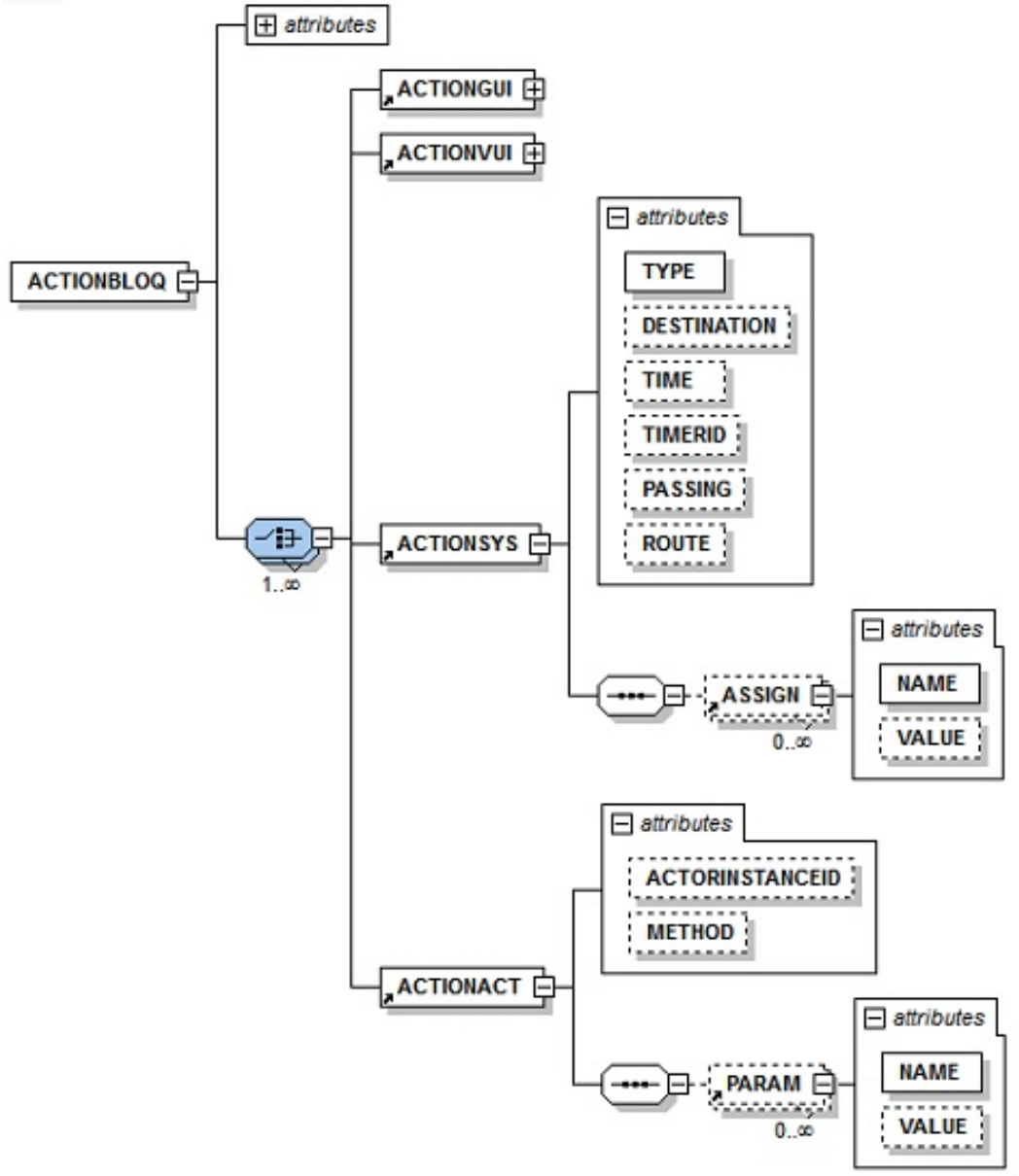

Figura 3.12: XMLSchema XMMVR: Elemento ACTION (parte 2)

3.2.3.2.2. Elemento cond En este elemento se especifican las condiciones que han de cumplirse para que se produzcan las acciones del ACTIONBLOQ ante un determinado evento. Se sigue una notación basada en ECMAScript (ECMA International, 2010) para poder reutilizar expresiones y conseguir que los ficheros XMMVR y VoiceXML sean más flexibles. Por ejemplo en la expresión 3.12 se indica que si el parámetro de salida "action" del diálogo es "go", el parámetro de salida "param1" de ese mismo diálogo es "first" y el parámetro de salida "param2" es "pedro", se ejecuten las acciones del ACTIONBLOQ.

Listado 3.12: Ejemplo de elemento COND

```
ACTIONBLOQ COND="$OUTDIALOG[action] ==?go? AND $OUTDIALOG[param1] ==?first? AND $OUTDIALOG[param2
    ] ==?pedro?">
```

3.2.4. Elemento context

Además del reparto y la secuencia de escenas, tenemos un elemento CONTEXT o contexto donde definimos VARIABLE, PARAM y DIALOG que son respectivamente variables, parámetros y diálogos propios de nuestro sistema. Cada elemento VARIABLE tendrá un nombre NAME que la identificará, será de un tipo TYPE y acumulará un valor VALUE. Se utilizarán para referenciar variables a las que se asignarán distintos valores durante la ejecución de nuestras aplicaciones. Los valores VALUE que podemos almacenar en los elementos VARIABLE podrán ser "int", "real", "string", "vector" o "color" según lo indiquemos en el atributo TYPE. Los elementos PARAM son pares nombre NAME y valor VALUE que referenciarán a parámetros a ser utilizados durante la ejecución de nuestras aplicaciones. Los elementos DIALOG podrán tener o no varios parámetros de entrada INPUT_PARAM o de salida OUTPUT_PARAM y siempre un identificador DIALOG_ID y un fichero con el código correspondiente SPEECHFILE en el lenguaje de especificación de diálogo elegido (VoiceXML, X+V, etc.). Se puede ver el XMLSchema del elemento CONTEXT en la figura 3.13 y su DTD en el listado 3.13.

Listado 3.13: DTD XMMVR: Elemento CONTEXT

```
<!ELEMENT CONTEXT (VARIABLE*,PARAM*,DIALOG*)>
<!ELEMENT VARIABLE EMPTY>
<!ATTLIST VARIABLE
          NAME CDATA #REQUIRED
          TYPE (int|real|string|vector|color) #REQUIRED
          VALUE CDATA #IMPLIED>
<!ELEMENT PARAM EMPTY>
<!ATTLIST PARAM
          NAME CDATA #REQUIRED
          VALUE CDATA #IMPLIED>
<!ELEMENT DIALOG (INPUT_PARAM*,OUTPUT_PARAM*)>
<!ATTLIST DIALOG
                DIALOG_ID CDATA #REQUIRED
        SPEECHFILE CDATA #REQUIRED>
 <!ELEMENT OUTPUT_PARAM EMPTY>
 <!ATTLIST OUTPUT_PARAM
                NAME CDATA #REQUIRED>
 <!ELEMENT INPUT_PARAM EMPTY>
 <!ATTLIST INPUT_PARAM
                NAME CDATA #REQUIRED>
```

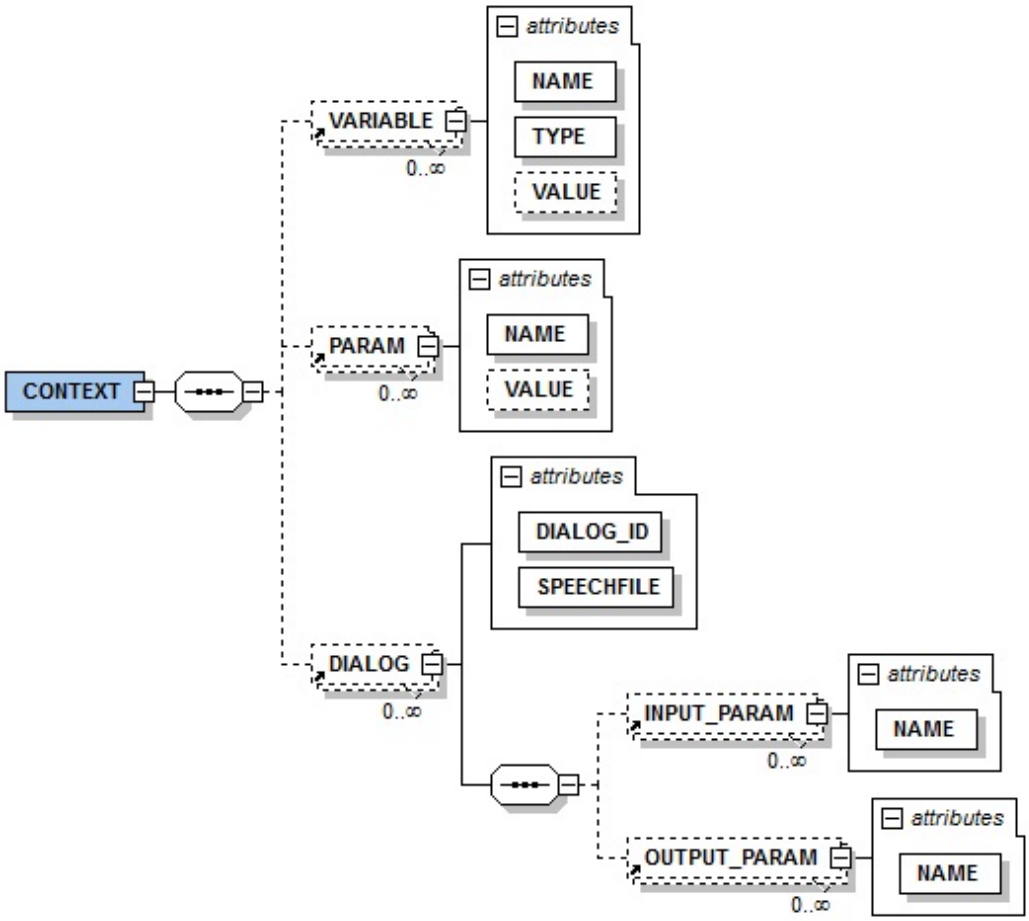

Figura 3.13: XMLSchema XMMVR: Elemento CONTEXT

3.3 Conclusiones

En este capítulo partiendo de que toda aplicación basada en mundos 3D donde se permite una interacción multimodal se basa en unos mismos principios, se ha descrito nuestro lenguaje XMMVR que se compone de distintos elementos para especificar los mundos 3D y las posibilidades de interacción multimodal que se le dan al usuario. Para ello se ha descrito cada uno de estos elementos, sus representaciones en formato DTD y sus XMLSchemas.

Vista la descripción del lenguaje se puede decir que se han cumplido los objetivos marcados. La abstracción se ha conseguido gracias a la utilización de la metáfora en la que nos basamos permitiéndonos modelar el problema que se quiere atacar. Al desarrollador se le ocultan explícitamente cuestiones prácticas que tienen que ver con la gestión de los diálogos o el control de los actores que pasan a depender de la plataforma concreta que interprete el documento XMMVR. La modularidad es obvia porque las distintas secciones se aíslan en forma de reparto de actores, secuencia de escenas, etc. haciendo que en futuras aplicaciones sea posible la reutilización de cualquiera de estos elementos. El encapsulamiento se observa en las agrupaciones de elementos definidas situándolos en el mismo nivel de abstracción. A su vez se produce un aislamiento de los elementos ya que otros elementos no pueden cambiar su estado interno de manera inesperada, únicamente a través de los eventos definidos. Al permitir que comportamientos diferentes, asociados a elementos distintos, puedan compartir el mismo nombre conseguimos el objetivo de polimorfismo. Finalmente, la herencia se aprecia en la definición de "ACTORINSTANCE" que son instancias de actores ACTOR que heredan las propiedades y el comportamiento de sus ancestros.

Todos estos objetivos se plasman en el DTD definido que se muestra en el listado 3.14. Pero también se va a definir cómo se enlaza todo esto definiendo los requisitos, la arquitectura y el control de flujo de la plataforma de implementación de la aplicación basada en el lenguaje de especificación de escena, comportamiento e interacción definido que describiremos en el capítulo 5.

En el capítulo 4 se verá cómo el lenguaje definido sirve para representar distintos ejemplos que a su vez ilustrarán cada una de las metáforas de interacción gráfica y vocal así como los posibles casos de cooperación entre modalidades descritos en el capítulo 2.

Listado 3.14: DTD XMMVR

```
<!ELEMENT XMMVR (CAST, SEQUENCE, CONTEXT)>
<!ELEMENT CAST (STAGE+, ACTOR+)>
<!ELEMENT SEQUENCE (SCENE+)>
<!ELEMENT SCENE (BEHAVIOUR*, ACTORINSTANCE*)>
<!ATTLIST SCENE
          STAGEID    IDREF #REQUIRED>
<!ELEMENT STAGE EMPTY>
<!ATTLIST STAGE
          STAGEID    ID #REQUIRED
          FILE3D CDATA #REQUIRED>
<!ELEMENT ACTOR (PARAMDECL*, METHODDECL*, BEHAVIOUR*)>
<!ATTLIST ACTOR
          ACTORID ID #REQUIRED
          FILE3D CDATA #REQUIRED>
<!ELEMENT METHODDECL (PARAMDECL*)>
<!ATTLIST METHODDECL
          NAME CDATA #REQUIRED>
<!ELEMENT PARAMDECL EMPTY>
<!ATTLIST PARAMDECL
          NAME CDATA #REQUIRED
          TYPE (int|real|string|vector|color) #REQUIRED
          VALUE CDATA #IMPLIED>
<!ELEMENT ACTORINSTANCE (PARAM*, BEHAVIOUR*)>
<!ATTLIST ACTORINSTANCE
          ACTORINSTANCEID ID #REQUIRED
          ACTORID   IDREF #REQUIRED>
<!ELEMENT BEHAVIOUR ((EVENTGUI|EVENTVUI|EVENTSYS|EVENTACT), ACTIONBLOQ)>
<!ATTLIST BEHAVIOUR
          BEHAVIOURID CDATA #REQUIRED>
<!ELEMENT EVENTGUI EMPTY>
<!ATTLIST EVENTGUI
          TYPE (mouseDown|arrowUP|arrowDOWN|arrowLEFT|arrowRIGHT|collision|onEnter) #REQUIRED>
<!ELEMENT EVENTVUI EMPTY>
<!ATTLIST EVENTVUI
    TYPE (launchedDialog|endedDialog) #REQUIRED>
<!ELEMENT EVENTSYS EMPTY>
<!ATTLIST EVENTSYS
    TYPE (onEnterScene|onQuitScene) #REQUIRED>
<!ELEMENT EVENTACT EMPTY>
<!ATTLIST EVENTACT
    ACTORINSTANCEID IDREF #REQUIRED>
<!ELEMENT ACTIONBLOQ (ACTIONGUI|ACTIONVUI|ACTIONSYS|ACTIONACT)+>
<!ATTLIST ACTIONBLOQ
   COND CDATA #IMPLIED>
<!ELEMENT ACTIONGUI (PARAM*)>
<!ATTLIST ACTIONGUI
          TYPE (callback) #REQUIRED
          DESTINATION CDATA #REQUIRED
          MESSAGE CDATA #REQUIRED>
<!ELEMENT ACTIONVUI (PARAM*)>
<!ATTLIST ACTIONVUI
    TYPE (launchDialog| stopDialog) #REQUIRED
    DIALOGID CDATA #REQUIRED>
<!ELEMENT ACTIONSYS (ASSIGN*)>
<!ATTLIST ACTIONSYS
    TYPE (assign|goToScene|quitScene|openWindow|trigger|routing|setTimer) #REQUIRED
    DESTINATION CDATA #IMPLIED
    TIME CDATA #IMPLIED
    TIMERID CDATA #IMPLIED
    PASSING CDATA #IMPLIED
    ROUTE CDATA #IMPLIED>
<!ELEMENT ASSIGN EMPTY>
<!ATTLIST ASSIGN
          NAME CDATA #REQUIRED
          VALUE CDATA #IMPLIED>
<!ELEMENT ACTIONACT (PARAM*)>
<!ATTLIST ACTIONACT
   ACTORINSTANCEID CDATA #IMPLIED
   METHOD CDATA #IMPLIED>
<!ELEMENT CONTEXT (VARIABLE*,PARAM*,DIALOG*)>
<!ELEMENT VARIABLE EMPTY>
<!ATTLIST VARIABLE
          NAME CDATA #REQUIRED
          TYPE (int|real|string|vector|color) #REQUIRED
          VALUE CDATA #IMPLIED>
<!ELEMENT PARAM EMPTY>
<!ATTLIST PARAM
          NAME CDATA #REQUIRED
          VALUE CDATA #IMPLIED>
<!ELEMENT DIALOG (INPUT_PARAM*,OUTPUT_PARAM*)>
<!ATTLIST DIALOG
                DIALOG_ID CDATA #REQUIRED
        SPEECHFILE CDATA #REQUIRED>
 <!ELEMENT OUTPUT_PARAM EMPTY>
 <!ATTLIST OUTPUT_PARAM
                NAME CDATA #REQUIRED>
 <!ELEMENT INPUT_PARAM EMPTY>
 <!ATTLIST INPUT_PARAM
                NAME CDATA #REQUIRED>
```

4

Uso de XMMVR y cobertura de metáforas

En este capítulo expondremos las capacidades del lenguaje propuesto para implementar cada una de la metáforas gráficas y vocales desarrolladas en el capítulo 2 así como las distintas posibilidades de cooperación entre ambas. Esto demostrará la capacidad del lenguaje propuesto para describir sistemas basados en aplicaciones que permitan una interacción multimodal con espacios virtuales. Para demostrar estas capacidades de nuestro lenguaje utilizaremos un ejemplo soporte que describiremos a continuación ya que es una forma práctica de dar a conocer las metáforas descritas, la cooperación entre modalidades y las capacidades de nuestro lenguaje para especificar todo ello.

4.1 Ejemplo soporte: Museo 3D Multimodal

Consideramos que un Museo 3D Multimodal es una buena plataforma para experimentar con XMMVR y la interacción multimodal porque ofrece escenarios potenciales para cada una de las metáforas de interacción gráfica, interacción vocal y cooperación entre ambas. En la actualidad existen multitud proyectos de museos virtuales de la informática en Internet, realizados de muy diversas maneras, conteniendo algunos de ellos visitas virtuales a partir de fotos y también con recorridos utilizando Adobe Flash (Adobe, 2010b). Pero son muy pocos los museos virtuales de informática que se adentran en la utilización de entornos tridimensionales y menos los que permiten una interacción vocal. El nuestro es un museo virtual de informática realizado con los programas Autodesk 3D Studio Max (Autodesk, 2010) y Adobe Director (Adobe, 2010a) principalmente. Se basa en el museo real de la Informática de la UVa situado en la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Valladolid que dispone de una extensa colección de las máquinas y periféricos más destacados en la historia de la informática expuestos en vitrinas que permiten que sean observados a la vez que están protegidos. La actual web del Museo de la Informática de la UVa, contiene información de todas las máquinas y periféricos que están expuestos en el museo real, conteniendo sus características técnicas, fotografías de los mismos y enlaces con más información de estos componentes. Además es posible realizar visitas virtuales que nos permiten ver los componentes informáticos de este museo en tres dimensiones, pudiendo interactuar con ellos para poder ver todos sus lados. Este museo virtual de informática es un edificio en 3D creado específicamente para exponer los componentes informáticos pertenecientes al museo. Fue necesario modelar un museo tridimensional y todos los componentes informáticos que serán expuestos en él. Para el modelado del museo y de los componentes informáticos se utilizó el programa Autodesk 3D Studio Max. La creación de la aplicación para las visitas virtuales se realizó a partir de los modelos en 3D diseñados con Autodesk 3D Studio Max y se utilizó el programa Adobe Director. Se crearon así escenas para las visitas virtuales en las se programaron unos comportamientos que permiten al usuario interactuar con los objetos 3D expuestos así como con el propio museo, permitiendo que el

usuario se desplace por las habitaciones y pasillos de éste. Para la realización de las visitas virtuales se utiliza Adobe Director Shockwave Player (Adobe, 2010c) que será el motor 3D que permita visionar las visitas virtuales del museo en Internet.

FIGURA 4.1: Objeto expuesto en el museo virtual de informática

Las fases básicas del diseño de esta aplicación ejemplo fueron (Sanz Prieto, 2009):

- Modelado en 3D de los objetos informáticos pertenecientes al Museo de la Informática de la UVa con el programa Autodesk 3D Studio Max. Un ejemplo se ve en la figura 4.1.
- Modelado en 3D de un edificio que realizara la función de museo virtual de informática con Autodesk 3D Studio Max. En este enlace (Olmedo-Rodríguez, 2010c) se puede ver un video donde se aprecian las características del edificio que alberga el museo virtual de informática.
- Desarrollo de la aplicación que permita interactuar con los componentes 3D modelados, así como con el museo modelado. Ésta se realiza utilizando el programa de Adobe Director y su aspecto se muestra en la figura 4.2.

El resultado de todo esto se puede ver en la URL siguiente: (Sanz Prieto, 2010) donde se puede interactuar de manera gráfica con el museo virtual de informática utilizando las teclas del cursor para desplazarse por sus pasillos y habitaciones permitiendo así observar las piezas exhibidas. Este escenario es ideal para demostrar la capacidad de XMMVR para definir nuevos comportamientos del museo y sus objetos ante los eventos gráficos y/o vocales emitidos por el usuario que mostraremos en el siguiente apartado ya que permite imaginar situaciones donde aparecerán las metáforas y la cooperación entre modalidades descritas. Para simplificar nuestros ejemplos hemos modelado el escenario común de todos ellos que será un museo simplificado que en la planta baja "Level0" tendrá tres habitaciones "Room0", "Room1" y "Room2", la entrada y un pasillo central "Corridor" que las une y en el que se encuentra un ascensor "Elevator" que nos moverá a las plantas primera "Level1" y segunda "Level2". Se muestra este escenario en la figura 4.3.

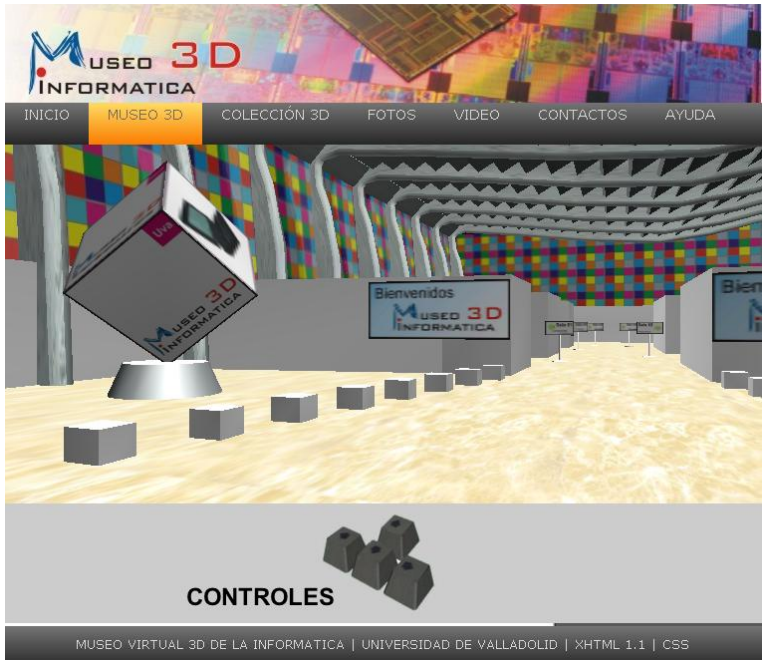

Figura 4.2: Aspecto de la aplicación del museo virtual de informática

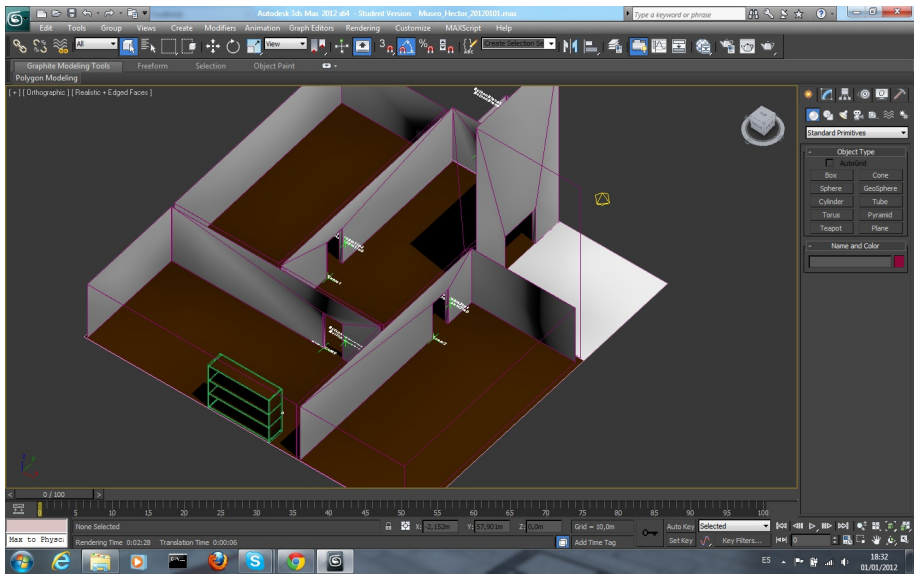

Figura 4.3: Vista del museo simplificado basado en la aplicación del museo virtual de informática

Expondremos ejemplos codificados en XMMVR para cada caso. Para simplificarlos sólo mostraremos las partes más relevantes del código en este capítulo y haremos referencia a los listados completos que se pueden ver en el apéndice A junto a los ficheros VoiceXML asociados.

4.2 Implementación de metáforas gráficas

Para demostrar la capacidad de XMMVR para especificar las metáforas de interacción gráfica, implementaremos casos concretos de nuestro ejemplo soporte. Pero antes recordemos las metáforas fundamentales de interacción gráfica que resumimos en la tabla 4.1 y que ya se presentaron en el capítulo 2.

Tabla 4.1: Metáforas fundamentales de interacción gráfica

Metáfora de teatro	El usuario tiene una posición estática y un punto de vista donde el mundo cambia a su alrededor
Metáfora de locomoción	El usuario tiene una posición dinámica y será trasladado a través de una estructura

4.2.1. Metáfora de teatro

El usuario tiene una representación estática del mundo, por lo que no se mueve la cámara. Para recrear esta metáfora en nuestro ejemplo soporte hemos definido un sencillo escenario. En éste se verá la sala principal del museo donde hay un número de piezas expuestas. El usuario podrá mover el puntero del ratón por la escena y seleccionar una de las piezas haciendo click sobre ella. En ese momento la pieza empezará a girar para que se puedan apreciar sus detalles.

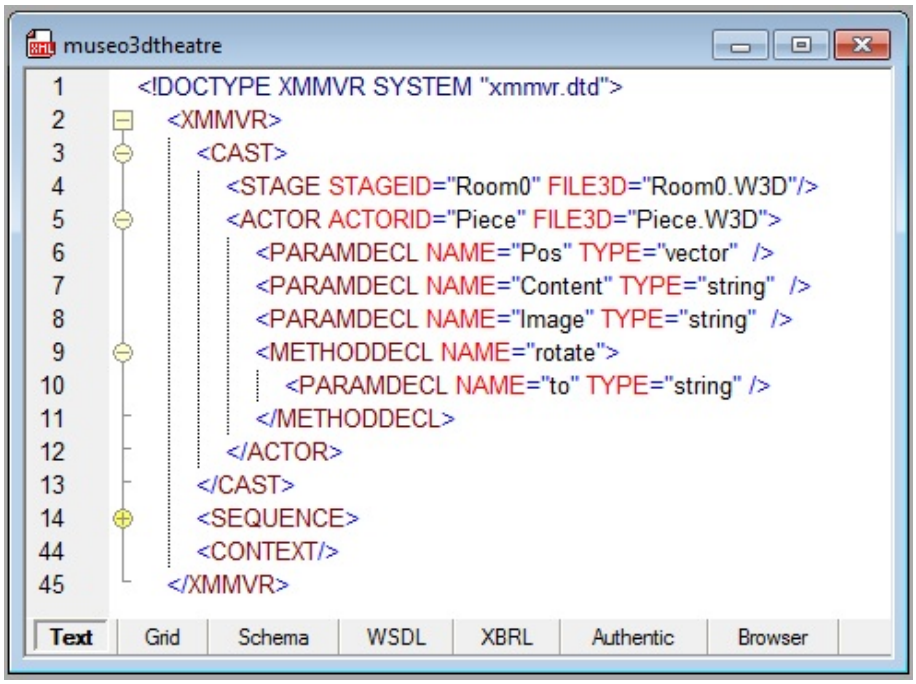

Figura 4.4: CAST de un ejemplo basado en la metáfora de teatro

Para ello se define un escenario STAGE cuyo identificador ”Room0” (línea 4 de la figura 4.4 y del listado A.1) hace referencia al lugar donde transcurre la escena es decir, el museo cuyos gráficos

3D están en el fichero ”Room0.W3D” referenciado con el parámetro ”FILE3D”. También se define un actor ACTOR cuyo identificador es ”Piece” (línea 5 de la figura 4.4 y del listado A.1) y sus gráficos 3D están en el fichero ”Piece.W3D” referenciado con el parámetro ”FILE3D”. Tiene tres parámetros que son ”Pos”, ”Content” e ”Image” que almacenarían la posición, una descripción y una imagen o textura asignada al actor, respectivamente. Además se ha definido un método ”rotate” que provoca que la pieza gire.

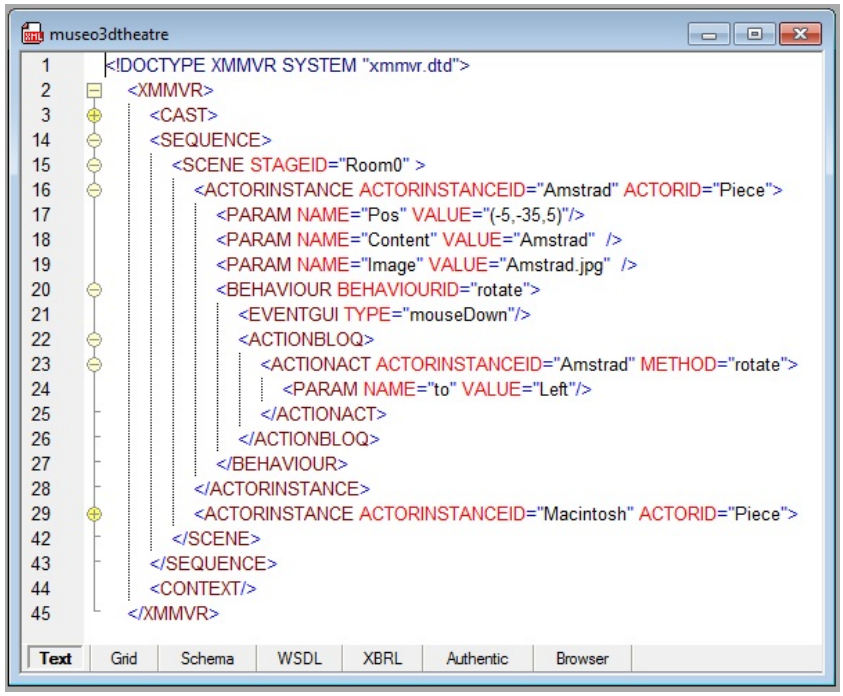

Figura 4.5: SCENE de un ejemplo basado en la metáfora de teatro

Definimos una secuencia SEQUENCE con una única escena SCENE basada en el escenario ”Room0” (línea 15 de la figura 4.5 y del listado A.1) en la que aparecen dos ACTORINSTANCE ”Amstrad” y ”Macintosh” (líneas 16 y 29 respectivamente de la figura 4.5 y del listado A.1) que son instancias del actor ”Piece”. Estos ACTORINSTANCE tienen un único comportamiento BEHAVIOUR ”rotate” definido en el atributo ”BEHAVIOURID” que hace que giren ante un evento ”mouseDown”, es decir cuando se hace click sobre ellos con el ratón (línea 21 de la figura 4.5 y líneas 21 y 34 respectivamente del listado A.1). Con este fichero XMMVR de 30 elementos XMMVR junto a los ficheros gráficos y de texturas relacionados en la tabla 4.2 será suficiente para definir una escena que recrea la metáfora de teatro en nuestro museo. Con ella un usuario podría interaccionar de manera gráfica desde un punto de vista estático.

Tabla 4.2: Ficheros del ejemplo ilustrativo de la metáfora gráfica fundamental de teatro

XMMVR	’’museo3dtheatre.xml’’
GRÁFICOS	’’Room0.W3D’’ ’’Piece.W3D’’
TEXTURAS	’’Amstrad.jpg’’ ’’Macintosh_Plus.jpg’’

4.2.2. Metáfora de locomoción

Ahora el escenario es el mismo pero ya no será una vista estática, puesto que la visión del usuario podrá desplazarse gracias a las teclas del cursor. Para recrear esta metáfora en nuestro museo hemos definido una escena donde se verá la sala principal del museo en la que hay un número de piezas expuestas, además de mover el ratón dentro de la sala, el usuario podrá mover la vista dentro del escenario usando las teclas del cursor. Se define un escenario "Room0", un actor "Piece" como en el ejemplo anterior pero además un actor "Camera" (línea 13 de la figura 4.6 y del listado A.2) con el valor "null" en el campo "FILE3D" que indica que no tiene apariencia.

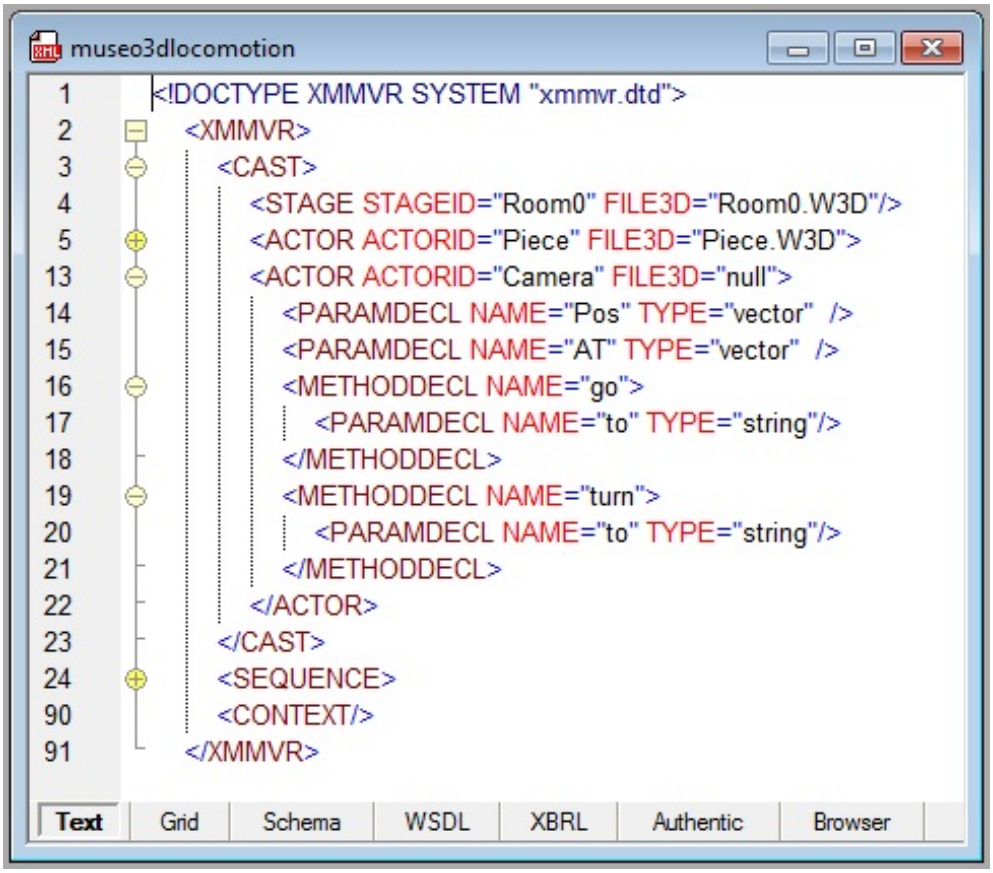

FIGURA 4.6: CAST de un ejemplo basado en la metáfora de locomoción

En la secuencia de escenas sólo se define una basada en el escenario "Room0" y en la que aparecen tres ACTORINSTANCE. Dos de ellos son instancias del actor "Piece": "Amstrad" y "Macintosh". Estos ACTORINSTANCE tienen un único comportamiento que hace que giren ante un evento "mouseDown", es decir cuando se hace click sobre ellos con el ratón. Se ha definido un ACTORINSTANCE denominado "User" (línea 26 de la figura 4.7 y del listado A.2) y será instancia del actor "Camera" que según le pongamos apariencia o no será un avatar del usuario o una cámara subjetiva respectivamente. Ante un evento "arrowUP" (tecla de cursor *UP* pulsada) hará que se avance dentro del escenario, ante un evento "arrowDOWN" (tecla de cursor *DOWN* pulsada) hará que se retroceda dentro del escenario, ante un evento "arrowLEFT" (tecla de cursor *LEFT* pulsada) se moverá hacia la izquierda y ante un evento "arrowRIGHT" (tecla de cursor *RIGHT* pulsada) se

moverá hacia la derecha (líneas 29, 37, 45 y 53 de la figura 4.7 y del listado A.2). Estos eventos provocan que el ACTORINSTANCE "User" ejecute el método "go" definido para el actor "Camera" que es en el que se basa, con el parámetro "to" y el valor correspondiente. Se hace así el habitual desplazamiento con las teclas de cursor.

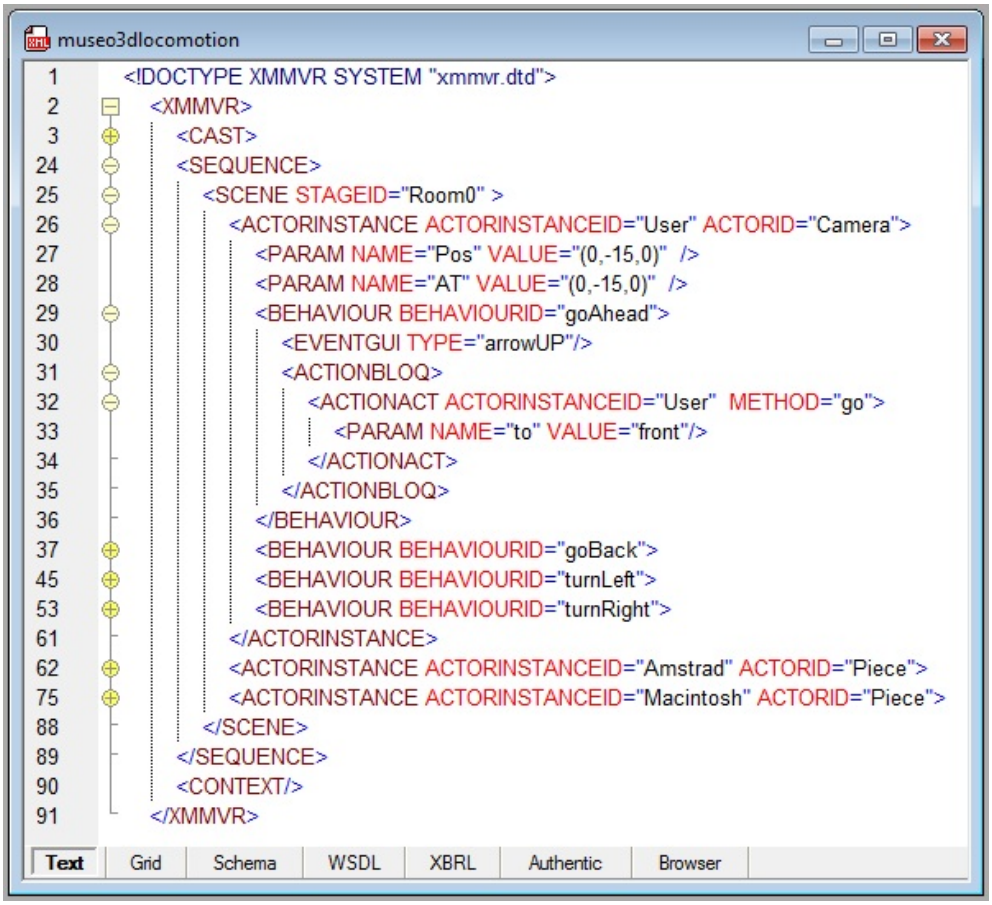

Figura 4.7: SCENE de un ejemplo basado en la metáfora de locomoción

En este caso hemos necesitado un fichero XMMVR de 60 elementos XMMVR junto a los ficheros gráficos y de texturas relacionados en la tabla 4.3 para definir una escena que recrea la metáfora de teatro en nuestro museo. Con ella un usuario podría interaccionar de manera gráfica desde un punto de vista dinámico.

Tabla 4.3: Ficheros del ejemplo ilustrativo de la metáfora gráfica fundamental de locomoción

XMMVR	''museo3dlocomotion.xml''
GRÁFICOS	''Room0.W3D'' ''Piece.W3D''
TEXTURAS	''Amstrad.jpg'' ''Macintosh_Plus.jpg''

Vistos los ejemplos de implementación de las metáforas fundamentales de interacción gráfica, recordamos en la tabla 4.4 las metáforas estructurales de interacción gráfica ya presentadas en el capítulo 2.

Tabla 4.4: Metáforas estructurales de interacción gráfica

Habitaciones	Pueden considerarse como partes de ambas metáforas fundamentales y se aplican en áreas como entrenamiento virtual, educación o comercio electrónico
Escenarios giratorios	Son ejemplos característicos de la metáfora de teatro, ya que el punto de vista del usuario se mantiene constante
Edificios con plantas, pasillos, niveles	Son ejemplos típicos de la metáfora de locomoción
Estaciones espaciales, moléculas, redes de burbujas	Pertenecen a la metáfora de locomoción. Permiten más ”libertad geométrica” que las anteriores. Esferas, habitaciones, burbujas o células tridimensionales son colocadas de manera no lineal creando estructuras moleculares, estaciones espaciales o redes conectadas por tubos por ejemplo
Metáforas urbanas	También son ejemplos de la metáfora de locomoción

4.2.3. Metáfora de habitaciones

Esta metáfora puede considerarse como parte de ambas metáforas fundamentales por lo que la recreación que hagamos de ella en nuestro museo reutilizará los elementos ya vistos. La figura 4.8 y el listado A.3 nos servirían para especificar un ejemplo donde nos movemos a través de habitaciones es decir, partiendo de un pasillo en el que se presentan varias puertas podremos entrar a través de ellas a cada una de las habitaciones que comunican. Se define la habitación inicial con el escenario ”Corridor” (línea 23 de la figura 4.8 y del listado A.3) que tendrá tres puertas: ”DoorRoom0”, ”DoorRoom1” y ”DoorRoom2” (líneas 24, 34 y 44 de la figura 4.8 y del listado A.3) que son ACTORINSTANCE del ACTOR ”Door” (línea 8 de la figura 4.8 y del listado A.3). Al hacer click sobre una de ellas nos llevará a la habitación correspondiente ”Room0”, ”Room1” o ”Room2”. En cada habitación aparecerán las piezas correspondientes si corresponde y una puerta ”DoorToCorridor0” (línea 82 de la figura 4.9 y del listado A.3), ”DoorToCorridor1” y ”DoorToCorridor2” (líneas 90 y 101 del listado A.3) que son ACTORINSTANCE del ACTOR ”Door” (línea 8 de la figura 4.8 y del listado A.3) y que al hacer click nos llevará de nuevo a la habitación ”Corridor”. Esto se realiza en base al elemento ACTIONSYS del tipo ”goToScene” con el parámetro DESTINATION con valor ”Corridor”. En este caso hemos necesitado un fichero XMMVR de 76 elementos XMMVR junto a los ficheros gráficos y de texturas relacionados en la tabla 4.5 para definir una escena que recrea la metáfora de habitaciones en nuestro museo y que combinada con otras puede ser la base del mundo virtual que nos ocupa.

museo3drooms

```
1   <!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
2   <XMMVR>
3     <CAST>
4       <STAGE STAGEID="Corridor" FILE3D="Corridor.W3D"/>
5       <STAGE STAGEID="Room0" FILE3D="Room0.W3D"/>
6       <STAGE STAGEID="Room1" FILE3D="Room1.W3D"/>
7       <STAGE STAGEID="Room2" FILE3D="Room2.W3D"/>
8       <ACTOR ACTORID="Door" FILE3D="Door.W3D">
13      <ACTOR ACTORID="Piece" FILE3D="Piece.W3D">
21    </CAST>
22    <SEQUENCE>
23      <SCENE STAGEID="Corridor">
24        <ACTORINSTANCE ACTORINSTANCEID="DoorRoom0" ACTORID="Door">
25          <PARAM NAME="Pos" VALUE="(0,-14,0)" />
26          <BEHAVIOUR BEHAVIOURID="enterRoom0">
27            <EVENTGUI TYPE="mouseDown"/>
28            <ACTIONBLOQ>
29              <ACTIONSYS TYPE="goToScene" DESTINATION="Room0"/>
30            </ACTIONBLOQ>
31          </BEHAVIOUR>
32        </ACTORINSTANCE>
33        <ACTORINSTANCE ACTORINSTANCEID="DoorRoom1" ACTORID="Door">
42        <ACTORINSTANCE ACTORINSTANCEID="DoorRoom2" ACTORID="Door">
51      </SCENE>
52      <SCENE STAGEID="Room0">
89      <SCENE STAGEID="Room1">
100     <SCENE STAGEID="Room2">
111   </SEQUENCE>
112   <CONTEXT/>
113 </XMMVR>
```

Text | Grid | Schema | WSDL | XBRL | Authentic | Browser

Figura 4.8: Ejemplo basado en la metáfora de habitaciones (Corridor)

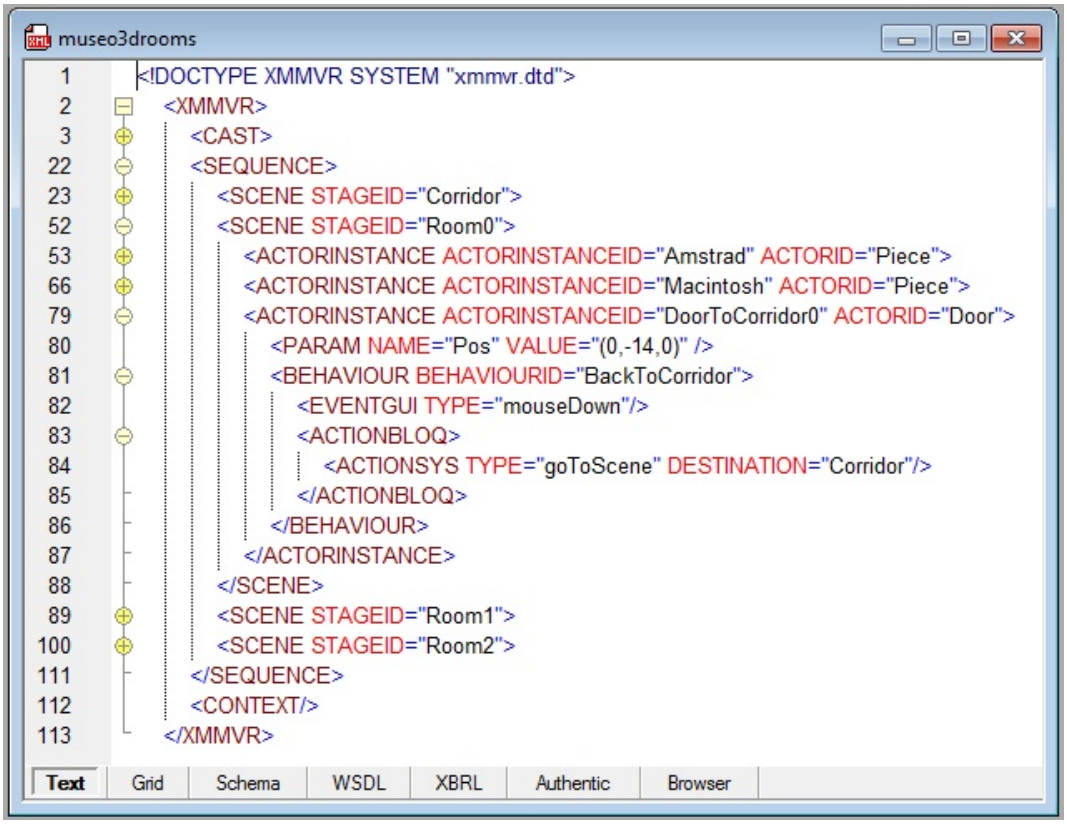

Figura 4.9: Ejemplo basado en la metáfora de habitaciones (Room0)

Tabla 4.5: Ficheros del ejemplo ilustrativo de la metáfora gráfica estructural de habitaciones

XMMVR	''museo3drooms.xml''
GRÁFICOS	''Room0.W3D'' ''Corridor.W3D'' ''Room1.W3D'' ''Piece.W3D'' ''Door.W3D'' ''Room2.W3D''
TEXTURAS	''Amstrad.jpg'' ''Macintosh_Plus.jpg''

4.2.4. Resto de metáforas estructurales

Ya que el resto de metáforas estructurales son evolucionadas de las ya vistas, en este apartado haremos una breve presentación de cada una de ellas sin presentar listados que repitan los conceptos ya vistos.

Metáfora de espacios giratorios: Para un escenario giratorio único, el mismo fichero que define la metáfora de teatro podría servirnos. Basta con que en el fichero que define el escenario se aplique una codificación que permita girar al escenario, por ejemplo si estuviera definido en VRML podríamos modelar un escenario animado usando rutas y eventos propios de VRML.

Metáfora de edificios con plantas, pasillos, niveles: Una variante del ejemplo de habitaciones con más escenarios añadidos serviría para implementar la metáfora de edificios con plantas, pasillos y niveles. Ahora cada escenario STAGE tendrá sus pasillos y habitaciones. Para cambiar de un nivel a otro podemos usar eventos del sistema que generen el salto a otro nivel vía un elevador o teletransportador.

Metáfora de estaciones espaciales, moléculas, redes de burbujas: Sería el mismo ejemplo que el anterior con un modelado más complejo de los escenarios.

Metáforas urbanas: Podría ser un gran escenario o como los ejemplos anteriores varios escenarios STAGE pero no dejaría de ser más que una evolución de la metáfora de habitaciones.

4.2.5. Metáforas navegacionales

Sigamos mostrando ejemplos de implementación de metáforas navegacionales de interacción gráfica, pero antes refrescamos en la tabla 4.6 su clasificación ya presentada en el capítulo 2.

Tabla 4.6: Metáforas navegacionales de interacción gráfica

Elevador / rampa hidráulica	Debido al movimiento lineal, soportan tareas secuenciales
Vehículo en vía / tren	Son posibles dos direcciones con esta metáfora secuencial en la mayoría de los casos
Deslizamiento / cable ferrocarril	Es una metáfora secuencial de un sentido donde sólo se permite una secuencia estricta de *action spaces* visitados
Silla voladora / alfombra	Pertenece al grupo de metáforas paralelas ya que el usuario puede volar a más de uno o dos *action spaces*
Tele-Transporte / "Beam me up"	Que a pesar de la desorientación, es una metáfora apropiada para estructuras completamente libres y permite la elección de varios espacios destino

Las metáforas navegacionales serán implementadas con gran esfuerzo de modelado. Asignaremos cada elemento de estas metáforas a cada elemento "ACTOR" de nuestro lenguaje. Así, la metáfora elevador podría ser un actor moviéndose a través de pisos (elemento "SCENE") resultando una combinación de metáfora navegacional (elevador) y metáfora estructural (edificio).

Elevador/rampa hidráulica: Un ascensor que nos desplaza a través de los pisos del edificio de nuestro museo, cada uno de ellos podría ser una evolución de la metáfora de habitaciones pero esta vez el movimiento es vertical en lugar de horizontal. Partiendo de la puerta del ascensor en una planta del museo, podemos subir o bajar a otro piso. En el listado A.4 y en la figura 4.10 podemos ver un ejemplo. Se definen tres niveles "Level0", "Level1" y "Level2" que son escenarios "SCENE" con un "STAGE" asociado (líneas 4, 5 y 6 del listado A.4 y de la figura 4.10).

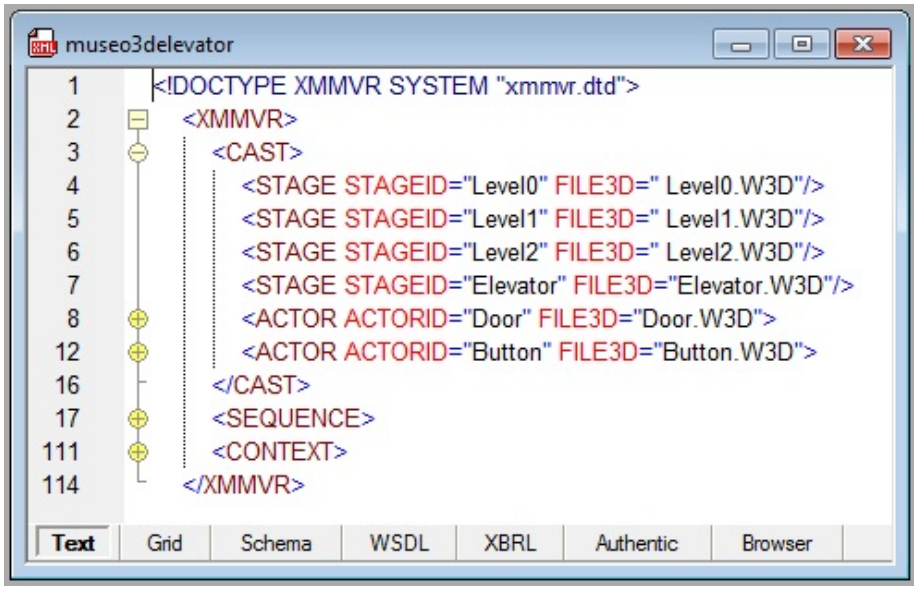

Figura 4.10: CAST de un ejemplo basado en la metáfora de ascensor

La secuencia "SEQUENCE" comienza en el "Level0" donde está la puerta del ascensor "DoorElevator0" y el botón "ButtonElevator01" que nos lleva al piso "Level1" y el botón "ButtonElevator02" que nos lleva al piso "Level2" según los comportamientos "go01" (línea 24 de la figura 4.11 y del listado A.4) y "go02" (línea 36 del listado A.4) respectivamente cuando hacemos click sobre ellos. En los pisos "Level1" y "Level2" (líneas 47 y 76 de la figura 4.11 y del listado A.4) aparecerán las puertas del ascensor con sus botones correspondientes que tendrán comportamientos similares. Durante la transición de un nivel a otro se nos mostrará el escenario "Elevator" (línea 105 de la figura 4.11 y del listado A.4) que representa el interior de la cabina del ascensor durante el trayecto. Se utiliza para ello el elemento "ACTIONSYS" de tipo "setTimer" (en la línea 28 de la figura 4.11 y del listado A.4 y 39 del listado A.4, revisar apartado 3.2.3.2.1 para más detalles).

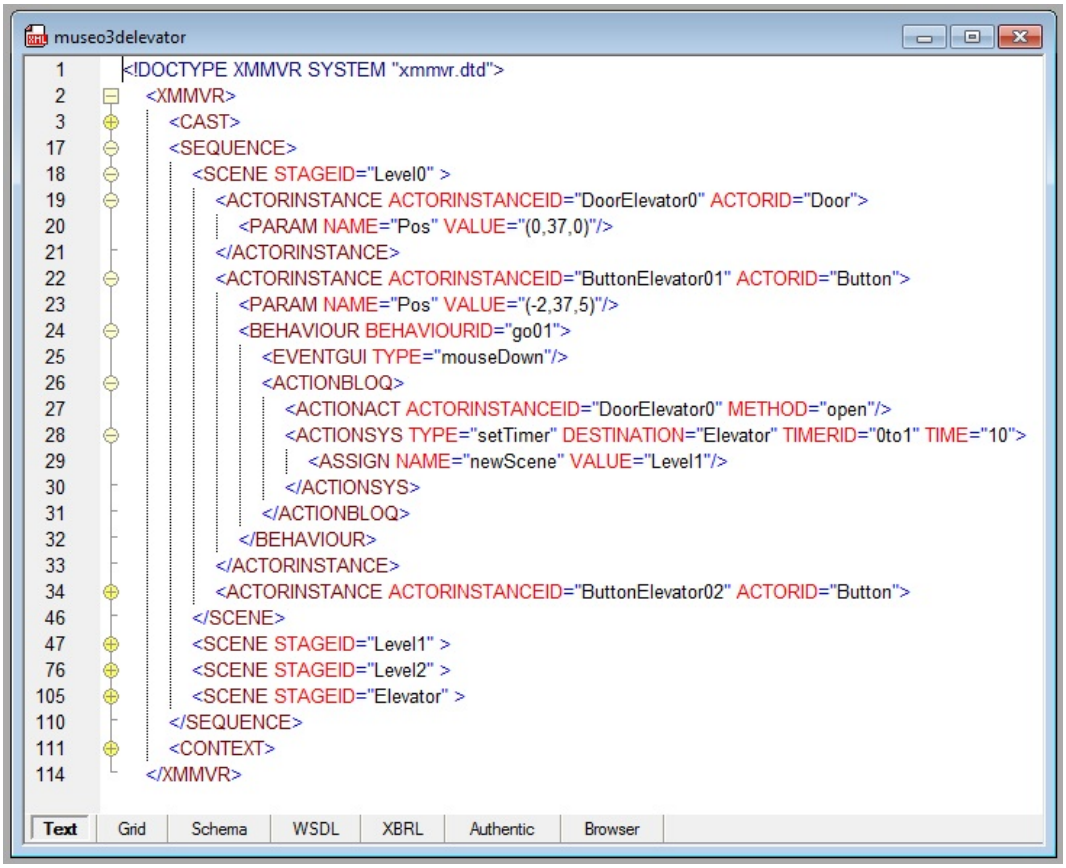

Figura 4.11: SCENE de un ejemplo basado en la metáfora de ascensor

En resumen, para subir o bajar hay un botón junto a la puerta del ascensor en cada piso. No se han colocado los botones dentro de la cabina del ascensor para dar simplicidad al ejemplo, por lo que sería realmente un montacargas. En este caso hemos necesitado un fichero XMMVR de 75 elementos XMMVR junto a los ficheros gráficos y de texturas relacionados en la tabla 4.7 para definir una escena que recrea la metáfora gráfica navegacional de elevador/rampa hidráulica en nuestro museo.

Tabla 4.7: Ficheros del ejemplo ilustrativo de la metáfora gráfica navegacional de ascensor

XMMVR	''museo3delevator.xml''	
GRÁFICOS	''Level0.W3D''	''Elevator.W3D''
	''Level1.W3D''	''Door.W3D''
	''Level2.W3D''	''Button.W3D''

Vehículo en vía/tren: Puesto que un tren no tendría mucho sentido dentro de nuestro museo, hemos optado por un sistema de desplazamiento del usuario que le lleva a las salas siguiendo rutas predefinidas tras pulsar botones que indican una u otra sala destino. Sería como un automatismo que desplaza al usuario a través de los pasillos del museo desde un lugar a otro como si estuviera sobre un raíl. Además el usuario puede desplazarse a través de los diferentes pasillos y habitaciones utilizando las teclas del cursor según se define en los comportamientos del ACTORINSTANCE "User" (línea 26 de la figura 4.12 y del listado A.4) definido en el único escenario "Museum" donde las habitaciones estarán ya modeladas (línea 25 de la figura 4.12 y del listado A.4).

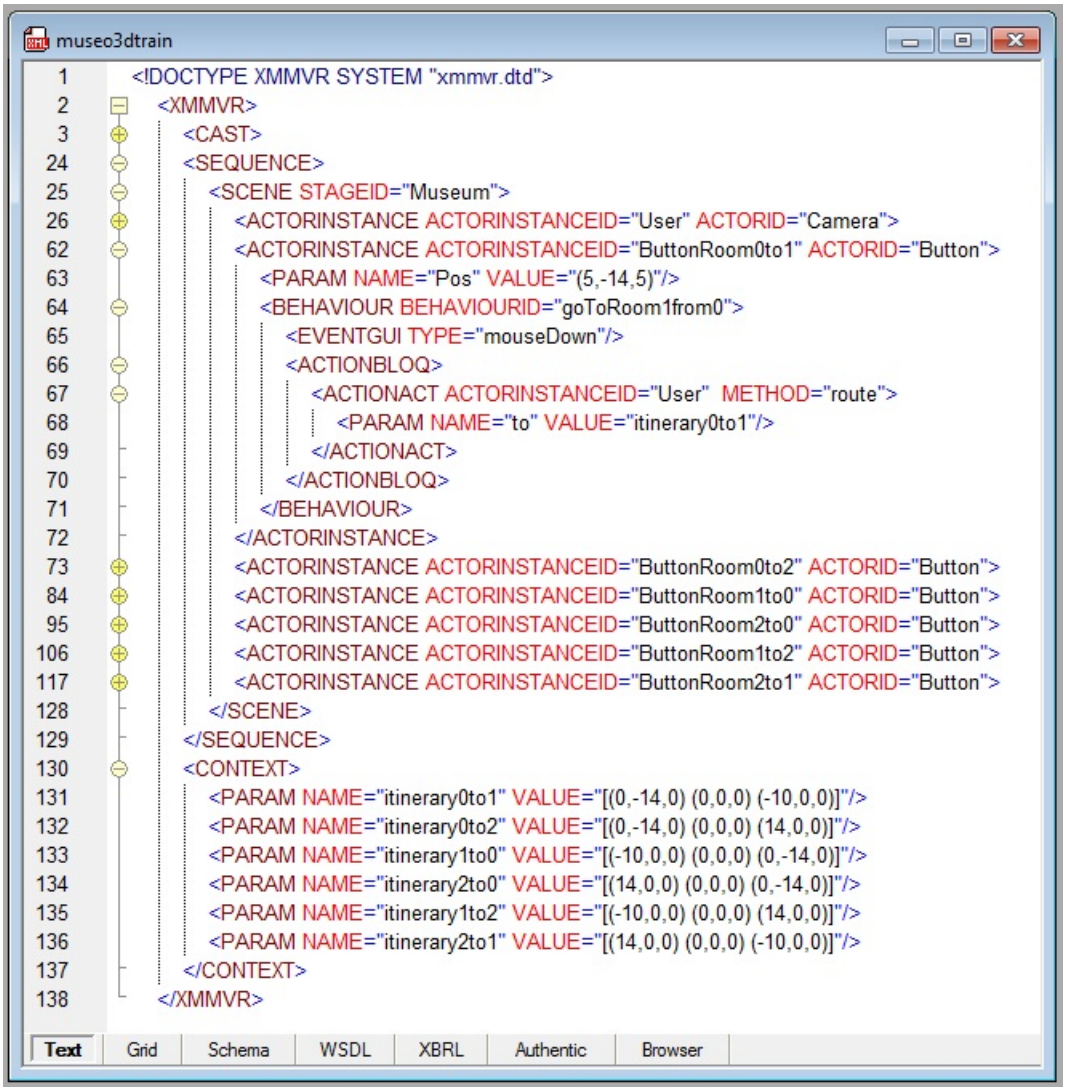

FIGURA 4.12: SCENE y CONTEXT de un ejemplo basado en la metáfora de tren

Pero cuando pulse un botón asociado a una sala destino, automáticamente se desplazará a esa sala siguiendo el itinerario pasado como parámetro al método "route" del ACTORINSTANCE

"User" (línea 67 de la figura 4.12 y líneas 67, 78, 89, 100 y 111 del listado A.4) y cuyos itinerarios asociados se definen en el CONTEXT (líneas 131, 132, 133, 134, 135 y 136 de la figura 4.12 del listado A.4). En este caso hemos necesitado un fichero XMMVR de 90 elementos XMMVR junto a los ficheros gráficos y de texturas relacionados en la tabla 4.8 para definir una escena que recrea la metáfora gráfica navegacional de vehículo en vía/tren en nuestro museo.

Tabla 4.8: Ficheros del ejemplo ilustrativo de la metáfora gráfica navegacional de vehículo en vía/tren

XMMVR	''museo3dtrain.xml''
GRÁFICOS	''Museum.W3D'' ''Button.W3D''

Deslizamiento/cable ferrocarril: Una leve variante de la anterior donde sólo se moverá el usuario si se hace click en los botones que tienen asociadas unas rutas o itinerarios definidos de la misma forma que el ejemplo anterior. Ver línea 30 de la figura 4.13 y líneas 30 y 41 del listado A.6.

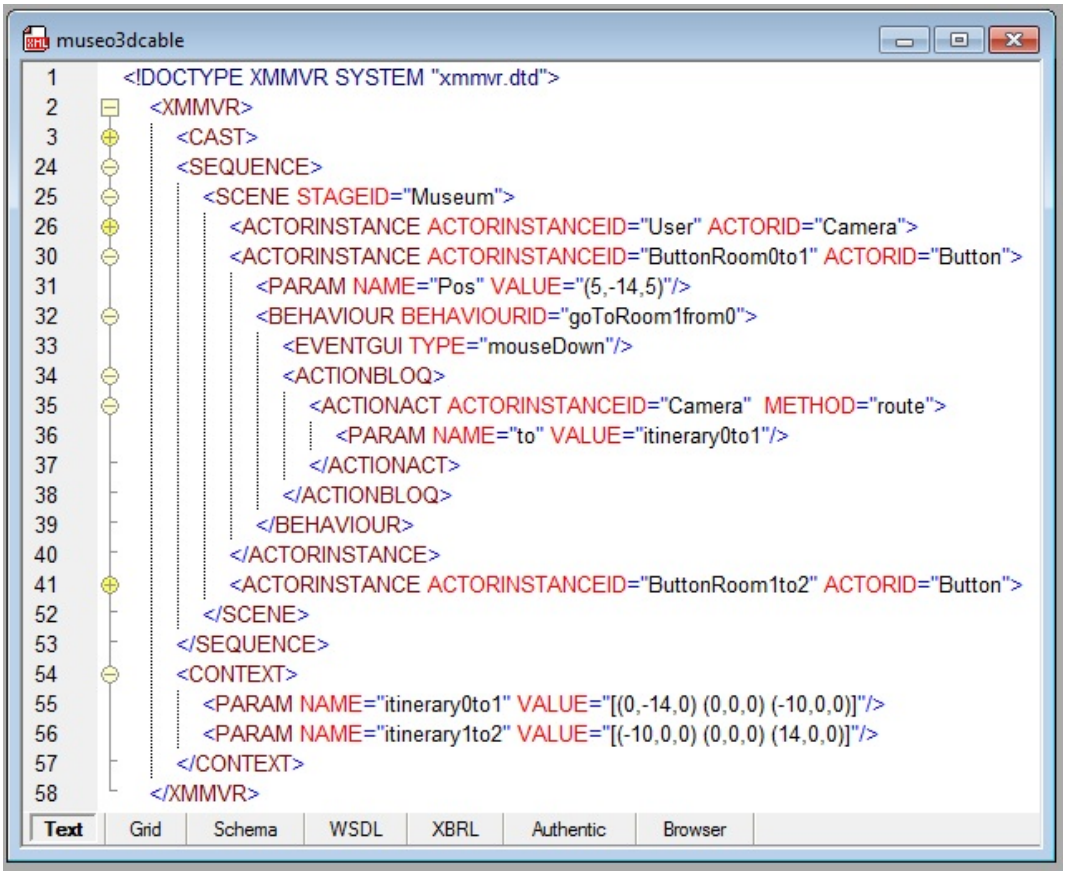

Figura 4.13: SCENE y CONTEXT de un ejemplo basado en la metáfora de cable

En este caso hemos necesitado un fichero XMMVR de 46 elementos XMMVR junto a los ficheros gráficos y de texturas relacionados en la tabla 4.9 para definir una escena que recrea

la metáfora gráfica navegacional de deslizamiento/cable ferrocarril en nuestro museo.

Tabla 4.9: Ficheros del ejemplo ilustrativo de la metáfora gráfica navegacional de deslizamiento/cable ferrocarril

XMMVR	''museo3dcable.xml''
GRÁFICOS	''Museum.W3D'' ''Button.W3D''

Silla voladora/alfombra: Aunque nuestra silla o alfombra voladora sea capaz de desplazarnos de un punto a otro del mundo, tendremos que tener un origen y un destino.

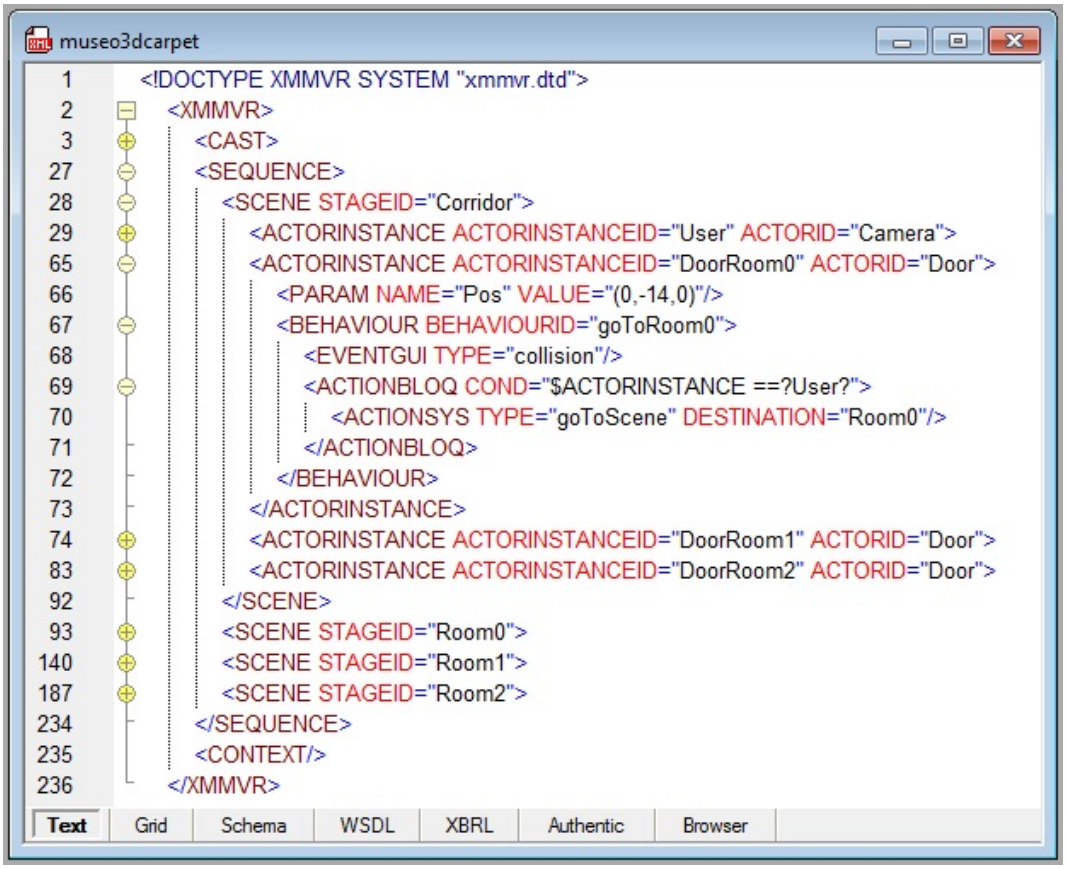

Figura 4.14: SCENE de un ejemplo basado en la metáfora de silla voladora

De nuevo éstos serán una evolución de la metáfora de habitaciones pero a diferencia de hacer click en la puerta para acceder a cada habitación, un nuevo evento gráfico ”collision” nos permitirá que el movimiento sea más similar al de una silla voladora ya que no habrá que abrir las puertas con una acción del usuario si no que se atravesarán suavemente haciendo que el desplazamiento no se interrumpa al pasar de una a otra habitación. Si usáramos por ejemplo VRML para modelar su apariencia, la detección de colisiones la realizaríamos a través de nodos VRML del tipo Collision que asociaríamos al evento gráfico ”collision” de XMMVR (ver línea 68 de la figura 4.14 y líneas 68, 86, 133, 180 y 227 del listado A.7. La

detección de colisiones indica al navegador VRML qué objetos de la escena no se van a poder atravesar. Esto permite evitar, por ejemplo, que los visitantes traspasen las paredes de un edificio. La respuesta a la colisión la define el navegador (haciendo que se rebote en el objeto, deteniéndose simplemente, etc.). En este caso hemos necesitado un fichero XMMVR de 153 elementos XMMVR junto a los ficheros gráficos y de texturas relacionados en la tabla 4.10 para definir una escena que recrea la metáfora gráfica navegacional de silla voladora/alfombra en nuestro museo.

Tabla 4.10: Ficheros del ejemplo ilustrativo de la metáfora gráfica navegacional de silla voladora/alfombra

XMMVR	''museo3dcarpet.xml''	
GRÁFICOS	''Room0.W3D''	''Corridor.W3D''
	''Room1.W3D''	''Door.W3D''
	''Room2.W3D''	

Tele-transporte/Beam me up: Nos volvemos a encontrar ante una evolución de la metáfora de habitaciones aunque en lugar de rutas usamos saltos entre escenarios.

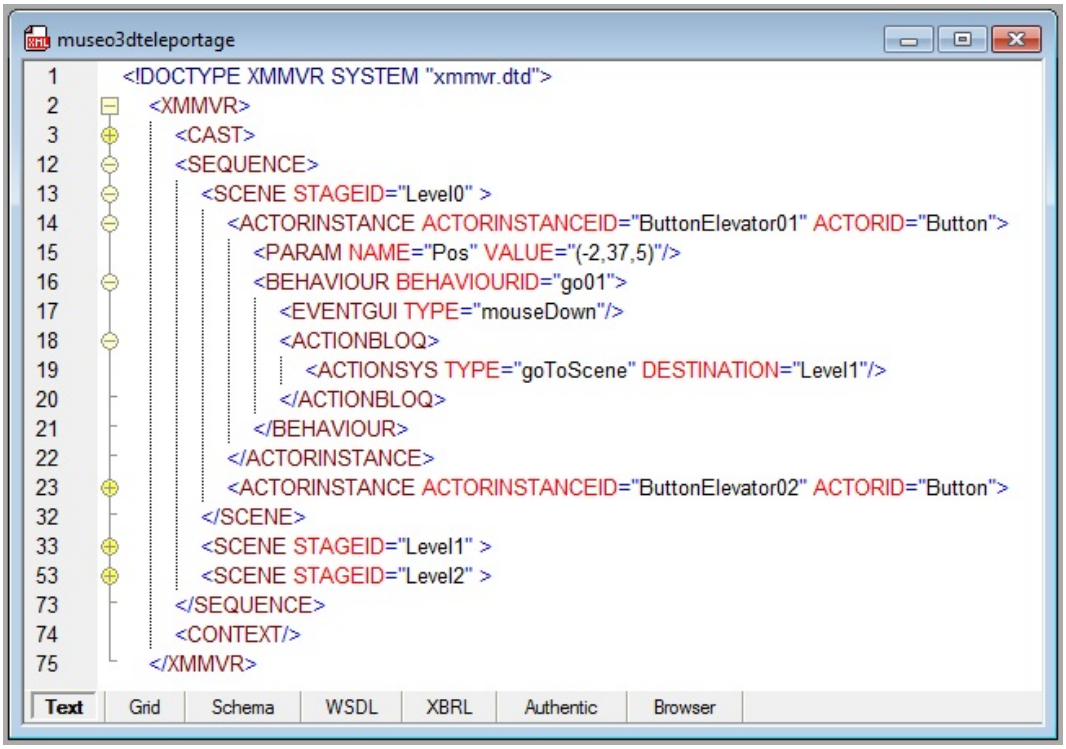

Figura 4.15: SCENE de un ejemplo basado en la metáfora de teletransporte

A diferencia del ejemplo del ascensor, aquí no introducimos una habitación de transición (interior del ascensor) para reforzar la idea de inmediatez del teletransporte (ver línea 19 en la figura 4.15 y líneas 19, 28, 39 , 48, 59 y 68 en el listado A.8). El teletransporte tan usado por los personajes de STAR TREK (IMDB, 1966) siempre tenía el mismo fin: desplazar a

los protagonistas de la serie desde el Enterprise hasta el lugar donde realizarían sus misiones. En este caso hemos necesitado un fichero XMMVR de 49 elementos XMMVR junto a los ficheros gráficos y de texturas relacionados en la tabla 4.11 para definir una escena que recrea la metáfora gráfica navegacional de teletransporte en nuestro museo.

Tabla 4.11: Ficheros del ejemplo ilustrativo de la metáfora gráfica navegacional de teletransporte

XMMVR	''museo3dteleportage.xml''
GRÁFICOS	''Level0.W3D'' ''Level1.W3D'' ''Level2.W3D'' ''Button.W3D''

4.2.6. Resumen

Metáforas estructurales y navegacionales podrían combinarse, así podríamos desplazarnos por una ciudad con una silla voladora utilizando lo implementado en los ejemplos anteriores correspondientes y haciendo uso de rutas combinándolas con los cambios de escenario cuando entremos en cada uno de los edificios de la ciudad, por ejemplo.

4.3 Implementación de metáforas vocales

Vistas las implementaciones de varios ejemplos basados en las metáforas de interacción gráfica, a continuación haremos lo mismo para cada una de las metáforas de interacción vocal presentadas en el capítulo 2, sirva la tabla 4.12 para enumerarlas.

Tabla 4.12: Metáforas de interacción vocal

Interface Agent o Agente Interfaz	El usuario se comunica con un agente, separado del mundo virtual, que ejecuta sus comandos, por ejemplo: mayordomo, quiero que la casa sea roja
Proxy o Delegado	El usuario puede tomar control de varios agentes (cambiar de agente, selección en la acción) en el mundo virtual e interactuar con el mundo virtual a través de ellos, por ejemplo: pintor, ¡pinta la casa de rojo!
Divinity o Divinidad	El usuario actúa como un dios y controla el mundo directamente, por ejemplo: que la casa sea roja! (sin foco)
Telekinesis	Los objetos y agentes en el mundo virtual pueden ser interlocutores de diálogo del usuario, por ejemplo: casa, ¡píntate de rojo!

4.3.1. Agente interfaz

En la metáfora vocal de agente interfaz el usuario se comunica con un agente, separado del mundo virtual, que ejecuta sus comandos. Para ilustrar esta metáfora vocal en nuestro museo hemos definido un guía virtual que no vemos y que nos recibe dándonos una bienvenida, tras ello nos ofrece las piezas del museo con las que podemos interactuar. Todo ello de una manera vocal a la que podemos responder también vocalmente. El escenario es el mismo que se ha presentado para recrear las metáforas gráficas fundamentales pero ahora un agente virtual comienza diciendo: *"Welcome to our museum. My name is Ana, I will help you in your visit. Tell me which piece you want to interact with: Amstrad or Macintosh. Say finish to finish your visit."*. Una vez respondamos al guía, se nos mostrará una u otra pieza. Si respondemos al guía *"Show me the Amstrad"* o cualquier variante contenida en el fichero con la gramática correspondiente, se nos mostrará la figura correspondiente y la voz nos indicará lo siguiente: *"OK. This is the Amstrad machine. Which is the next piece you want to see? "*. Así sucesivamente para que cuando indiquemos "finish" o cualquier variante contenida en el fichero de la gramática correspondiente, la voz nos despida de la manera siguiente: *"Bye. Thank you for visiting us. Come back again whenever you want."*

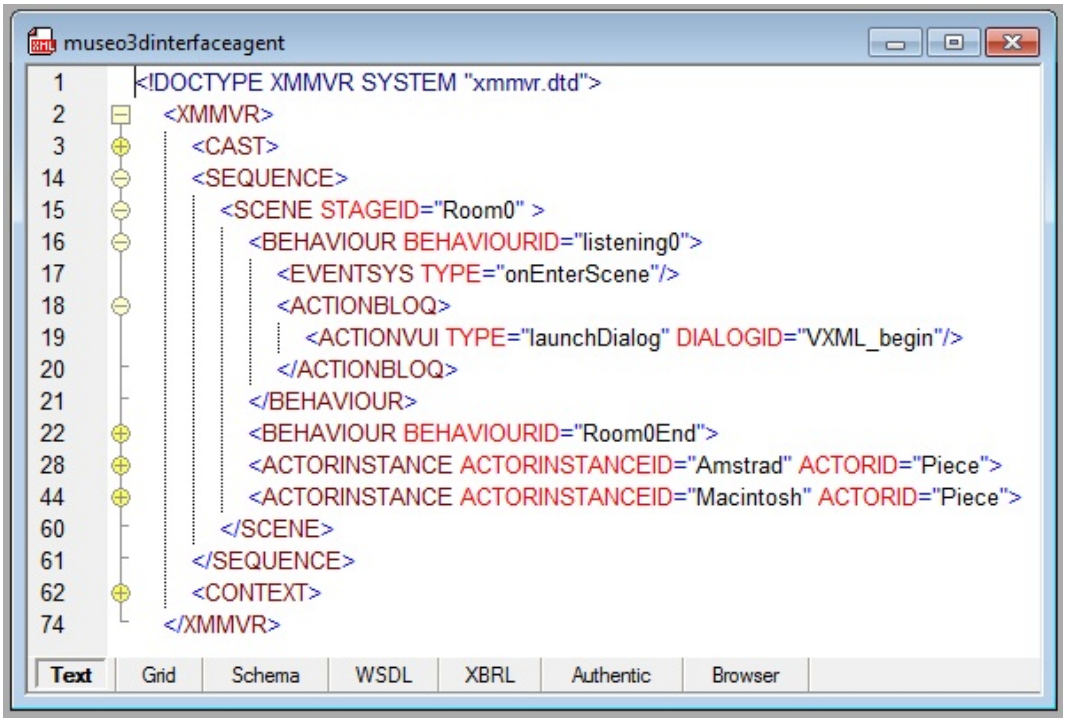

Figura 4.16: SCENE de un ejemplo basado en la metáfora de agente interfaz (listening0)

La representación gráfica se realiza de la misma forma que en los ejemplos de anteriores que recreaban las metáforas gráficas fundamentales pero la interacción vocal se especifica a través de las acciones vocales ACTIONVUI para lanzar diálogos por parte del sistema (ver línea 19 de la figura 4.16 y del listado A.9). En la figura 4.16 se ve cómo se lanza el diálogo de bienvenida descrito "VXML_begin" que corresponde con el fichero del listado A.10. A través de eventos vocales EVENTVUI se recogen los diálogos del usuario. La respuesta del usuario a la pregunta se recoge a través del evento vocal EVENTVUI (ver línea 33 de la figura 4.17 y del listado A.9) que a su vez genera una acción vocal ACTIONVUI que lanza el diálogo "VXML_piece" que corresponde con el

fichero del listado A.11.

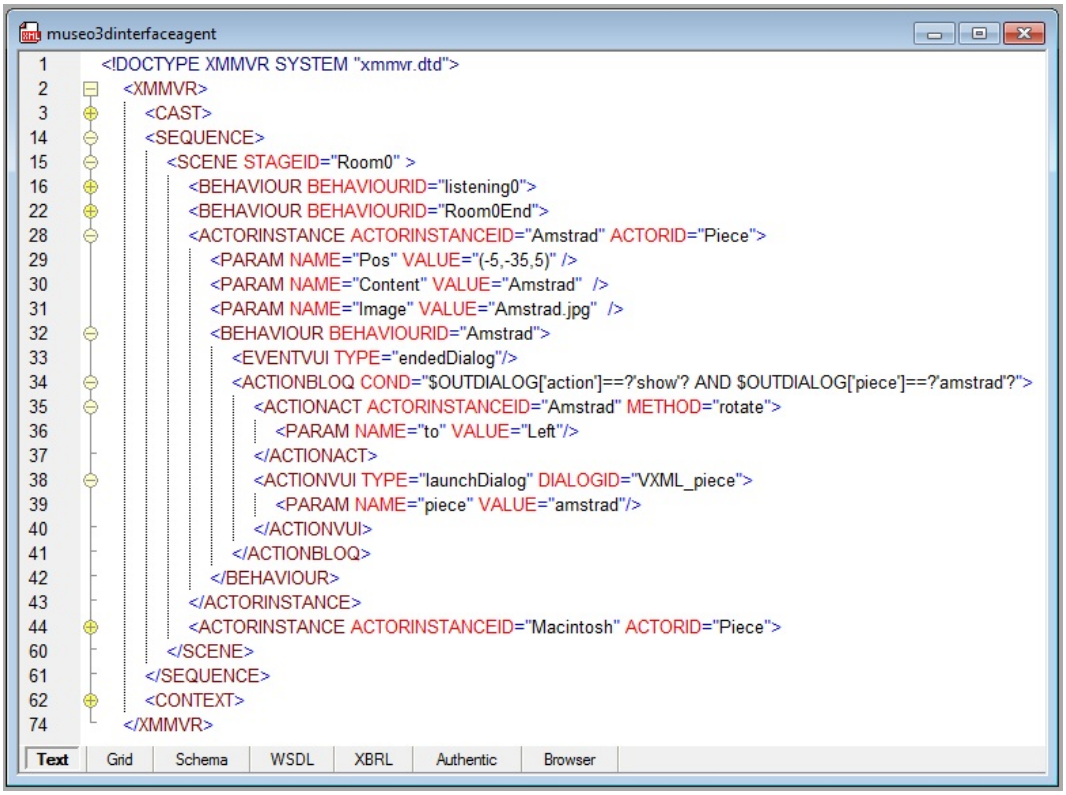

FIGURA 4.17: SCENE de un ejemplo basado en la metáfora de agente interfaz (Amstrad)

Cuando el usuario desea finalizar, un evento vocal EVENTVUI recoge su petición (ver línea 23 de la figura 4.18 y del listado A.9) que a su vez genera una acción vocal ACTIONVUI que lanza el diálogo "VXML_end" que corresponde con el fichero del listado A.12.

En resumen: tras cargarse el escenario descrito según indica la etiqueta "SCENE", comenzarán a ejecutarse comportamientos "BEHAVIOUR". En la línea 16 de la figura 4.16 y del listado A.9 vemos un comportamiento que provoca que nuestro sistema se quede escuchando los diálogos del usuario. En la línea 33 de la figura 4.16 y del listado A.9 se puede ver cómo se recoge el evento vocal con los valores correspondientes indicados. Es un evento vocal "EVENTVUI" del tipo "endedDialog" cuya evaluación provoca que obtengamos los valores "show" y "amstrad" en las variables "action" y "piece" respectivamente. Esto genera una acción "ACTIONACT" que supone la ejecución del método "rotate" por parte del "ACTORINSTANCEID" "Amstrad". En este caso hemos necesitado un fichero XMMVR de 51 elementos XMMVR que acompañado de los ficheros gráficos, de texturas, vocales y de gramática relacionados en la tabla 4.13 serán suficientes para definir una escena que recrea la metáfora de agente interfaz en nuestro museo.

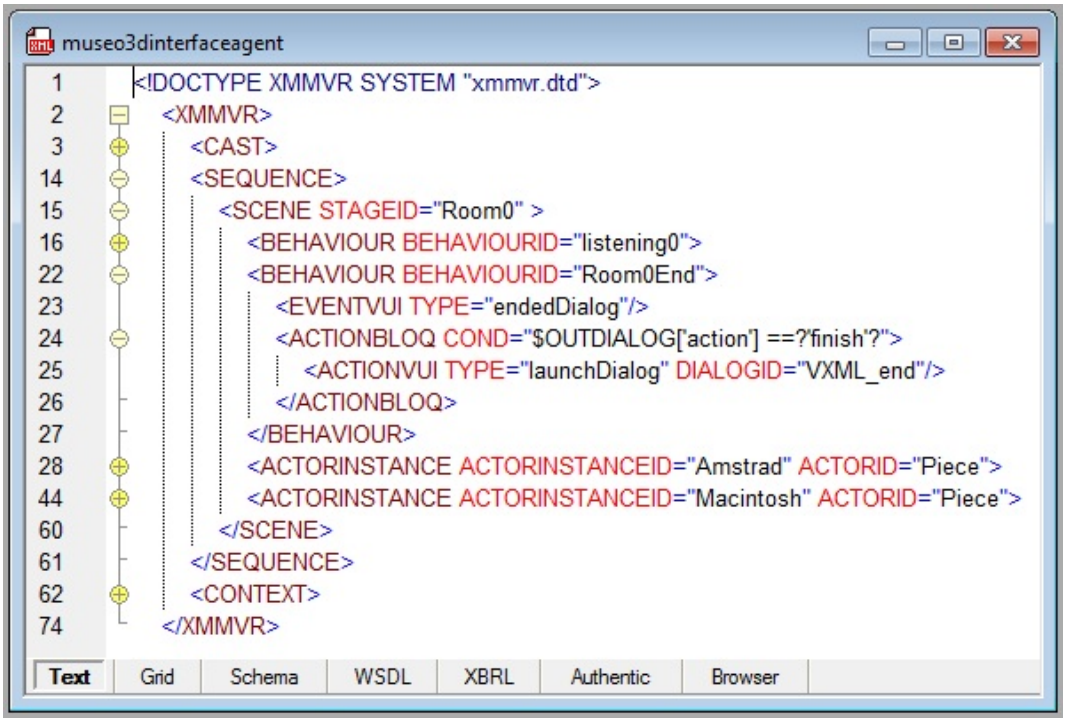

Figura 4.18: SCENE de un ejemplo basado en la metáfora de agente interfaz (Room0End)

Tabla 4.13: Ficheros del ejemplo ilustrativo de la metáfora vocal de agente interfaz

XMMVR	''museo3dinterfaceagent.xml''
GRÁFICOS	''Room0.W3D'' ''Piece.W3D''
TEXTURAS	''Amstrad.jpg'' ''Macintosh_Plus.jpg''
VOCALES	''museo3dinterfaceagent_begin.vxml'' ''museo3dinterfaceagent_piece.vxml'' ''museo3dinterfaceagent_end.vxml''
GRAMÁTICA	''interfaceagent.BNF''

4.3.2. Proxy o Delegado

En esta metáfora vocal el usuario puede tomar control de varios agentes (cambiar de agente, selección en la acción) en el mundo virtual e interactuar con el mundo virtual a través de ellos. Para recrearla definimos un escenario donde combinamos las metáforas gráficas descritas en los ejemplos anteriores pero donde habrá dos personajes: Ana y Pedro. Nos recibe Ana, la guía del museo que nos da la bienvenida y presenta a ambos. El escenario que veríamos durante todo esto sería la entrada al museo. El diálogo sería de la manera siguiente: *"Welcome to our museum. My name is Ana, I will help you in your visit. Tell me which piece you want to interact with: Amstrad or Macintosh. If you want to go to another floor, say it to Pedro. Say finish to finish your visit."*

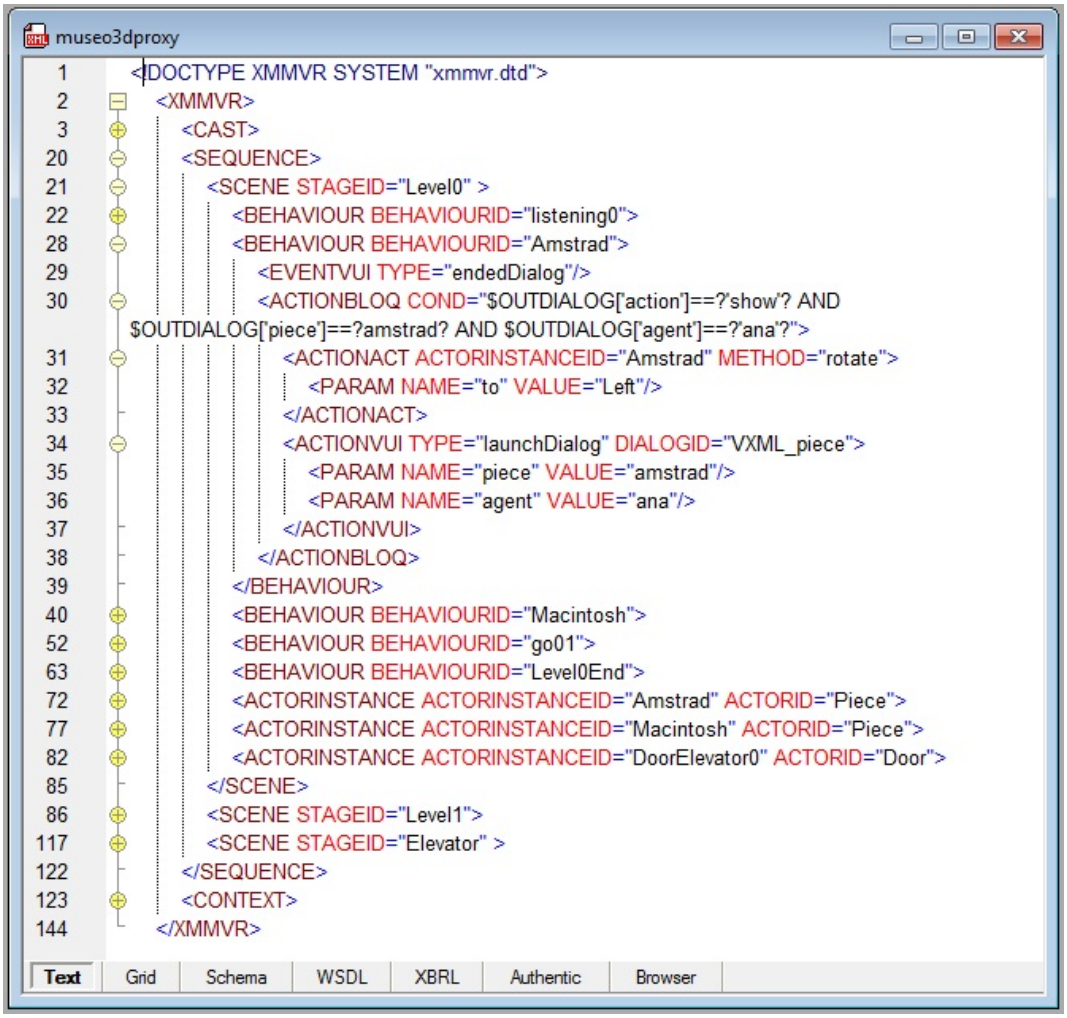

FIGURA 4.19: SCENE de un ejemplo basado en la metáfora de proxy o delegado (Ana)

Una vez respondamos por ejemplo *"Ana show me the Amstrad"* o cualquier variante incluida en la gramática correspondiente, se nos mostrará la pieza solicitada y la voz de Ana nos indicará lo siguiente: *"OK. This is the Amstrad machine. Which is the next piece you want to see? If you want to go to another floor, say it to Pedro"*. Si respondemos *"Pedro take me to the first"* o cualquier

variante incluida en la gramática correspondiente, Pedro nos subirá en ascensor al primer piso y dirá lo siguiente: *"Hi this is Pedro, you are on first floor, if you want to go back to floor zero just tell me"*. Podremos indicar a Ana que nos muestre otra pieza, a Pedro que nos suba de nuevo o terminar diciendo *"finish"* o cualquier variante incluida en la gramática correspondiente. Será entonces cuando la voz de Ana nos despedirá de la manera siguiente: *"Bye. Thank you for visiting us. Come back again whenever you want"*.

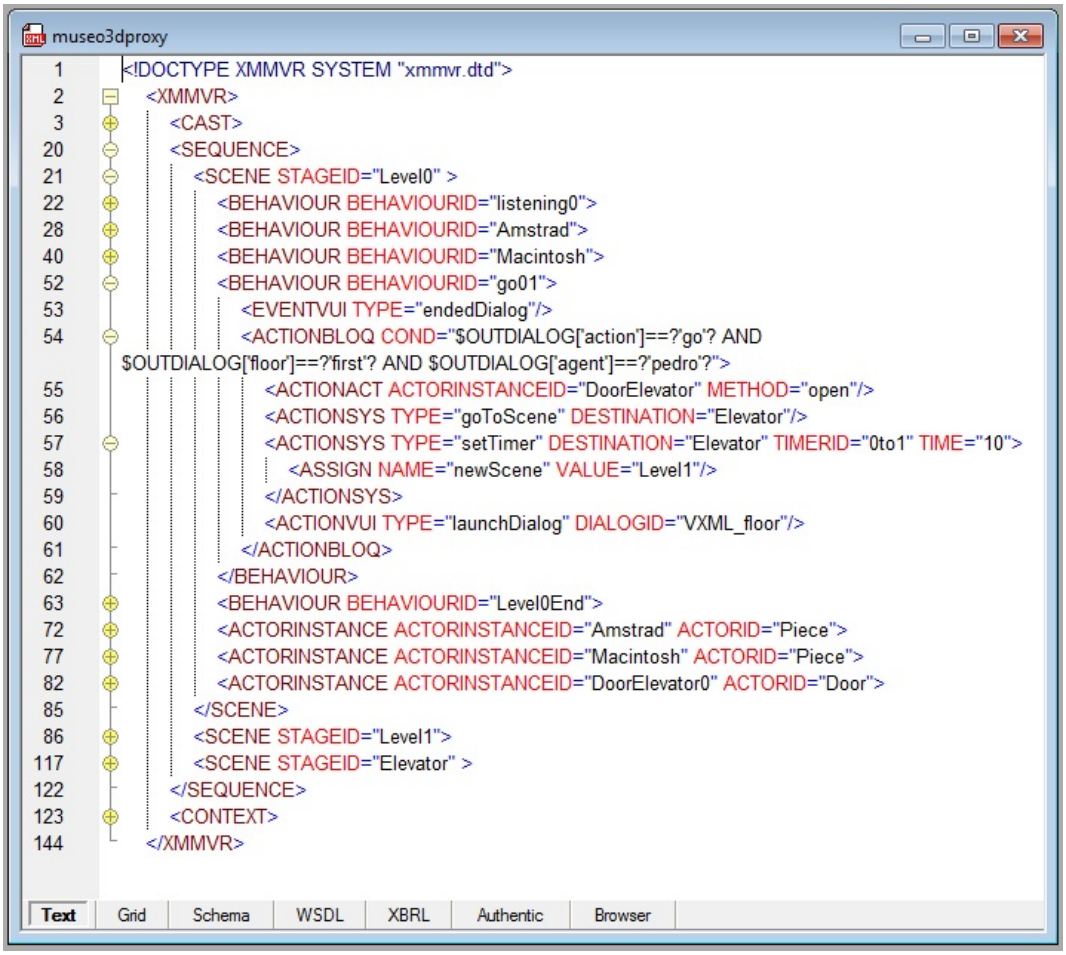

Figura 4.20: SCENE de un ejemplo basado en la metáfora de proxy o delegado (Pedro)

La forma de lanzar y gestionar los diálogos se basará en eventos y acciones vocales tal y como se ha descrito en el ejemplo anterior. Tras cargarse el escenario descrito según indica la etiqueta "SCENE", comenzarán a ejecutarse comportamientos "BEHAVIOUR". En la línea 22 de la figura 4.19 y del listado A.14 vemos un comportamiento que provoca que nuestro sistema se quede escuchando los diálogos del usuario. En la línea 29 de la figura 4.19 y del listado A.14 se puede ver cómo se recoge el evento vocal con los valores correspondientes indicados. Es un evento vocal "EVENTVUI" del tipo "endedDialog" cuya evaluación provoca que obtengamos los valores "show", "amstrad" y "ana" en las variables "action", "piece" y "agent" respectivamente. Esto genera una acción "ACTIONACT" que supone la ejecución del método "rotate" por parte del "ACTORINSTANCEID" "Amstrad". Podremos movernos por el primer piso y cuando queramos volver a la planta baja indicaremos:

"Pedro take me to the zero". Ya de nuevo en la planta baja Pedro nos indicará: *"You are on zero floor again, if you want to go back to the first just tell me. Tell Ana which piece you want to interact with"*. En la línea 53 de la figura 4.20 y del listado A.14 se puede ver cómo se recoge el evento vocal con los valores correspondientes indicados. Es un evento vocal "EVENTVUI" del tipo "endedDialog" cuya evaluación provoca que obtengamos los valores "go", "first" y "pedro" en las variables "action", "floor" y "agent" respectivamente. Esto genera una acción "ACTIONACT" que supone la ejecución del método "open" por parte del "ACTORINSTANCEID" "DoorElevator" es decir, se abre la puerta del ascensor. Tras ello se ejecutan dos acciones del sistema "ACTIONSYS". La primera es del tipo "goToScene" con destino al escenario "Elevator" (ver línea 56 de la figura 4.20 y del listado A.14), lo que provoca que el usuario se vea por un momento en el interior del ascensor simulando el trayecto. La duración de ese trayecto la indica la siguiente "ACTIONSYS" del tipo "setTimer" que tras el tiempo establecido por el parámetro "TIME" nos llevará al escenario "Level1" es decir, habremos llegado al primer piso. Finalmente una acción vocal nos informará de que hemos llegado a nuestro destino (línea 60 de la figura 4.20 y del listado A.14). En este caso hemos necesitado un fichero XMMVR de 101 elementos XMMVR que acompañado de los ficheros gráficos, de texturas, vocales y de gramática relacionados en la tabla 4.14 serán suficientes para definir una escena que recrea la metáfora de agente interfaz en nuestro museo.

Tabla 4.14: Ficheros del ejemplo ilustrativo de la metáfora vocal de proxy o delegado

XMMVR	''museo3dproxy.xml''
GRÁFICOS	''Level0.W3D'' ''Level1.W3D'' ''Elevator.W3D'' ''Piece.W3D''
TEXTURAS	''Amstrad.jpg'' ''Macintosh_Plus.jpg''
VOCALES	''museo3dproxy_begin.vxml'' ''museo3dproxy_piece.vxml'' ''museo3dproxy_floor.vxml'' ''museo3dproxy_end.vxml''
GRAMÁTICA	''proxy.BNF''

4.3.3. Divinidad

En esta metáfora vocal el usuario actúa como un dios y controla el mundo directamente. Hemos definido un escenario que intenta emularla en nuestro museo. Lo que veríamos nada más empezar sería la sala del museo. El sistema estaría a la espera de que el usuario diga algo. Si indicamos *"show Amstrad"*, se nos mostrará la figura Amstrad. Si indicamos *"finish"*, finalizará la aplicación. Aquí habrá un diálogo que podríamos denominar monólogo ya que el usuario es quien enuncia lo que desea y sus deseos se convierten en acciones que se realizan sin que nadie le responda por eso los ficheros VoiceXML (ver listados A.22 y A.23) estarán prácticamente vacíos de manera que no existe una realimentación vocal por parte del sistema.

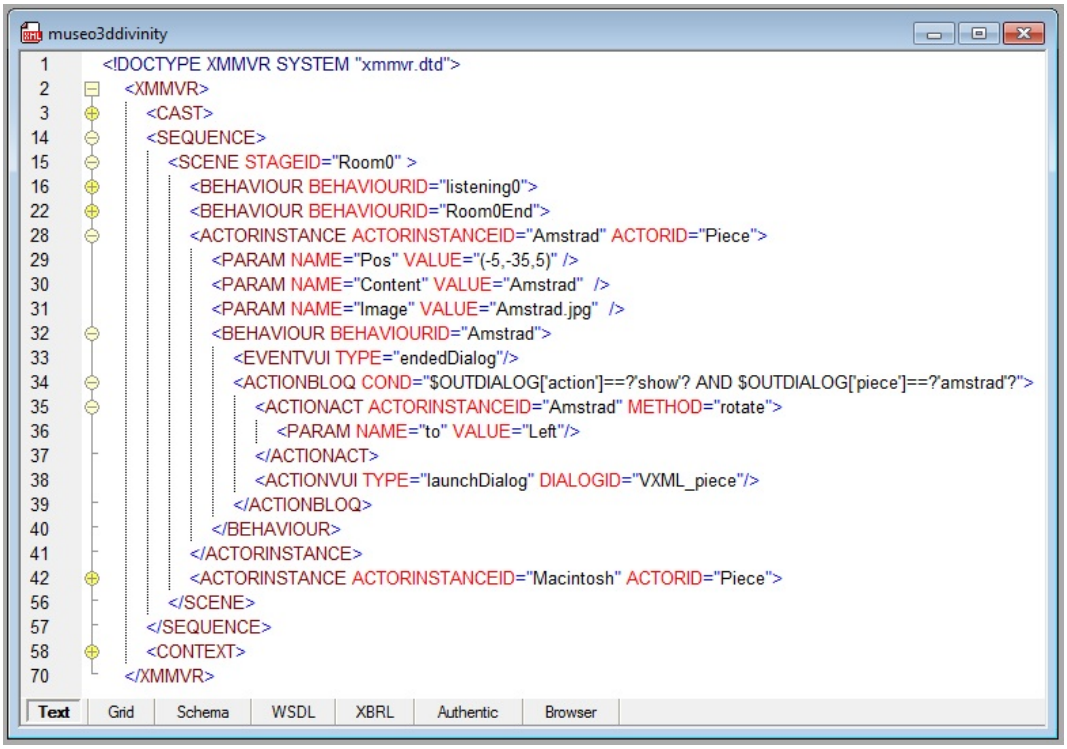

Figura 4.21: SCENE de un ejemplo basado en la metáfora de divinidad

La gestión de diálogo sería análoga a los ejemplos anteriores y los escenarios se implementarían de la misma forma. Tras cargarse el escenario descrito según indica la etiqueta "SCENE", comenzarán a ejecutarse comportamientos "BEHAVIOUR". En la línea 16 de la figura 4.21 y del listado A.20 vemos un comportamiento que provoca que nuestro sistema se quede escuchando los diálogos del usuario. En la línea 33 de la figura 4.21 y del listado A.20 se puede ver cómo se recoge el evento vocal con los valores correspondientes indicados. Es un evento vocal "EVENTVUI" del tipo "endedDialog" cuya evaluación provoca que obtengamos los valores "show" y "amstrad" en las variables "action" y "piece" respectivamente. Esto genera una acción "ACTIONACT" que supone la ejecución del método "rotate" por parte del "ACTORINSTANCEID" "Amstrad". En este caso hemos necesitado un fichero XMMVR de 48 elementos XMMVR que acompañado de los ficheros gráficos, de texturas, vocales y de gramática relacionados en la tabla 4.15 serán suficientes para definir una escena que recrea la metáfora de agente interfaz en nuestro museo.

Tabla 4.15: Ficheros del ejemplo ilustrativo de la metáfora vocal de divinidad

XMMVR	''museo3ddivinity.xml''
GRÁFICOS	''Room0.W3D'' ''Piece.W3D''
TEXTURAS	''Amstrad.jpg'' ''Macintosh_Plus.jpg''
VOCALES	''museo3ddivinity_begin.vxml'' ''museo3ddivinity_piece.vxml'' ''museo3ddivinity_end.vxml''
GRAMÁTICA	''divinity.BNF''

4.3.4. Telekinesis

Esta metáfora vocal permite que los objetos y agentes en el mundo virtual sean interlocutores de diálogo del usuario, como si el usuario hablase con los objetos y éstos obedecieran.

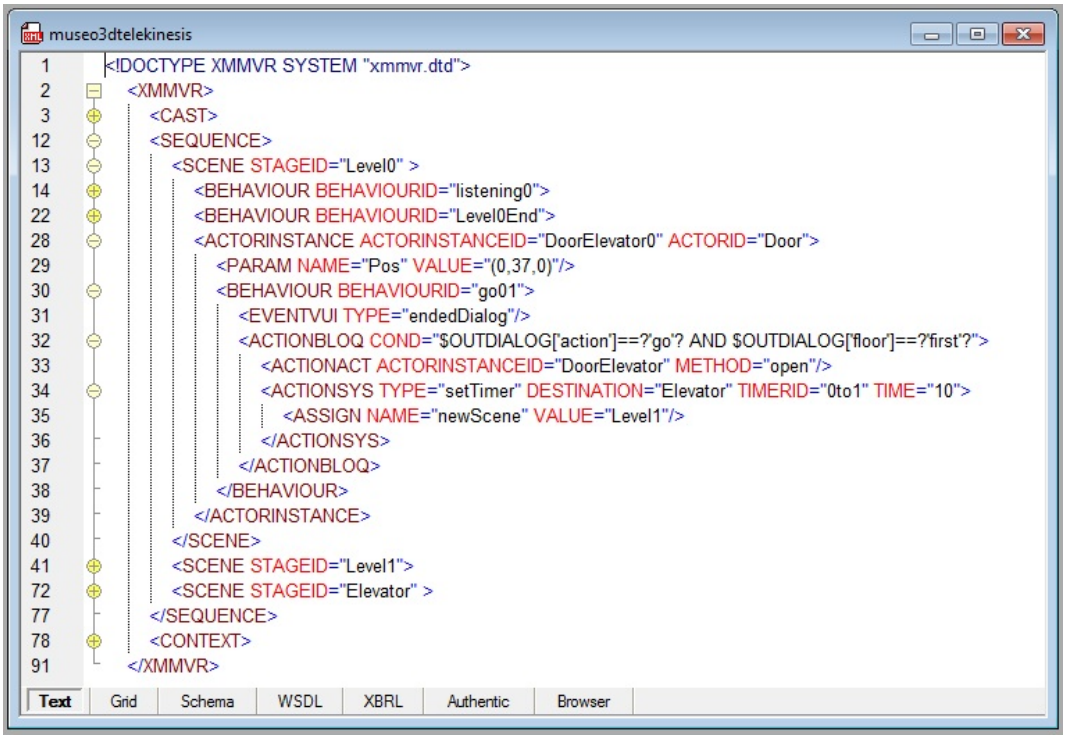

Figura 4.22: SCENE de un ejemplo basado en la metáfora de telekinesis

El escenario que hemos definido en nuestro museo para recrear esta metáfora nos mostraría nada más empezar la puerta de un ascensor en la planta baja del museo que está a la espera de nuestras órdenes. Si indicamos *"Elevator to first"*, seremos llevados al primer piso. Podremos volver a la planta baja o finalizar indicando *"Elevator to the zero"* o *"finish"* respectivamente. De nuevo

la gestión de diálogo sería análoga a los ejemplos anteriores y los escenarios se implementarían de la misma forma. Tras cargarse el escenario descrito según indica la etiqueta ”SCENE”, comenzarán a ejecutarse comportamientos ”BEHAVIOUR”. En la línea 14 de la figura 4.22 y del listado A.25 vemos un comportamiento que provoca que nuestro sistema se quede escuchando los diálogos del usuario. En la línea 31 de la figura 4.22 y del listado A.25 se puede ver cómo se recoge el evento vocal con los valores correspondientes indicados. Es un evento vocal ”EVENTVUI” del tipo ”endedDialog” cuya evaluación provoca que obtengamos los valores ”go” y ”first” en las variables ”action” y ”floor” respectivamente. Esto genera una acción ”ACTIONACT” que supone la ejecución del método ”open” por parte del ”ACTORINSTANCEID” ”DoorElevator” que se traduce en que la puerta del ascensor se abre. Tras ello se ejecuta la acción del sistema ”ACTIONSYS” del tipo ”setTimer” que tras el tiempo establecido por el parámetro ”TIME” nos llevará al escenario ”Level1” es decir, habremos llegado al primer piso (ver línea 34 de la figura 4.22 y del listado A.25). Al igual que la metáfora anterior no tenemos una retroalimentación vocal por parte del sistema ya que hemos considerado que el ascensor no habla pero emulando algunos ascensores del mundo real que emiten grabaciones para informarnos del piso al que llegamos, podríamos incluir diálogos emitidos por parte del ascensor. En este caso hemos necesitado un fichero XMMVR de 60 elementos XMMVR que acompañado de los ficheros gráficos, vocales y de gramática relacionados en la tabla 4.16 serán suficientes para definir una escena que recrea la metáfora de agente interfaz en nuestro museo.

Tabla 4.16: Ficheros del ejemplo ilustrativo de la metáfora vocal de telekinesis

XMMVR	’’museo3dtelekinesis.xml’’
GRÁFICOS	’’Level0.W3D’’ ’’Level1.W3D’’ ’’Elevator.W3D’’ ’’Door.W3D’’
VOCALES	’’museo3dtelekinesis_begin.vxml’’ ’’museo3dtelekinesis_floor.vxml’’ ’’museo3dtelekinesis_end.vxml’’
GRAMÁTICA	’’telekinesis.BNF’’

4.4 Implementación de cooperación entre modalidades

Demostradas las capacidades para implementar cada una de las metáforas de interacción gráfica y vocal por separado, haremos lo mismo con la cooperación entre modalidades. Para ello en la tabla 4.17 recordamos las posibilidades que tenemos.

Tabla 4.17: Cooperación entre modalidades

Especialización	Un determinado tipo de información es siempre procesada por la misma modalidad
Equivalencia	Ambos modos podrían tratar la misma información
Redundancia	La misma información es procesada por ambas modalidades
Transferencia	Parte de la información producida por una modalidad es usada por otra modalidad
Complementariedad	Diferentes partes de información son procesadas por cada modalidad pero tienen que ser combinadas

4.4.1. Especialización

En la especialización, un determinado tipo de información es siempre procesada por la misma modalidad.

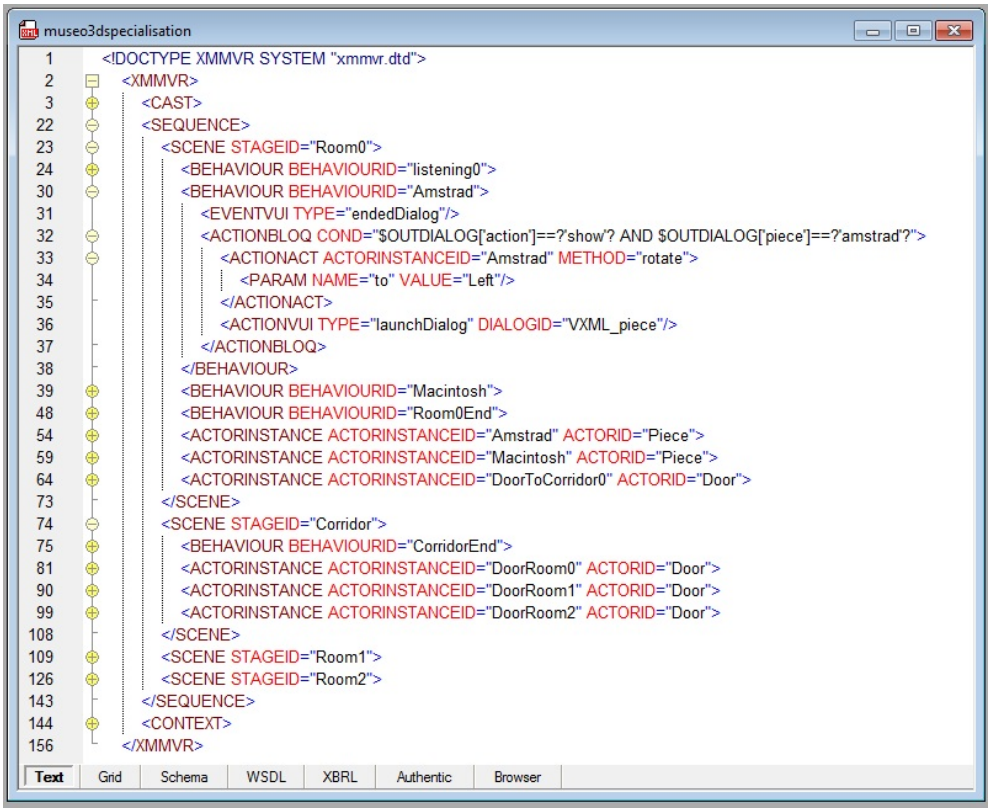

Figura 4.23: Ejemplo basado en la cooperación por especialización (VUI)

Para demostrar que nuestra propuesta es capaz de implementarla bastaría con demostrar que se puede implementar cada una de las metáforas consideradas para cada modalidad. Tras ello bastaría dejar a elección del programador qué modalidad utilizar ante una determinada situación. No obstante hemos implementado un ejemplo donde podemos movernos entre habitaciones con una interacción gráfica y donde en una habitación dada podemos solicitar con que piezas interactuar con una modalidad vocal.

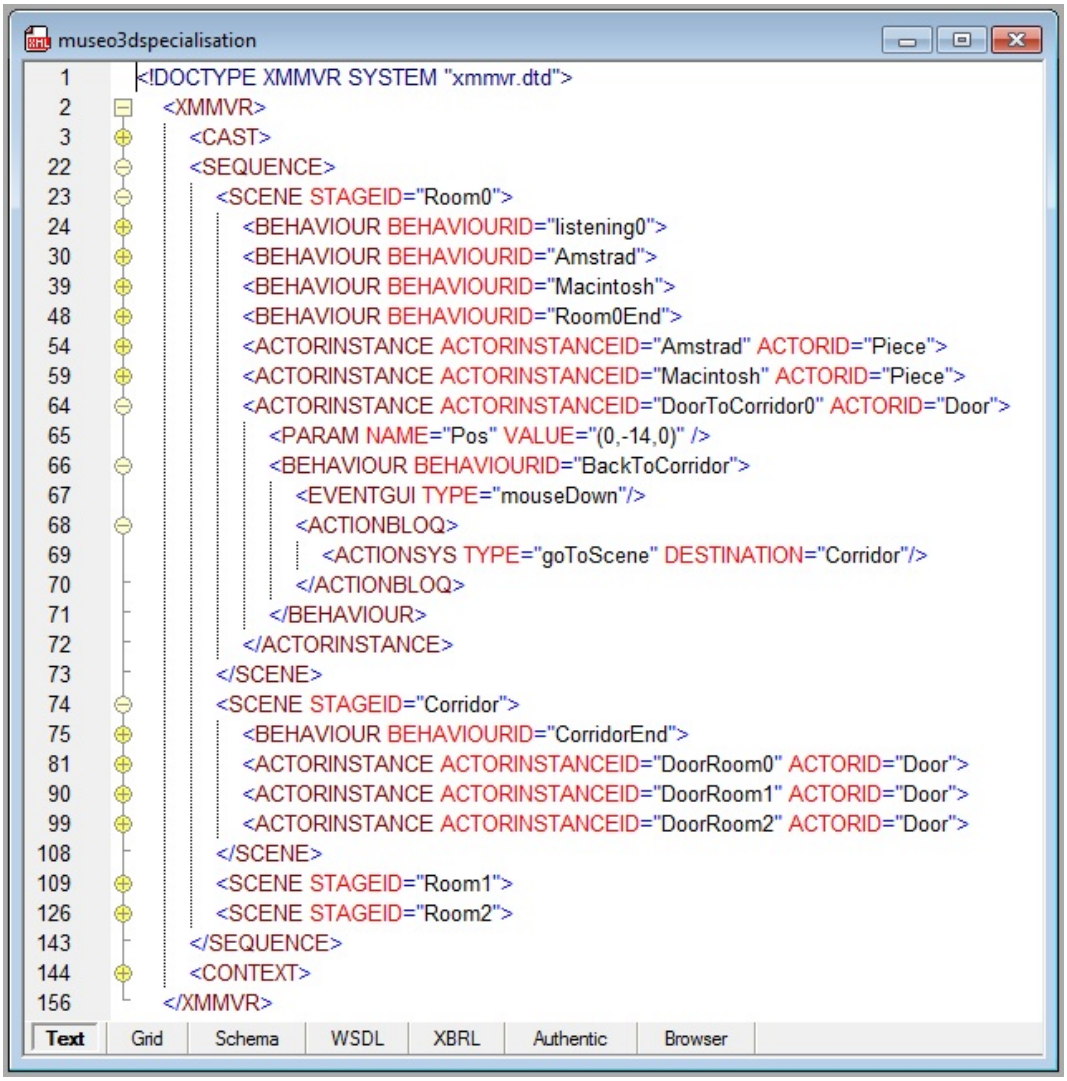

Figura 4.24: CAST y SCENE de un ejemplo basado en la cooperación por especialización

La representación gráfica se realiza de la misma forma que en los ejemplos anteriores que recreaban metáforas gráficas y la interacción vocal de la forma descrita en los ejemplos que recreaban metáforas vocales. Se iniciará la escena en la habitación "Room0" donde un agente iniciará un diálogo indicándonos *"Welcome to our museum. My name is Ana, I will help you in your visit. Tell me which piece you want to interact with: Amstrad or Macintosh. Click on a door to go to another room. Say finish to finish your visit."*. Seleccionaremos vocalmente la pieza que queremos ver (ver

línea 31 de la figura 4.23 y del listado A.30). Cuando hagamos click en la puerta nos moveremos a la habitación siguiente (ver línea 64 de la figura 4.24 y del listado A.30). En cualquier habitación podremos finalizar la sesión diciendo "Finish" o cualquier variante incluida en la gramática correspondiente. En este caso hemos necesitado un fichero XMMVR de 106 elementos XMMVR que acompañado de los ficheros gráficos, de texturas, vocales y de gramática relacionados en la tabla 4.18 serán suficientes para definir una escena que recrea la cooperación entre metáforas gráfica y vocal por especialización en nuestro museo.

Tabla 4.18: Ficheros del ejemplo ilustrativo de la cooperación entre modalidades por especialización

XMMVR	''museo3dspecialisation.xml''
GRÁFICOS	''Corridor.W3D'' ''Room0.W3D'' ''Room1.W3D'' ''Room2.W3D'' ''Door.W3D'' ''Piece.W3D''
TEXTURAS	''Amstrad.jpg'' ''Macintosh_Plus.jpg''
VOCALES	''museo3dspecialisation_begin.vxml'' ''museo3dspecialisation_piece.vxml'' ''museo3dspecialisation_end.vxml''
GRAMÁTICA	''specialisation.BNF''

4.4.2. Equivalencia / Redundancia

Cuando la entrada se realiza a través de varios modos, puede darse el caso de que se interfieran. Existen varias posibles estrategias para resolver este problema (Beringer, Kartal, Louka, Schiel, Tuerk & Trk, 2002) que se indican a continuación:

- 1. First match: la información reconocida primero es tomada para ser procesada por el sistema, sin tener en cuenta el método de reconocimiento. Ésta no es la mejor solución para el procesamiento multimodal pero sí es la opción más sencilla de implementar.
- 2. "Mean" match: el sistema toma la información que es común a ambos módulos de reconocimiento. Esto podría denominarse verificación multimodal.
- 3. Additional match: toma toda la información dada por ambos reconocedores para ser procesada por el sistema. Ésta sería la mejor solución si asumimos que ambos reconocedores son altamente precisos.

La equivalencia entre modalidades supone que ambos modos podrían tratar la misma información. En la redundancia se da que la misma información es procesada por ambas modalidades, supondría establecer una prioridad de una modalidad sobre otra. Nosotros implementaremos un ejemplo basado en la primera estrategia (first match) ya que como hemos indicado es la más sencilla de implementar y cualquiera de las otras dos estrategias son evolucionadas de ésta y se implementarían bajo el criterio del diseñador según los requisitos impuestos. Para ello usamos unas variables o "semáforos" que establecen la prioridad entre una u otra modalidad. La representación gráfica se realiza de la misma forma que en los ejemplos anteriores que recreaban metáforas gráficas y la interacción vocal de la forma descrita en los ejemplos que recreaban metáforas vocales. Hemos definido un ejemplo similar al anterior donde un guía virtual nos recibe en la habitación "Room0" y

nos indica *"Welcome to our museum. My name is Ana, I will help you in your visit. Tell me which piece you want to interact with: Amstrad or Macintosh. You can also click on any piece. Maybe you want to visit another room, just tell me which one or click on the door. Say finish to finish your visit."*. De esta forma podremos seleccionar la pieza que queremos ver de manera vocal (ver línea 50 de la figura 4.25 y del listado A.35):

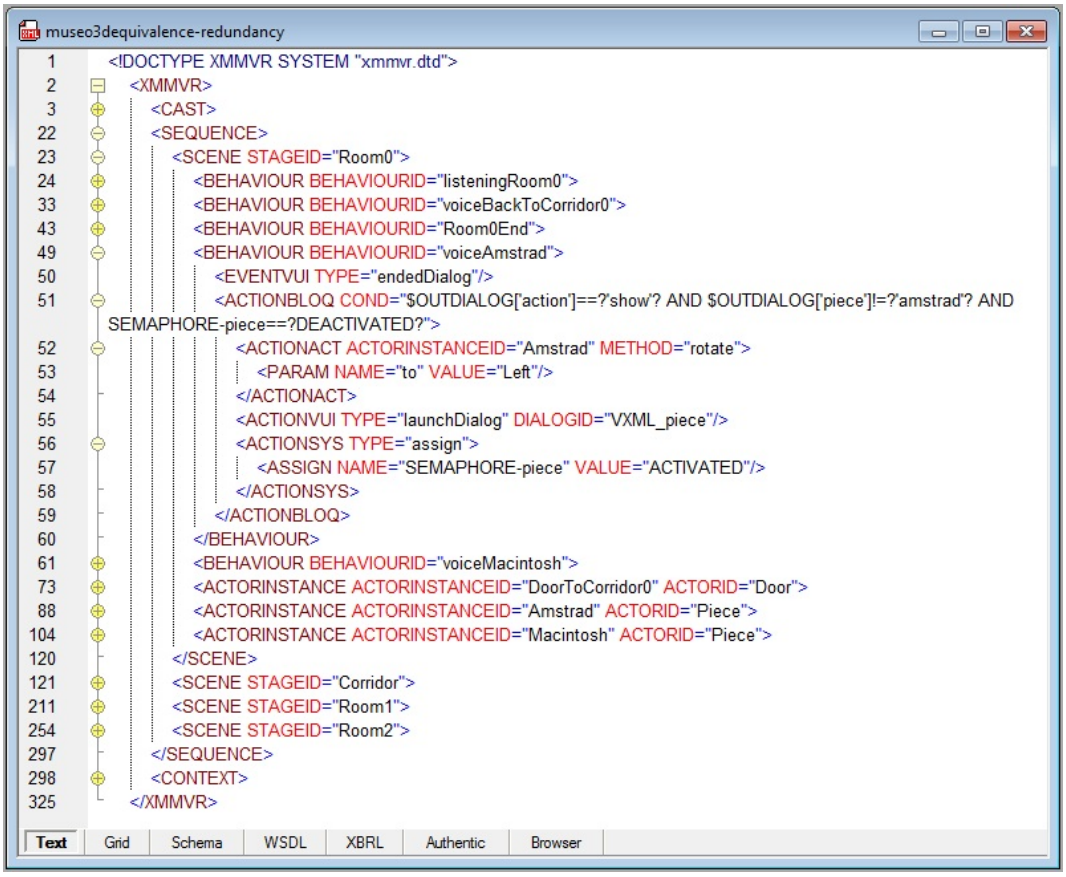

Figura 4.25: Ejemplo basado en la cooperación por equivalencia/redundancia (VUI)

O bien de manera gráfica (ver línea 92 de la figura 4.26 y del listado A.35). Asimismo, podremos ir a otra habitación haciendo click en la puerta o indicando el nombre de la habitación a visitar donde se presentarán otras piezas o se ofrecerán otras posibilidades.

En este caso hemos necesitado un fichero XMMVR de 227 elementos XMMVR que acompañado de los ficheros gráficos, de texturas, vocales y de gramática relacionados en la tabla 4.19 serán suficientes para definir una escena que recrea la cooperación entre metáforas gráfica y vocal por equivalencia/redundancia en nuestro museo.

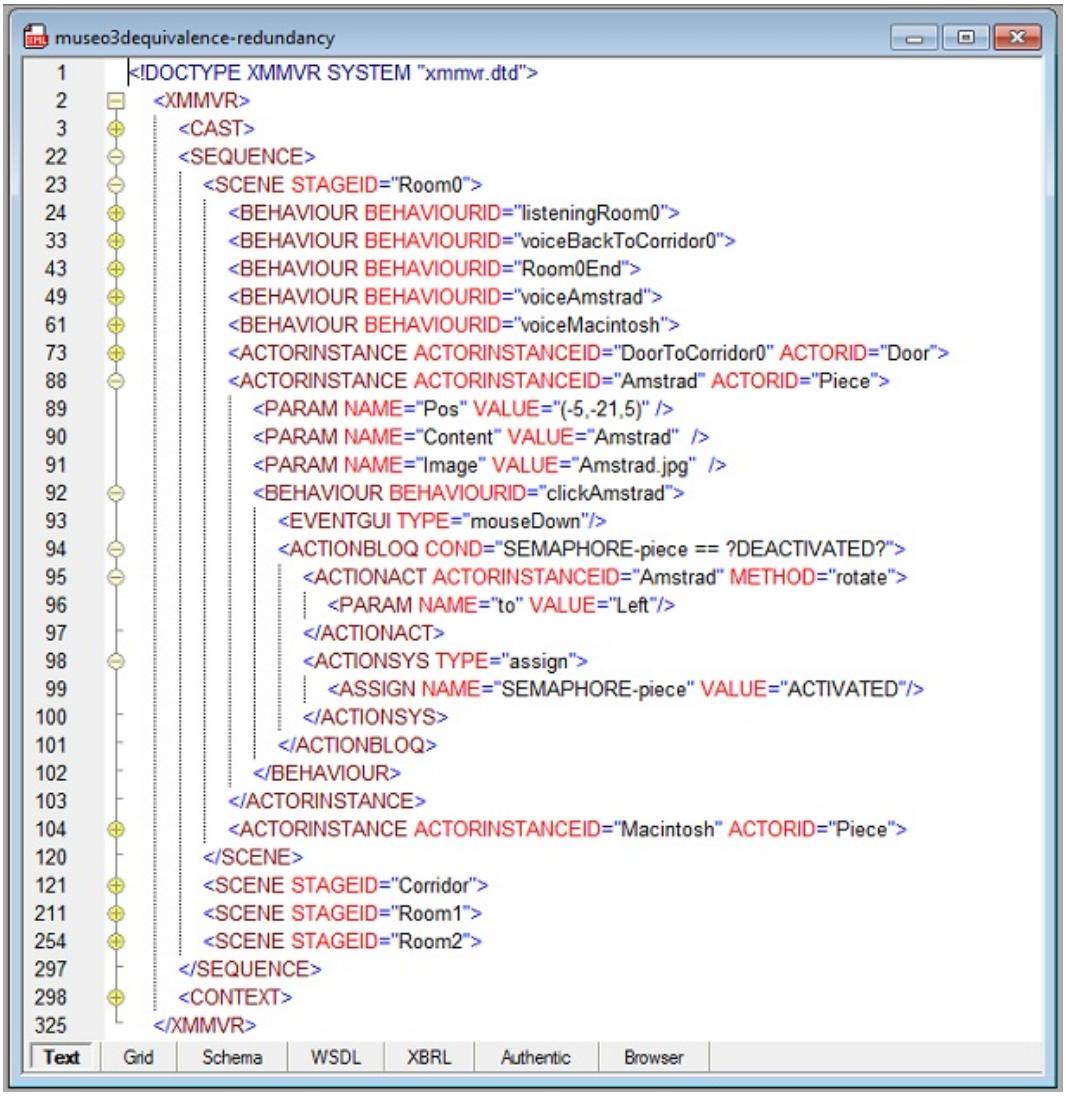

Figura 4.26: Ejemplo basado en la cooperación por equivalencia/redundancia (GUI)

Tabla 4.19: Ficheros del ejemplo ilustrativo de la cooperación entre modalidades por equivalencia/redundancia

XMMVR	''museo3dequivalence-redundancy.xml''
GRÁFICOS	''Corridor.W3D'' ''Room0.W3D'' ''Room1.W3D'' ''Room2.W3D'' ''Door.W3D'' ''Piece.W3D''
TEXTURAS	''Amstrad.jpg'' ''Macintosh_Plus.jpg''
VOCALES	''museo3dequivalence-redundancy_begin.vxml'' ''museo3dequivalence-redundancy_piece.vxml'' ''museo3dequivalence-redundancy_room.vxml'' ''museo3dequivalence-redundancy_end.vxml''
GRAMÁTICA	''equivalence-redundancy.BNF''

4.4.3. Transferencia / Complementariedad

En la transferencia parte de la información producida por una modalidad es usada por otra modalidad y en la complementariedad diferentes partes de información son procesadas por cada modalidad pero tienen que ser combinadas. No obstante, consideramos que ambos son casos particulares de uno mismo. Para ilustrarlo desarrollamos un ejemplo donde se presentan unas estanterías del museo donde podemos colocar las piezas a exhibir. Nos hemos basado en la idea del ”Put-that-there” (Bolt, 1980) y la metáfora elegida para la parte vocal sería la de ”divinidad” mientras que para la parte gráfica estaríamos usando una metáfora de teatro. Podemos indicar las piezas a colocar (what) y las estanterías que ocuparán (where) de una manera gráfica y/o vocal. Para ello implementamos unas variables ”semáforos” asociadas a cada una de las peticiones realizadas por el usuario de manera gráfica y/o vocal (”SEMAPHORE-what” y ”SEMAPHORE-where”) que se activan o desactivan a medida que se hayan asignado valores a las variables correspondientes a través de cualquiera de las dos modalidades.

Por ejemplo, si queremos colocar la pieza Amstrad en la estantería superior podríamos indicar vocalmente ”Put Amstrad up" (VUI+VUI) o podríamos hacer click en el Amstrad y click en la estantería de arriba indicando a la vez ”Put that there” (GUI+GUI). También podríamos indicar ”Put Amstrad there” haciendo click en la estantería de arriba (VUI+GUI). Finalmente podríamos indicar ”Put that up” haciendo click en el Amstrad (GUI+VUI). Explicaremos a continuación cada uno de los casos revisando el código XMMVR correspondiente partiendo de una situación en la que los semáforos están desactivados inicialmente según se indica en las líneas 26 y 27 de la figura 4.27 y del listado A.42. La representación gráfica se realiza de la misma forma que en los ejemplos anteriores que recreaban metáforas gráficas y la interacción vocal de la forma descrita en los ejemplos que recreaban metáforas vocales.

4.4.3.1. Transferencia-Complementariedad VUI+VUI

En este caso el usuario selecciona la pieza de manera vocal e indica que la quiere situar en la estantería de arriba de manera vocal también, para ello dice la frase ”Put Amstrad up”. La captura de esta frase se especifica en la línea 33 de la figura 4.28 y del listado A.42 que comprueba que las variables de salida del diálogo ”what” y ”where” sean distintas de ”that” y ”there” respectivamente. Desencadenará una acción del sistema ”ACTIONSYS” que activará las variables ”SEMAPHORE-what” y ”SEMAPHORE-where”. Puesto que los valores correspondientes a las variables ”what” y

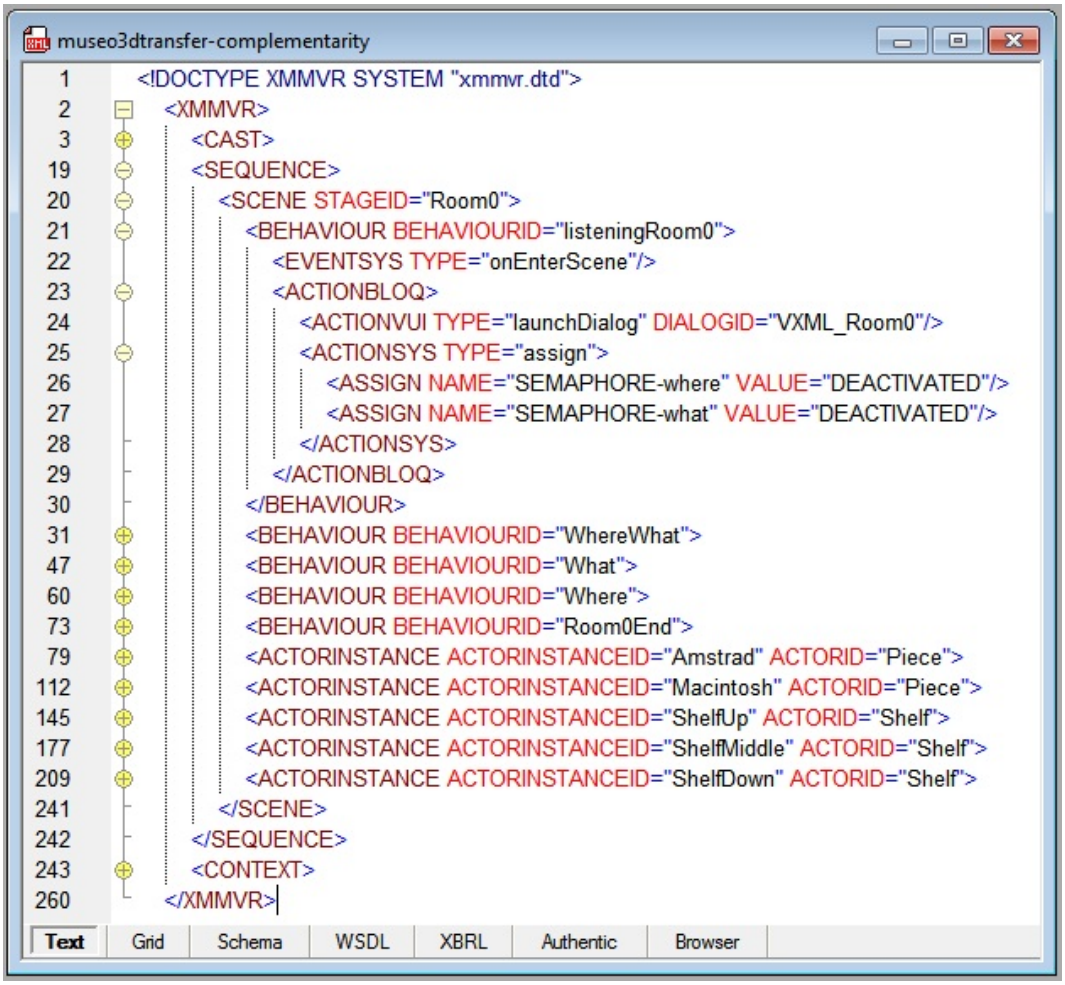

FIGURA 4.27: Inicialización de semáforos en el ejemplo basado en la cooperación por transferencia/redundancia

"where" ya han sido recogidos, generará el diálogo correspondiente (ver línea 38 de la figura 4.28 y del listado A.42) y se desencadenará la acción sobre la pieza que provocará que se mueva a la posición de la estantería indicada (ver línea 42 de la figura 4.28 y del listado A.42).

4.4.3.2. Transferencia-Complementariedad GUI+VUI

En este caso el usuario selecciona la pieza haciendo click sobre ella. Ya que la quiere situar en la estantería de arriba, para ello dice la frase "Put that up".

Este diálogo se recoge en la línea 48 de la figura 4.29 y del listado A.42. Se provoca la recogida del parámetro "where" con el valor "up" indicado y se asigna "NULL" el valor del parámetro "what" según se indica en las líneas 51 y 52 de la figura 4.29 y del listado A.42. Además se activa la variable "SEMAPHORE-where" y se desactiva la variable "SEMAPHORE-what" según se indica en las líneas 55 y 56 de la figura 4.29 y del listado A.42. El click se especifica en la línea 84 de la figura 4.30 y del listado A.42 que comprueba que esté activada la variable "SEMAPHORE-where". Desencadenará

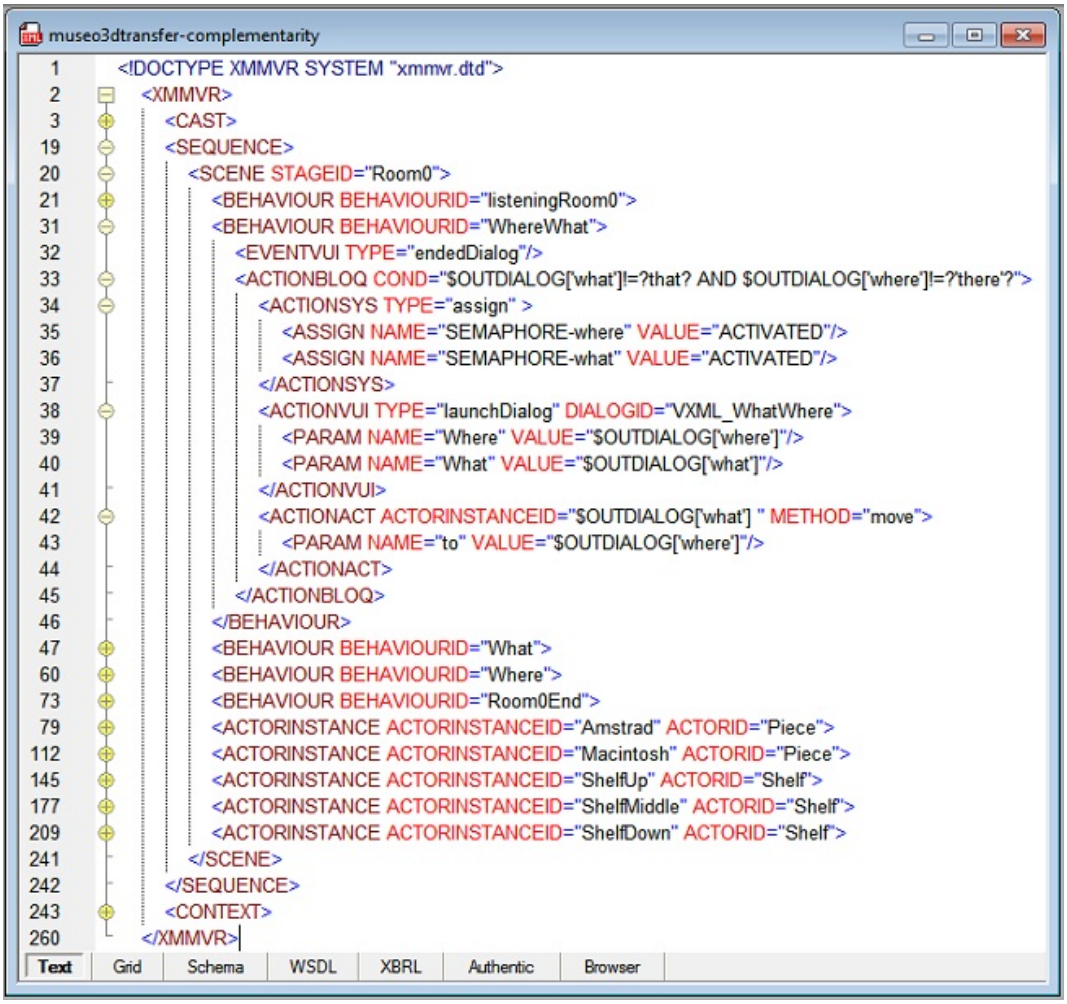

FIGURA 4.28: "Put Amstrad up" en el ejemplo basado en la cooperación por transferencia/redundancia (VUI+VUI)

la acción sobre la pieza que provocará que se mueva a la posición de la estantería indicada (ver línea 86 de la figura 4.30 y del listado A.42), se generará el diálogo correspondiente (ver línea 89 de la figura 4.30 y del listado A.42) y se harán las asignaciones correspondientes a los semáforos (ver línea 93 de la figura 4.30 y del listado A.42).

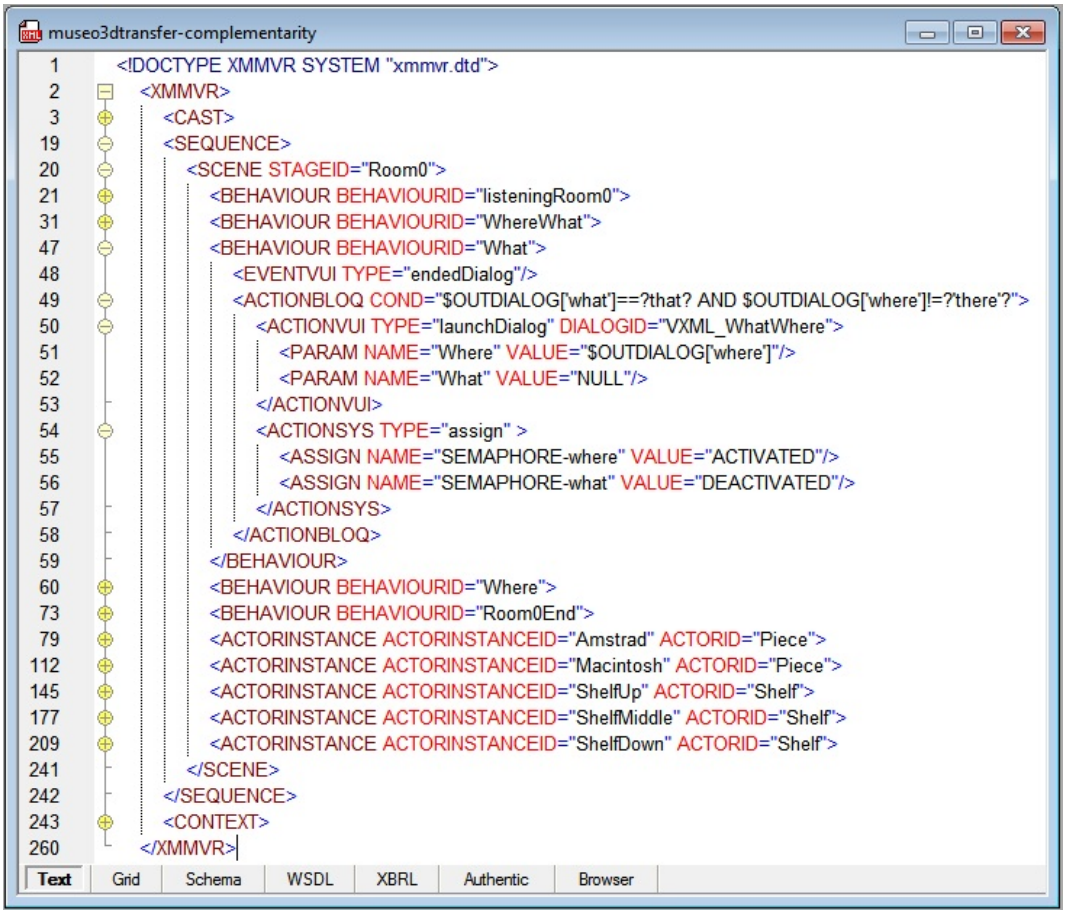

FIGURA 4.29: "Put that up" en el ejemplo basado en la cooperación por transferencia/redundancia (VUI)

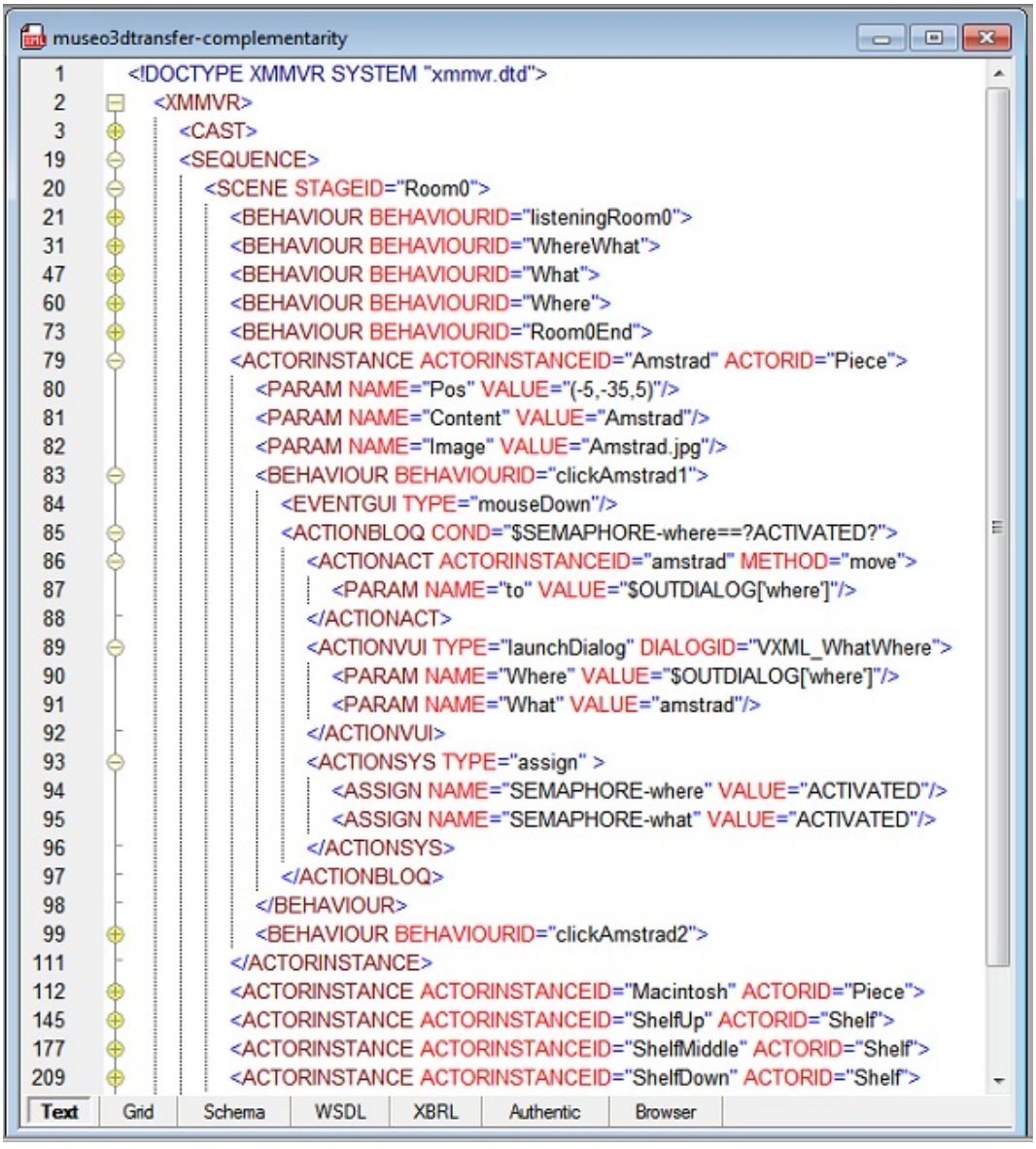

Figura 4.30: ”Put that up” en el ejemplo basado en la cooperación por transferencia/redundancia (GUI)

4.4.3.3. Transferencia-Complementariedad VUI+GUI

En este caso el usuario selecciona la pieza de manera vocal e indica que la quiere situar en la estantería de arriba seleccionando dicha estantería haciendo click sobre ella mientras dice la frase "Put Amstrad there". La captura de esta frase se especifica en la línea 62 de la figura 4.31 y del listado A.42 que comprueba que las variables de salida del diálogo "what" y "where" sean distintas de "that" y igual a "there" respectivamente. Desencadenará un diálogo que tendrá el parámetro "where" a "NULL" (ver línea 63 de la figura 4.31 y del listado A.42). Una acción del sistema "ACTIONSYS" activará la variable "SEMAPHORE-what" puesto que su valor ya ha sido recogido (ver línea 69 de la figura 4.31 y del listado A.42) y desactivará la variable "SEMAPHORE-where" (ver línea 68 de la figura 4.31 y del listado A.42).

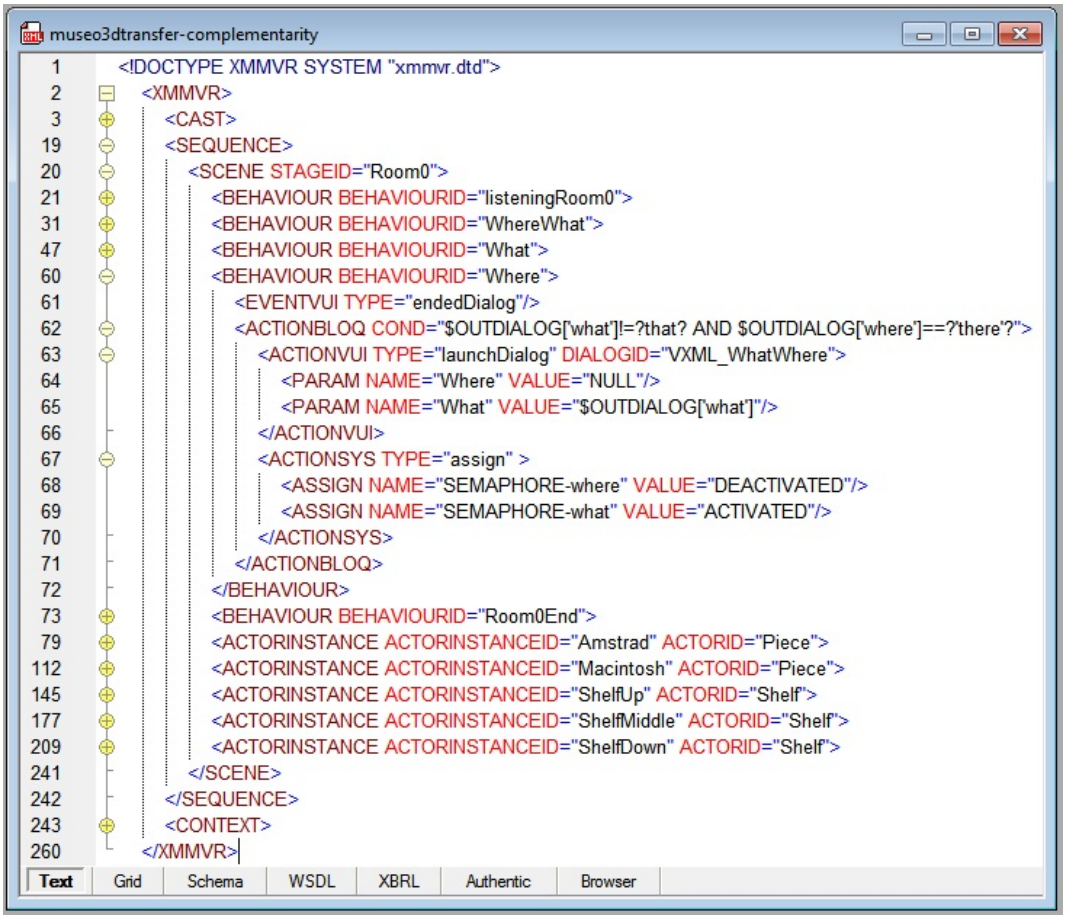

FIGURA 4.31: "Put Amstrad there" en el ejemplo basado en la cooperación por transferencia/redundancia (VUI)

La especificación del click se realiza en la línea 149 de la figura 4.32 y del listado A.42) donde se ejecuta la acción que mueve la pieza indicada a la estantería seleccionada (ver línea 152 de la figura 4.32 y del listado A.42), se genera el diálogo correspondiente (ver línea 155 de la figura 4.32 y del listado A.42) y se activan los semáforos (ver línea 159 de la figura 4.32 y del listado A.42).

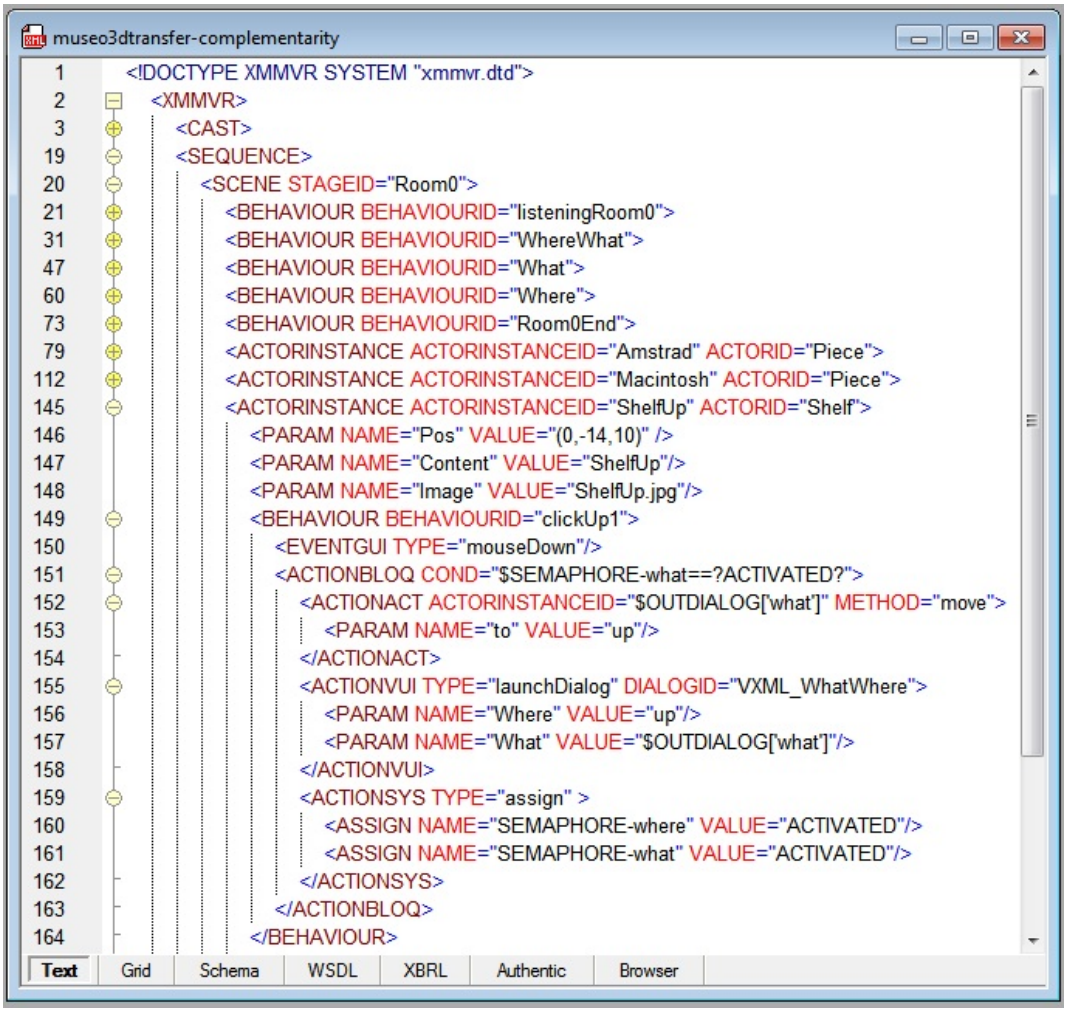

FIGURA 4.32: "Put Amstrad there" en el ejemplo basado en la cooperación por transferencia/redundancia (GUI)

4.4.3.4. Transferencia-Complementariedad GUI+GUI

En este caso el usuario selecciona la pieza haciendo click sobre ella. Tras ello, ya que la quiere situar en la estantería de arriba, hace click sobre dicha estantería. A la vez dice la frase "Put that there". Tendríamos que recoger los dos clicks.

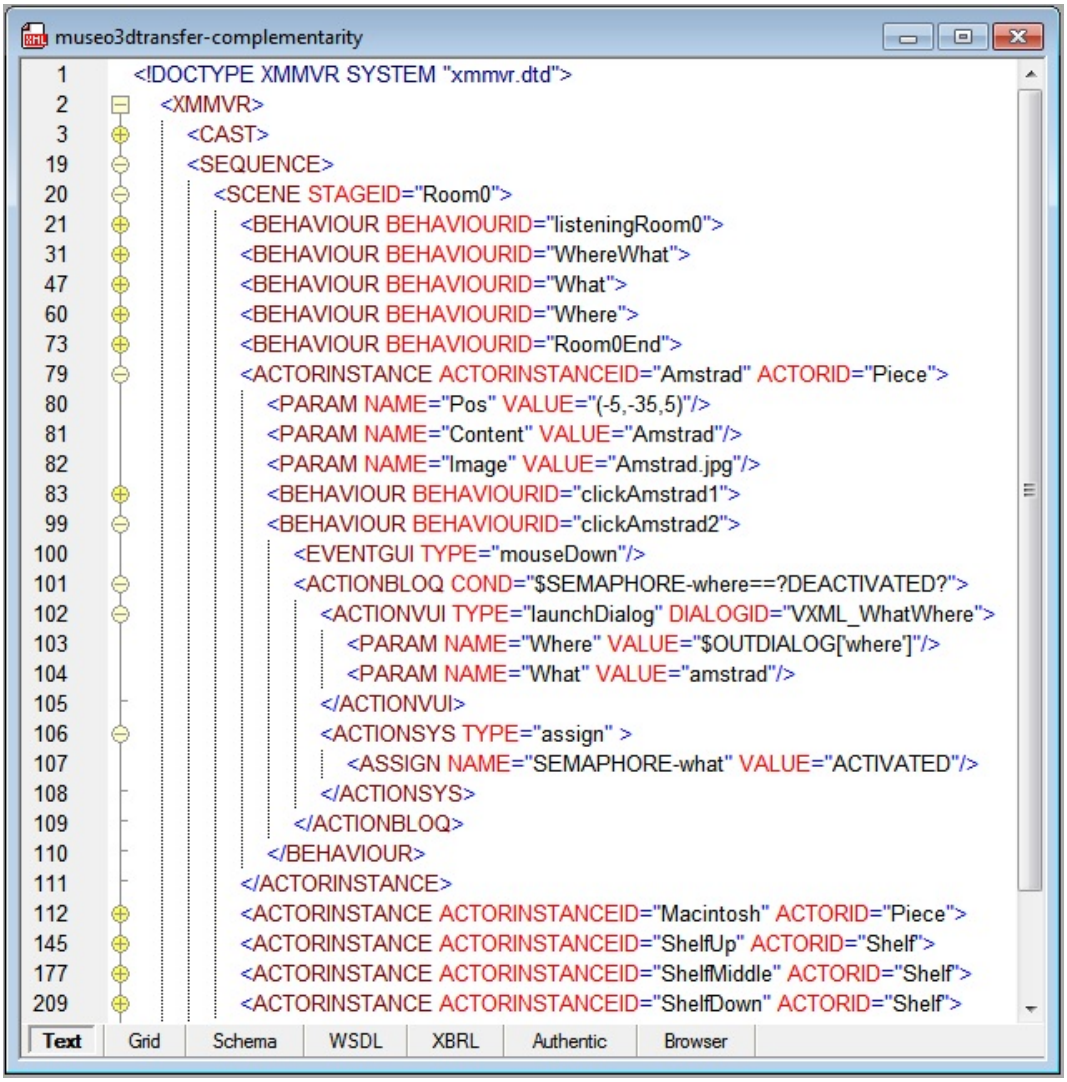

FIGURA 4.33: "Put that there" en el ejemplo basado en la cooperación por transferencia/redundancia (what)

El primer click se especifica en la línea 99 de la figura 4.33 y del listado A.42 que comprueba que esté desactivada la variable "SEMAPHORE-where". Desencadenará el diálogo correspondiente (ver línea 102 de la figura 4.33 y del listado A.42) para recoger el destino (where) que ya tendrá asignado el valor de la pieza (what=Amstrad). Se harán las asignaciones correspondientes a los semáforos (ver línea 106 de la figura 4.33 y del listado A.42). En este caso sólo se activará la variable "SEMAPHORE-what".

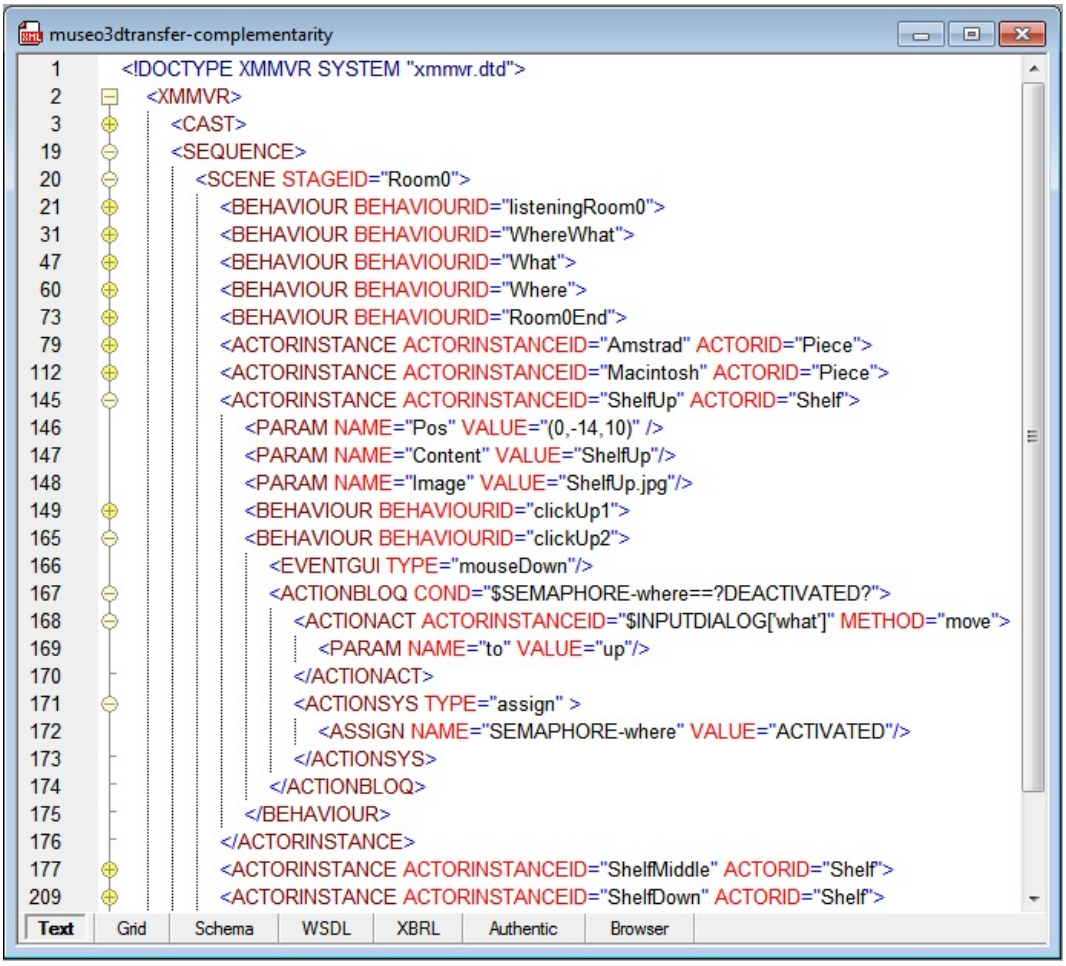

FIGURA 4.34: "Put that there" en el ejemplo basado en la cooperación por transferencia/redundancia (where)

TABLA 4.20: Ficheros del ejemplo ilustrativo de la cooperación entre modalidades por transferencia/complementariedad

XMMVR	''museo3dtransfer-complementarity.xml''
GRÁFICOS	''Room0.W3D'' ''Shelf.W3D'' ''Piece.W3D''
TEXTURAS	''Amstrad.jpg'' ''Macintosh_Plus.jpg''
VOCALES	''museo3dtransfer-complementarity_begin.vxml'' ''museo3dtransfer-complementarity_whatwhere.vxml'' ''museo3dtransfer-complementarity_end.vxml''
GRAMÁTICA	''transfer-complementarity.BNF''

El segundo click se especifica en la línea 165 de la figura 4.34 y del listado A.42 que comprueba que esté desactivada la variable "SEMAPHORE-where". Desencadenará la acción que provocará el movimiento de la pieza a la estantería seleccionada (ver línea 168 de la figura 4.34 y del listado A.42). Se harán las asignaciones correspondientes a los semáforos (ver línea 171 de la figura 4.34 y del listado A.42). La selección del resto de piezas y estanterías se realizaría de forma análoga en ambas modalidades como se indica en el listado A.42.

En este caso hemos necesitado un fichero XMMVR de 181 elementos XMMVR que acompañado de los ficheros gráficos, de texturas, vocales y de gramática relacionados en la tabla 4.20 serán suficientes para definir una escena que recrea la cooperación entre metáforas gráfica y vocal por transferencia/complementariedad en nuestro museo.

4.5 Conclusiones

En este capítulo hemos visto que nuestra propuesta es capaz de especificar cada metáfora gráfica. Las metáforas estructurales son desarrolladas rápidamente con nuestra solución y las metáforas navegacionales también pero con un gran esfuerzo de modelado 3D para que los ficheros de gráficos en los que se apoyan tengan el realismo suficiente como para provocar que el usuario reconozca el modo de desplazamiento representado por dicha metáfora navegacional. Las metáforas vocales también pueden ser implementadas fácilmente en base a definir ficheros vocales y de gramática adecuados. También hemos demostrado que la cooperación entre modalidades se puede implementar con nuestra propuesta. Para ello hemos presentado ejemplos de cada caso basados en nuestra aplicación ejemplo que demuestran las capacidades de nuestro lenguaje.

Tabla 4.21: Elementos XMMVR utilizados en la implementación de cada ejemplo

Metáfora de teatro	30
Metáfora de locomoción	60
Metáfora de habitaciones	76
Metáfora de elevador o rampa hidráulica	75
Metáfora de vehículo en vía o tren	90
Metáfora de deslizamiento o cable ferrocarril	46
Metáfora de silla voladora o alfombra	153
Metáfora de tele-transporte o beam me up	49
Metáfora de agente interfaz	51
Metáfora de proxy o delegado	101
Metáfora de divinidad	48
Metáfora de telekinesis	60
Cooperación por especialización	106
Cooperación por equivalencia o redundancia	227
Cooperación por transferencia o complementariedad	181

Como se puede apreciar en la tabla 4.21, si asociamos la complejidad de los ficheros con el número de elementos XMMVR utilizados en cada fichero de ejemplo, la complejidad de los ficheros XMMVR es relativamente sencilla al especificar metáforas fundamentales, incrementándose al implementar metáforas gráficas, manteniéndose al implementar las metáforas de interacción vocal y haciéndose más compleja cuando representamos la cooperación entre modalidades. Pero lo que también hemos visto en los ejemplos descritos es que las posibilidades de reutilización de código y recursos son enormes con nuestra propuesta. En el capítulo 5 describiremos cómo ha de ser la arquitectura software que soportará la puesta en marcha de los ejemplos descritos con XMMVR.

5

Arquitectura de la plataforma XMMVR

En capítulos anteriores se ha definido un lenguaje, se han mostrado ejemplos que muestran las capacidades del lenguaje para describir las metáforas de interacción gráficas y vocales así como sus posibilidades de cooperación. Para interpretar los ficheros XMMVR y poner en marcha los mundos virtuales descritos así como la multimodalidad deseada, se ha de definir una arquitectura software capaz de hacer estas funciones. En este capítulo se detallan los aspectos arquitectónicos destacando que nuestra propuesta no construye una aplicación desde cero si no que reutiliza tecnologías que cumplen los requisitos definidos. Estos requisitos se clasifican en funcionales y no funcionales. Los requisitos funcionales describen la funcionalidad o los servicios que se espera que el sistema provea (funcionalidad de alto nivel) y habrá también unos requisitos no funcionales que serán las propiedades emergentes del sistema como tiempo de respuesta, fiabilidad, portabilidad, etc. A continuación enumeraremos los requisitos funcionales que se presentan en la figura 5.1:

- Al iniciar el subsistema, éste automáticamente creará una jerarquía de planificación de eventos y un manejador para cada elemento de interés del mundo virtual y regularizará la visualización de este elemento con respecto a su estado inicial dentro de lo que sería el ”gestor del mundo”.
- Se necesitará un mecanismo de ”recepción de eventos” de forma que éstos puedan ser recibidos desde diferentes fuentes. Los eventos recibidos serán descompuestos en acciones de una complejidad arbitraria.
- Habrá que proporcionar un mecanismo de ”planificación” para la ejecución de las acciones de complejidad arbitraria procedentes de los eventos. Estas acciones podrán afectar a uno o varios elementos del mundo virtual.
- Se hará un ”tratamiento de eventos” en base a ficheros de especificación de escenas, diálogos y comportamientos que serán interpretados y cargados en memoria.
- Enviará una ”salida gráfica y/o vocal” y ejecutará las acciones procedentes de los eventos sobre los representantes de los elementos virtuales que actualice la representación visual.
- El subsistema almacenará la ”situación del mundo” manteniendo una ”concordancia estado mundo”.

Los requisitos no funcionales complementan a los anteriores y optimizan la funcionalidad deseada. Se detallan a continuación:

- El tiempo de respuesta a la llegada de eventos debe ser lo más breve posible, pero sin dejar de producir una respuesta óptima. Ya que se trata de una aplicación de realidad virtual. Uno de los factores más importantes que condicionan ese realismo es la acción en tiempo real (Vince, 1995).

Figura 5.1: Esquema de integración del marco de trabajo con una plataforma

- El subsistema debe ser multiplataforma para no condicionar a la aplicación que lo contenga.
- Debido a la posible naturaleza web de la aplicación, el subsistema debe ser lo más compacto posible para una rápida transferencia y carga en una máquina cliente.
- Ya que este subsistema se diseña con intención de incluirlo en una aplicación completa, es un punto muy importante la correcta documentación del mismo para que pueda ser entendido y aprovechado al máximo por las personas que vayan a hacer uso de él.
- Es necesario potenciar al máximo su grado de reutilización, en orden a que debe ser un subsistema que se pueda incluir en una aplicación completa sin necesidad de modificarlo.
- Debe presentar un gran nivel de encapsulación, o lo que es lo mismo, su dependencia con el mundo virtual que controlará debe ser mínima, ya que de lo contrario no sería fácil de adaptar a mundos diferentes y la finalidad del proyecto quedaría truncada.
- Debe exigir pocos requisitos hardware/software para poder ser ejecutado con resultados satisfactorios.

Como resultado del desarrollo de todos estos objetivos se debe obtener la aplicación adecuada (vía web) para el control de un mundo virtual por parte de un usuario, usando su voz y/o el teclado y ratón del ordenador. Se debe definir de una manera totalmente transparente al usuario, que no debe notar ni la distribución física, ni la lógica, ni el lenguaje de definición del mundo virtual.

5.1 Módulos de la arquitectura deseada

Para obtener la arquitectura que cumpla los requisitos descritos, hemos definido una serie de módulos a desarrollar necesarios para poder dar solución a la funcionalidad requerida y que pasamos a detallar a continuación.

5.1.1. Gestión de gráficos 3D

El módulo encargado de la gestión de gráficos 3D de la plataforma ha de tener la capacidad de definir y representar elementos 3D y gestionar archivos que almacenen esos elementos 3D con sus características y métodos asociados tanto para escenarios como actores permitiendo así el completo desarrollo de sus comportamientos gráficos asociados. Siguiendo el marco conceptual para gráficos interactivos descrito en (Foley, van Dam, Feiner & Hughes, 1990) estaríamos hablando de un sistema que integre el programa de aplicación y el sistema de gráficos que sustentan el marco referenciado. El modelo de aplicación lo constituirían los ficheros VRML, X3D o W3D que contienen los actores y escenarios. Debe permitirnos definir "Modelos 3D, luces y cámaras" así como permitirnos incluir "Texturas" en dichos elementos. También necesitaremos representar "Animaciones", definir "Grupos" de elementos 3D. Características como la representación de "Cinemática inversa", la posibilidad de representar "Dinámicas y partículas", etc. serían deseables también. Por supuesto, las posibilidades de interacción gráfica básicas (teclado y ratón) son indispensables, considerándose positivamente la posibilidad de hacer uso de "Dispositivos de interacción complejos". Conseguir "Visualización estereoscópica" y "Capacidades de pantalla completa para dispositivos HMD" serían interesantes posibilidades a añadir a nuestras necesidades. Se detallarán las funcionalidades básicas que ha de realizar este módulo y en el apartado 5.3 se ilustrará cómo se ejecutan en las arquitecturas implementadas.

- Representación y sustitución de escenarios: ante eventos provocados por ambos modos de interacción de usuario y situaciones tales como colisiones entre actores, según se especifique en el fichero XMMVR de la aplicación concreta a través de las etiquetas "ACTIONSYS" y "goToScene", se ha de poder ir de un escenario a otro es decir, cambiar el escenario. Serán utilizadas en aplicaciones basadas en metáforas de interacción gráfica tales como la metáfora de habitaciones (ver listado A.3), la de silla voladora (ver listado A.7) y la de tele-transporte (ver listado A.8) pero también en aplicaciones basadas en la metáfora vocal de proxy (ver listado A.14) y en aplicaciones donde se da la cooperación de modalidad gráfica y vocal por especialización y equivalencia-redundancia (ver listado A.35).
- Representación de actores: los objetos, entes animados, etc. que definimos en nuestros ficheros XMMVR con la etiqueta "ACTOR", deben ser situados en el mundo virtual de nuestras aplicaciones con la etiqueta "ACTORINSTANCE" de la forma que se indique permitiendo dar nuevas características en cada instancia de éstos, tal y como se puede ver en todos los ejemplos del capítulo 4.
- Cámara, movimientos y actor asociado: al movernos por el mundo y desplazarnos a lo largo de éste debemos poder dar al usuario una visión que se asemeje a la que le dan sus ojos al desplazarse por un mundo real. Esto se consigue usando cámaras subjetivas o representando al usuario con un actor dentro del mundo. En las diferentes implementaciones realizadas en los ejemplos basados en la metáfora de teatro (ver listado A.1) y locomoción (ver listado A.2) donde se define un actor "Camera" que al ser instanciado es denominado "User", se demuestra la capacidad de nuestro lenguaje para referenciar esta funcionalidad. Asimismo a través de eventos gráficos "EVENTGUI" de tipo "arrowUP", "arrowDOWN", "arrowLEFT" y "arrowRIGHT" definimos su desplazamiento.
- Colisiones entre actores: cuando nos desplazamos por el mundo virtual, algunos elementos no nos deberían permitir atravesarlos haciendo que nuestra experiencia sea así más realista y reflejando en el mundo virtual las leyes de la física que se dan en el mundo real. En el ejemplo que ilustra la metáfora gráfica navegacional de silla o alfombra voladora (ver listado A.7) definimos con la etiqueta "EVENTGUI" del tipo "collision" el salto a otra habitación cuando se produce un choque con una puerta (ver apartado 3.2.3.1).
- Interacción: por supuesto la sensibilidad de los gráficos 3D a eventos del ratón y teclado es básica para crear una interacción gráfica con el usuario. Por ello este módulo ha de ser capaz

de recoger esos eventos y hacer que se redirijan al módulo de la arquitectura que los asocie con las acciones que se indiquen en el fichero XMMVR definido. Con la etiqueta "EVENT-GUI" de tipo "mouseDown" en varias implementaciones de metáforas gráficas y cooperación entre modalidades gráfica y vocal especificamos esta funcionalidad (ver listados XMMVR del apartado 4.2 y listados XMMVR de 4.4 donde se permite seleccionar objetos de la manera gráfica descrita).

- Rutas: la capacidad de definición de rutas sobre las que se desplazaran algunos actores es necesaria para implementar metáforas navegacionales como vehículo en vía (ver listado A.5), deslizamiento (ver listado A.6) o silla voladora (ver listado A.7) donde se definen métodos "route" asociados a un evento gráfico y que siguen un itinerario definido.
- Interfaz web: para evitar una engorrosa instalación y buscando una arquitectura multiplataforma, este módulo debería ser capaz de representar el mundo virtual de la aplicación en un entorno web.

5.1.2. Gestión de diálogos

Este módulo se ha de encargar de lanzar los diálogos generados por el mundo virtual y/o sus elementos así como de interpretar los diálogos que genere el usuario. Enviará los eventos generados al módulo que ejecute las acciones asociadas a esos eventos vocales según se indique en el fichero XMMVR correspondiente al mundo descrito en cada aplicación.

- Lanzamiento de diálogos: según se indique en el fichero XMMVR mediante las etiquetas "ACTIONVUI" y "launchDialog" este módulo emitirá ciertos diálogos con las características indicadas. En algunos casos será un monólogo como en el caso de implementación de las metáforas de divinidad (ver listado A.20) y telekinesis (ver listado A.25). Pudiera ser un diálogo entre usuario y un actor del mundo para la implementación de la metáfora del agente interfaz (ver listado A.9) o un diálogo entre usuario y varios actores del mundo si implementamos la metáfora de proxy o delegado (ver listado A.14). Además, se debe asociar a los diferentes agentes en el mundo virtual con los que interlocuta el usuario diferentes voces según las características de éstos (por ejemplo asociar una voz femenina a un actor de ese género).
- Parada de diálogos: ante ciertos eventos se debe parar los diálogos que se estén emitiendo ya que podría ser que no tuvieran sentido una vez producidos dichos eventos. Esto se indicaría en el fichero XMMVR con las etiquetas "ACTIONVUI" y "stopDialog".
- Comunicación con los diálogos: los diálogos han de permitir obtener unos datos y eventos asociados y de cara a reutilizar diálogos han de permitirnos que les enviemos ciertos parámetros según se define en el fichero XMMVR con la etiqueta "DIALOG" (ver apartado 3.2.4) y sus parámetros tal y como podemos ver en los ejemplos de los listados A.9, A.14, A.20, A.25, A.30, A.35 y A.42.

5.1.3. Gestión de comportamientos

El mundo virtual definido y sus elementos están regidos por la asociación de eventos y acciones especificadas en el fichero XMMVR asociado. Este módulo se encargará de ejecutar dichas acciones implementando una serie de mecanismos que se describen a continuación y apoyándose en la capacidad del sistema para ejecutar funciones o métodos empleando literales es decir, strings con los nombres de las funciones o métodos.

- Gestión de eventos: se debe dar una recepción centralizada y se debe crear, mantener y gestionar una cola o sistema de colas de eventos para garantizar el correcto tratamiento de éstos y así garantizar los requisitos de rapidez de respuesta y coherencia del estado del mundo ya que si no fuera así perderíamos eventos o no mantendríamos un historial coherente del

devenir de nuestro mundo virtual. En el fichero XMMVR se especifican con las etiquetas "EVENTGUI", "EVENTVUI", EVENTSYS" y "EVENTACT" tal y como se puede ver en todos los ejemplos del capítulo 4 y se introdujo en el apartado 3.2.3.1.

- Resolución de acciones: cada acción o grupo de acciones asociados a cada evento suponen la ejecución de código que modifique la apariencia del mundo virtual, la emisión de nuevos diálogos, etc. En el fichero XMMVR se especifican entre las etiquetas "ACTIONBLOQ" tal y como se puede ver en todos los ejemplos del capítulo 4 y se introdujo en el apartado 3.2.3.2 .
- Resolución de expresiones lógicas: en los ficheros XMMVR para dar más dinamismo, permitir la reutilización y en aras de simplificar los ficheros XMMVR se implementan ciertas condiciones asociadas a eventos que han de ser decodificadas por este módulo para la correcta ejecución. Si no lo hiciésemos de esta forma, los ficheros ejemplo de implementación de cooperación entre modalidades serían mucho más largos y complejos (ver listados A.30, A.35 y A.42 y apartado 3.2.3.2.2).

5.1.4. Control del estado

Además de los cambios en el mundo virtual y la generación de diálogos producidos según la llegada de eventos y como resultado de la ejecución de sus correspondientes acciones, se debe mantener un histórico de lo sucedido y la situación del mundo en todo momento. Necesitamos una tabla de símbolos para las variables del mundo donde guardemos el estado de cada una de ellas. Esto nos permitirá, entre otras cosas, implementar la cooperación entre modalidades por transferencia o complementariedad. Asimismo necesitaremos poder implementar semáforos para facilitar la implementación de cooperaciones entre modalidades por equivalencia o redundancia (ver listados A.35 y A.42 y apartado 3.2.3.2.1).

5.1.5. Gestión de ficheros XMMVR/Intérprete XML

La información del fichero XMMVR ha de estar residente en memoria durante todo el tiempo que nuestra aplicación se esté ejecutando. Para ello hemos de leer este fichero y guardarlo en la estructura de datos correspondiente. Por ello, es necesario disponer de funciones de interpretación de ficheros XML.

5.1.6. Comunicación entre los mundos

Como se ha visto, la plataforma consta, conceptualmente hablando, de dos partes muy bien definidas que deben comunicarse entre sí. La parte gráfica que representa el escenario o mundo virtual junto a los actores con los que el usuario va a interactuar gráficamente, y la parte vocal que es el mecanismo usado por dicho usuario para interactuar vocalmente con el mundo y los actores virtuales. Esta necesidad de comunicación entre ambas partes da lugar a definir un elemento específico. Se debe encontrar una solución que permita que dos aplicaciones distintas que no comparten, en principio, ninguna característica de implementación puedan intercambiar información. Es decir, que dos filosofías diferentes (la gráfica y la vocal) se complementen pudiéndose relacionar con algún estándar o protocolo de cara a conseguir la fusión multimodal. Este objetivo hace surgir una nueva capa en nuestra arquitectura.

Así, la capa de comunicación entre la parte gráfica y la parte vocal es la encargada de posibilitar el intercambio de información entre ambas partes. La aplicación gráfica recoge la información de usuario y se la envía a la parte vocal a través de la capa de comunicación y a través de ésta la parte vocal la envía a la aplicación gráfica la información de usuario recogida. La capa de comunicación se encargará de formatear la información a un "lenguaje" común para ambas partes. Posteriormente deberá transformarla en información que la aplicación del otro lado entenderá. Es decir, traduce

del mundo gráfico al lenguaje estándar de comunicación y de éste al mundo vocal y viceversa, compartiendo la información según un estándar interno.

5.2 Arquitectura final

El esquema de la arquitectura deseada se representa en la figura 5.2.

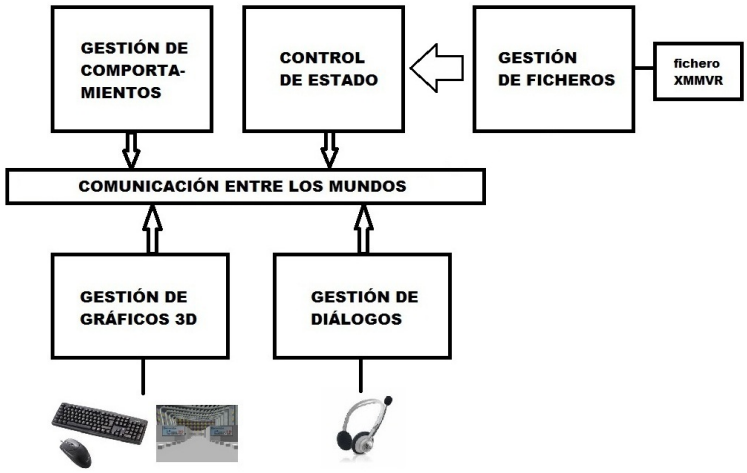

Figura 5.2: Esquema de la arquitectura XMMVR

En el capítulo 2 introdujimos la propuesta del grupo MMI del W3C para cualquier arquitectura multimodal donde se indicaban los elementos que deberían aparecer o constituents que recordamos a continuación:

- Interaction Manager: coordina las diferentes modalidades. Sería el Controlador en el paradigma MVC [1].
- Data Component: provee el modelo de datos común y sería el Modelo en el paradigma MVC.
- Modality Components: proveen capacidades de interacción específicas de cada modalidad. Serían las vistas en el paradigma MVC.
- Event transport layer: provee el canal de comunicación entre el Interaction Manager y los Modality Components.
- Runtime Framework: provee la infraestructura básica y habilita la comunicación entre el resto de los constituents.

Existe un paralelismo entre nuestra propuesta de arquitectura y la del grupo MMI ya que los módulos "Gestión de comportamiento" y "Control de estado" realizan tareas propias del "Interaction Manager" al coordinar las modalidades. Nuestros ficheros XMMVR y el módulo de "Gestión de ficheros" serían el "Data Component" ya que proveen el modelo de datos común a nuestras aplicaciones basadas en la arquitectura descrita. Los módulos "Gestión de gráficos 3D" y "Gestión de diálogos"

[1] Según se describe en la propuesta de la arquitectura MMI (Barnett, Dahl, Kliche, Tumuluri, Yudkowsky, Bodell, Porter, Raggett, Raman & Wahbe, 2008)

serían los "Modality Components" para la modalidad gráfica y para la modalidad vocal respectivamente. El módulo descrito como "comunicación entre los mundos" en nuestra arquitectura sería la "Event transport layer" de la propuesta del grupo MMI ya que es el mecanismo de comunicación entre los módulos descritos. Todos los módulos de nuestra arquitectura se corresponderían con el "Runtime Framework" de la propuesta del MMI. La relación de los módulos de nuestra arquitectura deseada con los módulos de la arquitectura propuesta por el grupo MMI podría representarse en la figura 5.3 según lo descrito.

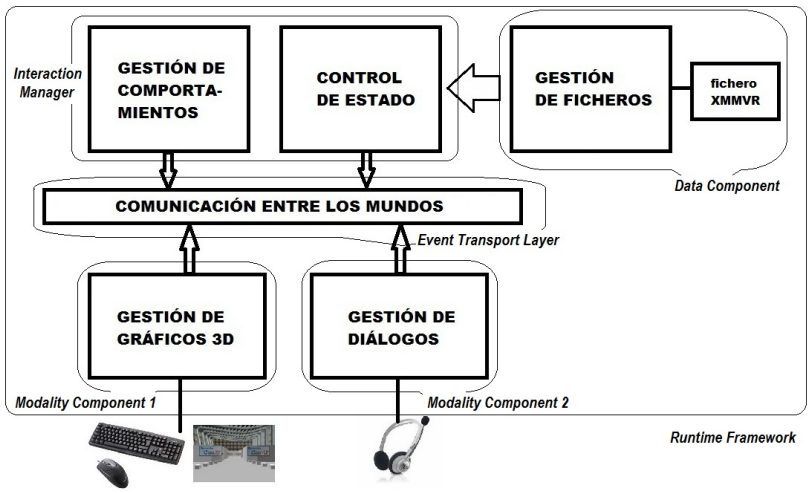

FIGURA 5.3: Esquema de la arquitectura XMMVR y relación con la arquitectura MMI

Siguiendo esta arquitectura hemos desarrollado y en el futuro desarrollaremos aplicaciones basándonos en las tecnologías existentes en cada momento pero siempre con las metas básicas de diseño que motivaron la definición del estándar de la arquitectura MMI:

- Encapsulación. La arquitectura no debería hacer asunciones acerca de la implementación interna de los componentes, que se tratarán como cajas negras.
- Distribución. La arquitectura debería soportar implementaciones distribuidas y co-alojadas.
- Extensibilidad. La arquitectura debería facilitar la integración de componentes para nuevas modalidades. Por ejemplo, dada una implementación existente con componentes vocales y gráficos, sería posible añadir un nuevo componente (por ejemplo, un componente de seguridad biométrica) sin modificar los componentes existentes.
- Recursividad. La arquitectura debería permitir anidamiento, de manera que una instancia del framework consistente de varios componentes podría ser empaquetada hasta aparecer como un solo componente en una instancia de más alto nivel en la arquitectura.
- Modularidad. La arquitectura debería proveer una separación entre datos, control y presentación.

5.3 Ejemplos de Arquitecturas

Para poner en práctica la arquitectura propuesta, se han realizado dos prototipos con diferentes tecnologías. La primera implementación se basó en el navegador VRML, CORTONA VRML VIEWER que utilizaba un API EAI basado en la Máquina Virtual Java de Microsoft, pero debido a la

demanda por parte de Sun, dejó de estar soportada (Microsoft, 2012). Por ello, aprovechando la experiencia en desarrollo de aplicaciones 3D con Adobe Director, se ha implementado la arquitectura propuesta con dicha tecnología. Ambas implementaciones comparten el mismo módulo de gestión de diálogos basado en la plataforma vocal VERBIO (Verbio, 2012) que implementa un navegador VoiceXML de desarrollo propio.

5.3.1. Arquitectura basada en CORTONA VRML VIEWER

A continuación describiremos cómo se implementan los módulos definidos en la arquitectura basada en CORTONA VRML VIEWER (Cortona, 2010) que se representa en la figura 5.4. Debido a la necesidad de usar Java para desarrollar programas basados en el API EAI (Phelps, 2010) que provee Cortona, toda la arquitectura descrita a excepción de la gestión de diálogos quedará embebida en un applet que a su vez estará dividido en varios paquetes que ejecutan las funciones de los módulos descritos a continuación.

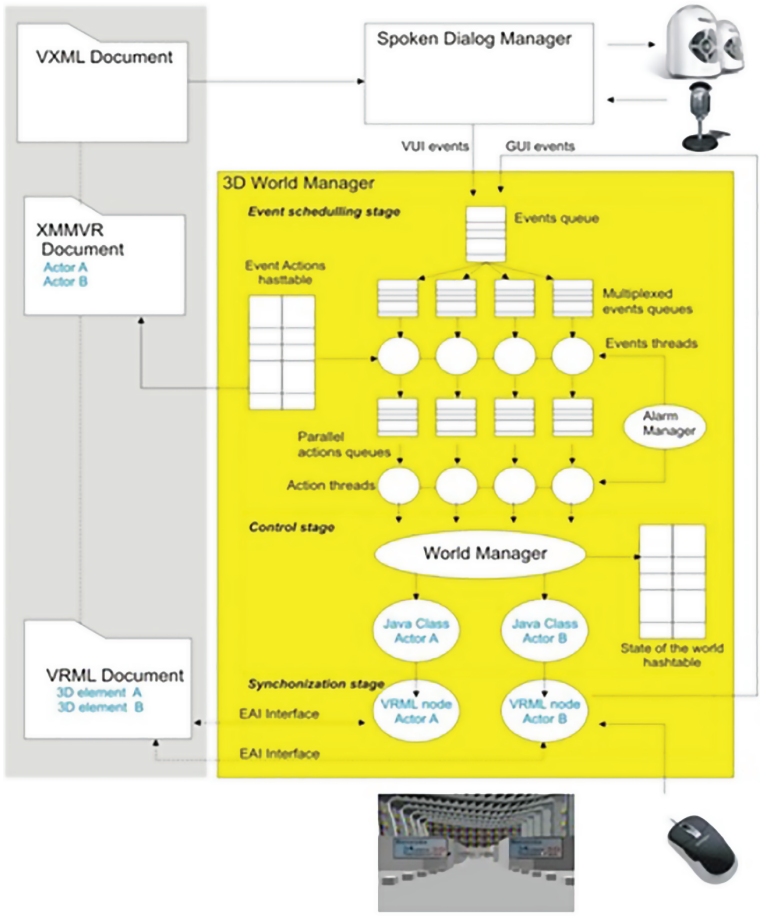

FIGURA 5.4: Esquema de la arquitectura XMMVR basada en CORTONA VRML VIEWER

Gestión de gráficos 3D: El módulo encargado de la gestión de gráficos 3D de nuestra plataforma basada en CORTONA VRML VIEWER con su API EAI (Phelps, 2010) tiene las funcionalidades básicas que ha de realizar este módulo que introducimos a continuación.

- Representación y sustitución de escenarios: se carga el primer escenario al iniciarse el applet y se presenta en el navegador VRML embebido en la interfaz gráfica definida en el propio applet.
- Representación de actores, movimientos, rutas y colisiones entre ellos: cada actor tiene su fichero VRML asociado que se representa en el navegador VRML que tiene implementadas por defecto las capacidades requeridas.
- Cámara y actor asociado a ésta: el navegador VRML tiene implementadas por defecto las capacidades requeridas.
- Interacción: nos da las capacidades requeridas ya que las tiene implementadas por defecto recogiendo eventos de teclado y ratón u otra interfaz similar. Además, se ha desarrollado en el applet correspondiente basado en el API EAI los mecanismos de recogida de eventos pertinentes.
- Interfaz web: el plug-in sólo funciona con Microsoft Internet Explorer 6 con la máquina virtual Java de Microsoft.

Gestión de diálogos: Este módulo es común en ambas plataformas pero la comunicación y el envío del primer diálogo se realizan de diferente forma.

- Lanzamiento de diálogos: el primer diálogo lo envía el applet vía socket a la plataforma vocal.
- Comunicación con los diálogos: los ficheros VoiceXML lanzados a medida que se produce la respuesta del usuario, llaman a otro servlet que vía socket envía los datos recogidos de dicha respuesta al applet. Para el paso de parámetros a los ficheros VoiceXML se aprovechan las capacidades propias de VoiceXML de asignar valores a variables de los ficheros VoiceXML llamados a través de la etiqueta ”subdialog”.

Gestión de comportamientos: Este módulo integrado en el applet desarrollado, se encargará de la asociación de eventos y acciones especificadas en el fichero XMMVR asociado. Sus funciones son:

- Gestión de eventos: se implementa un sistema de colas para gestionar los eventos y compararlos con la hashtable ”Configuración” donde se ha almacenado el contenido del fichero XMMVR, es decir los posibles eventos y sus acciones asociadas.
- Resolución de acciones: cada instancia de actor tiene una clase asociada con los métodos correspondientes.

Control del estado: Se debe mantener un histórico de lo sucedido y la situación del mundo en todo momento. Necesitamos una tabla de símbolos para las variables del mundo donde guardemos el estado de cada una de ellas. Mantenemos todo en la hashtable ”Estado del mundo” o ”State of the world” declarada con el lenguaje de programación Java y que se representa en la figura 5.4.

Gestión de ficheros XMMVR/Intérprete XML: Usamos el API SAX (Megginson, 2010) implementado en lenguaje Java para leer el fichero y cargarlo en la hashtable ”Configuración” o ”Event actions” en la figura 5.4.

Comunicación entre los mundos: Ha de realizarse a través de elementos tales como servlets y sockets que nos permiten intercambiar información entre los módulos vistos. Para ello nos servimos de un servidor Apache Tomcat (Apache Foundation, 2012) que hace las veces

de almacén de nuestros ficheros de especificación de gráficos, diálogos y comportamiento y alberga los ficheros de aplicación con los scripts desarrollados. En las respuestas de usuario a los diálogos se llamará a servlets que ejecutarán sockets contra el applet para enviar los datos obtenidos.

5.3.2. Arquitectura basada en ADOBE DIRECTOR SHOCKWAVE

Describiremos cómo se implementan los módulos definidos en la arquitectura basada en ADOBE DIRECTOR SHOCKWAVE que se muestra en la figura 5.5. Se va a emplear metodología muy ligada al programa que puede resultar complicada para los lectores no familiarizados con éste por lo que se recomienda la lectura de la documentación facilitada en (Adobe, 2010a).

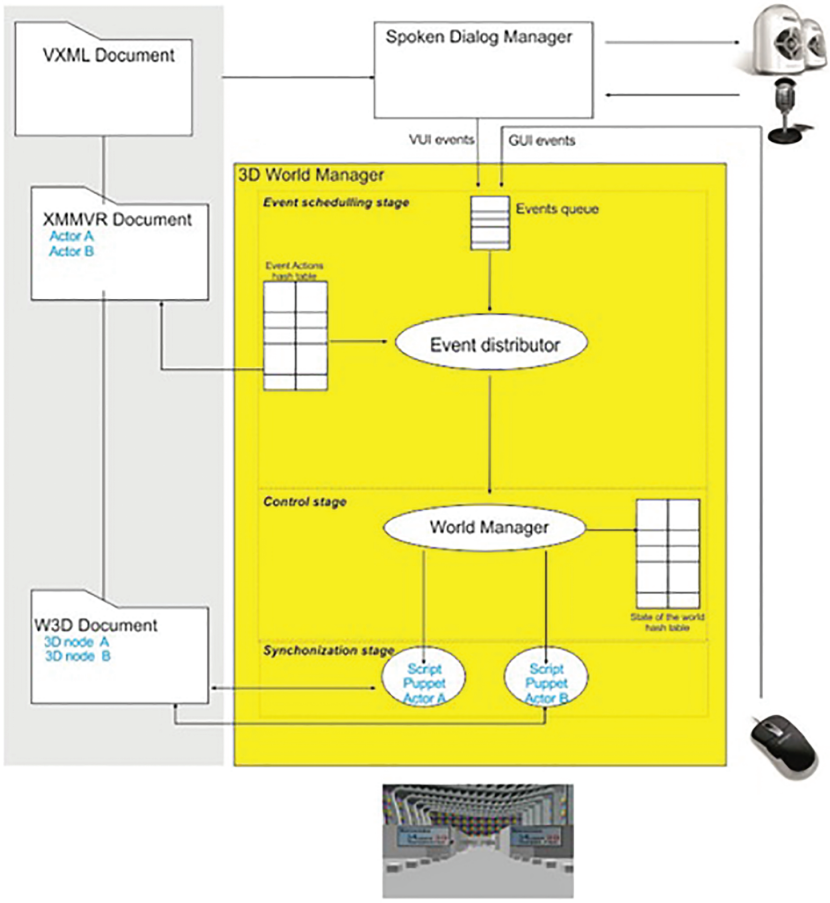

Figura 5.5: Esquema de la arquitectura XMMVR basada en SHOCKWAVE

A diferencia de la solución anterior que integraba toda la aplicación en un applet Java, la solución aquí implementada será parte de un módulo Adobe Director Shockwave que tendrá un "internal cast" donde se definirán básicamente los elementos de control, planificación, gestión de ficheros y gestión de gráficos 3D. Por cada "actor" del mundo habrá un "external cast" asociado que

se emplea para hacer programación orientada al objeto siguiendo la filosofía descrita pero siendo ahora scripts programados en el lenguaje de programación Lingo propio de Adobe Director para que cada actor pueda realizar su propia funcionalidad. Se emplean recursos propios de Adobe Director como el acceso a elementos gráficos y ficheros externos junto a la comunicación con aplicaciones externas intentando así dar cobertura a la arquitectura propuesta.

Gestión de gráficos 3D: ADOBE DIRECTOR SHOCKWAVE es una aplicación para desarrollo de entornos multimedia que incluye, desde la versión 8.5 la posibilidad de incluir elementos 3D y de gestionar la interacción con ellos a través del lenguaje de programación. Aprovechándonos de esta posibilidad hemos desarrollado el módulo encargado de la gestión de gráficos 3D de nuestra plataforma que nos permite realizar sus funcionalidades básicas de la manera siguiente.

- Representación y sustitución de escenarios: en el "internal cast" tendremos scripts Lingo que generarán el mundo virtual en base a los ficheros W3D definidos en el propio "cast" y referenciados en el fichero XMMVR para cargarse y representarse en tiempo de ejecución. En el "stage" hay un "sprite" que contiene un escenario genérico. Para cargar un nuevo escenario asociamos a dicho sprite el "3D cast member" del nuevo escenario. Se carga el primer escenario al iniciarse la película cuando el fichero XMMVR es interpretado. Se pueden cargar nuevos escenarios según se produzcan acciones del tipo "ACTIONSYS/goToScene". Se emplean ficheros W3D para representar los escenarios que se modelan directamente en AUTODESK 3D STUDIO MAX (Autodesk, 2010) y se guardan en el formato W3D. Cuando el interprete de XMMVR encuentra un escenario carga el fichero W3D correspondiente en el "internal cast".
- Representación de actores: cada actor tiene su fichero W3D asociado y programamos sus comportamientos a través de los scripts Lingo. Al igual que los escenarios, los actores XMMVR se modelan en AUTODESK 3D STUDIO MAX y tendrán asociado un fichero W3D que se carga en forma de "3D cast member" de Adobe Director cada vez que se emplee este elemento. Además, la lógica del actor precisa de la presencia un "external cast" específico como se explica en la sección de comportamientos de este apartado. Los parámetros y propiedades declarados en el actor XMMVR se actualizan desde el "external cast" cuando el actor XMMVR se representa en un escenario. Para representar un actor XMMVR en un escenario, se añaden los modelos del "3D cast member" correspondiente al "3D cast member" que está en el sprite principal del escenario.
- Representación de rutas, movimientos y colisiones entre actores: el lenguaje de programación Lingo de Adobe Director permite acceder directamente a las propiedades de los miembros gráficos de los "3D cast member", entre ellas la posición y la esfera envolvente. Es necesario programar los métodos asociados a los comportamientos de los actores XMMVR tal y como se explica más adelante en el apartado "Gestión de comportamientos".
- Cámara y actor asociado: resolvemos este requisito a través de los scripts Lingo desarrollados para tal efecto. Cada "3D cast member" tiene al menos un elemento cámara incluido. Esta cámara tiene propiedades de posición, orientación y enfoque que pueden ser manipuladas directamente desde el lenguaje y cuyo efecto es inmediato. Por tanto, los comportamientos y propiedades asociadas al actor cámara de XMMVR son proyectadas directamente a la cámara activa del "3D cast member" que esté en cada momento en el escenario activo.
- Interacción: resolvemos este requisito a través de los scripts Lingo desarrollados para tal efecto. Adobe Director gestiona los eventos del ratón y del teclado. Además, informa de la posición del ratón y del "sprite" gráfico más cercano a la cámara sobre el que se ha hecho click.

- Interfaz web: el archivo Adobe Director Shockwave generado para ejecutarse desde un navegador con el plugin Adobe Director Shockwave o su ejecutable equivalente accede a ficheros donde se comparten datos para permitir la interacción con los otros módulos de la arquitectura. El uso de ficheros externos implica retardos debido a que hasta que no están cargados el programa no es operativo.

Gestión de diálogos: Este módulo es común en ambas arquitecturas pero varía la forma en la que se produce el primer diálogo y la comunicación con la plataforma vocal.

- Lanzamiento de diálogos: el primer diálogo lo envía el Adobe Director Shockwave llamando a un servlet utilizando el método "PostNetText" (Adobe, 2010h) propio del lenguaje Lingo. Dicho servlet indica el fichero de diálogo que debe lanzar el intérprete de VoiceXML
- Comunicación con los diálogos: los ficheros VoiceXML lanzados a medida que se produce la respuesta del usuario, llaman a otro servlet que escribe en un fichero los datos recogidos de dicha respuesta y que serán leídos por el Adobe Director Shockwave que estará monitorizándolo constantemente. Para el paso de parámetros a los ficheros VoiceXML se aprovechan las capacidades propias de VoiceXML de asignar valores a variables de los ficheros VoiceXML llamados a través de la etiqueta "subdialog".

Gestión de comportamientos: Este módulo se encargará de la asociación de eventos y acciones especificadas en el fichero XMMVR asociado implementando una serie de mecanismos en los scripts en Lingo.

- Gestión de eventos: a partir del fichero XMMVR se ha creado una tabla donde se especifica la relación entre eventos y acciones relativas a cada actor. Una vez recibido el evento, se interpreta identificando su tipo, se consulta la tabla, se identifica el actor al que hace referencia y se invoca el método correspondiente. Se utilizan los recursos de Adobe Director de identificación de eventos y de llamada a métodos empleando el string con el nombre del mismo.
- Resolución de acciones: cada actor XMMVR tiene un "external cast" asociado. En este "external cast" habrá dos members: un "3D cast member" y un script de tipo "parent". Los scripts parent se utilizan para hacer programación orientada al objeto. Significa que se pueden crear múltiples instancias (child scripts) empleando el mismo "external cast". Esto se empleará para crear las instancias de los actores XMMVR que están en el escenario. El script contiene las propiedades externas (parámetros del actor que aparecen en el fichero XMMVR) y otras propiedades o variables de uso interno en la clase. El constructor de la clase se encarga de representar el modelo 3D en el escenario empleando los valores de los parámetros declarados en el fichero XMMVR. Un método específico se debe encargar de eliminar los modelos 3D del "3D cast member" del escenario. En el parent script se definen los métodos del actor que hayan sido declarados en el documento XMMVR y otros que se puedan necesitar.
- Resolución de expresiones lógicas: Adobe Director permite interpretar strings como comandos por lo que su uso sería inmediato.

Control del estado: Se define con el lenguaje de programación Lingo una tabla donde se almacena el valor de cada variable declarada en el fichero XMMVR para mantener un histórico de lo sucedido y la situación del mundo en todo momento basándose en las estructuras de información acerca de los actores y escenarios que componen el mundo.

Gestión de ficheros XMMVR/Intérprete XML: El fichero XMMVR puede estar residente en memoria durante todo el tiempo que nuestra aplicación se esté ejecutando. Para ello hemos de leer este fichero y guardarlo en la estructura de datos correspondiente. En el "Internal

cast" usamos el xtra "xmlparser" que a través scripts Lingo lee el fichero XMMVR y carga con el contenido del fichero varias hashtables que son estructuras list de Lingo. Se generarán estructuras de datos donde estará la relación evento-acciones en base al fichero XMMVR definido y otras con información acerca del estado de los actores, escenarios, etc. que componen el mundo en todo momento durante la ejecución del programa. La primera servirá para que los eventos sean tratados correctamente y generen las acciones correspondientes y las siguientes para mantener el "estado del mundo".

Comunicación entre los mundos: Ha de realizarse a través de elementos tales como servlets y ficheros que nos permiten intercambiar información entre los módulos vistos. Para ello nos servimos de un servidor Apache Tomcat (Apache Foundation, 2012) que hace las veces de almacén de nuestros ficheros de especificación de gráficos, diálogos y comportamiento y alberga los ficheros de aplicación con los scripts desarrollados. Estos servlets escribirán las respuestas de usuario a los diálogos en ficheros que leerá el Adobe Director Shockwave. Se definirán además scripts que gestionen la interacción gráfica y la interacción vocal: los primeros íntegramente codificados en Lingo y los segundos comunicándose con la plataforma vocal basada en VoiceXML.

Tabla 5.1: Arquitectura MMI, XMMVR, CORTONA y SHOCKWAVE

MMI	XMMVR	CORTONA VRML VIEWER	ADOBE DIRECTOR SHOCKWAVE
DATA COMPONENT	Gestión del fichero XMMVR	API SAX	xtra "xmlparser"
INTERACTION MANAGER	Gestión de comportamiento Gestión de estado	Jerarquías de planificación y control	getNetText() gotoNetPage() netTextResult() postNetText() Internal cast
MODALITY COMPONENT 1	Gestión de gráficos 3D	CORTONA VRML VIEWER y API EAI	Dolly Camera Pan Camera Horizontal Keyboard Input External cast
MODALITY COMPONENT 2	Gestión de diálogos	VERBIO y VoiceXML	VERBIO y VoiceXML
EVENT TRANSPORT LAYER	Comunicación entre los mundos	SERVLET y SOCKETS	SERVLET y FICHEROS

En la tabla 5.1 y observando las figuras 5.4 y 5.5 vemos que la implementación de la arquitectura y la definición de ésta tienen una correspondencia con los módulos descritos en la figura 5.3. A todos los niveles la Gestión del fichero XMMVR es el Data Component de la arquitectura que se implementa a través del API SAX en la arquitectura basada en CORTONA y a través del xtra "xmlparser" en la arquitectura basada en Adobe Director Shockwave. El Interaction Manager que se encarga de la Gestión de comportamiento y Gestión de estado, se implementa a través de las jerarquías de planificación y control definidas en la arquitectura basada en CORTONA y a través de

scripts Lingo que utilizan funciones propias de Adobe Director en la arquitectura basada en Adobe Director Shockwave. Algo similar ocurre con la Gestión de gráficos 3D que podríamos asumir como el Modality Component 1 ya que en la arquitectura basada en CORTONA se implementa con aplicaciones propias mientras que en la arquitectura basada en Adobe Director Shockwave son funciones propias de Adobe Director las que se encargan de ello. Para el Modality Component 2 al que asignamos la Gestión de diálogos optamos en ambas arquitecturas por la plataforma vocal basada en VERBIO (Verbio, 2012) y su intérprete VoiceXML. La capa de comunicación o Event Transport Layer se implementa con sevlets en ambas arquitecturas pero además se usan sockets para implementarla en la arquitectura basada en CORTONA en lugar de los ficheros usados en la arquitectura basada en Adobe Director Shockwave.

Vemos que de esta manera se consigue el objetivo de modularidad y encapsulamiento en cada módulo de la arquitectura pero permitiendo a la vez que dichos módulos sean extensibles de cara a la aparición de nuevas funcionalidades haciendo incluso posible una recursividad en su implementación.

Se pueden observar varios videos de demo en (Olmedo-Rodríguez, 2010a) y (Olmedo-Rodríguez, 2010b) para la arquitectura basada en CORTONA y la arquitectura basada en Adobe Director Shockwave respectivamente.

5.4 Conclusiones

En este capítulo hemos revisado las necesidades de una arquitectura que soporte nuestra propuesta y cada uno de los módulos que la componen con sus funciones. Tras ello hemos visto que estos módulos encajan con la propuesta de arquitectura para aplicaciones multimodales del grupo MMI del W3C. Finalmente hemos presentado dos implementaciones que usan tecnologías distintas pero que son válidas para definir aplicaciones multimodales que permiten interacción con mundos 3D según nuestra propuesta.

Respecto al cumplimiento de la especificación del grupo MMI del W3C para toda aplicación multimodal, hemos de matizar que a día de hoy ninguna de las aproximaciones tecnológicas de la arquitectura presentada lo cumplen al 100 %. Sólo nos hemos basado en su filosofía de diseño modular, quedando pendiente para el futuro el cumplimiento de los estándares que propone para protocolos de comunicación entre módulos y tratamiento de eventos.

En el siguiente capítulo para dar más fuerza a lo presentado en éste, realizaremos la evaluación de una de las aplicaciones desarrolladas usando una de las aproximaciones tecnológicas para la arquitectura presentada.

6

Evaluación de un escenario de uso

En este capítulo se exponen los resultados de evaluar la experiencia de usuario al interactuar con la aplicación ejemplo Museo 3D Multimodal realizada basándonos en nuestra propuesta y ya descrita en el capítulo 4. Como se presentó en el capítulo 2, la evaluación de usabilidad de aplicaciones en entornos virtuales es todavía un campo de investigación poco desarrollado, con las excepciones de trabajos tales como los mostrados en 2.7.2. Se utiliza una aproximación sistemática inspirada en metodologías de ingeniería de usabilidad que propone usar cuatro fases (Gabbard, Hix & Swan, 1999):

1. *Análisis de tareas de usuario* es el proceso de identificar una completa descripción de tareas, subtareas y métodos requeridos para usar un sistema, así como otros recursos necesarios para que el usuario o usuarios y el sistema ejecuten tareas cooperativamente.
2. *Evaluación basada en pautas expertas* (evaluación heurística o inspección de usabilidad) apunta a identificar problemas potenciales de usabilidad comparando un diseño de interacción (existente o evolucionado) con pautas de diseño de usabilidad establecidas.
3. *Evaluación formativa centrada en usuario* es un método de evaluación empírico y observacional que asegura la usabilidad de sistemas interactivos. Incluye usuarios temprana y continuamente durante el desarrollo de la interacción de usuario es decir, que desde las primeras versiones de la aplicación se pone a los usuarios a interactuar con ésta para detectar las carencias en la interacción y de este modo poder corregirlas a lo largo de futuras versiones.
4. *Evaluación comparativa global* es una valoración empírica de un diseño de interacción en comparación con otros diseños de interacción maduros para ejecutar las mismas tareas de usuario.

Presentadas estas cuatro fases, describiremos cómo relacionamos esta metodología con lo presentado hasta ahora. El lenguaje XMMVR presentado en el capítulo 3 ha sido diseñado como resultado de tener en cuenta las necesidades de un usuario que interactúa de manera gráfica y vocal con una aplicación basada en un mundo 3D. Por esto decimos que es el resultado de un análisis de tareas de usuario (1) ya que hemos identificado una completa descripción de tareas, subtareas y métodos requeridos para usar un sistema que permita interacción multimodal, gráfica y vocal con un mundo 3D. En el capítulo 4 revisando las modalidades y la cooperación entre ambas e implementándolas con XMMVR, ejecutamos la evaluación basada en pautas expertas (2). Así demostramos que nuestra propuesta es capaz de implementar cada metáfora gráfica y resolver las metáforas vocales así como permitir la cooperación entre modalidades. Identificamos problemas potenciales de usabilidad comparando un diseño de interacción con pautas de diseño de usabilidad establecidas (las metáforas gráficas y vocales junto a las posibilidades de cooperación entre ambas). Para ilustrar la evaluación formativa centrada en usuario (3) a continuación en el apartado 6.1.1 presentaremos la evaluación de una de las aplicaciones desarrolladas, aplicando la metodología USE introducida en

2.7.1.3. Finalmente y para cerrar el ciclo de cuatro fases en el que se basa la aproximación sistemática inspirada en metodologías de ingeniería de usabilidad, una evaluación comparativa global (4) de nuestra propuesta será presentada en el capítulo 7.

6.1 Procedimento experimental

Para realizar la fase de evaluación de nuestra propuesta hemos elegido utilizar el USE Questionnaire (Lund, 2001) presentado en el capítulo 2. Realizar un estudio basado en PROMISE, también descrito en el capítulo 2, sólo tendría relevancia para comparar la implementación de nuestra aplicación con distintos motores de voz. PROMISE se basa en PARADISE que como indicamos también en el capítulo 2 evalúa sistemas de diálogo hablado es decir, mide el rendimiento global de un conjunto de diálogos o subdiálogos y especifica la contribución relativa de diferentes factores al rendimiento global. Nuestro objetivo es evaluar la experiencia de usuario al interactuar con nuestra propuesta, ya que creemos que una evaluación subjetiva es más adecuada que una evaluación objetiva.

6.1.1. Caso de uso

Hay tres escenarios de evaluación correspondientes a tres modos de interacción, serán tres posibles situaciones que podrían darse en la aplicación Museo 3D Multimodal:

- Escenario basado en la metáfora de interacción gráfica de ”habitaciones” donde el usuario se mueve por los pasillos del museo usando las teclas de cursor y seleccionando las piezas a interactuar con clicks de ratón. El usuario parte de la entrada del museo y con las teclas del cursor lo recorre hasta que encuentra una de las piezas y si quiere verla en detalle la selecciona haciendo click con el ratón sobre ella. Es entonces cuando se le presenta la pieza hasta que con el ratón hace click en el elemento que le devuelve al museo para poder continuar su visita haciendo lo mismo con otras piezas.
- Escenario basado en la metáfora de interacción vocal de ”agente interfaz” donde el usuario indica vocalmente con qué piezas quiere interactuar. El usuario parte de la entrada del museo y con las teclas del cursor lo recorre pero una voz le ofrece visitar las piezas a ver en detalle simplemente indicando vocalmente el nombre de la pieza. Cuando ha visto todas las piezas puede indicar el fin de la visita también de manera vocal.
- Escenario basado en la complementariedad entre la metáfora de interacción gráfica de ”habitaciones” y la metáfora de interacción vocal de ”agente interfaz” donde el usuario selecciona la pieza con clicks de ratón e indica la acción a realizar sobre ella de manera vocal. El usuario parte de la entrada del museo y con las teclas del cursor lo recorre hasta que encuentra una de las piezas y si quiere verla en detalle la selecciona haciendo click con el ratón sobre ella. Es entonces cuando se le presenta la pieza a la que puede indicar la acción a realizar sobre ella tal y como le ha explicado una voz. Por ejemplo, puede encender o apagar cada modelo de ordenador con tan sólo indicarlo vocalmente como se muestra en la figura 6.1.

6.1.2. Cuestionario y metodología

Nos centraremos en cuatro factores: Utilidad, Facilidad de uso, Facilidad de aprendizaje y Satisfacción. Cada uno de estos factores sirve para recoger información relativa a partir de los usuarios que tras tener una experiencia de uso con nuestra aplicación responderán a las cuestiones correspondientes acerca de los cuatro factores según se muestra en la tabla 6.1.

Tabla 6.1: Factores evaluados con el cuestionario USE (Lund, 2001)

Utilidad	Me ayuda a ser más efectivo. Me ayuda a ser más productivo. Es útil. Me da más control sobre las actividades de mi vida. Hace las cosas a conseguir más fáciles de realizar. Me ahorra tiempo cuando lo utilizo. Satisface mis necesidades. Hace todo los que podría esperar.
Facilidad de uso	Fácil de usar. Simple de usar. Amigable al usuario. Requiere el menor número de pasos posible para realizar lo que quiero con él. Es flexible. Se utiliza sin esfuerzo. Puedo utilizarlo sin instrucciones escritas. No noto ninguna inconsistencia cuando lo utilizo. Gustará por igual a usuarios regulares y ocasionales. Puedo recuperarme fácil y rápidamente de errores. Puedo utilizarlo con éxito en cualquier momento.
Facilidad de aprendizaje	Aprendí rápido a utilizarlo. Fácilmente recuerdo cómo utilizarlo. Es fácil aprender a usarlo. Rápidamente me hice diestro en él.
Satisfacción	Estoy satisfecho con él. Lo recomendaría a un amigo. Es divertido utilizarlo. Funciona del modo que quiero que funcione. Es maravilloso. Siento que necesito tenerlo. Es agradable utilizarlo.

Un grupo de usuarios tras interactuar con los escenarios presentados rellenarán el cuestionario B.1 del apéndice B que se basa en varias afirmaciones a las que se responde con valores del 1 al 7 siendo 1 ”Totalmente en desacuerdo” y 7 ”Totalmente de acuerdo”. Tras ello se les pregunta por el modo de interacción que más les satisface.

El objetivo de realizar pruebas con usuarios fue observar a los usuarios interactuando con los tres escenarios presentados y valorar su nivel de satisfacción ante las afirmaciones del cuestionario USE para el sistema además de obtener su predilección con respecto a los tres escenarios. El test se desarrolló para explorar aspectos subjetivos con respecto a los cuatro factores presentados en los que se basa el cuestionario USE. La finalidad era explorar el nivel de aceptación de cada una de las modalidades gráfica y vocal por separado así como la cooperación entre ambas.

6.1.3. Usuarios de prueba

En cuanto a los participantes, se han evaluado 20 usuarios, de ellos 13 eran hombres y 7 mujeres. Todos con una edad comprendida entre los 25 y 42 años teniendo 15 de ellos estudios superiores y 5 con estudios medios. A pesar de que varios tenían profesiones relacionadas con la informática, ninguno era experto en interfaces, también había algún usuario poco habituado al uso de herramientas

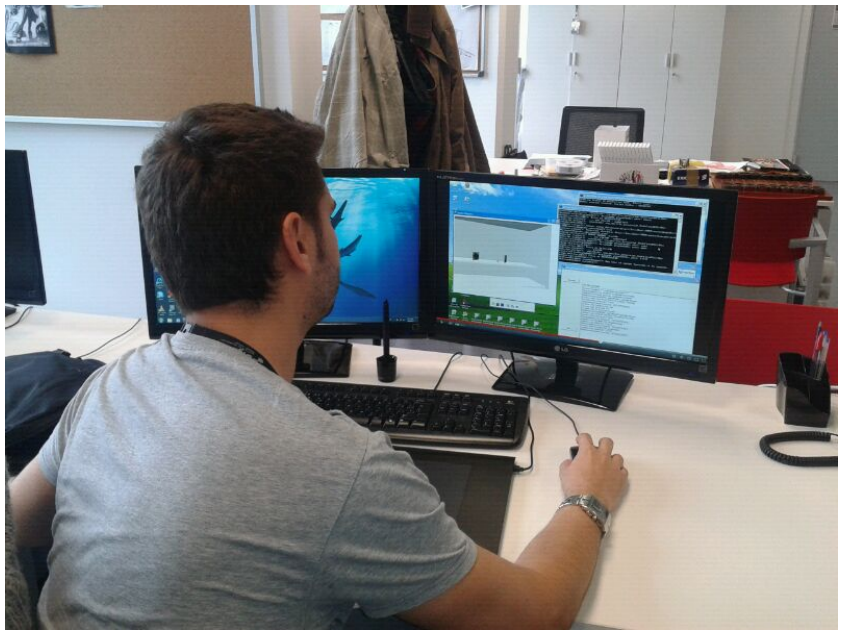

FIGURA 6.1: Usuario interactuando con la aplicación Museo 3D Multimodal

informáticas junto al resto que eran usuarios medios. Se explicaba a los usuarios el funcionamiento de la aplicación y los objetivos a realizar en cada uno de los escenarios. Acerca de los dispositivos utilizados, para la prueba se utilizó un portátil con la aplicación Museo 3D Multimodal instalada que fue ejecutada por cada usuario interactuando con cada uno de los escenarios presentados en el orden presentado en el apartado 6.1.1. Sobre el espacio donde se desarrollaron las pruebas, al estar la aplicación instalada en un portátil se podía interactuar con ella en la ubicación que decidiese el usuario.

Para describir brevemente el protocolo de desarrollo de la sesión y tareas, indicar que primeramente había una fase de familiarización y entrenamiento con los dispositivos, seguida de la interacción con los 3 escenarios y tras ello el usuario rellenaba el cuestionario presentado en B.1.

6.2 Resultados obtenidos

Antes de interpretar los resultados de la encuesta, que se presentan en las tablas B.1, B.2, B.3 y B.4 del apéndice B, hemos de resaltar que tras realizar el cuestionario basado en USE, se presentaron una pregunta y tres afirmaciones a los usuarios encuestados cuyas respuestas se presentan en la tabla 6.2.

Tabla 6.2: Pregunta y afirmaciones abiertas

PREGUNTA/AFIRMACIÓN	RESPUESTA DE LOS USUARIOS
¿Cuál le ha aportado la mejor experiencia?	Dos indicaron que la interacción vocal, ninguno optó por la interacción gráfica, trece se decantaron por la combinación de modalidades gráfica y vocal.
La interacción vocal era más sencilla que la interacción gráfica	Diez usuarios estaban de acuerdo, ocho se mostraron en contra.
La interacción gráfica era más sencilla que la vocal	Cuatro lo apoyaban, quince se mostraban en contra.
La combinación de ambas era la mejor	Dieciséis usuarios estaban de acuerdo, dos usuarios fueron discrepantes.

De todo ello se deduce que la combinación de modalidades fue preferida por los usuarios frente a usar una sola de las modalidades, habiendo una mayor aceptación por la modalidad vocal sobre la gráfica. Dentro del colectivo de usuarios de sexo masculino, el 69,23 % prefirió la complementariedad de modalidades frente a una sola modalidad, en el grupo de sexo femenino fue el 57,14 %. El 61,54 % de los hombres consideraron que la interacción vocal es más sencilla que la gráfica, mientras que sólo el 28,57 % de las mujeres lo pensó así. Por el contrario, sólo el 7,69 % de los hombres indicó que la interacción gráfica es más sencilla que la vocal, ante el 42,86 % de las mujeres que opinaron que la interacción gráfica es más sencilla que la vocal. Respecto a la combinación de modalidades, los hombres en un 92,31 % opinaron que es la mejor ante el 57,14 % de las mujeres que opinaron lo mismo.

Las áreas en las que los usuarios ven que podría ser más interesante utilizar esta tecnología además de las propuestas (museos virtuales, ayuda a discapacitados y juegos) son tales como aplicaciones industriales, tiendas online de productos intangibles (compra de billetes, venta de entradas, búsqueda de hoteles) y de productos tangibles (ropa, alimentación, etc.), visualización de video sin mando a distancia, cambio de canales de TV, arquitectura, arqueología, arte, turismo, educación, sanidad, ayuntamientos, gobiernos y búsqueda de información, sexo virtual e interacciones remotas muy difíciles de plasmar en un sistema informático tradicional.

Observaciones negativas de los usuarios se han referido a la apariencia gráfica, la no inclusión de un avatar de la guía del museo, los problemas de interacción gráfica y vocal, etc. Consideramos que son opiniones muy ligadas al prototipo presentado. No obstante se han dado opiniones apoyando la necesidad de combinar ambas modalidades destacando la sencillez y el potencial de la propuesta.

Tras recoger las valoraciones de los usuarios a las afirmaciones del cuestionario USE con respecto a la aplicación evaluada, se realizaron análisis descriptivos de las respuestas usando diagramas de cajas o boxplots y se obtuvieron las gráficas que se presentarán a continuación en base a las respuestas a cada una de las afirmaciones presentadas. Éstas pueden tener una respuesta del 1 al 7 correspondiéndose con "Totalmente en desacuerdo" y "Totalmente de acuerdo" respectivamente.

Exceptuando las afirmaciones "Puedo recuperarme fácil y rápidamente de errores" y "Es fácil aprender a usarlo", todas tienen una respuesta media igual o por encima del 4. A continuación analizaremos las afirmaciones relacionadas con cada uno de los factores a evaluar: utilidad, facilidad de uso, facilidad de aprendizaje y satisfacción.

Respecto a la utilidad del sistema en la gráfica de la figura 6.2 vemos que la media de las puntuaciones asignadas están por encima del valor 6 a excepción de la afirmación UT7. Esto indica que la mayoría de usuarios tiene claro que el sistema evaluado tiene utilidad siendo un poco menor la valoración respecto al ahorro de tiempo (UT7) pero siendo mucho más que aceptable ya que la aplicación no se desarrolló para conseguir que los usuarios ahorraran tiempo.

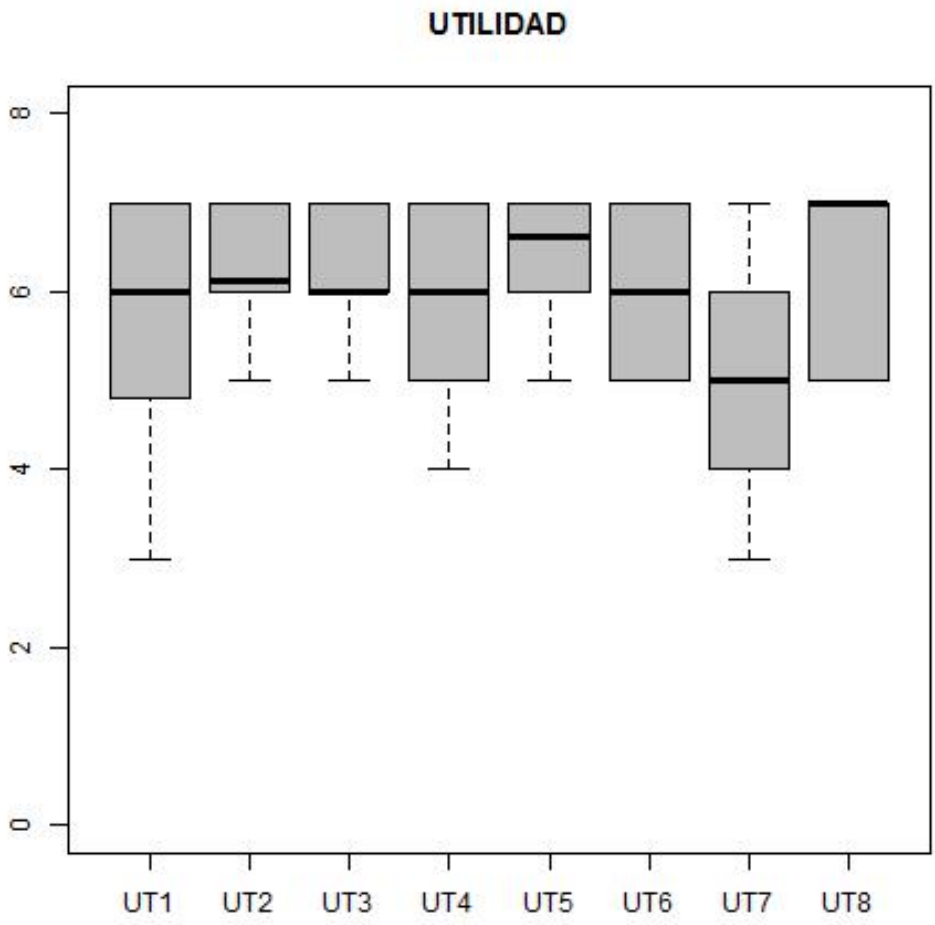

Figura 6.2: Utilidad de la aplicación Museo 3D Multimodal

UT1. Es útil.
UT2. Hace todo lo que podría esperar que haga.
UT3. Me daría más control sobre las actividades de mi vida.
UT4. Haría las cosas que quiero conseguir más fáciles de realizar.
UT5. Me ayudaría a ser más efectivo.
UT6. Satisface mis necesidades.
UT7. Me ahorraría tiempo cuando lo utilizo.
UT8. Me ayudaría a ser más productivo.

Vemos en la gráfica de la figura 6.3 que siendo menos valorada, se considera que el sistema ofrece una facilidad de uso. Siendo la simpleza, flexibilidad y recuperación de errores (FU2, FU7, FU8) las cuestiones peor valoradas sobre todo por los usuarios menos acostumbrados a utilizar aplicaciones basadas en mundos 3D, según lo que podríamos intuir de sus profesiones.

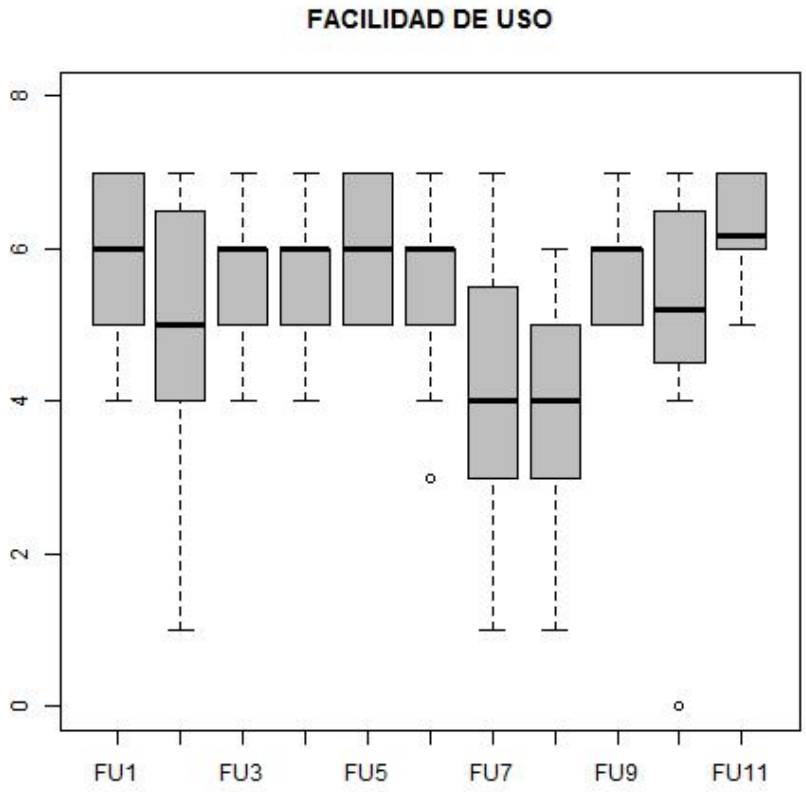

Figura 6.3: Facilidad de uso de la aplicación Museo 3D Multimodal

FU1. Gustaría tanto a usuarios regulares como ocasionales.
FU2. Es simple de utilizar.
FU3. No noto inconsistencias cuando lo uso.
FU4. Requiere el menor número de pasos posible para realizar lo que quiero con él.
FU5. Es amigable al usuario.
FU6. Puedo utilizarlo con éxito en cualquier momento.
FU7. Es flexible.
FU8. Puedo recuperarme fácil y rápidamente de errores.
FU9. Se utiliza sin esfuerzo.
FU10. Puedo utilizarlo sin instrucciones escritas.
FU11. Es fácil de usar.

Analizando la gráfica de la figura 6.4 deducimos que es fácil aprender a utilizar el sistema para los usuarios evaluados, incluso para aquellos menos habituados a este tipo de aplicaciones los resultados son notables. De nuevo nos basamos en sus profesiones.

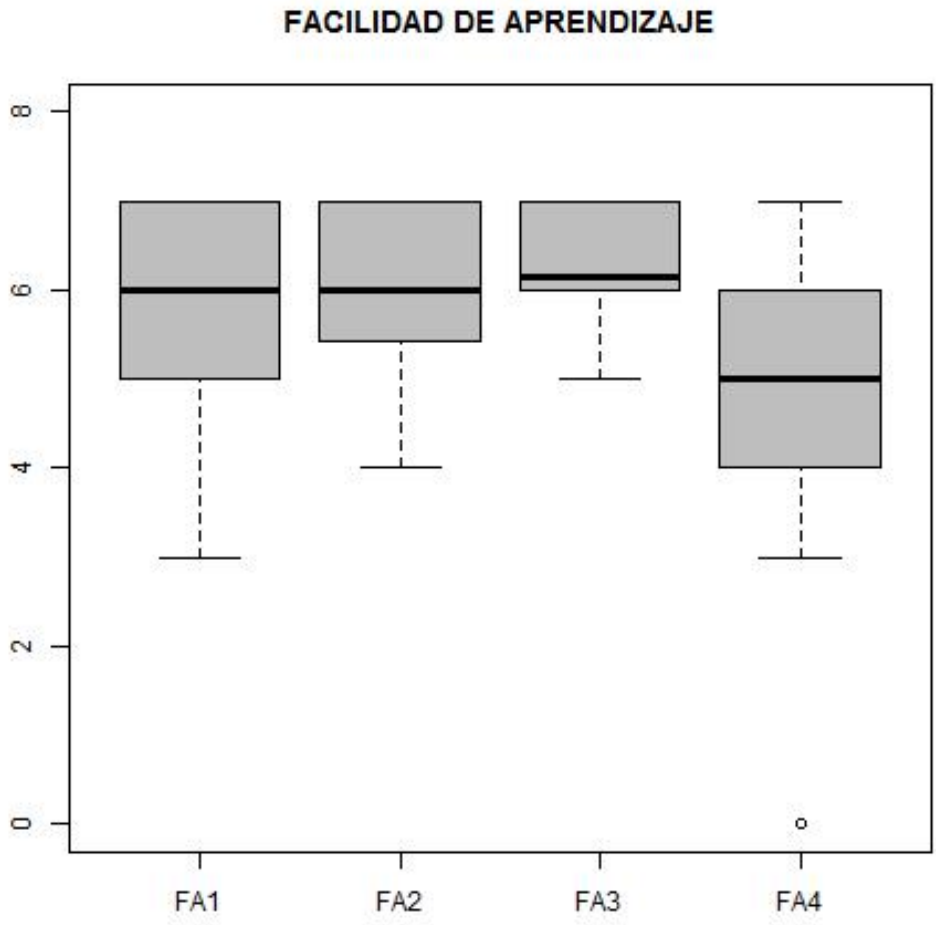

Figura 6.4: Facilidad de aprendizaje de la aplicación Museo 3D Multimodal

FA1. Aprendí rápido a utilizarlo.
FA2. Fácilmente recuerdo cómo utilizarlo.
FA3. Rápidamente me hice diestro en él.
FA4. Es fácil aprender a usarlo.

La gráfica de la figura 6.5 muestra que los usuarios están satisfechos con el sistema aunque sin ser considerado maravilloso unánimemente, ni verse como sobresaliente la diversión y la agradabilidad que produce al utilizarlo, algo que se justifica por la naturaleza de la aplicación que para nada tiene como objetivo entretener.

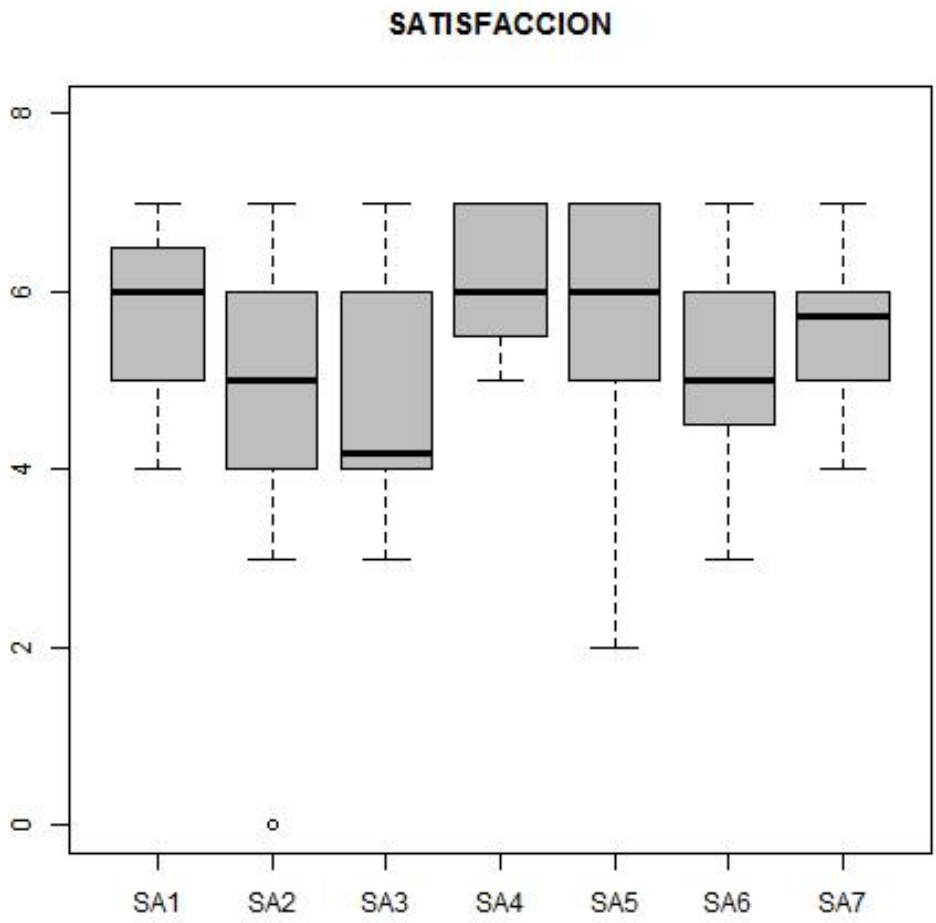

Figura 6.5: Satisfacción de la aplicación Museo 3D Multimodal

SA1. Estoy satisfecho con él.
SA2. Es maravilloso.
SA3. Es divertido usarlo.
SA4. Siento que necesito tenerlo.
SA5. Funciona del modo que quiero que funcione.
SA6. Es agradable utilizarlo.
SA7. Lo recomendaría a un amigo.

6.3 Resumen y conclusiones

Tras diferenciar la evaluación del lenguaje XMMVR de la evaluación del escenario de uso, hemos presentado la metodología para la evaluación de dicho escenario de uso así como los resultados obtenidos. Hemos visto que la calificación media es alta a la hora de apoyar las afirmaciones del cuestionario USE ya que la mayoría de usuarios están de acuerdo con dichas afirmaciones y tampoco hay un gran número de usuarios muy en desacuerdo. Es de remarcar que a pesar de que la utilidad no es lo más valorado, sí que se considera que se tiene un mayor control sobre la tarea utilizando un sistema como éste. Por el contrario, aunque la mayoría de usuarios coinciden en la facilidad de uso del sistema, muchos usuarios discrepan de la flexibilidad y de la capacidad de recuperarse ante errores. En cuanto a la facilidad de aprendizaje, a pesar de que la mayoría de usuarios no afirman totalmente que sea fácil aprender a utilizarlo, sí que en su mayoría admiten que se hicieron diestros con él y que no tardaron en aprender a usarlo. En cuanto al nivel de satisfacción, a la mayoría de usuarios les provoca tener algo similar. Es curioso el estudio por sexos realizado del que podríamos deducir que existe una mayor inquietud en el sexo masculino por la complementariedad y la combinación de modalidades así como un interés mayor por la modalidad vocal respecto a la gráfica. Esto contrasta con la posición conservadora de las mujeres hacia la interacción gráfica clásica mostrando nulo interés por la interacción vocal así como combinaciones entre ambas modalidades.

7 Conclusión

7.1 Conclusiones

En el capítulo 1 tras una introducción en la que se presentaba la interacción multimodal y se establecía su relación con los mundos virtuales, se definió la necesidad de establecer un marco común con un lenguaje de especificación y una arquitectura soporte que ayude a implementar aplicaciones de interacción multimodal con espacios virtuales.

En el capítulo 2 se han introducido lenguajes de marcado que permiten especificar diálogo, escena o comportamiento de manera única (dedicados) y otros que permiten especificar una combinación de más de una de estas modalidades e incluso otras como son el gesto y/o la escritura caligráfica o anotación manual de datos (híbridos). Observando que todos estos lenguajes declarativos se basan en marcas es decir, son aplicaciones XML, la primera condición para el lenguaje que se adaptó fue que se basara en XML. Puesto que ya existían lenguajes de especificación básicos para definir parte de lo que queríamos (X3D y VRML para escenas 3D y VoiceXML para diálogos), optar por definir un lenguaje de especificación "híbrido" utilizando lenguajes de especificación "básicos" facilitaría mucho los cosas y permitiría reutilizar módulos ya definidos.

Por ello, la segunda condición del lenguaje fue que fuera "híbrido". Tras estudiar los diferentes tipos de lenguajes de marcado "básicos" (para especificar interacción vocal o diálogo, imágenes 2D, escenas 3D y comportamiento), "híbridos" y "dedicados", se dedujo que no existía ninguno que se ajustara a la necesidad: especificar espacios virtuales con interacción multimodal (gráfica y vocal). Así la tercera condición del lenguaje y la más importante sería que permitiera especificar diálogo, escenas y comportamiento en un marco común.

Vistas todas las posibilidades y las características propias de los modos de interacción y su naturaleza, los problemas a resolver fueron: definir un sistema multimodal capaz de implementar cada una de las metáforas de interacción gráfica y vocal así como todas las posibles cooperaciones entre ambas e integrarlos en aplicaciones basadas en espacios virtuales. La propuesta de arquitectura del grupo MMI del W3C sirvió como base de los requisitos de la arquitectura del sistema multimodal a definir ya que todas las demás propuestas realizadas anteriormente eran pequeños prototipos de laboratorio que se introdujeron en el apartado 2.1 y que han sido solamente intentos de resolver problemas concretos pero sin ser capaces de dar una respuesta general. Conocidas las arquitecturas de los sistemas de diálogo existentes y vista la necesidad de utilizar VoiceXML para resolver el problema, se concluyó que el único requisito del sistema de diálogo a integrar en la arquitectura propuesta fue que se basara en un navegador VoiceXML. Revisadas cada una de las tecnologías RIA, sus características y capacidades 3D, se concluyó que cada tecnología presentada ofrecía ventajas e inconvenientes según el criterio básico a considerar pero teniendo en cuenta que la riqueza de todas residía en la posibilidad de combinarlas y de integrarlas con otras tecnologías. Por ello, la elección de una tecnología RIA para integrarla en el sistema se basó además en la experiencia en su utilización.

Finalmente se ha mostrado que la evaluación de sistemas multimodales puede ser vista como una evolución de la evaluación de sistemas unimodales es decir, evaluación de sistemas de diálogo ya que muchas características se comparten en ambos. También se observó que en los sistemas basados en entornos de realidad virtual, se han desarrollado pocas pero algunas pautas para la evaluación de este tipo de sistemas, estando la evaluación subjetiva más desarrollada que la evaluación objetiva.

En el capítulo 3 partiendo de que toda aplicación basada en mundos 3D donde se permite una interacción multimodal se basa en unos mismos principios, se ha descrito el lenguaje XMMVR que se compone de distintos elementos para especificar los mundos 3D y las posibilidades de interacción multimodal que se le dan al usuario. Para ello han descrito cada uno de estos elementos, sus representaciones en formato DTD y sus XMLSchemas. Vista la descripción del lenguaje se puede decir que se han cumplido los objetivos marcados: abstracción, modularidad, encapsulamiento, polimorfismo y herencia.

En el capítulo 4 se ha visto que la propuesta es capaz de especificar cada metáfora gráfica. Las metáforas estructurales son desarrolladas rápidamente y las metáforas navegacionales también pero descargando el esfuerzo en el diseño gráfico. Las metáforas vocales también pueden ser implementadas fácilmente en base a definir ficheros vocales y de gramática adecuados. Se ha demostrado que la cooperación entre modalidades se puede implementar. Para ello se han presentado ejemplos de cada caso basados en la aplicación ejemplo que demuestran las capacidades del lenguaje. La complejidad de los ficheros XMMVR es relativamente baja al especificar metáforas gráficas y se incrementa al añadir las metáforas de interacción vocal haciéndose mayor cuando se incluye la cooperación entre modalidades. Pero también se ha visto en los ejemplos descritos que las posibilidades de reutilización de código y recursos son enormes.

En el capítulo 5 se han revisado las necesidades de una arquitectura que soporte la propuesta y cada uno de los módulos que la componen con sus funciones. Tras ello se ha visto que estos módulos encajan con la propuesta de arquitectura para aplicaciones multimodales del grupo MMI del W3C. Finalmente se han presentado dos implementaciones que usan tecnologías distintas pero que son válidas para definir aplicaciones multimodales que permiten interacción con mundos 3D según el marco propuesto. Respecto al cumplimiento de la especificación del grupo MMI del W3C para toda aplicación multimodal, a día de hoy ninguna de las aproximaciones tecnológicas de la arquitectura presentada lo cumplen al 100 %. Su filosofía de diseño modular ha sido la base de éstas, quedando pendiente para el futuro el cumplimiento de los estándares que propone para protocolos de comunicación entre módulos y tratamiento de eventos.

En el capítulo 6 tras diferenciar la evaluación del lenguaje XMMVR de la evaluación del escenario de uso, hemos presentado la metodología para la evaluación de dicho escenario de uso así como los resultados obtenidos. Hemos visto que la calificación media es alta a la hora de apoyar las afirmaciones del cuestionario USE ya que la mayoría de usuarios están de acuerdo con dichas afirmaciones y tampoco hay un gran número de usuarios muy en desacuerdo. Utilidad, facilidad de uso del sistema, facilidad de aprendizaje y satisfacción son los factores evaluados. Como evaluación comparativa global que correspondería a la cuarta fase de la metodología propuesta en (Gabbard, Hix & Swan, 1999), se concluye con que se deben hacer aplicaciones más flexibles, capaces de recuperarse ante errores y que además tengan una utilidad real para el usuario.

7.2 Perspectivas de trabajo futuro

Como trabajo futuro se plantean cinco líneas de investigación claramente definidas pero a la vez interrelacionadas que detallamos a continuación:

Evolución de los dispositivos de entrada no vocal El uso del teclado y el ratón como se han conocido hasta ahora, ha sido adoptado como algo básico pero la forma de utilizarlos ha evolucionado. La aparición de teclados virtuales y la evolución de los ratones clásicos a elementos más potentes, unidas al uso de dispositivos hápticos es algo que se debe considerar. Aunque

la base es la misma, nuevas características de estos dispositivos deberán ser consideradas al ser integrados en el marco propuesto.

Integración de la modalidad gestual Durante el desarrollo de la investigación en la que se ha centrado este trabajo de tesis, han aparecido y se han popularizado dispositivos que permiten interactuar con los sistemas a través de una hasta hace poco novedosa modalidad: la modalidad gestual. Esta nueva modalidad está claramente indicada para interactuar con mundos de realidad virtual ya que aporta una naturalidad extrema a la interacción del usuario con los entornos virtuales. Por eso y al igual que hemos realizado para cada modalidad gráfica y vocal, deberíamos identificar las metáforas de interacción gestual más extendidas. Tras ello hemos de incluir en nuestro lenguaje XMMVR una extensión que permita especificar éstas e implementar ejemplos que las pongan en práctica. Finalmente y tras adaptar nuestra arquitectura al tratamiento de esta nueva modalidad teniendo en cuenta las posibilidades de cooperación de ésta con las otras dos modalidades presentadas, desarrollaríamos nuevas aplicaciones que integren la modalidad gestual para ser evaluadas con usuarios finales.

Integración de la RA/RM Cada vez son más las aplicaciones que nos permiten interactuar con mundos virtuales y extensiones del mundo real. Se ha producido una explosión de aplicaciones de interacción con mundos de realidad aumentada y realidad mixta gracias a la popularidad de los smartphones y los nuevos elementos de interacción desarrollados para consolas de videojuegos. Esto nos indica que debemos adaptar lo descrito en esta memoria de tesis aplicado a mundos de realidad virtual a entornos de realidad aumentada y realidad mixta. La primera tarea a realizar para ello será identificar las peculiaridades de estos entornos así como los formatos de fichero que representan los nuevos mundos de realidad aumentada y realidad mixta.

Integración de nuevas tecnologías software Que duda cabe de que los fabricantes de software para desarrollo de realidad virtual, realidad aumentada, realidad mixta, reconocimiento y síntesis de habla y reconocimiento gestual están constantemente presentado nuevos productos. Igualmente los lenguajes de marcas han evolucionado y se presentan cada vez nuevos lenguajes de marcas más específicos. Por ello una línea de investigación clara que debemos mantener es la de recopilar todas estas nuevas tecnologías software y aplicarlas allí donde podamos integrarlas bien sea para nuevos objetivos o para mejorar los ya conseguidos.

Integración de nuevas tecnologías hardware Como hemos indicado, la modalidad gestual se ha generalizado cada vez más gracias a la integración de elementos hardware dedicados a ella en las consolas de videojuegos, en los acelerómetros de los smartphones e incluso en los ordenadores personales. Por otro lado la realidad aumentada exige nuevos elementos de posicionamiento y de reconocimiento de imágenes que ya se han integrado en el nuevo hardware de videoconsolas, smartphones y ordenadores personales. Es por ello que para poder desarrollar aplicaciones que aprovechen todo esto debemos conocer profundamente el hardware de todos estos nuevos dispositivos.

Las publicaciones Olmedo-Rodríguez & Augusto (2012), Olmedo-Rodríguez & Augusto (2013) y (Olmedo-Rodríguez, 2013) son los primeros resultados de estas líneas de investigación.

Apéndices

A

Ejemplos XMMVR completos

A.1 Implementación de metáforas gráficas

A.1.1. Metáfora de teatro

Listado A.1: Implementación de la metáfora de teatro en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Room0" FILE3D="Room0.W3D"/>
            <ACTOR ACTORID="Piece" FILE3D="Piece.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
                <METHODDECL NAME="rotate">
                    <PARAMDECL NAME="to" TYPE="string" />
                </METHODDECL>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Room0" >
                <ACTORINSTANCE ACTORINSTANCEID="Amstrad" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(-5,-35,5)"/>
                    <PARAM NAME="Content" VALUE="Amstrad" />
                    <PARAM NAME="Image" VALUE="Amstrad.jpg" />
                    <BEHAVIOUR BEHAVIOURID="rotate">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="Amstrad" METHOD="rotate">
                                <PARAM NAME="to" VALUE="Left"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="Macintosh" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Macintosh" />
                    <PARAM NAME="Image" VALUE="Macintosh_Plus.jpg" />
                    <BEHAVIOUR BEHAVIOURID="rotate">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="Macintosh" METHOD="rotate">
                                <PARAM NAME="to" VALUE="Right" />
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
        </SEQUENCE>
        <CONTEXT/>
    </XMMVR>
```

A.1.2. Metáfora de locomoción

Listado A.2: Implementación de la metáfora de locomoción en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Room0" FILE3D="Room0.W3D"/>
            <ACTOR ACTORID="Piece" FILE3D="Piece.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
                <METHODDECL NAME="rotate">
                    <PARAMDECL NAME="to" TYPE="string" />
                </METHODDECL>
            </ACTOR>
            <ACTOR ACTORID="Camera" FILE3D="null">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="AT" TYPE="vector" />
                <METHODDECL NAME="go">
                    <PARAMDECL NAME="to" TYPE="string"/>
                </METHODDECL>
                <METHODDECL NAME="turn">
                    <PARAMDECL NAME="to" TYPE="string"/>
                </METHODDECL>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Room0" >
                <ACTORINSTANCE ACTORINSTANCEID="User" ACTORID="Camera">
                    <PARAM NAME="Pos" VALUE="(0,-15,0)" />
                    <PARAM NAME="AT" VALUE="(0,-15,0)" />
                    <BEHAVIOUR BEHAVIOURID="goAhead">
                        <EVENTGUI TYPE="arrowUP"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="User" METHOD="go">
                                <PARAM NAME="to" VALUE="front"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                    <BEHAVIOUR BEHAVIOURID="goBack">
                        <EVENTGUI TYPE="arrowDOWN"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="User" METHOD="go">
                                <PARAM NAME="to" VALUE="back"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                    <BEHAVIOUR BEHAVIOURID="turnLeft">
                        <EVENTGUI TYPE="arrowLEFT"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="User" METHOD="turn">
                                <PARAM NAME="to" VALUE="left"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                    <BEHAVIOUR BEHAVIOURID="turnRight">
                        <EVENTGUI TYPE="arrowRIGHT"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="User" METHOD="turn">
                                <PARAM NAME="to" VALUE="right"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="Amstrad" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(-5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Amstrad" />
                    <PARAM NAME="Image" VALUE="Amstrad.jpg" />
                    <BEHAVIOUR BEHAVIOURID="rotate">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="Amstrad" METHOD="rotate">
                                <PARAM NAME="to" VALUE="Left" />
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="Macintosh" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Macintosh" />
                    <PARAM NAME="Image" VALUE="Macintosh_Plus.jpg" />
                    <BEHAVIOUR BEHAVIOURID="rotate">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="Macintosh" METHOD="rotate">
                                <PARAM NAME="to" VALUE="Right" />
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
```

```
                </SCENE>
            </SEQUENCE>
            <CONTEXT/>
        </XMMVR>
```

A.1.3. Metáfora de habitaciones

LISTADO A.3: Implementación de la metáfora de habitaciones en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Corridor" FILE3D="Corridor.W3D"/>
            <STAGE STAGEID="Room0" FILE3D="Room0.W3D"/>
            <STAGE STAGEID="Room1" FILE3D="Room1.W3D"/>
            <STAGE STAGEID="Room2" FILE3D="Room2.W3D"/>
            <ACTOR ACTORID="Door" FILE3D="Door.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
            </ACTOR>
            <ACTOR ACTORID="Piece" FILE3D="Piece.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
                <METHODDECL NAME="rotate">
                    <PARAMDECL NAME="to" TYPE="string" />
                </METHODDECL>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Corridor">
                <ACTORINSTANCE ACTORINSTANCEID="DoorRoom0" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(0,-14,0)" />
                    <BEHAVIOUR BEHAVIOURID="enterRoom0">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Room0"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="DoorRoom1" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(-10,0,0)" />
                    <BEHAVIOUR BEHAVIOURID="enterRoom1">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Room1"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="DoorRoom2" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(14,0,0)" />
                    <BEHAVIOUR BEHAVIOURID="enterRoom2">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Room2"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
            <SCENE STAGEID="Room0">
                <ACTORINSTANCE ACTORINSTANCEID="Amstrad" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(-5,-35,5)"/>
                    <PARAM NAME="Content" VALUE="Amstrad" />
                    <PARAM NAME="Image" VALUE="Amstrad.jpg" />
                    <BEHAVIOUR BEHAVIOURID="rotate">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="Amstrad" METHOD="rotate">
                                <PARAM NAME="to" VALUE="Left"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="Macintosh" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Macintosh" />
                    <PARAM NAME="Image" VALUE="Macintosh_Plus.jpg" />
                    <BEHAVIOUR BEHAVIOURID="rotate">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="Macintosh" METHOD="rotate">
                                <PARAM NAME="to" VALUE="Right" />
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor0" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(0,-14,0)" />
                    <BEHAVIOUR BEHAVIOURID="BackToCorridor">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
```

```
            </SCENE>
            <SCENE STAGEID="Room1">
                <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor1" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(-10,0,0)" />
                    <BEHAVIOUR BEHAVIOURID="BackToCorridor">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
            <SCENE STAGEID="Room2">
                <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor2" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(14,0,0)" />
                    <BEHAVIOUR BEHAVIOURID="BackToCorridor">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
        </SEQUENCE>
        <CONTEXT/>
    </XMMVR>
```

A.1.4. Metáforas navegacionales

A.1.4.1. Elevador/rampa hidráulica

Listado A.4: Implementación de la metáfora de ascensor en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
<XMMVR>
    <CAST>
        <STAGE STAGEID="Level0" FILE3D="Level0.W3D"/>
        <STAGE STAGEID="Level1" FILE3D="Level1.W3D"/>
        <STAGE STAGEID="Level2" FILE3D="Level2.W3D"/>
        <STAGE STAGEID="Elevator" FILE3D="Elevator.W3D"/>
        <ACTOR ACTORID="Door" FILE3D="Door.W3D">
            <PARAMDECL NAME="Pos" TYPE="vector" />
            <METHODDECL NAME="open"/>
        </ACTOR>
        <ACTOR ACTORID="Button" FILE3D="Button.W3D">
            <PARAMDECL NAME="Pos" TYPE="vector" />
            <PARAMDECL NAME="Destiny" TYPE="string" />
        </ACTOR>
    </CAST>
    <SEQUENCE>
        <SCENE STAGEID="Level0" >
            <ACTORINSTANCE ACTORINSTANCEID="DoorElevator0" ACTORID="Door">
                <PARAM NAME="Pos" VALUE="(0,37,0)"/>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator01" ACTORID="Button">
                <PARAM NAME="Pos" VALUE="(-2,37,5)"/>
                <BEHAVIOUR BEHAVIOURID="go01">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="DoorElevator0" METHOD="open"/>
                        <ACTIONSYS TYPE="setTimer" DESTINATION="Elevator" TIMERID="0to1" TIME
                            ="10">
                            <ASSIGN NAME="newScene" VALUE="Level1"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator02" ACTORID="Button">
                <PARAM NAME="Pos" VALUE="(-2,37,6)"/>
                <BEHAVIOUR BEHAVIOURID="go02">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="DoorElevator0" METHOD="open"/>
                        <ACTIONSYS TYPE="setTimer" DESTINATION="Elevator" TIMERID="0to2" TIME
                            ="20">
                            <ASSIGN NAME="newScene" VALUE="Level2"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
        </SCENE>
        <SCENE STAGEID="Level1" >
            <ACTORINSTANCE ACTORINSTANCEID="DoorElevator1" ACTORID="Door">
                <PARAM NAME="Pos" VALUE="(0,37,15)"/>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator10" ACTORID="Button">
                <PARAM NAME="Pos" VALUE="(-2,37,20)"/>
                <BEHAVIOUR BEHAVIOURID="go10">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="DoorElevator1" METHOD="open"/>
                        <ACTIONSYS TYPE="setTimer" DESTINATION="Elevator" TIMERID="1to0" TIME
                            ="10">
                            <ASSIGN NAME="newScene" VALUE="Level0"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator12" ACTORID="Button">
                <PARAM NAME="Pos" VALUE="(-2,37,21)"/>
                <BEHAVIOUR BEHAVIOURID="go12">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="DoorElevator1" METHOD="open"/>
                        <ACTIONSYS TYPE="setTimer" DESTINATION="Elevator" TIMERID="1to2" TIME
                            ="10">
                            <ASSIGN NAME="newScene" VALUE="Level2"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
        </SCENE>
        <SCENE STAGEID="Level2" >
            <ACTORINSTANCE ACTORINSTANCEID="DoorElevator2" ACTORID="Door">
                <PARAM NAME="Pos" VALUE="(0,37,30)"/>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator21" ACTORID="Button">
```

```
                    <PARAM NAME="Pos" VALUE="(-2,37,35)"/>
82                  <BEHAVIOUR BEHAVIOURID="go21">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="DoorElevator2" METHOD="open"/>
                            <ACTIONSYS TYPE="setTimer" DESTINATION="Elevator" TIMERID="2to1" TIME
                                ="20">
87                              <ASSIGN NAME="newScene" VALUE="Level1"/>
                            </ACTIONSYS>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
92              <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator20" ACTORID="Button">
                    <PARAM NAME="Pos" VALUE="(-2,37,36)"/>
                    <BEHAVIOUR BEHAVIOURID="go20">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
97                          <ACTIONACT ACTORINSTANCEID="DoorElevator2" METHOD="open"/>
                            <ACTIONSYS TYPE="setTimer" DESTINATION="Elevator" TIMERID="2to0" TIME
                                ="10">
                                <ASSIGN NAME="newScene" VALUE="Level0"/>
                            </ACTIONSYS>
                        </ACTIONBLOQ>
102                 </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
            <SCENE STAGEID="Elevator" >
                <ACTORINSTANCE ACTORINSTANCEID="DoorElevator" ACTORID="Door">
107                 <PARAM NAME="Pos " VALUE=" (0,37,0)" />
                </ACTORINSTANCE>
            </SCENE>
        </SEQUENCE>
        <CONTEXT>
112         <VARIABLE NAME="newScene" TYPE="string" VALUE="Level0"/>
        </CONTEXT>
    </XMMVR>
```

A.1.4.2. Vehículo en vía/tren

Listado A.5: Implementación de la metáfora de tren en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Museum" FILE3D="Museum.W3D"/>
            <ACTOR ACTORID="Button" FILE3D="Button.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
            </ACTOR>
            <ACTOR ACTORID="Camera" FILE3D="null">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="AT" TYPE="vector" />
                <METHODDECL NAME="go">
                    <PARAMDECL NAME="to" TYPE="string"/>
                </METHODDECL>
                <METHODDECL NAME="turn">
                    <PARAMDECL NAME="to" TYPE="string"/>
                </METHODDECL>
                <METHODDECL NAME="route">
                    <PARAMDECL NAME="to" TYPE="string"/>
                </METHODDECL>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Museum">
                <ACTORINSTANCE ACTORINSTANCEID="User" ACTORID="Camera">
                    <PARAM NAME="Pos" VALUE="(0,0,0)" />
                    <PARAM NAME="AT" VALUE="(0,0,1)" />
                    <BEHAVIOUR BEHAVIOURID="goAhead">
                        <EVENTGUI TYPE="arrowUP"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="User" METHOD="go">
                                <PARAM NAME="to" VALUE="front"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                    <BEHAVIOUR BEHAVIOURID="goBack">
                        <EVENTGUI TYPE="arrowDOWN"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="User" METHOD="go">
                                <PARAM NAME="to" VALUE="back"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                    <BEHAVIOUR BEHAVIOURID="turnLeft">
                        <EVENTGUI TYPE="arrowLEFT"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="User" METHOD="turn">
                                <PARAM NAME="to" VALUE="left"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                    <BEHAVIOUR BEHAVIOURID="turnRight">
                        <EVENTGUI TYPE="arrowRIGHT"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="User" METHOD="turn">
                                <PARAM NAME="to" VALUE="right"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="ButtonRoom0to1" ACTORID="Button">
                    <PARAM NAME="Pos" VALUE="(5,-14,5)"/>
                    <BEHAVIOUR BEHAVIOURID="goToRoom1from0">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="User" METHOD="route">
                                <PARAM NAME="to" VALUE="itinerary0to1"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="ButtonRoom0to2" ACTORID="Button">
                    <PARAM NAME="Pos" VALUE="(5,-14,6)"/>
                    <BEHAVIOUR BEHAVIOURID="goToRoom2from1">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONACT ACTORINSTANCEID="User" METHOD="route">
                                <PARAM NAME="to" VALUE="itinerary0to2"/>
                            </ACTIONACT>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="ButtonRoom1to0" ACTORID="Button">
                    <PARAM NAME="Pos" VALUE="(-10,5,5)"/>
                    <BEHAVIOUR BEHAVIOURID="goToRoom0from1">
                        <EVENTGUI TYPE="mouseDown"/>
```

```
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="User"  METHOD="route">
                            <PARAM NAME="to" VALUE="itinerary1to0"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ButtonRoom2to0" ACTORID="Button">
                <PARAM NAME="Pos" VALUE="(14,5,5)"/>
                <BEHAVIOUR BEHAVIOURID="goToRoom0from2">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="User"  METHOD="route">
                            <PARAM NAME="to" VALUE="itinerary2to0"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ButtonRoom1to2" ACTORID="Button">
                <PARAM NAME="Pos" VALUE="(-10,5,6)"/>
                <BEHAVIOUR BEHAVIOURID="goToRoom2from1">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="User"  METHOD="route">
                            <PARAM NAME="to" VALUE="itinerary1to2"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ButtonRoom2to1" ACTORID="Button">
                <PARAM NAME="Pos" VALUE="(14,5,6)"/>
                <BEHAVIOUR BEHAVIOURID="goToRoom1from2">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="User"  METHOD="route">
                            <PARAM NAME="to" VALUE="itinerary2to1"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
        </SCENE>
    </SEQUENCE>
    <CONTEXT>
        <PARAM NAME="itinerary0to1" VALUE="[(0,-14,0) (0,0,0) (-10,0,0)]"/>
        <PARAM NAME="itinerary0to2" VALUE="[(0,-14,0) (0,0,0) (14,0,0)]"/>
        <PARAM NAME="itinerary1to0" VALUE="[(-10,0,0) (0,0,0) (0,-14,0)]"/>
        <PARAM NAME="itinerary2to0" VALUE="[(14,0,0) (0,0,0) (0,-14,0)]"/>
        <PARAM NAME="itinerary1to2" VALUE="[(-10,0,0) (0,0,0) (14,0,0)]"/>
        <PARAM NAME="itinerary2to1" VALUE="[(14,0,0) (0,0,0) (-10,0,0)]"/>
    </CONTEXT>
</XMMVR>
```

A.1.4.3. Deslizamiento/cable ferrocarril

Listado A.6: Implementación de la metáfora de cable en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
<XMMVR>
    <CAST>
        <STAGE STAGEID="Museum" FILE3D="Museum.W3D"/>
        <ACTOR ACTORID="Button" FILE3D="Button.W3D">
            <PARAMDECL NAME="Pos" TYPE="vector" />
            <PARAMDECL NAME="Content" TYPE="string" />
            <PARAMDECL NAME="Image" TYPE="string" />
        </ACTOR>
        <ACTOR ACTORID="Camera" FILE3D="null">
            <PARAMDECL NAME="Pos" TYPE="vector" />
            <PARAMDECL NAME="AT" TYPE="vector" />
            <METHODDECL NAME="go">
                <PARAMDECL NAME="to" TYPE="string"/>
            </METHODDECL>
            <METHODDECL NAME="turn">
                <PARAMDECL NAME="to" TYPE="string"/>
            </METHODDECL>
            <METHODDECL NAME="route">
                <PARAMDECL NAME="to" TYPE="string"/>
            </METHODDECL>
        </ACTOR>
    </CAST>
    <SEQUENCE>
        <SCENE STAGEID="Museum">
            <ACTORINSTANCE ACTORINSTANCEID="User" ACTORID="Camera">
                <PARAM NAME="Pos" VALUE="(0,0,0)" />
                <PARAM NAME="AT" VALUE="(0,0,1)" />
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ButtonRoom0to1" ACTORID="Button">
                <PARAM NAME="Pos" VALUE="(5,-14,5)"/>
                <BEHAVIOUR BEHAVIOURID="goToRoom1from0">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="Camera" METHOD="route">
                            <PARAM NAME="to" VALUE="itinerary0to1"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ButtonRoom1to2" ACTORID="Button">
                <PARAM NAME="Pos" VALUE="(-10,5,6)"/>
                <BEHAVIOUR BEHAVIOURID="goToRoom2from1">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="Camera" METHOD="route">
                            <PARAM NAME="to" VALUE="itinerary1to2"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
        </SCENE>
    </SEQUENCE>
    <CONTEXT>
        <PARAM NAME="itinerary0to1" VALUE="[(0,-14,0) (0,0,0) (-10,0,0)]"/>
        <PARAM NAME="itinerary1to2" VALUE="[(-10,0,0) (0,0,0) (14,0,0)]"/>
    </CONTEXT>
</XMMVR>
```

A.1.4.4. Silla voladora/alfombra

Listado A.7: Implementación de la metáfora de alfombra en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
<XMMVR>
  <CAST>
    <STAGE STAGEID="Corridor" FILE3D="Corridor.W3D"/>
    <STAGE STAGEID="Room0" FILE3D="Room0.W3D"/>
    <STAGE STAGEID="Room1" FILE3D="Room1.W3D"/>
    <STAGE STAGEID="Room2" FILE3D="Room2.W3D"/>
    <ACTOR ACTORID="Door" FILE3D="Door.W3D">
      <PARAMDECL NAME="Pos" TYPE="vector" />
      <PARAMDECL NAME="Content" TYPE="string" />
      <PARAMDECL NAME="Image" TYPE="string" />
    </ACTOR>
    <ACTOR ACTORID="Camera" FILE3D="null">
      <PARAMDECL NAME="Pos" TYPE="vector" />
      <PARAMDECL NAME="AT" TYPE="vector" />
      <METHODDECL NAME="go">
        <PARAMDECL NAME="to" TYPE="string"/>
      </METHODDECL>
      <METHODDECL NAME="turn">
        <PARAMDECL NAME="to" TYPE="string"/>
      </METHODDECL>
      <METHODDECL NAME="route">
        <PARAMDECL NAME="to" TYPE="string"/>
      </METHODDECL>
    </ACTOR>
  </CAST>
  <SEQUENCE>
    <SCENE STAGEID="Corridor">
      <ACTORINSTANCE ACTORINSTANCEID="User" ACTORID="Camera">
        <PARAM NAME="Pos" VALUE="(0,0,0)" />
        <PARAM NAME="AT" VALUE="(0,0,1)" />
        <BEHAVIOUR BEHAVIOURID="goAhead">
          <EVENTGUI TYPE="arrowUP"/>
          <ACTIONBLOQ>
            <ACTIONACT ACTORINSTANCEID="User" METHOD="go">
              <PARAM NAME="to" VALUE="front"/>
            </ACTIONACT>
          </ACTIONBLOQ>
        </BEHAVIOUR>
        <BEHAVIOUR BEHAVIOURID="goBack">
          <EVENTGUI TYPE="arrowDOWN"/>
          <ACTIONBLOQ>
            <ACTIONACT ACTORINSTANCEID="User" METHOD="go">
              <PARAM NAME="to" VALUE="back"/>
            </ACTIONACT>
          </ACTIONBLOQ>
        </BEHAVIOUR>
        <BEHAVIOUR BEHAVIOURID="turnLeft">
          <EVENTGUI TYPE="arrowLEFT"/>
          <ACTIONBLOQ>
            <ACTIONACT ACTORINSTANCEID="User" METHOD="turn">
              <PARAM NAME="to" VALUE="left"/>
            </ACTIONACT>
          </ACTIONBLOQ>
        </BEHAVIOUR>
        <BEHAVIOUR BEHAVIOURID="turnRight">
          <EVENTGUI TYPE="arrowRIGHT"/>
          <ACTIONBLOQ>
            <ACTIONACT ACTORINSTANCEID="User" METHOD="turn">
              <PARAM NAME="to" VALUE="right"/>
            </ACTIONACT>
          </ACTIONBLOQ>
        </BEHAVIOUR>
      </ACTORINSTANCE>
      <ACTORINSTANCE ACTORINSTANCEID="DoorRoom0" ACTORID="Door">
        <PARAM NAME="Pos" VALUE="(0,-14,0)"/>
        <BEHAVIOUR BEHAVIOURID="goToRoom0">
          <EVENTGUI TYPE="collision"/>
          <ACTIONBLOQ COND="$ACTORINSTANCE ==?User?">
            <ACTIONSYS TYPE="goToScene" DESTINATION="Room0"/>
          </ACTIONBLOQ>
        </BEHAVIOUR>
      </ACTORINSTANCE>
      <ACTORINSTANCE ACTORINSTANCEID="DoorRoom1" ACTORID="Door">
        <PARAM NAME="Pos" VALUE="(-10,0,0)"/>
        <BEHAVIOUR BEHAVIOURID="goToRoom1">
          <EVENTGUI TYPE="collision"/>
          <ACTIONBLOQ COND="$ACTORINSTANCE ==?User?">
            <ACTIONSYS TYPE="goToScene" DESTINATION="Room1"/>
          </ACTIONBLOQ>
        </BEHAVIOUR>
      </ACTORINSTANCE>
      <ACTORINSTANCE ACTORINSTANCEID="DoorRoom2" ACTORID="Door">
        <PARAM NAME="Pos" VALUE="(14,0,0)"/>
        <BEHAVIOUR BEHAVIOURID="goToRoom2">
          <EVENTGUI TYPE="collision"/>
          <ACTIONBLOQ COND="$ACTORINSTANCE ==?User?">
```

```
                        <ACTIONSYS TYPE="goToScene" DESTINATION="Room2"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
92      </SCENE>
        <SCENE STAGEID="Room0">
            <ACTORINSTANCE ACTORINSTANCEID="User0" ACTORID="Camera">
                <PARAM NAME="Pos" VALUE="(0,-15,0)" />
                <PARAM NAME="AT" VALUE="(0,-15,0)" />
97              <BEHAVIOUR BEHAVIOURID="goAhead">
                    <EVENTGUI TYPE="arrowUP"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="User0" METHOD="go">
                            <PARAM NAME="to" VALUE="front"/>
102                     </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="goBack">
                    <EVENTGUI TYPE="arrowDOWN"/>
107                 <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="User0" METHOD="go">
                            <PARAM NAME="to" VALUE="back"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
112             </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="turnLeft">
                    <EVENTGUI TYPE="arrowLEFT"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="User0" METHOD="turn">
117                         <PARAM NAME="to" VALUE="left"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="turnRight">
122                 <EVENTGUI TYPE="arrowRIGHT"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="User0" METHOD="turn">
                            <PARAM NAME="to" VALUE="right"/>
                        </ACTIONACT>
127                 </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor0" ACTORID="Door">
                <PARAM NAME="Pos" VALUE="(0,-14,0)"/>
132             <BEHAVIOUR BEHAVIOURID="goToCorridor">
                    <EVENTGUI TYPE="collision"/>
                    <ACTIONBLOQ COND="$ACTORINSTANCE ==?User0?">
                        <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                    </ACTIONBLOQ>
137             </BEHAVIOUR>
            </ACTORINSTANCE>
        </SCENE>
        <SCENE STAGEID="Room1">
            <ACTORINSTANCE ACTORINSTANCEID="User1" ACTORID="Camera">
142             <PARAM NAME="Pos" VALUE="(-11,0,0)" />
                <PARAM NAME="AT" VALUE="(-11,0,0)" />
                <BEHAVIOUR BEHAVIOURID="goAhead">
                    <EVENTGUI TYPE="arrowUP"/>
                    <ACTIONBLOQ>
147                     <ACTIONACT ACTORINSTANCEID="User1" METHOD="go">
                            <PARAM NAME="to" VALUE="front"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
152             <BEHAVIOUR BEHAVIOURID="goBack">
                    <EVENTGUI TYPE="arrowDOWN"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="User1" METHOD="go">
                            <PARAM NAME="to" VALUE="back"/>
157                     </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="turnLeft">
162                 <EVENTGUI TYPE="arrowLEFT"/>
                    <ACTIONBLOQ>
                        <ACTIONACT ACTORINSTANCEID="User1" METHOD="turn">
                            <PARAM NAME="to" VALUE="left"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
167             </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="turnRight">
                    <EVENTGUI TYPE="arrowRIGHT"/>
                    <ACTIONBLOQ>
172                     <ACTIONACT ACTORINSTANCEID="User1" METHOD="turn">
                            <PARAM NAME="to" VALUE="right"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
177         <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor1" ACTORID="Door">
                <PARAM NAME="Pos" VALUE="(-10,0,0)"/>
                <BEHAVIOUR BEHAVIOURID="goToCorridor">
                    <EVENTGUI TYPE="collision"/>
                    <ACTIONBLOQ COND="$ACTORINSTANCE ==?User1?">
```

```
                                    <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                               </ACTIONBLOQ>
                          </BEHAVIOUR>
                     </ACTORINSTANCE>
                </SCENE>
                <SCENE STAGEID="Room2">
                     <ACTORINSTANCE ACTORINSTANCEID="User2" ACTORID="Camera">
                          <PARAM NAME="Pos" VALUE="(15,0,0)"  />
                          <PARAM NAME="AT" VALUE="(15,0,0)"  />
                          <BEHAVIOUR BEHAVIOURID="goAhead">
                               <EVENTGUI TYPE="arrowUP"/>
                               <ACTIONBLOQ>
                                    <ACTIONACT ACTORINSTANCEID="User2"  METHOD="go">
                                         <PARAM NAME="to" VALUE="front"/>
                                    </ACTIONACT>
                               </ACTIONBLOQ>
                          </BEHAVIOUR>
                          <BEHAVIOUR BEHAVIOURID="goBack">
                               <EVENTGUI TYPE="arrowDOWN"/>
                               <ACTIONBLOQ>
                                    <ACTIONACT ACTORINSTANCEID="User2" METHOD="go">
                                         <PARAM NAME="to" VALUE="back"/>
                                    </ACTIONACT>
                               </ACTIONBLOQ>
                          </BEHAVIOUR>
                          <BEHAVIOUR BEHAVIOURID="turnLeft">
                               <EVENTGUI TYPE="arrowLEFT"/>
                               <ACTIONBLOQ>
                                    <ACTIONACT ACTORINSTANCEID="User2" METHOD="turn">
                                         <PARAM NAME="to" VALUE="left"/>
                                    </ACTIONACT>
                               </ACTIONBLOQ>
                          </BEHAVIOUR>
                          <BEHAVIOUR BEHAVIOURID="turnRight">
                               <EVENTGUI TYPE="arrowRIGHT"/>
                               <ACTIONBLOQ>
                                    <ACTIONACT ACTORINSTANCEID="User2" METHOD="turn">
                                         <PARAM NAME="to" VALUE="right"/>
                                    </ACTIONACT>
                               </ACTIONBLOQ>
                          </BEHAVIOUR>
                     </ACTORINSTANCE>
                     <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor2" ACTORID="Door">
                          <PARAM NAME="Pos" VALUE="(14,0,0)"/>
                          <BEHAVIOUR BEHAVIOURID="goToCorridor">
                               <EVENTGUI TYPE="collision"/>
                               <ACTIONBLOQ COND="$ACTORINSTANCE ==?User2?">
                                    <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                               </ACTIONBLOQ>
                          </BEHAVIOUR>
                     </ACTORINSTANCE>
                </SCENE>
           </SEQUENCE>
           <CONTEXT/>
      </XMMVR>
```

A.1.4.5. Tele-transporte/Beam me up

Listado A.8: Implementación de la metáfora de tele-transporte en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Level0" FILE3D="  Level0.W3D"/>
            <STAGE STAGEID="Level1" FILE3D="  Level1.W3D"/>
            <STAGE STAGEID="Level2" FILE3D="  Level2.W3D"/>
            <ACTOR ACTORID="Button" FILE3D="Button.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Destiny" TYPE="string" />
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Level0" >
                <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator01" ACTORID="Button">
                    <PARAM NAME="Pos" VALUE="(-2,37,5)"/>
                    <BEHAVIOUR BEHAVIOURID="go01">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Level1"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator02" ACTORID="Button">
                    <PARAM NAME="Pos" VALUE="(-2,37,6)"/>
                    <BEHAVIOUR BEHAVIOURID="go02">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Level2"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
            <SCENE STAGEID="Level1" >
                <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator10" ACTORID="Button">
                    <PARAM NAME="Pos" VALUE="(-2,37,20)"/>
                    <BEHAVIOUR BEHAVIOURID="go10">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Level0"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator12" ACTORID="Button">
                    <PARAM NAME="Pos" VALUE="(-2,37,20)"/>
                    <BEHAVIOUR BEHAVIOURID="go12">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Level2"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
            <SCENE STAGEID="Level2" >
                <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator21" ACTORID="Button">
                    <PARAM NAME="Pos" VALUE="(-2,37,35)"/>
                    <BEHAVIOUR BEHAVIOURID="go21">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Level1"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="ButtonElevator20" ACTORID="Button">
                    <PARAM NAME="Pos" VALUE="(-2,37,36)"/>
                    <BEHAVIOUR BEHAVIOURID="go20">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Level0"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
        </SEQUENCE>
        <CONTEXT/>
    </XMMVR>
```

A.2 Implementación de metáforas vocales

A.2.1. Agente interfaz

Listado A.9: Implementación de la metáfora de agente interfaz en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Room0" FILE3D="Room0.W3D"/>
            <ACTOR ACTORID="Piece" FILE3D="Piece.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
                <METHODDECL NAME="rotate">
                    <PARAMDECL NAME="to" TYPE="string" />
                </METHODDECL>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Room0" >
                <BEHAVIOUR BEHAVIOURID="listening0">
                    <EVENTSYS TYPE="onEnterScene"/>
                    <ACTIONBLOQ>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_begin"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Room0End">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action'] ==?'finish'?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <ACTORINSTANCE ACTORINSTANCEID="Amstrad" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(-5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Amstrad" />
                    <PARAM NAME="Image" VALUE="Amstrad.jpg" />
                    <BEHAVIOUR BEHAVIOURID="Amstrad">
                        <EVENTVUI TYPE="endedDialog"/>
                        <ACTIONBLOQ COND="$OUTDIALOG['action']==?'show'? AND $OUTDIALOG['piece
                            ']==?'amstrad'?">
                            <ACTIONACT ACTORINSTANCEID="Amstrad" METHOD="rotate">
                                <PARAM NAME="to" VALUE="Left"/>
                            </ACTIONACT>
                            <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_piece">
                                <PARAM NAME="piece" VALUE="amstrad"/>
                            </ACTIONVUI>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="Macintosh" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Macintosh" />
                    <PARAM NAME="Image" VALUE="Macintosh_Plus.jpg" />
                    <BEHAVIOUR BEHAVIOURID="Macintosh">
                        <EVENTVUI TYPE="endedDialog"/>
                        <ACTIONBLOQ COND="$OUTDIALOG['action']==?'show'? AND $OUTDIALOG['piece
                            ']==?'macintosh'?">
                            <ACTIONACT ACTORINSTANCEID="Macintosh" METHOD="rotate">
                                <PARAM NAME="to" VALUE="Right" />
                            </ACTIONACT>
                            <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_piece">
                                <PARAM NAME="piece" VALUE="macintosh"/>
                            </ACTIONVUI>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
        </SEQUENCE>
        <CONTEXT>
            <DIALOG DIALOG_ID="VXML_begin" SPEECHFILE="http://localhost:8080/
                museo3dinterfaceagent/voiceXML/museo3dinterfaceagent_begin.vxml">
                <OUTPUT_PARAM NAME="action"/>
                <OUTPUT_PARAM NAME="piece"/>
            </DIALOG>
            <DIALOG DIALOG_ID="VXML_piece" SPEECHFILE="http://localhost:8080/
                museo3dinterfaceagent/voiceXML/museo3dinterfaceagent_piece.vxml">
                <INPUT_PARAM NAME="piece"/>
                <OUTPUT_PARAM NAME="action"/>
                <OUTPUT_PARAM NAME="piece"/>
            </DIALOG>
            <DIALOG DIALOG_ID="VXML_end" SPEECHFILE="http://localhost:8080/museo3dinterfaceagent/
                voiceXML/museo3dinterfaceagent_end.vxml"/>
        </CONTEXT>
    </XMMVR>
```

Listado A.10: Diálogo I de la metáfora de agente interfaz en el Museo 3D Multimodal (VXML_begin)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
      XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <block>
            <prompt bargein="false" xml:lang="en-us">Welcome to our museum. My name is Ana, I
                will help you in your visit. </prompt>
        </block>
        <field name="piece">
            <grammar src="interfaceagent.BNF"/>
            <prompt xml:lang="en-us">Tell me which piece you want to interact with: Amstrad or
                Macintosh. Say finish to finish your visit</prompt>
        </field>
            <field name="finish">
            <grammar src="interfaceagent.BNF"/>
        </field>
        <subdialog name="checkpiece" src="museo3dinterfaceagent_piece.vxml">
            <param name="piece" expr="piece"/>
                    <param name="finish" expr="finish"/>
        </subdialog>
    </form>
</vxml>
```

Listado A.11: Diálogo II de la metáfora de agente interfaz en el Museo 3D Multimodal (VXML_piece)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
      XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
      voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <var name="piece"/>
        <var name="finish"/>
        <block>
            <if cond="finish=='finish'">
                <submit next="museo3dinterfaceagent_end.vxml"/>
            </if>
            <if cond="piece=='amstrad'">
                <submit next="http://localhost:8080/museo3dinterfaceagent/Dispatcher?action=
                    rotate&amp;piece=amstrad" method="post"/>
                <prompt xml:lang="en-us"> OK. This is the <value expr="piece"/> machine. It was
                    very popular in the eighties. </prompt>
            </if>
            <if cond="piece=='macintosh'">
                <submit next="http://localhost:8080/museo3dinterfaceagent/Dispatcher?action=
                    rotate&amp;piece=macintosh" method="post"/>
                <prompt xml:lang="en-us"> OK. This is the <value expr="piece"/> machine. It is
                    the enemy of PC. </prompt>
            </if>
        </block>
    </form>
</vxml>
```

Listado A.12: Diálogo III de la metáfora de agente interfaz en el Museo 3D Multimodal (VXML_end)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
      XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
      voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <block>
            <prompt xml:lang="en-us">Bye. Thank you for visiting us. Come back again whenever you
                want.</prompt>
        </block>
    </form>
</vxml>
```

Listado A.13: Gramática utilizada para la metáfora de agente interfaz en el Museo 3D Multimodal (interfaceagent.BNF)

```
#ABNF 1.0 ISO8859-1;

root $action = [[[$show] [$piece]] | [$finish] ] ;

$show = show me the | show | let's see | let's see the ;
$piece = amstrad | macintosh ;
$finish = ( finish | end | stop | bye ) {"finish"} ;
```

A.2.2. Proxy o Delegado

Listado A.14: Implementación de la metáfora de delegado en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Level0" FILE3D="Level0.W3D"/>
            <STAGE STAGEID="Level1" FILE3D="Level1.W3D"/>
            <STAGE STAGEID="Elevator" FILE3D="Elevator.W3D"/>
            <ACTOR ACTORID="Piece" FILE3D="Piece.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
                <METHODDECL NAME="rotate">
                    <PARAMDECL NAME="to" TYPE="string" />
                </METHODDECL>
            </ACTOR>
            <ACTOR ACTORID="Door" FILE3D="Door.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <METHODDECL NAME="open"/>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Level0" >
                <BEHAVIOUR BEHAVIOURID="listening0">
                    <EVENTSYS TYPE="onEnterScene"/>
                    <ACTIONBLOQ>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_begin"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Amstrad">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'show'? AND $OUTDIALOG['piece']==?
                        amstrad? AND $OUTDIALOG['agent']==?'ana'?">
                        <ACTIONACT ACTORINSTANCEID="Amstrad" METHOD="rotate">
                            <PARAM NAME="to" VALUE="Left"/>
                        </ACTIONACT>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_piece">
                            <PARAM NAME="piece" VALUE="amstrad"/>
                            <PARAM NAME="agent" VALUE="ana"/>
                        </ACTIONVUI>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Macintosh">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'show'? AND $OUTDIALOG['piece']==?
                        macintosh? AND $OUTDIALOG['agent']==?'ana'?">
                        <ACTIONACT ACTORINSTANCEID="Macintosh" METHOD="rotate">
                            <PARAM NAME="to" VALUE="Right" />
                        </ACTIONACT>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_piece">
                            <PARAM NAME="piece" VALUE="macintosh"/>
                            <PARAM NAME="agent" VALUE="ana"/>
                        </ACTIONVUI>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="go01">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'go'? AND $OUTDIALOG['floor']==?'
                        first'? AND $OUTDIALOG['agent']==?'pedro'?">
                        <ACTIONACT ACTORINSTANCEID="DoorElevator" METHOD="open"/>
                        <ACTIONSYS TYPE="goToScene" DESTINATION="Elevator"/>
                        <ACTIONSYS TYPE="setTimer" DESTINATION="Elevator" TIMERID="0to1" TIME="10
                            ">
                            <ASSIGN NAME="newScene" VALUE="Level1"/>
                        </ACTIONSYS>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_floor"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Level0End">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action'] ==?'finish'?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end">
                            <PARAM NAME="floor" VALUE="zero"/>
                            <PARAM NAME="agent" VALUE="pedro"/>
                        </ACTIONVUI>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <ACTORINSTANCE ACTORINSTANCEID="Amstrad" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(-5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Amstrad" />
                    <PARAM NAME="Image" VALUE="Amstrad.jpg" />
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="Macintosh" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Macintosh" />
                    <PARAM NAME="Image" VALUE="Macintosh_Plus.jpg" />
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="DoorElevator0" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(0,37,0)"/>
```

```
                </ACTORINSTANCE>
            </SCENE>
            <SCENE STAGEID="Level1">
                <BEHAVIOUR BEHAVIOURID="listening1">
                    <EVENTSYS TYPE="onEnterScene"/>
                    <ACTIONBLOQ>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_floor">
                            <PARAM NAME="floor" VALUE="first"/>
                            <PARAM NAME="agent" VALUE="pedro"/>
                        </ACTIONVUI>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="go10">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'go'? AND $OUTDIALOG['floor']==?'
                        zero'? AND $OUTDIALOG['agent']==?'pedro'?">
                        <ACTIONACT ACTORINSTANCEID="DoorElevator" METHOD="open"/>
                        <ACTIONSYS TYPE="goToScene" DESTINATION="Elevator"/>
                        <ACTIONSYS TYPE="setTimer" DESTINATION="Elevator" TIMERID="1to0" TIME="10
                            ">
                            <ASSIGN NAME="newScene" VALUE="Level0"/>
                        </ACTIONSYS>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_floor"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Level1End">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action'] ==?'finish'?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <ACTORINSTANCE ACTORINSTANCEID="DoorElevator1" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(0,37,15)"/>
                </ACTORINSTANCE>
            </SCENE>
            <SCENE STAGEID="Elevator" >
                <ACTORINSTANCE ACTORINSTANCEID="DoorElevatorA" ACTORID="Door">
                    <PARAM NAME="Pos " VALUE=" (0,37,0)" />
                </ACTORINSTANCE>
            </SCENE>
        </SEQUENCE>
        <CONTEXT>
            <VARIABLE NAME="newScene" TYPE="string" VALUE="Level0"/>
            <DIALOG DIALOG_ID="VXML_begin" SPEECHFILE="http://localhost:8080/museo3dproxy/
                voiceXML/museo3dproxy_begin.vxml">
                <OUTPUT_PARAM NAME="action"/>
                <OUTPUT_PARAM NAME="piece"/>
                <OUTPUT_PARAM NAME="agent"/>
            </DIALOG>
            <DIALOG DIALOG_ID="VXML_piece" SPEECHFILE="http://localhost:8080/museo3dproxy/
                voiceXML/museo3dproxy_piece.vxml">
                <INPUT_PARAM NAME="piece"/>
                <OUTPUT_PARAM NAME="action"/>
                <OUTPUT_PARAM NAME="piece"/>
                <OUTPUT_PARAM NAME="agent"/>
            </DIALOG>
            <DIALOG DIALOG_ID="VXML_floor" SPEECHFILE="http://localhost:8080/museo3dproxy/
                voiceXML/museo3dproxy_floor.vxml">
                <INPUT_PARAM NAME="floor"/>
                <OUTPUT_PARAM NAME="action"/>
                <OUTPUT_PARAM NAME="floor"/>
                <OUTPUT_PARAM NAME="agent"/>
            </DIALOG>
            <DIALOG DIALOG_ID="VXML_end" SPEECHFILE="http://localhost:8080/museo3dproxy/voiceXML/
                museo3dproxy_end.vxml"/>
        </CONTEXT>
    </XMMVR>
```

LISTADO A.15: Diálogo I de la metáfora de delegado en el Museo 3D Multimodal (VXML_begin)

```
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <block>
            <prompt bargein="false" xml:lang="en-us">Welcome to our museum. My name is Ana, I
                will help you in your visit. </prompt>
        </block>
        <field name="finish">
            <grammar src="proxy.BNF"/>
        </field>
        <field name="piece">
            <grammar src="proxy.BNF"/>
            <prompt xml:lang="en-us">Tell me which piece you want to interact with: Amstrad or
                Macintosh. Say finish to finish your visit</prompt>
        </field>
        <subdialog name="checkpiece" src="museo3dproxy_piece.vxml">
            <param name="piece" expr="piece"/>
            <param name="finish" expr="finish"/>
        </subdialog>
        <field name="floor">
```

```
            <grammar src="proxy.BNF"/>
            <prompt xml:lang="en-us"> If you want to go to another floor, say it to Pedro. Say
                finish to finish your visit.</prompt>
        </field>
        <subdialog name="checkfloor" src="museo3dproxy_floor.vxml">
            <param name="floor" expr="floor"/>
            <param name="finish" expr="finish"/>
        </subdialog>
    </form>
</vxml>
```

Listado A.16: Diálogo II de la metáfora de delegado en el Museo 3D Multimodal (VXML_piece)

```
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <var name="piece"/>
        <var name="action"/>
        <block>
            <if cond="action=='finish'">
                <submit next="museo3dproxy_end.vxml"/>
            </if>
            <if cond="piece=='amstrad'">
                <submit next="http://localhost:8080/museo3dproxy/Dispatcher?action=rotate&amp;
                    piece=amstrad" method="post"/>
                <prompt xml:lang="en-us"> Amstrad computer was very popular in the eighties </
                    prompt>
            </if>
            <if cond="piece=='macintosh'">
                <submit next="http://localhost:8080/museo3dproxy/Dispatcher?action=rotate&amp;
                    piece=macintosh" method="post"/>
                <prompt xml:lang="en-us"> Macintosh computer is the enemy of PC </prompt>
            </if>
            <prompt xml:lang="en-us">OK. This is the <value expr="piece"/> machine.</prompt>
        </block>
    </form>
</vxml>
```

Listado A.17: Diálogo III de la metáfora de delegado en el Museo 3D Multimodal (VXML_floor)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <var name="finish"/>
        <var name="floor"/>
        <block>
            <if cond="finish=='finish'">
                <submit next="museo3dproxy_end.vxml"/>
            </if>
            <if cond="floor=='zero'">
                <submit next="http://localhost:8080/museo3dproxy/Dispatcher?action=go&amp;floor=
                    zero" method="post"/>
                <prompt/>
            </if>
            <if cond="floor=='first'">
                <submit next="http://localhost:8080/museo3dproxy/Dispatcher?action=go&amp;piece=
                    first" method="post"/>
                <prompt/>
            </if>
            <prompt>OK. This is the <value expr="floor"/> floor. </prompt>
        </block>
    </form>
</vxml>
```

Listado A.18: Diálogo IV de la metáfora de delegado en el Museo 3D Multimodal (VXML_end)

```
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <block>
            <prompt xml:lang="en-us">Bye. Thank you for visiting us. Come back again whenever you
                want.</prompt>
        </block>
    </form>
</vxml>
```

LISTADO A.19: Gramática I utilizada para la metáfora de delegado en el Museo 3D Multimodal (proxy.BNF)

```
#ABNF 1.0 ISO8859-1;

root $action = [ [[ana] [$show] [$piece]] | [[pedro] [$go] [$floor]] [$finish] ] ;

$show = show me the | show | let's see | let's see the ;
$piece = amstrad | macintosh ;
$finish = ( finish | end | stop | bye ) {"finish"} ;
$go = go to the | take me to the | to the | let's go to the ;
$floor = zero | first ;

#ABNF 1.0;
language es;
mode voice;
root $piece;
public $piece =
root $floor;
public $floor =

ana show (amstrad | macintosh) | pedro go (zero | first) | finish
```

A.2.3. Divinidad

Listado A.20: Implementación de la metáfora de divinidad en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Room0" FILE3D="Room0.W3D"/>
            <ACTOR ACTORID="Piece" FILE3D="Piece.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
                <METHODDECL NAME="rotate">
                    <PARAMDECL NAME="to" TYPE="string" />
                </METHODDECL>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Room0" >
                <BEHAVIOUR BEHAVIOURID="listening0">
                    <EVENTSYS TYPE="onEnterScene"/>
                    <ACTIONBLOQ>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_begin"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Room0End">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action'] ==?'finish'?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <ACTORINSTANCE ACTORINSTANCEID="Amstrad" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(-5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Amstrad" />
                    <PARAM NAME="Image" VALUE="Amstrad.jpg" />
                    <BEHAVIOUR BEHAVIOURID="Amstrad">
                        <EVENTVUI TYPE="endedDialog"/>
                        <ACTIONBLOQ COND="$OUTDIALOG['action']==?'show'? AND $OUTDIALOG['piece
                            ']==?'amstrad'?">
                            <ACTIONACT ACTORINSTANCEID="Amstrad" METHOD="rotate">
                                <PARAM NAME="to" VALUE="Left"/>
                            </ACTIONACT>
                            <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_piece"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="Macintosh" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Macintosh" />
                    <PARAM NAME="Image" VALUE="Macintosh_Plus.jpg" />
                    <BEHAVIOUR BEHAVIOURID="Macintosh">
                        <EVENTVUI TYPE="endedDialog"/>
                        <ACTIONBLOQ COND="$OUTDIALOG['action']==?'show'? AND $OUTDIALOG['piece
                            ']==?'macintosh'?">
                            <ACTIONACT ACTORINSTANCEID="Macintosh" METHOD="rotate">
                                <PARAM NAME="to" VALUE="Right" />
                            </ACTIONACT>
                            <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_piece"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
        </SEQUENCE>
        <CONTEXT>
            <DIALOG DIALOG_ID="VXML_begin" SPEECHFILE="http://localhost:8080/museo3ddivinity/
                voiceXML/museo3ddivinity_begin.vxml">
                <OUTPUT_PARAM NAME="action"/>
                <OUTPUT_PARAM NAME="piece"/>
            </DIALOG>
            <DIALOG DIALOG_ID="VXML_piece" SPEECHFILE="http://localhost:8080/museo3ddivinity/
                voiceXML/museo3ddivinity_piece.vxml">
                <INPUT_PARAM NAME="piece"/>
                <OUTPUT_PARAM NAME="action"/>
                <OUTPUT_PARAM NAME="piece"/>
            </DIALOG>
            <DIALOG DIALOG_ID="VXML_end" SPEECHFILE="http://localhost:8080/museo3ddivinity/
                voiceXML/museo3ddivinity_end.vxml"/>
        </CONTEXT>
    </XMMVR>
```

LISTADO A.21: Diálogo I de la metáfora de divinidad en el Museo 3D Multimodal (VXML_begin)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
  <form id="seccion">
    <block>
      <prompt/>
    </block>
    <field name="finish">
      <grammar src="divinity.BNF"/>
    </field>
    <field name="piece">
      <grammar src="divinity.BNF"/>
    </field>
    <subdialog name="checkmachines" src="museo3ddivinity_piece.vxml">
      <param name="piece" expr="piece"/>
      <param name="finish" expr="finish"/>
    </subdialog>
  </form>
</vxml>
```

LISTADO A.22: Diálogo II de la metáfora de divinidad en el Museo 3D Multimodal (VXML_piece)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
  <form id="seccion">
    <var name="piece"/>
    <var name="action"/>
    <block>
      <if cond="finish=='finish'">
        <submit next="museo3ddivinity_end.vxml"/>
      </if>
      <if cond="piece=='macintosh'">
        <submit next="http://localhost:8080/museo3ddivinity/Dispatcher?action=show&amp;
            piece=macintosh" method="post"/>
        <prompt/>
      </if>
      <if cond="piece=='amstrad'">
        <submit next="http://localhost:8080/museo3ddivinity/Dispatcher?action=show&amp;
            piece=amstrad" method="post"/>
        <prompt/>
      </if>
    </block>
  </form>
</vxml>
```

LISTADO A.23: Diálogo III de la metáfora de divinidad en el Museo 3D Multimodal (VXML_end)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
  <form id="seccion">
    <block/>
  </form>
</vxml>
```

LISTADO A.24: Gramática utilizada para la metáfora de divinidad en el Museo 3D Multimodal (divinity.BNF)

```
#ABNF 1.0 ISO8859-1;

root $action = [ [[$show] [$piece]] | [$finish] ] ;

$show = show me the | show | let's see | let's see the ;
$piece = amstrad | macintosh ;
$finish = ( finish | end | stop | bye ) {"finish"} ;


#ABNF 1.0;
language es;
mode voice;
root $piece;
public $piece =

show (amstrad | macintosh) | finish
```

A.2.4. Telekinesis

Listado A.25: Implementación de la metáfora de telekinesis en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Level0" FILE3D="Level0.W3D"/>
            <STAGE STAGEID="Level1" FILE3D="Level1.W3D"/>
            <STAGE STAGEID="Elevator" FILE3D="Elevator.W3D"/>
            <ACTOR ACTORID="Door" FILE3D="Door.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <METHODDECL NAME="open"/>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Level0" >
                <BEHAVIOUR BEHAVIOURID="listening0">
                    <EVENTSYS TYPE="onEnterScene"/>
                    <ACTIONBLOQ>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_begin">
                            <PARAM NAME="floor" VALUE="zero"/>
                        </ACTIONVUI>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Level0End">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']=='exit'?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <ACTORINSTANCE ACTORINSTANCEID="DoorElevator0" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(0,37,0)"/>
                    <BEHAVIOUR BEHAVIOURID="go01">
                        <EVENTVUI TYPE="endedDialog"/>
                        <ACTIONBLOQ COND="$OUTDIALOG['action']=='go'? AND $OUTDIALOG['floor
                            ']=='first'?">
                            <ACTIONACT ACTORINSTANCEID="DoorElevator" METHOD="open"/>
                            <ACTIONSYS TYPE="setTimer" DESTINATION="Elevator" TIMERID="0to1" TIME
                                ="10">
                                <ASSIGN NAME="newScene" VALUE="Level1"/>
                            </ACTIONSYS>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
            <SCENE STAGEID="Level1">
                <BEHAVIOUR BEHAVIOURID="listening1">
                    <EVENTSYS TYPE="onEnterScene"/>
                    <ACTIONBLOQ>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_floor">
                            <PARAM NAME="floor" VALUE="first"/>
                        </ACTIONVUI>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Level1End">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']=='exit'?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end">
                            <PARAM NAME="floor" VALUE="zero"/>
                        </ACTIONVUI>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <ACTORINSTANCE ACTORINSTANCEID="DoorElevator1" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(0,37,15)"/>
                    <BEHAVIOUR BEHAVIOURID="go10">
                        <EVENTVUI TYPE="endedDialog"/>
                        <ACTIONBLOQ COND="$OUTDIALOG['action']=='go'? AND $OUTDIALOG['floor
                            ']=='zero'?">
                            <ACTIONACT ACTORINSTANCEID="DoorElevator" METHOD="open"/>
                            <ACTIONSYS TYPE="setTimer" DESTINATION="Elevator" TIMERID="1to0" TIME
                                ="10">
                                <ASSIGN NAME="newScene" VALUE="Level0"/>
                            </ACTIONSYS>
                            <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_floor"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
            <SCENE STAGEID="Elevator" >
                <ACTORINSTANCE ACTORINSTANCEID="DoorElevatorA" ACTORID="Door">
                    <PARAM NAME="Pos " VALUE=" (0,37,0)" />
                </ACTORINSTANCE>
            </SCENE>
        </SEQUENCE>
        <CONTEXT>
            <VARIABLE NAME="newScene" TYPE="string" VALUE="Level0"/>
            <DIALOG DIALOG_ID="VXML_begin" SPEECHFILE="http://localhost:8080/museo3dtelekinesis/
                voiceXML/museo3dtelekinesis_begin.vxml">
                <OUTPUT_PARAM NAME="action"/>
                <OUTPUT_PARAM NAME="floor"/>
```

```
            </DIALOG>
            <DIALOG DIALOG_ID="VXML_floor" SPEECHFILE="http://localhost:8080/museo3dtelekinesis/
                voiceXML/museo3dtelekinesis_floor.vxml">
                <INPUT_PARAM NAME="floor"/>
                <OUTPUT_PARAM NAME="action"/>
                <OUTPUT_PARAM NAME="floor"/>
            </DIALOG>
            <DIALOG DIALOG_ID="VXML_end" SPEECHFILE="http://localhost:8080/museo3dtelekinesis/
                voiceXML/museo3dtelekinesis_end.vxml"/>
        </CONTEXT>
    </XMMVR>
```

Listado A.26: Diálogo I de la metáfora de telekinesis en el Museo 3D Multimodal (VXML_begin)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <block>
            <prompt/>
        </block>
        <field name="finish">
            <grammar src="telekinesis.BNF"/>
        </field>
        <field name="floor">
            <grammar src="telekinesis.BNF"/>
        </field>
        <subdialog name="checkfloor" src="museo3dtelekinesis_floor.vxml">
            <param name="floor" expr="floor"/>
            <param name="finish" expr="finish"/>
        </subdialog>
    </form>
</vxml>
```

Listado A.27: Diálogo II de la metáfora de telekinesis en el Museo 3D Multimodal (VXML_floor)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <var name="floor"/>
        <var name="finish"/>
        <block>
            <if cond="finish=='finish'">
                <submit next="museo3dinterfaceagent_end.vxml"/>
            </if>
            <if cond="floor=='zero'">
                <submit next="http://localhost:8080/museo3dinterfaceagent/Dispatcher?action=go&
                    amp;floor=zero" method="post"/>
                <prompt/>
            </if>
            <if cond="floor=='first'">
                <submit next="http://localhost:8080/museo3dinterfaceagent/Dispatcher?action=go&
                    amp;piece=first" method="post"/>
                <prompt/>
            </if>
        </block>
    </form>
</vxml>
```

Listado A.28: Diálogo III de la metáfora de telekinesis en el Museo 3D Multimodal (VXML_end)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <block/>
    </form>
</vxml>
```

Listado A.29: Gramática utilizada para la metáfora de telekinesis en el Museo 3D Multimodal (telekinesis.BNF)

```
#ABNF 1.0 ISO8859-1;

root $action = [[[$go] [$floor]] | [$finish] ] ;

$go = go to the | take me to the | to the | let's go to the ;
$floor = zero | first ;
$finish = ( finish | end | stop | bye ) {"finish"} ;


#ABNF 1.0;
language es;
mode voice;
root $floor;
public $floor =

go (zero | first) | finish
```

A.3 Implementación de cooperación entre modalidades

A.3.1. Especialización

Listado A.30: Implementación de la cooperación entre modalidades por especialización en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Corridor" FILE3D="Corridor.W3D"/>
            <STAGE STAGEID="Room0" FILE3D="Room0.W3D"/>
            <STAGE STAGEID="Room1" FILE3D="Room1.W3D"/>
            <STAGE STAGEID="Room2" FILE3D="Room2.W3D"/>
            <ACTOR ACTORID="Door" FILE3D="Door.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
            </ACTOR>
            <ACTOR ACTORID="Piece" FILE3D="Piece.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
                <METHODDECL NAME="rotate">
                    <PARAMDECL NAME="to" TYPE="string" />
                </METHODDECL>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Room0">
                <BEHAVIOUR BEHAVIOURID="listening0">
                    <EVENTSYS TYPE="onEnterScene"/>
                    <ACTIONBLOQ>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_begin"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Amstrad">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'show'? AND $OUTDIALOG['piece']==?'
                        amstrad'?">
                        <ACTIONACT ACTORINSTANCEID="Amstrad" METHOD="rotate">
                            <PARAM NAME="to" VALUE="Left"/>
                        </ACTIONACT>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_piece"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Macintosh">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'show'? AND $OUTDIALOG['piece']==?'
                        macintosh'?">
                        <ACTIONACT ACTORINSTANCEID="Macintosh" METHOD="rotate">
                            <PARAM NAME="to" VALUE="Right" />
                        </ACTIONACT>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_piece"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Room0End">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'finish'?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <ACTORINSTANCE ACTORINSTANCEID="Amstrad" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(-5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Amstrad" />
                    <PARAM NAME="Image" VALUE="Amstrad.jpg" />
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="Macintosh" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(5,-35,5)" />
                    <PARAM NAME="Content" VALUE="Macintosh" />
                    <PARAM NAME="Image" VALUE="Macintosh_Plus.jpg" />
                </ACTORINSTANCE>
                <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor0" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(0,-14,0)" />
                    <BEHAVIOUR BEHAVIOURID="BackToCorridor">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                </ACTORINSTANCE>
            </SCENE>
            <SCENE STAGEID="Corridor">
                <BEHAVIOUR BEHAVIOURID="CorridorEnd">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'finish'?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
```

```
                    <ACTORINSTANCE ACTORINSTANCEID="DoorRoom0" ACTORID="Door">
                        <PARAM NAME="Pos" VALUE="(0,-14,0)" />
                        <BEHAVIOUR BEHAVIOURID="enterRoom0">
                            <EVENTGUI TYPE="mouseDown"/>
                            <ACTIONBLOQ>
                                <ACTIONSYS TYPE="goToScene" DESTINATION="Room0"/>
                            </ACTIONBLOQ>
                        </BEHAVIOUR>
                    </ACTORINSTANCE>
                    <ACTORINSTANCE ACTORINSTANCEID="DoorRoom1" ACTORID="Door">
                        <PARAM NAME="Pos" VALUE="(-10,0,0)" />
                        <BEHAVIOUR BEHAVIOURID="enterRoom1">
                            <EVENTGUI TYPE="mouseDown"/>
                            <ACTIONBLOQ>
                                <ACTIONSYS TYPE="goToScene" DESTINATION="Room1"/>
                            </ACTIONBLOQ>
                        </BEHAVIOUR>
                    </ACTORINSTANCE>
                    <ACTORINSTANCE ACTORINSTANCEID="DoorRoom2" ACTORID="Door">
                        <PARAM NAME="Pos" VALUE="(14,0,0)" />
                        <BEHAVIOUR BEHAVIOURID="enterRoom2">
                            <EVENTGUI TYPE="mouseDown"/>
                            <ACTIONBLOQ>
                                <ACTIONSYS TYPE="goToScene" DESTINATION="Room2"/>
                            </ACTIONBLOQ>
                        </BEHAVIOUR>
                    </ACTORINSTANCE>
                </SCENE>
                <SCENE STAGEID="Room1">
                    <BEHAVIOUR BEHAVIOURID="Room1End">
                        <EVENTVUI TYPE="endedDialog"/>
                        <ACTIONBLOQ COND="$OUTDIALOG['action']==?'finish'?">
                            <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                    <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor1" ACTORID="Door">
                        <PARAM NAME="Pos" VALUE="(-10,0,0)" />
                        <BEHAVIOUR BEHAVIOURID="BackToCorridor">
                            <EVENTGUI TYPE="mouseDown"/>
                            <ACTIONBLOQ>
                                <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                            </ACTIONBLOQ>
                        </BEHAVIOUR>
                    </ACTORINSTANCE>
                </SCENE>
                <SCENE STAGEID="Room2">
                    <BEHAVIOUR BEHAVIOURID="Room2End">
                        <EVENTVUI TYPE="endedDialog"/>
                        <ACTIONBLOQ COND="$OUTDIALOG['action']==?'finish'?">
                            <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                        </ACTIONBLOQ>
                    </BEHAVIOUR>
                    <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor2" ACTORID="Door">
                        <PARAM NAME="Pos" VALUE="(14,0,0)" />
                        <BEHAVIOUR BEHAVIOURID="BackToCorridor">
                            <EVENTGUI TYPE="mouseDown"/>
                            <ACTIONBLOQ>
                                <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                            </ACTIONBLOQ>
                        </BEHAVIOUR>
                    </ACTORINSTANCE>
                </SCENE>
            </SEQUENCE>
            <CONTEXT>
                <DIALOG DIALOG_ID="VXML_begin" SPEECHFILE="http://localhost:8080/
                    museo3dspecialisation/voiceXML/museo3dspecialisation_begin.vxml">
                    <OUTPUT_PARAM NAME="action"/>
                    <OUTPUT_PARAM NAME="piece"/>
                </DIALOG>
                <DIALOG DIALOG_ID="VXML_piece" SPEECHFILE="http://localhost:8080/
                    museo3dspecialisation/voiceXML/museo3dspecialisation_piece.vxml">
                    <INPUT_PARAM NAME="piece"/>
                    <OUTPUT_PARAM NAME="action"/>
                    <OUTPUT_PARAM NAME="piece"/>
                </DIALOG>
                <DIALOG DIALOG_ID="VXML_end" SPEECHFILE="http://localhost:8080/museo3dspecialisation/
                    voiceXML/museo3dspecialisation_end.vxml"/>
            </CONTEXT>
        </XMMVR>
```

Listado A.31: Diálogo I de la cooperación entre modalidades por especialización en el Museo 3D Multimodal (VXML_begin)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
      XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <block>
            <prompt bargein="false" xml:lang="en-us">Welcome to our museum. My name is Ana, I
                will help you in your visit. </prompt>
```

```
        </block>
        <field name="finish">
            <grammar src="specialisation.BNF"/>
        </field>
        <field name="piece">
            <grammar src="specialisation.BNF"/>
            <prompt xml:lang="en-us">Tell me which piece you want to interact with: Amstrad or
                Macintosh. Click on a door to go to another room. Say finish to finish your
                visit</prompt>
        </field>
        <subdialog name="checkpiece" src="museo3dspecialisation_piece.vxml">
            <param name="piece" expr="piece"/>
            <param name="finish" expr="finish"/>
        </subdialog>
    </form>
</vxml>
```

Listado A.32: Diálogo II de la cooperación entre modalidades por especialización en el Museo 3D Multimodal (VXML_piece)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <var name="piece"/>
        <var name="action"/>
        <block>
            <if cond="finish=='finish'">
                <submit next="museo3dspecialisation_end.vxml"/>
            </if>
            <if cond="piece=='amstrad'">
                <submit next="http://localhost:8080/museo3dspecialisation/Dispatcher?action=
                    rotate&amp;piece=amstrad" method="post"/>
                <prompt xml:lang="en-us"> OK. This is the <value expr="piece"/> machine. It was
                    very popular in the eighties. </prompt>
            </if>
            <if cond="piece=='macintosh'">
                <submit next="http://localhost:8080/museo3dspecialisation/Dispatcher?action=
                    rotate&amp;piece=macintosh" method="post"/>
                <prompt xml:lang="en-us"> OK. This is the <value expr="piece"/> machine. It is
                    the enemy of PC. </prompt>
            </if>
        </block>
    </form>
</vxml>
```

Listado A.33: Diálogo III de la cooperación entre modalidades por especialización en el Museo 3D Multimodal (VXML_end)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
        <form id="seccion">
            <block>Bye. Thank you for visiting us. Come back again whenever you want. </block>
        </form>
    </vxml>
```

Listado A.34: Gramática utilizada en la cooperación entre modalidades por especialización en el Museo 3D Multimodal (specialisation.BNF)

```
#ABNF 1.0 ISO8859-1;

root $action = [[[$show] [$piece]] | [$finish] ] ;

$show = show me the | show | let's see | let's see the ;
$piece = amstrad | macintosh ;
$finish = ( finish | end | stop | bye ) {"finish"} ;

#ABNF 1.0;
language es;
mode voice;
root $piece;
public $piece =

show (amstrad | macintosh) | finish
```

A.3.2. Equivalencia / Redundancia

Listado A.35: Implementación de la cooperación entre modalidades por equivalencia/redundancia en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Corridor" FILE3D="Corridor.W3D"/>
            <STAGE STAGEID="Room0" FILE3D="Room0.W3D"/>
            <STAGE STAGEID="Room1" FILE3D="Room1.W3D"/>
            <STAGE STAGEID="Room2" FILE3D="Room2.W3D"/>
            <ACTOR ACTORID="Door" FILE3D="Door.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
            </ACTOR>
            <ACTOR ACTORID="Piece" FILE3D="Piece.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
                <METHODDECL NAME="rotate">
                    <PARAMDECL NAME="to" TYPE="string" />
                </METHODDECL>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Room0">
                <BEHAVIOUR BEHAVIOURID="listeningRoom0">
                    <EVENTSYS TYPE="onEnterScene"/>
                    <ACTIONBLOQ>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_begin"/>
                        <ACTIONSYS TYPE="assign">
                            <ASSIGN NAME="SEMAPHORE-doortocorridor" VALUE="DEACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="voiceBackToCorridor0">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'go'? AND $OUTDIALOG['room']!=?'
                        corridor'? AND SEMAPHORE-doortocorridor==?DEACTIVATED?">
                        <ACTIONACT ACTORINSTANCEID="DoorToCorridor0" METHOD="open"/>
                        <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                        <ACTIONSYS TYPE="assign">
                            <ASSIGN NAME="SEMAPHORE-doortocorridor" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Room0End">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action'] ==?'finish'?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="voiceAmstrad">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'show'? AND $OUTDIALOG['piece']!=?'
                        amstrad'? AND SEMAPHORE-piece==?DEACTIVATED?">
                        <ACTIONACT ACTORINSTANCEID="Amstrad" METHOD="rotate">
                            <PARAM NAME="to" VALUE="Left"/>
                        </ACTIONACT>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_piece"/>
                        <ACTIONSYS TYPE="assign">
                            <ASSIGN NAME="SEMAPHORE-piece" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="voiceMacintosh">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['action']==?'show'? AND $OUTDIALOG['piece']!=?'
                        macintosh'? AND SEMAPHORE-piece==?DEACTIVATED?">
                        <ACTIONACT ACTORINSTANCEID="Macintosh" METHOD="rotate">
                            <PARAM NAME="to" VALUE="Right" />
                        </ACTIONACT>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_piece"/>
                        <ACTIONSYS TYPE="assign">
                            <ASSIGN NAME="SEMAPHORE-piece" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor0" ACTORID="Door">
                    <PARAM NAME="Pos" VALUE="(0,-14,0)" />
                    <PARAM NAME="Content" VALUE="DoorToCorridor0"/>
                    <PARAM NAME="Image" VALUE="DoorToCorridor0.jpg"/>
                    <BEHAVIOUR BEHAVIOURID="clickBackToCorridor">
                        <EVENTGUI TYPE="mouseDown"/>
                        <ACTIONBLOQ COND="SEMAPHORE-doortocorridor == ?DEACTIVATED?">
                            <ACTIONACT ACTORINSTANCEID="DoorToCorridor0" METHOD="open"/>
                            <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                            <ACTIONSYS TYPE="assign">
                                <ASSIGN NAME="SEMAPHORE-doortocorridor0" VALUE="ACTIVATED"/>
```

```
                    </ACTIONSYS>
                </ACTIONBLOQ>
            </BEHAVIOUR>
        </ACTORINSTANCE>
        <ACTORINSTANCE ACTORINSTANCEID="Amstrad" ACTORID="Piece">
            <PARAM NAME="Pos" VALUE="(-5,-21,5)" />
            <PARAM NAME="Content" VALUE="Amstrad" />
            <PARAM NAME="Image" VALUE="Amstrad.jpg" />
            <BEHAVIOUR BEHAVIOURID="clickAmstrad">
                <EVENTGUI TYPE="mouseDown"/>
                <ACTIONBLOQ COND="SEMAPHORE-piece == ?DEACTIVATED?">
                    <ACTIONACT ACTORINSTANCEID="Amstrad" METHOD="rotate">
                        <PARAM NAME="to" VALUE="Left"/>
                    </ACTIONACT>
                    <ACTIONSYS TYPE="assign">
                        <ASSIGN NAME="SEMAPHORE-piece" VALUE="ACTIVATED"/>
                    </ACTIONSYS>
                </ACTIONBLOQ>
            </BEHAVIOUR>
        </ACTORINSTANCE>
        <ACTORINSTANCE ACTORINSTANCEID="Macintosh" ACTORID="Piece">
            <PARAM NAME="Pos" VALUE="(5,-21,5)" />
            <PARAM NAME="Content" VALUE="Macintosh" />
            <PARAM NAME="Image" VALUE="Macintosh_Plus.jpg" />
            <BEHAVIOUR BEHAVIOURID="clickMacintosh">
                <EVENTGUI TYPE="mouseDown"/>
                <ACTIONBLOQ COND="SEMAPHORE-piece == ?DEACTIVATED?">
                    <ACTIONACT ACTORINSTANCEID="Macintosh" METHOD="rotate">
                        <PARAM NAME="to" VALUE="Right" />
                    </ACTIONACT>
                    <ACTIONSYS TYPE="assign">
                        <ASSIGN NAME="SEMAPHORE-piece" VALUE="ACTIVATED"/>
                    </ACTIONSYS>
                </ACTIONBLOQ>
            </BEHAVIOUR>
        </ACTORINSTANCE>
    </SCENE>
    <SCENE STAGEID="Corridor">
        <BEHAVIOUR BEHAVIOURID="listeningCorridor">
            <EVENTSYS TYPE="onEnterScene"/>
            <ACTIONBLOQ>
                <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_begin"/>
                <ACTIONSYS TYPE="assign">
                    <ASSIGN NAME="SEMAPHORE-piece" VALUE="DEACTIVATED"/>
                    <ASSIGN NAME="SEMAPHORE-room" VALUE="DEACTIVATED"/>
                </ACTIONSYS>
            </ACTIONBLOQ>
        </BEHAVIOUR>
        <BEHAVIOUR BEHAVIOURID="voiceRoom0">
            <EVENTVUI TYPE="endedDialog"/>
            <ACTIONBLOQ COND="$OUTDIALOG['action']==?'go'? AND $OUTDIALOG['room']!=?zero?
                AND SEMAPHORE-room==?DEACTIVATED?">
                <ACTIONACT ACTORINSTANCEID="DoorRoom0" METHOD="open"/>
                <ACTIONSYS TYPE="goToScene" DESTINATION="Room0"/>
                <ACTIONSYS TYPE="assign">
                    <ASSIGN NAME="SEMAPHORE-room" VALUE="ACTIVATED"/>
                </ACTIONSYS>
            </ACTIONBLOQ>
        </BEHAVIOUR>
        <BEHAVIOUR BEHAVIOURID="voiceRoom1">
            <EVENTVUI TYPE="endedDialog"/>
            <ACTIONBLOQ COND="$OUTDIALOG['action']==?'go'? AND $OUTDIALOG['room']!=?one?
                AND SEMAPHORE-room==?DEACTIVATED?">
                <ACTIONACT ACTORINSTANCEID="DoorRoom1" METHOD="open"/>
                <ACTIONSYS TYPE="goToScene" DESTINATION="Room1"/>
                <ACTIONSYS TYPE="assign">
                    <ASSIGN NAME="SEMAPHORE-room" VALUE="ACTIVATED"/>
                </ACTIONSYS>
            </ACTIONBLOQ>
        </BEHAVIOUR>
        <BEHAVIOUR BEHAVIOURID="voiceRoom2">
            <EVENTVUI TYPE="endedDialog"/>
            <ACTIONBLOQ COND="$OUTDIALOG['action']==?'go'? AND $OUTDIALOG['room']!=?two?
                AND SEMAPHORE-room==?DEACTIVATED?">
                <ACTIONACT ACTORINSTANCEID="DoorRoom2" METHOD="open"/>
                <ACTIONSYS TYPE="goToScene" DESTINATION="Room2"/>
                <ACTIONSYS TYPE="assign">
                    <ASSIGN NAME="SEMAPHORE-room" VALUE="ACTIVATED"/>
                </ACTIONSYS>
            </ACTIONBLOQ>
        </BEHAVIOUR>
        <BEHAVIOUR BEHAVIOURID="CorridorEnd">
            <EVENTVUI TYPE="endedDialog"/>
            <ACTIONBLOQ COND="$OUTDIALOG['action'] ==?'finish'?">
                <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
            </ACTIONBLOQ>
        </BEHAVIOUR>
        <ACTORINSTANCE ACTORINSTANCEID="DoorRoom0" ACTORID="Door">
            <PARAM NAME="Pos" VALUE="(0,-14,0)" />
            <PARAM NAME="Content" VALUE="DoorRoom0"/>
            <PARAM NAME="Image" VALUE="DoorRoom0.jpg"/>
            <BEHAVIOUR BEHAVIOURID="clickRoom0">
                <EVENTGUI TYPE="mouseDown"/>
                <ACTIONBLOQ COND="SEMAPHORE-room == ?DEACTIVATED?">
```

```
                    <ACTIONSYS TYPE="goToScene" DESTINATION="Room0"/>
                    <ACTIONSYS TYPE="assign">
                        <ASSIGN NAME="SEMAPHORE-room" VALUE="ACTIVATED"/>
                    </ACTIONSYS>
                </ACTIONBLOQ>
            </BEHAVIOUR>
        </ACTORINSTANCE>
        <ACTORINSTANCE ACTORINSTANCEID="DoorRoom1" ACTORID="Door">
            <PARAM NAME="Pos" VALUE="(-10,0,0)" />
            <PARAM NAME="Content" VALUE="DoorRoom1"/>
            <PARAM NAME="Image" VALUE="DoorRoom1.jpg"/>
            <BEHAVIOUR BEHAVIOURID="clickRoom1">
                <EVENTGUI TYPE="mouseDown"/>
                <ACTIONBLOQ COND="SEMAPHORE-room == ?DEACTIVATED?">
                    <ACTIONSYS TYPE="goToScene" DESTINATION="Room1"/>
                    <ACTIONSYS TYPE="assign">
                        <ASSIGN NAME="SEMAPHORE-room" VALUE="ACTIVATED"/>
                    </ACTIONSYS>
                </ACTIONBLOQ>
            </BEHAVIOUR>
        </ACTORINSTANCE>
        <ACTORINSTANCE ACTORINSTANCEID="DoorRoom2" ACTORID="Door">
            <PARAM NAME="Pos" VALUE="(14,0,0)" />
            <PARAM NAME="Content" VALUE="DoorRoom2"/>
            <PARAM NAME="Image" VALUE="DoorRoom2.jpg"/>
            <BEHAVIOUR BEHAVIOURID="clickRoom2">
                <EVENTGUI TYPE="mouseDown"/>
                <ACTIONBLOQ COND="SEMAPHORE-room == ?DEACTIVATED?">
                    <ACTIONSYS TYPE="goToScene" DESTINATION="Room2"/>
                    <ACTIONSYS TYPE="assign">
                        <ASSIGN NAME="SEMAPHORE-room2" VALUE="ACTIVATED"/>
                    </ACTIONSYS>
                </ACTIONBLOQ>
            </BEHAVIOUR>
        </ACTORINSTANCE>
    </SCENE>
    <SCENE STAGEID="Room1">
        <BEHAVIOUR BEHAVIOURID="listeningRoom1">
            <EVENTSYS TYPE="onEnterScene"/>
            <ACTIONBLOQ>
                <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_room"/>
                <ACTIONSYS TYPE="assign">
                    <ASSIGN NAME="SEMAPHORE-doortocorridor1" VALUE="DEACTIVATED"/>
                </ACTIONSYS>
            </ACTIONBLOQ>
        </BEHAVIOUR>
        <BEHAVIOUR BEHAVIOURID="voiceBackToCorridor1">
            <EVENTVUI TYPE="endedDialog"/>
            <ACTIONBLOQ COND="$OUTDIALOG['action']==?'go'? AND $OUTDIALOG['room']!=?'
                corridor'? AND SEMAPHORE-doortocorridor==?DEACTIVATED?">
                <ACTIONACT ACTORINSTANCEID="DoorToCorridor1" METHOD="open"/>
                <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                <ACTIONSYS TYPE="assign">
                    <ASSIGN NAME="SEMAPHORE-doortocorridor" VALUE="ACTIVATED"/>
                </ACTIONSYS>
            </ACTIONBLOQ>
        </BEHAVIOUR>
        <BEHAVIOUR BEHAVIOURID="Room1End">
            <EVENTVUI TYPE="endedDialog"/>
            <ACTIONBLOQ COND="$OUTDIALOG['action'] ==?'finish'?">
                <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
            </ACTIONBLOQ>
        </BEHAVIOUR>
        <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor1" ACTORID="Door">
            <PARAM NAME="Pos" VALUE="(-10,0,0)" />
            <PARAM NAME="Content" VALUE="DoorToCorridor1"/>
            <PARAM NAME="Image" VALUE="DoorToCorridor1.jpg"/>
            <BEHAVIOUR BEHAVIOURID="clickBackToCorridor">
                <EVENTGUI TYPE="mouseDown"/>
                <ACTIONBLOQ COND="SEMAPHORE-doortocorridor == ?DEACTIVATED?">
                    <ACTIONACT ACTORINSTANCEID="DoorToCorridor1" METHOD="open"/>
                    <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor" >
                    </ACTIONSYS>
                    <ACTIONSYS TYPE="assign">
                        <ASSIGN NAME="SEMAPHORE-doortocorridor1" VALUE="ACTIVATED"/>
                    </ACTIONSYS>
                </ACTIONBLOQ>
            </BEHAVIOUR>
        </ACTORINSTANCE>
    </SCENE>
    <SCENE STAGEID="Room2">
        <BEHAVIOUR BEHAVIOURID="listeningRoom2">
            <EVENTSYS TYPE="onEnterScene"/>
            <ACTIONBLOQ>
                <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_room"/>
                <ACTIONSYS TYPE="assign">
                    <ASSIGN NAME="SEMAPHORE-doortocorridor2" VALUE="DEACTIVATED"/>
                </ACTIONSYS>
            </ACTIONBLOQ>
        </BEHAVIOUR>
        <BEHAVIOUR BEHAVIOURID="voiceBackToCorridor2">
            <EVENTVUI TYPE="endedDialog"/>
            <ACTIONBLOQ COND="$OUTDIALOG['action']==?'go'? AND $OUTDIALOG['room']!=?'
                corridor'? AND SEMAPHORE-doortocorridor==?DEACTIVATED?">
```

```
                    <ACTIONACT ACTORINSTANCEID="DoorToCorridor2" METHOD="open"/>
                    <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor"/>
                    <ACTIONSYS TYPE="assign">
                        <ASSIGN NAME="SEMAPHORE-doortocorridor2" VALUE="ACTIVATED"/>
                    </ACTIONSYS>
                </ACTIONBLOQ>
            </BEHAVIOUR>
            <BEHAVIOUR BEHAVIOURID="Room2End">
                <EVENTVUI TYPE="endedDialog"/>
                <ACTIONBLOQ COND="$OUTDIALOG['action'] ==?'finish'?">
                    <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                </ACTIONBLOQ>
            </BEHAVIOUR>
            <ACTORINSTANCE ACTORINSTANCEID="DoorToCorridor2" ACTORID="Door">
                <PARAM NAME="Pos" VALUE="(14,0,0)" />
                <PARAM NAME="Content" VALUE="DoorToCorridor2"/>
                <PARAM NAME="Image" VALUE="DoorToCorridor2.jpg"/>
                <BEHAVIOUR BEHAVIOURID="clickBackToCorridor">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ COND="SEMAPHORE-doortocorridor == ?DEACTIVATED?">
                        <ACTIONACT ACTORINSTANCEID="DoorToCorridor2" METHOD="open"/>
                        <ACTIONSYS TYPE="goToScene" DESTINATION="Corridor" >
                        </ACTIONSYS>
                        <ACTIONSYS TYPE="assign">
                            <ASSIGN NAME="SEMAPHORE-doortocorridor2" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
        </SCENE>
    </SEQUENCE>
    <CONTEXT>
        <VARIABLE NAME="SEMAPHORE-piece" TYPE="string" VALUE="DEACTIVATED"/>
        <VARIABLE NAME="SEMAPHORE-room" TYPE="string" VALUE="DEACTIVATED"/>
        <VARIABLE NAME="SEMAPHORE-doortocorridor" TYPE="string" VALUE="DEACTIVATED"/>
        <DIALOG DIALOG_ID="VXML_begin" SPEECHFILE="http://localhost:8080/museo3dequivalence-
            redundancy/voiceXML/museo3dequivalence-redundancy_begin.vxml">
            <OUTPUT_PARAM NAME="action"/>
            <OUTPUT_PARAM NAME="room"/>
                        <OUTPUT_PARAM NAME="piece"/>
        </DIALOG>
        <DIALOG DIALOG_ID="VXML_piece" SPEECHFILE="http://localhost:8080/museo3dequivalence-
            redundancy/voiceXML/museo3dequivalence-redundancy_piece.vxml">
            <INPUT_PARAM NAME="piece"/>
            <OUTPUT_PARAM NAME="action"/>
            <OUTPUT_PARAM NAME="room"/>
                        <OUTPUT_PARAM NAME="piece"/>
        </DIALOG>
        <DIALOG DIALOG_ID="VXML_room" SPEECHFILE="http://localhost:8080/museo3dequivalence-
            redundancy/voiceXML/museo3dequivalence-redundancy_room.vxml">
            <INPUT_PARAM NAME="room"/>
            <OUTPUT_PARAM NAME="action"/>
            <OUTPUT_PARAM NAME="room"/>
        </DIALOG>
        <DIALOG DIALOG_ID="VXML_corridor" SPEECHFILE="http://localhost:8080/
            museo3dequivalence-redundancy/voiceXML/museo3dequivalence-redundancy_corridor.
            vxml">
            <INPUT_PARAM NAME="room"/>
            <OUTPUT_PARAM NAME="action"/>
            <OUTPUT_PARAM NAME="room"/>
        </DIALOG>
        <DIALOG DIALOG_ID="VXML_end" SPEECHFILE="http://localhost:8080/museo3dequivalence-
            redundancy/voiceXML/museo3dequivalence-redundancy_end.vxml"/>
    </CONTEXT>
</XMMVR>
```

Listado A.36: Diálogo I de la cooperación entre modalidades por equivalencia/redundancia en el Museo 3D Multimodal (VXML_begin)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <block>
            <prompt bargein="false" xml:lang="en-us">Welcome to our museum. My name is Ana, I
                will help you in your visit. </prompt>
        </block>
        <field name="finish">
            <grammar src="equivalence-redundancy.BNF"/>
        </field>
        <field name="piece">
            <grammar src="equivalence-redundancy.BNF"/>
            <prompt xml:lang="en-us">Tell me which piece you want to interact with: Amstrad or
                Macintosh. You can also click on every piece. </prompt>
        </field>
        <subdialog name="checkpiece" src="museo3dequivalence-redundancy_piece.vxml">
            <param name="room" expr="room"/>
            <param name="piece" expr="piece"/>
            <param name="finish" expr="finish"/>
```

```
        </subdialog>
        <block>
            <prompt bargein="false" xml:lang="en-us"> Maybe you want to visit another room, just
                tell me which one or click on the door. </prompt>
        </block>
        <field name="room">
            <grammar src="equivalence-redundancy.BNF"/>
            <prompt xml:lang="en-us">Say finish to finish your visit. </prompt>
        </field>
        <subdialog name="checkroom" src="museo3dequivalence-redundancy_room.vxml">
            <param name="room" expr="room"/>
            <param name="finish" expr="finish"/>
        </subdialog>
    </form>
</vxml>
```

Listado A.37: Diálogo II de la cooperación entre modalidades por equivalencia/redundancia en el Museo 3D Multimodal (VXML_piece)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
            <var name="room"/>
        <var name="piece"/>
        <var name="action"/>
        <block>
            <if cond="finish=='finish'">
                <submit next="museo3dequivalence-redundancy_end.vxml"/>
            </if>
            <if cond="piece=='amstrad'">
                <submit next="http://localhost:8080/museo3dequivalence-redundancy/Dispatcher?
                    action=rotate&amp;piece=amstrad" method="post"/>
                <prompt xml:lang="en-us"> OK. This is the <value expr="piece"/> machine. It was
                    very popular in the eighties. </prompt>
            </if>
            <if cond="piece=='macintosh'">
                <submit next="http://localhost:8080/museo3dequivalence-redundancy/Dispatcher?
                    action=rotate&amp;piece=macintosh" method="post"/>
                <prompt xml:lang="en-us"> OK. This is the <value expr="piece"/> machine. It is
                    the enemy of PC. </prompt>
            </if>
                    <if cond="room=='corridor'">
                <submit next="http://localhost:8080/museo3dequivalence-redundancy/Dispatcher?
                    action=go&amp;room=corridor" method="post"/>
                <prompt/>
            </if>
        </block>
    </form>
</vxml>
```

Listado A.38: Diálogo III de la cooperación entre modalidades por equivalencia/redundancia en el Museo 3D Multimodal (VXML_room)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <var name="room"/>
        <var name="finish"/>
        <block>
            <if cond="finish=='finish'">
                <submit next="museo3dequivalence-redundancy_end.vxml"/>
            </if>
            <if cond="room=='zero'">
                <submit next="http://localhost:8080/museo3dequivalence-redundancy/Dispatcher?
                    action=go&amp;room=zero" method="post"/>
                <prompt/>
            </if>
            <if cond="room=='one'">
                <submit next="http://localhost:8080/museo3dequivalence-redundancy/Dispatcher?
                    action=go&amp;room=one" method="post"/>
                <prompt/>
            </if>
            <if cond="room=='two'">
                <submit next="http://localhost:8080/museo3dequivalence-redundancy/Dispatcher?
                    action=go&amp;room=two" method="post"/>
                <prompt/>
            </if>
            <if cond="room=='corridor'">
                <submit next="http://localhost:8080/museo3dequivalence-redundancy/Dispatcher?
                    action=go&amp;room=corridor" method="post"/>
                <prompt/>
            </if>
            <prompt>OK. This is the <value expr="room"/> room. </prompt>
```

```
        </block>
    </form>
</vxml>
```

Listado A.39: Diálogo IV de la cooperación entre modalidades por equivalencia/redundancia en el Museo 3D Multimodal (VXML_corridor)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <block>
            <prompt bargein="false" xml:lang="en-us"> You are in the corridor. Maybe you want to
                visit another room, just tell me which one or click on the door. </prompt>
        </block>
        <field name="room">
            <grammar src="equivalence-redundancy.BNF"/>
            <prompt xml:lang="en-us">Say finish to finish your visit. </prompt>
        </field>
        <subdialog name="checkroom" src="museo3dequivalence-redundancy_room.vxml">
            <param name="room" expr="room"/>
        </subdialog>
    </form>
</vxml>
```

Listado A.40: Diálogo V de la cooperación entre modalidades por equivalencia/redundancia en el Museo 3D Multimodal (VXML_end)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
        <form id="seccion">
            <block> Bye. Thank you for visiting us. Come back again whenever you want. </block>
        </form>
    </vxml>
```

Listado A.41: Gramática utilizada en la cooperación entre modalidades por equivalencia/redundancia en el Museo 3D Multimodal (equivalence-redundancy.BNF)

```
#ABNF 1.0 ISO8859-1;

root $action = [ [[$go] [$room]] | [[$show] [$piece]] | [$finish] ] ;

$go = go to the | take me to the | to the | let's go to the ;
$room = zero | one | two | corridor ;
$show = show me the | show | let's see | let's see the ;
$piece = amstrad | macintosh ;
$finish = ( finish | end | stop | bye ) {"finish"} ;

#ABNF 1.0;
language es;
mode voice;
root $piece;
public $piece =
root $room;
public $room =

show (amstrad | macintosh) | go (zero | one | two | corridor) | finish
```

A.3.3. Transferencia / Complementariedad

Listado A.42: Implementación de la cooperación entre modalidades por transferencia/complementariedad en el Museo 3D Multimodal

```
<!DOCTYPE XMMVR SYSTEM "xmmvr.dtd">
    <XMMVR>
        <CAST>
            <STAGE STAGEID="Room0" FILE3D="Room0.W3D"/>
            <ACTOR ACTORID="Shelf" FILE3D="Shelf.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
            </ACTOR>
            <ACTOR ACTORID="Piece" FILE3D="Piece.W3D">
                <PARAMDECL NAME="Pos" TYPE="vector" />
                <PARAMDECL NAME="Content" TYPE="string" />
                <PARAMDECL NAME="Image" TYPE="string" />
                <METHODDECL NAME="move">
                    <PARAMDECL NAME="to" TYPE="string" />
                </METHODDECL>
            </ACTOR>
        </CAST>
        <SEQUENCE>
            <SCENE STAGEID="Room0">
                <BEHAVIOUR BEHAVIOURID="listeningRoom0">
                    <EVENTSYS TYPE="onEnterScene"/>
                    <ACTIONBLOQ>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_Room0"/>
                        <ACTIONSYS TYPE="assign">
                            <ASSIGN NAME="SEMAPHORE-where" VALUE="DEACTIVATED"/>
                            <ASSIGN NAME="SEMAPHORE-what" VALUE="DEACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="WhereWhat">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['what']!=?that? AND $OUTDIALOG['where']!=?'there
                        '?">
                        <ACTIONSYS TYPE="assign" >
                            <ASSIGN NAME="SEMAPHORE-where" VALUE="ACTIVATED"/>
                            <ASSIGN NAME="SEMAPHORE-what" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_WhatWhere">
                            <PARAM NAME="Where" VALUE="$OUTDIALOG['where']"/>
                            <PARAM NAME="What" VALUE="$OUTDIALOG['what']"/>
                        </ACTIONVUI>
                        <ACTIONACT ACTORINSTANCEID="$OUTDIALOG['what'] " METHOD="move">
                            <PARAM NAME="to" VALUE="$OUTDIALOG['where']"/>
                        </ACTIONACT>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="What">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['what']==?that? AND $OUTDIALOG['where']!=?'there
                        '?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_WhatWhere">
                            <PARAM NAME="Where" VALUE="$OUTDIALOG['where']"/>
                            <PARAM NAME="What" VALUE="NULL"/>
                        </ACTIONVUI>
                        <ACTIONSYS TYPE="assign" >
                            <ASSIGN NAME="SEMAPHORE-where" VALUE="ACTIVATED"/>
                            <ASSIGN NAME="SEMAPHORE-what" VALUE="DEACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Where">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG['what']!=?that? AND $OUTDIALOG['where']==?'there
                        '?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_WhatWhere">
                            <PARAM NAME="Where" VALUE="NULL"/>
                            <PARAM NAME="What" VALUE="$OUTDIALOG['what']"/>
                        </ACTIONVUI>
                        <ACTIONSYS TYPE="assign" >
                            <ASSIGN NAME="SEMAPHORE-where" VALUE="DEACTIVATED"/>
                            <ASSIGN NAME="SEMAPHORE-what" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="Room0End">
                    <EVENTVUI TYPE="endedDialog"/>
                    <ACTIONBLOQ COND="$OUTDIALOG[action]==?finish?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_end"/>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <ACTORINSTANCE ACTORINSTANCEID="Amstrad" ACTORID="Piece">
                    <PARAM NAME="Pos" VALUE="(-5,-35,5)"/>
                    <PARAM NAME="Content" VALUE="Amstrad"/>
                    <PARAM NAME="Image" VALUE="Amstrad.jpg"/>
                    <BEHAVIOUR BEHAVIOURID="clickAmstrad1">
```

```
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ COND="$SEMAPHORE-where==?ACTIVATED?">
                        <ACTIONACT ACTORINSTANCEID="amstrad" METHOD="move">
                            <PARAM NAME="to" VALUE="$OUTDIALOG['where']"/>
                        </ACTIONACT>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_WhatWhere">
                            <PARAM NAME="Where" VALUE="$OUTDIALOG['where']"/>
                            <PARAM NAME="What" VALUE="amstrad"/>
                        </ACTIONVUI>
                        <ACTIONSYS TYPE="assign" >
                            <ASSIGN NAME="SEMAPHORE-where" VALUE="ACTIVATED"/>
                            <ASSIGN NAME="SEMAPHORE-what" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="clickAmstrad2">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ COND="$SEMAPHORE-where==?DEACTIVATED?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_WhatWhere">
                            <PARAM NAME="Where" VALUE="$OUTDIALOG['where']"/>
                            <PARAM NAME="What" VALUE="amstrad"/>
                        </ACTIONVUI>
                        <ACTIONSYS TYPE="assign" >
                            <ASSIGN NAME="SEMAPHORE-what" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="Macintosh" ACTORID="Piece">
                <PARAM NAME="Pos" VALUE="(5,-35,5)" />
                <PARAM NAME="Content" VALUE="Macintosh" />
                <PARAM NAME="Image" VALUE="Macintosh_Plus.jpg" />
                <BEHAVIOUR BEHAVIOURID="clickMacintosh1">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ COND="$SEMAPHORE-where==?ACTIVATED?">
                        <ACTIONACT ACTORINSTANCEID="Macintosh" METHOD="move">
                            <PARAM NAME="to" VALUE="$OUTDIALOG['where']"/>
                        </ACTIONACT>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_WhatWhere">
                            <PARAM NAME="Where" VALUE="$OUTDIALOG['where']"/>
                            <PARAM NAME="What" VALUE="Macintosh"/>
                        </ACTIONVUI>
                        <ACTIONSYS TYPE="assign" >
                            <ASSIGN NAME="SEMAPHORE-where" VALUE="ACTIVATED"/>
                            <ASSIGN NAME="SEMAPHORE-what" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="clickMacintosh2">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ COND="$SEMAPHORE-where==?DEACTIVATED?">
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_WhatWhere">
                            <PARAM NAME="Where" VALUE="$OUTDIALOG['where']"/>
                            <PARAM NAME="What" VALUE="Macintosh"/>
                        </ACTIONVUI>
                        <ACTIONSYS TYPE="assign" >
                            <ASSIGN NAME="SEMAPHORE-what" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
            <ACTORINSTANCE ACTORINSTANCEID="ShelfUp" ACTORID="Shelf">
                <PARAM NAME="Pos" VALUE="(0,-14,10)" />
                <PARAM NAME="Content" VALUE="ShelfUp"/>
                <PARAM NAME="Image" VALUE="ShelfUp.jpg"/>
                <BEHAVIOUR BEHAVIOURID="clickUp1">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ COND="$SEMAPHORE-what==?ACTIVATED?">
                        <ACTIONACT ACTORINSTANCEID="$OUTDIALOG['what']" METHOD="move">
                            <PARAM NAME="to" VALUE="up"/>
                        </ACTIONACT>
                        <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_WhatWhere">
                            <PARAM NAME="Where" VALUE="up"/>
                            <PARAM NAME="What" VALUE="$OUTDIALOG['what']"/>
                        </ACTIONVUI>
                        <ACTIONSYS TYPE="assign" >
                            <ASSIGN NAME="SEMAPHORE-where" VALUE="ACTIVATED"/>
                            <ASSIGN NAME="SEMAPHORE-what" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
                <BEHAVIOUR BEHAVIOURID="clickUp2">
                    <EVENTGUI TYPE="mouseDown"/>
                    <ACTIONBLOQ COND="$SEMAPHORE-where==?DEACTIVATED?">
                                                    <ACTIONACT ACTORINSTANCEID="$INPUTDIALOG
                                                        ['what']" METHOD="move">
                                                        <PARAM NAME="to" VALUE="up"/>
                                                    </ACTIONACT>
                        <ACTIONSYS TYPE="assign" >
                            <ASSIGN NAME="SEMAPHORE-where" VALUE="ACTIVATED"/>
                        </ACTIONSYS>
                    </ACTIONBLOQ>
                </BEHAVIOUR>
            </ACTORINSTANCE>
```

```
                    <ACTORINSTANCE ACTORINSTANCEID="ShelfMiddle" ACTORID="Shelf">
                        <PARAM NAME="Pos" VALUE="(0,-14,5)" />
                        <PARAM NAME="Content" VALUE="ShelfMiddle"/>
                        <PARAM NAME="Image" VALUE="ShelfMiddle.jpg"/>
                        <BEHAVIOUR BEHAVIOURID="clickMiddle1">
                            <EVENTGUI TYPE="mouseDown"/>
                            <ACTIONBLOQ COND="$SEMAPHORE-what==?ACTIVATED?">
                                <ACTIONACT ACTORINSTANCEID="$OUTDIALOG['what']" METHOD="move">
                                    <PARAM NAME="to" VALUE="Middle"/>
                                </ACTIONACT>
                                <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_WhatWhere">
                                    <PARAM NAME="Where" VALUE="Middle"/>
                                    <PARAM NAME="What" VALUE="$OUTDIALOG['what']"/>
                                </ACTIONVUI>
                                <ACTIONSYS TYPE="assign" >
                                    <ASSIGN NAME="SEMAPHORE-where" VALUE="ACTIVATED"/>
                                    <ASSIGN NAME="SEMAPHORE-what" VALUE="ACTIVATED"/>
                                </ACTIONSYS>
                            </ACTIONBLOQ>
                        </BEHAVIOUR>
                        <BEHAVIOUR BEHAVIOURID="clickMiddle2">
                            <EVENTGUI TYPE="mouseDown"/>
                            <ACTIONBLOQ COND="$SEMAPHORE-where==?DEACTIVATED?">
                                                    <ACTIONACT ACTORINSTANCEID="$INPUTDIALOG
                                                        ['what']" METHOD="move">
                                                        <PARAM NAME="to" VALUE="Middle"/>
                                                    </ACTIONACT>
                                <ACTIONSYS TYPE="assign" >
                                    <ASSIGN NAME="SEMAPHORE-where" VALUE="ACTIVATED"/>
                                </ACTIONSYS>
                            </ACTIONBLOQ>
                        </BEHAVIOUR>
                    </ACTORINSTANCE>
                    <ACTORINSTANCE ACTORINSTANCEID="ShelfDown" ACTORID="Shelf">
                        <PARAM NAME="Pos" VALUE="(0,-14,0)" />
                        <PARAM NAME="Content" VALUE="ShelfDown"/>
                        <PARAM NAME="Image" VALUE="ShelfDown.jpg"/>
                        <BEHAVIOUR BEHAVIOURID="clickDown1">
                            <EVENTGUI TYPE="mouseDown"/>
                            <ACTIONBLOQ COND="$SEMAPHORE-what==?ACTIVATED?">
                                <ACTIONACT ACTORINSTANCEID="$OUTDIALOG['what']" METHOD="move">
                                    <PARAM NAME="to" VALUE="Down"/>
                                </ACTIONACT>
                                <ACTIONVUI TYPE="launchDialog" DIALOGID="VXML_WhatWhere">
                                    <PARAM NAME="Where" VALUE="Down"/>
                                    <PARAM NAME="What" VALUE="$OUTDIALOG['what']"/>
                                </ACTIONVUI>
                                <ACTIONSYS TYPE="assign" >
                                    <ASSIGN NAME="SEMAPHORE-where" VALUE="ACTIVATED"/>
                                    <ASSIGN NAME="SEMAPHORE-what" VALUE="ACTIVATED"/>
                                </ACTIONSYS>
                            </ACTIONBLOQ>
                        </BEHAVIOUR>
                        <BEHAVIOUR BEHAVIOURID="clickDown2">
                            <EVENTGUI TYPE="mouseDown"/>
                            <ACTIONBLOQ COND="$SEMAPHORE-where==?DEACTIVATED?">
                                                    <ACTIONACT ACTORINSTANCEID="$INPUTDIALOG
                                                        ['what']" METHOD="move">
                                                        <PARAM NAME="to" VALUE="Down"/>
                                                    </ACTIONACT>
                                <ACTIONSYS TYPE="assign" >
                                    <ASSIGN NAME="SEMAPHORE-where" VALUE="ACTIVATED"/>
                                </ACTIONSYS>
                            </ACTIONBLOQ>
                        </BEHAVIOUR>
                    </ACTORINSTANCE>
                </SCENE>
            </SEQUENCE>
            <CONTEXT>
                <VARIABLE NAME="Where" TYPE="string"/>
                <VARIABLE NAME="What" TYPE="string"/>
                <VARIABLE NAME="SEMAPHORE-where" TYPE="string" VALUE="DEACTIVATED"/>
                <VARIABLE NAME="SEMAPHORE-what" TYPE="string" VALUE="DEACTIVATED"/>
                <DIALOG DIALOG_ID="VXML_Room0" SPEECHFILE="http://localhost:8080/museo3dtransfer-
                    complementarity/voiceXML/museo3dtransfer-complementarity_begin.vxml">
                    <OUTPUT_PARAM NAME="what"/>
                    <OUTPUT_PARAM NAME="where"/>
                </DIALOG>
                <DIALOG DIALOG_ID="VXML_WhatWhere" SPEECHFILE="http://localhost:8080/museo3dtransfer-
                    complementarity/voiceXML/museo3dtransfer-complementarity_whatwhere.vxml">
                    <INPUT_PARAM NAME="what"/>
                    <INPUT_PARAM NAME="where"/>
                    <OUTPUT_PARAM NAME="what"/>
                    <OUTPUT_PARAM NAME="where"/>
                </DIALOG>
                <DIALOG DIALOG_ID="VXML_end" SPEECHFILE="http://localhost:8080/museo3dtransfer-
                    complementarity/voiceXML/museo3dtransfer-complementarity_end.vxml"/>
            </CONTEXT>
        </XMMVR>
```

LISTADO A.43: Diálogo I de la cooperación entre modalidades por transferencia/complementariedad en el Museo 3D Multimodal (VXML_Room0)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <block>
            <prompt bargein="false" xml:lang="en-us"> </prompt>
        </block>
        <field name="finish">
            <grammar src="transfer-complementarity.BNF"/>
            <prompt xml:lang="en-us"> </prompt>
        </field>
        <field name="what">
            <grammar src="transfer-complementarity.BNF"/>
            <prompt xml:lang="en-us"> </prompt>
        </field>
        <field name="where">
            <grammar src="transfer-complementarity.BNF"/>
            <prompt xml:lang="en-us"> </prompt>
        </field>
        <subdialog name="checkwhatwhere" src="museo3dtransfer-complementarity_whatwhere.vxml">
            <param name="what" expr="what"/>
            <param name="where" expr="where"/>
                    <param name="finish" expr="finish"/>
        </subdialog>
    </form>
</vxml>
```

LISTADO A.44: Diálogo II de la cooperación entre modalidades por transferencia/complementariedad en el Museo 3D Multimodal (VXML_WhatWhere)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/
    voicexml20/vxml.xsd" xml:lang="es">
    <form id="seccion">
        <var name="finish" />
        <var name="what" expr="what" />
        <var name="where" expr="where" />
        <block>
            <if cond="finish=='finish'">
                <submit next="museo3dinterfaceagent_end.vxml" />
            </if>
            <if cond="what=='amstrad'">
                <if cond="where=='up'">
                    <submit next="http://localhost:8080/museo3dtransfer-complementarity/
                        Dispatcher?what=amstrad&amp;where=up" method="post" />
                    <prompt />
                </if>
                <if cond="where=='middle'">
                    <submit next="http://localhost:8080/museo3dtransfer-complementarity/
                        Dispatcher?what=amstrad&amp;where=middle" method="post" />
                    <prompt />
                </if>
                <if cond="where=='down'">
                    <submit next="http://localhost:8080/museo3dtransfer-complementarity/
                        Dispatcher?what=amstrad&amp;where=down" method="post" />
                    <prompt />
                </if>
            </if>
            <if cond="what=='macintosh'">
                <if cond="where=='up'">
                    <submit next="http://localhost:8080/museo3dtransfer-complementarity/
                        Dispatcher?what=macintosh&amp;where=up" method="post" />
                    <prompt />
                </if>
                <if cond="where=='middle'">
                    <submit next="http://localhost:8080/museo3dtransfer-complementarity/
                        Dispatcher?what=macintosh&amp;where=middle" method="post" />
                    <prompt />
                </if>
                <if cond="where=='down'">
                    <submit next="http://localhost:8080/museo3dtransfer-complementarity/
                        Dispatcher?what=macintosh&amp;where=down" method="post" />
                    <prompt />
                </if>
            </if>
            <prompt xml:lang="en-us">OK. Piece <value expr="what" /> is on the shelf at <value
                expr="where" />. Where do you want to move it now?</prompt>
        </block>
    </form>
</vxml>
```

LISTADO A.45: Diálogo III de la cooperación entre modalidades por transferencia/complementariedad en el Museo 3D Multimodal (VXML_end)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd" xml:lang="es">
        <form id="seccion">
            <block>
            </block>
        </form>
    </vxml>
```

LISTADO A.46: Gramática utilizada en la cooperación entre modalidades por transferencia/complementariedad en el Museo 3D Multimodal (transfer-complementarity.BNF)

```
#ABNF 1.0 ISO8859-1;

root $action = [ [put] [$what] [$where]] | [$finish] ] ;

$what = amstrad | macintosh | that ;
$where = middle | up | down | there ;
$finish = ( finish | end | stop | bye ) {"finish"} ;


#ABNF 1.0;
language es;
mode voice;
root $what;
public $what =
root $where;
public $where =

put (amstrad | macintosh | that)  (middle | up | down | there)
```

B

Cuestionario USE y resultados

En este apéndice se presenta el cuestionario y los resultados de la encuesta descrita en 6.1.1.

B.1 Cuestionario USE

Listado B.1: Cuestionario USE

```
Instrucciones:

Basándote en tu experiencia con productos similares, categoriza tu nivel de acuerdo con los
    siguientes enunciados acerca de cómo te sientes en general cuando utilizas el "Museo Virtual
    3D Mutimodal". Simplemente marca con un círculo o una X el nivel de acuerdo que aplique (
    donde 1 significa Totalmente en desacuerdo, 4 significa Ni de acuerdo, ni en desacuerdo, y 7
    significa Totalmente de acuerdo; y NA significa No aplica), como en el ejemplo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

**********Enunciados acerca del producto usado para evaluar.**********

Gustaría tanto a usuarios regulares como ocasionales.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo          NA

Estoy satisfecho con él.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo          NA

Es simple de utilizar.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo          NA

Es maravilloso.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo          NA

Es útil.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo          NA

No noto inconsistencias cuando lo uso.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo          NA

Es divertido usarlo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo          NA

Hace todo lo que podría esperar que haga.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo          NA

Requiere el menor número de pasos posible para realizar lo que quiero con él.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo          NA

Es amigable al usuario.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo          NA

Me daría más control sobre las actividades de mi vida.
```

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Siento que necesito tenerlo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Puedo utilizarlo con éxito en cualquier momento.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Es flexible.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Aprendí rápido a utilizarlo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Haría las cosas que quiero conseguir más fáciles de realizar.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Puedo recuperarme fácil y rápidamente de errores.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Me ayudaría a ser más efectivo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Se utiliza sin esfuerzo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Puedo utilizarlo sin instrucciones escritas.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Satisface mis necesidades.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Funciona del modo que quiero que funcione.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Me ahorraría tiempo cuando lo utilizo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Fácilmente recuerdo cómo utilizarlo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Rápidamente me hice diestro en él.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Es agradable utilizarlo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Es fácil de usar.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Lo recomendaría a un amigo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Me ayudaría a ser más productivo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

Es fácil aprender a usarlo.

Totalmente en desacuerdo 1---2---3---4---5---6---7 Totalmente de acuerdo NA

DATOS PERSONALES

NOMBRE:

APELLIDOS:

EDAD:

SEXO:

NIVEL DE ESTUDIOS:

1. Interacción gráfica

2. Interacción vocal

```
3. Interacción gráfica y vocal
¿Cuál le ha aportado la mejor experiencia?
La interacción vocal era más sencilla que la interacción gráfica
La interacción gráfica era más sencilla que la vocal
La combinación de ambas era la mejor
Áreas en las que ven que podría ser más interesante utilizar esta tecnología:
juegos
museos
ayuda a discapacitados
otro
OBSERVACIONES:
```

B.2 Resultados

B.2.1. Utilidad

En la tabla B.1 cada fila representa la valoración del 1 al 7 asignada por cada uno de los usuarios a cada una de las afirmaciones relativas a la Utilidad. La última fila representa la media de cada cuestión.

Tabla B.1: Resultados del cuestionario USE (Utilidad)

UT1	UT2	UT3	UT4	UT5	UT6	UT7	UT8
3	4	6	6	4	5	3	5
4	7	7	5	7	7	4	7
3	6	6	5	6	5	3	5
5	6	7	7	7	6	5	7
5	7	7	5	7	5	4	7
4	6	6	5	6	5	4	6
7	7	7	7	7	7	7	7
6	7	7	6	6	6	5	7
7	7	7	7	7	6	6	7
7	7	7	7	7	7	6	7
6	6	6	6	6	6	3	6
7	7	6	6	7	5	4	7
7	6	5	7	5	7	6	5
7	7	6	7	5	7	7	5
5	5	5	5	6	5	6	5
4	6	7	7	7	7	5	7
6	6	6	7	7	6	5	7
5	5	5	5	6	5	6	5
6	6	6	5	5	6	5	5
6	7	6	4	7	5	6	7
4.63	6.25	6.63	5.75	6.25	5.90	5.00	6.38

UT1. Es útil.
UT2. Hace todo lo que podría esperar que haga.
UT3. Me daría más control sobre las actividades de mi vida.
UT4. Haría las cosas que quiero conseguir más fáciles de realizar.
UT5. Me ayudaría a ser más efectivo.
UT6. Satisface mis necesidades.
UT7. Me ahorraría tiempo cuando lo utilizo.
UT8. Me ayudaría a ser más productivo.

B.2.2. Facilidad de uso

En la tabla B.2 cada fila representa la valoración del 1 al 7 asignada por cada uno de los usuarios a cada una de las afirmaciones relativas a la Facilidad. La última fila representa la media de cada cuestión.

Tabla B.2: Resultados del cuestionario USE (Facilidad de uso)

FU1	FU2	FU3	FU4	FU5	FU6	FU7	FU8	FU9	FU10	FU11
4	3	7	3	5	6	4	1	4	4	5
7	7	5	6	7	5	2	4	7	0	7
4	5	6	5	5	4	3	3	5	5	6
5	7	6	6	6	6	4	4	6	6	7
5	3	6	5	6	6	3	2	5	4	7
5	6	5	5	5	6	3	3	5	5	6
7	4	6	6	7	7	7	6	6	4	7
7	5	6	4	5	6	6	5	6	6	7
7	5	4	6	7	6	3	4	6	7	7
6	6	6	6	6	6	6	6	6	7	7
6	6	6	6	5	6	N/A	1	6	6	6
7	5	6	6	7	6	1	2	6	7	6
7	7	7	7	7	6	5	5	7	7	5
7	7	7	7	7	6	5	6	7	7	6
7	1	7	6	5	5	7	5	5	5	6
5	2	5	5	7	7	3	2	6	4	7
6	4	6	6	5	7	5	3	6	6	6
7	1	7	6	5	5	7	5	5	5	6
5	N/A	4	N/A	N/A	3	3	3	N/A	N/A	6
5	7	4	7	7	5	4	6	5	4	7
5.95	5.00	5.88	5.00	5.75	5.75	4.00	3.50	5.74	5.21	6.35

FU1. Gustaría tanto a usuarios regulares como ocasionales.
FU2. Es simple de utilizar.
FU3. No noto inconsistencias cuando lo uso.
FU4. Requiere el menor número de pasos posible para realizar lo que quiero con él.
FU5. Es amigable al usuario.
FU6. Puedo utilizarlo con éxito en cualquier momento.
FU7. Es flexible.
FU8. Puedo recuperarme fácil y rápidamente de errores.
FU9. Se utiliza sin esfuerzo.
FU10. Puedo utilizarlo sin instrucciones escritas.
FU11. Es fácil de usar.

B.2.3. Facilidad de aprendizaje

En la tabla B.3 cada fila representa la valoración del 1 al 7 asignada por cada uno de los usuarios a cada una de las afirmaciones relativas a la Facilidad de aprendizaje. La última fila representa la media de cada cuestión.

Tabla B.3: Resultados del cuestionario USE (Facilidad de aprendizaje)

FA1	FA2	FA3	FA4
6	5	6	3
7	6	7	0
5	4	5	3
3	6	6	5
6	6	7	4
5	4	6	4
7	6	7	6
6	7	7	6
4	5	6	5
7	6	6	6
6	6	6	4
7	7	7	6
6	6	5	6
7	7	7	6
7	7	7	7
7	5	7	5
4	6	6	4
7	7	7	7
7	7	6	4
4	4	5	4
5.63	5.85	6.30	3.88

FA1. Aprendí rápido a utilizarlo.
FA2. Fácilmente recuerdo cómo utilizarlo.
FA3. Rápidamente me hice diestro en él.
FA4. Es fácil aprender a usarlo.

B.2.4. Satisfacción

En la tabla B.4 cada fila representa la valoración del 1 al 7 asignada por cada uno de los usuarios a cada una de las afirmaciones relativas a la Satisfacción. La última fila representa la media de cada cuestión.

Tabla B.4: Resultados del cuestionario USE (Satisfacción)

SA1	SA2	SA3	SA4	SA5	SA6	SA7
0	5	3	5	5	4	2
4	0	4	7	2	7	7
5	3	3	6	3	4	4
6	6	4	6	5	6	6
5	4	4	7	7	5	5
5	6	4	6	6	4	5
7	7	7	7	7	6	6
6	4	6	6	6	5	5
7	6	6	7	7	6	6
6	6	6	7	7	6	6
6	6	4	6	5	3	6
7	6	3	5	6	6	6
7	6	7	7	4	7	7
7	7	7	7	6	7	7
6	5	5	5	5	5	5
4	5	3	5	7	4	7
5	4	4	6	6	5	5
6	5	5	5	5	5	5
6	N/A	4	6	6	4	4
5	4	6	5	7	5	5
4.75	4.38	4.38	6.25	5.60	5.20	5.45

SA1. Estoy satisfecho con él.
SA2. Es maravilloso.
SA3. Es divertido usarlo.
SA4. Siento que necesito tenerlo.
SA5. Funciona del modo que quiero que funcione.
SA6. Es agradable utilizarlo.
SA7. Lo recomendaría a un amigo.

C
Publicaciones

En este apéndice se presentan las publicaciones realizadas durante el periodo de desarrollo de la tesis.

Tabla C.1: Participación en congresos y publicaciones

Noviembre 2006:	Primeros Pasos Hacia la Especificación Formal de Interacción Multimodal en Escenarios 3D. INTERACCION 2006, Puertollano 2006. (I.S.B.N.: 84-690-1613-X, páginas 509-512) (Olmedo-Rodríguez, Escudero-Mancebo, González-Escribano, González-Ferreras & Cardeñoso-Payo, 2006)
Septiembre 2007:	XMMVR: Especificación y Arquitectura para el desarrollo de aplicaciones de Interacción Multimodal en Escenarios 3D. INTERACCION 2007, Zaragoza 2007. (I.S.B.N.: 978-84-9732-596-7, páginas 221-229) (Olmedo-Rodríguez, Escudero-Mancebo, González-Escribano, González-Ferreras & Cardeñoso-Payo, 2007b)
Septiembre 2007:	Propuesta de Especificación y Arquitectura para aplicaciones de Interacción Multimodal en Escenarios 3D. CEIG 2007, Zaragoza 2007. (I.S.B.N.: 978-84-9732-595-0, páginas 287-290) (Olmedo-Rodríguez, Escudero, González, González & Cardeñoso, 2007a)
Febrero 2008:	A framework for the development of applications allowing Multimodal Interaction with Virtual Reality worlds. WSCG 2008, Plzen 2008, República Checa. (I.S.B.N.: 978-80-86943-16-9, páginas 79-85) (Olmedo-Rodríguez, Cardeñoso-Payo & Escudero-Mancebo, 2008a)
Junio 2008:	3D en las Rich Internet Applications: comparativa de opciones tecnológicas. JOREVIR 2008, Albacete 2008. (I.S.B.N.: 978-84-691-4028-4, páginas 145-152) (Olmedo-Rodríguez, Escudero-Mancebo & Cardeñoso-Payo, 2008b)
Julio 2008:	A simple and effective system for Computer-Assisted Semen Analysis. MEDSIP 2008, Santa Margarita Ligure 2008. (I.S.B.N.: 978-0-86341-934-8) (Pascual-Gaspar, Olmedo-Rodríguez, Exposito, Exposito & Finat, 2008)
Noviembre 2008:	Arquitectura multimodal controlada por voz: revisión de metáforas de interacción. V Jornadas en Tecnologías del Habla, Bilbao 2008. (I.S.B.N.: 978-84-9860-169-5, páginas 175-178) (Escudero-Mancebo, Olmedo-Rodríguez & Cardeñoso-Payo, 2008)
Marzo 2009:	Conceptual and practical framework for the integration of multimodal interaction in 3D worlds. New Trends on Human-Computer Interaction, Springer 2008. (I.S.B.N.: 978-1-84882-351-8, páginas 87-95) (Olmedo-Rodríguez, Escudero-Mancebo, Cardeñoso-Payo, González-Ferreras & González-Escribano, 2009c)
Julio 2009:	Evaluation proposal of a framework for the integration of multimodal interaction in 3D worlds. HCI International 2009, San Diego, CA, USA 2009. (I.S.B.N.: 978-3-642-02576-1, páginas 84-92) (Olmedo-Rodríguez, Escudero-Mancebo & Cardeñoso-Payo, 2009a)
Septiembre 2009:	Interacción Multimodal con Espacios Virtuales. INTERACCIÓN 2009, Barcelona 2009. (I.S.B.N.-13: 978-84-692-5005-1) (Olmedo-Rodríguez, Escudero-Mancebo & Cardeñoso-Payo, 2009b)
Septiembre 2010:	Interacción Multimodal con Espacios Virtuales, un caso de estudio: Museo Virtual 3D MultiModal. INTERACCIÓN 2010, Valencia 2010. (I.S.B.N.: 978-84-92812-52-3, páginas 201-204) (Olmedo-Rodríguez, Sanz Prieto, Escudero-Mancebo & Cardeñoso-Payo, 2010)
Octubre 2012:	De la realidad aumentada a la realidad mixta: opciones tecnológicas. INTERACCIÓN 2012, Elche 2012. (I.S.B.N.: 978-84-695-4477-8, páginas 293-296) (Olmedo-Rodríguez & Augusto, 2012)
Septiembre 2013:	Towards the commodification of augmented reality: tools and platforms. New trends in interaction, virtual reality and modeling, Springer 2013. (I.S.B.N.: 978-1-4471-5444-0, páginas 63-72) (Olmedo-Rodríguez & Augusto, 2013)
Noviembre 2013:	Virtuality Continuum's State of the Art. VARE2013 "Virtual and Augmented Reality in Education", Puerto de la Cruz 2013. (I.S.B.N.: pendiente) (Olmedo-Rodríguez, 2013)

D
Glosario

Acrónimos

3DML: 3D Markup Language.
ABNF: Augmented BNF.
AIML: Artificial Intelligence Markup Language.
AJAX: Asynchronous JavaScript and XML.
API: Application Programming Interface.
AVI: Audio Video Interleaved.
Behavior3D: Behavior 3D.
BNF: Backus-Naur form.
CCXML: Call Control XML.
DOM: Document Object Model.
DTD: Document Type Definition.
DTMF: Dual Tone Multi-Frequency.
EAI: External Authoring Interface.
ECA-SIMM: Entornos de Computación Avanzada y Sistemas de Interacción MultiModal.
ECMAScript: European Computer Manufacturers Association (ECMA) Script.
EMMA: Extensible MultiModal Annotation markup language.
EmotionML: Emotion Markup Language.
HMD: Head-Mounted Display.
HTML: HyperText Markup Language.
JPEG: Joint Photographic Experts Group.
JSGF: Java Speech Grammar Format.
MIML: Multimodal Interaction Markup Language.
MMS: Multimedia Messaging Service.
MP3: MPEG-1 or MPEG-2 Audio Layer III (audio).
MPG: MPEG-1 or MPEG-2 Audio Layer III (video).
MPEG: Moving Picture Experts Group.

MP4: MPEG-4.
MPML: Multimodal Presentation Markup Language.
MPML-VR: Multimodal Presentation Markup Language for Virtual Reality.
MVC: Modelo Vista Controlador.
MXML: Macromedia eXtensible Markup Language.
OAA: Open Agent Architecture.
PNG: Portable Network Graphics.
POMPD: Partially Observable Markov Decission Processes.
RIA: Rich Internet Applications.
RT: RealText Streaming Text File.
SAI: Scene Authoring Interface.
SALT: Speech Application Language Tags.
SCXML: State Chart XML o State Machine Notation for Control Abstraction.
SDK: Software Development Kit.
SMIL: Synchronized Multimedia Integration Language.
SRGS: Speech Recognition Grammar Specification.
SSML: Speech Synthesis Markup Language.
SUB: MicroDVD Subtitle file format.
SVG: Scalable Vector Graphics.
tATN: Augmented Transition Network .
TCP/IP: Transmission Control Protocol (TCP) / Internet Protocol (IP).
UML: Unified Modeling Language.
VHML: Virtual Human Markup Language.
VML: Vector Markup Language.
VRML: Virtual Reality Markup Language.
VXML: Voice eXtensible Markup Language.
W3C MMI WG: MultiModal Interaction Working Group.
W3C SYMM: Synchronized Multimedia Working Group.
W3C VBWG: Voice Browser Working Group.
W3C: World Wide Web Consortium.
W3D: 3D Scene Export File.
WAV: Waveform Audio File Format.
WPF: Windows Presentation Foundation.
X+V: XHTML+Voice Profile.
X3D: Extensible 3D.
XAML: eXtensible Application Markup Language.
XHTML: Extensible HyperText Markup Language.
XML: eXtensible Markup Language.
XMMVR: eXtensible markup language for MultiModal interaction with Virtual Reality worlds.

Términos

Accesibilidad: Grado en el que todas las personas pueden utilizar un objeto, visitar un lugar o acceder a un servicio, independientemente de sus capacidades técnicas, cognitivas o físicas. En informática, la accesibilidad incluye ayudas como las tipografías de alto contraste o gran tamaño, magnificadores de pantalla, lectores y revisores de pantalla, programas de reconocimiento de voz, teclados adaptados, y otros dispositivos apuntadores y de entrada de información. La accesibilidad aplicada al contenido de Internet se denomina accesibilidad web. En la Web, el W3C ha desarrollado directrices o pautas específicas para permitir y asegurar este tipo de accesiblidad. El grupo de trabajo dentro del W3C encargado de promoverla es el WAI (Web Accessibility Initiative), elaborando para ello unas Pautas de Accesibilidad al contenido Web 1.0, WCAG.

Action spaces: Aproximaciones metafóricas para estructurar los interfaces de usuario tridimensionales.

Chatterbot: Programa de ordenador diseñado para simular el diálogo pero sólo en apariencia, comparando las entradas mediante reglas y escogiendo la salida de una lista de frases predefinidas.

Constituent(s): Componente(s) de la arquitectura propuesta por el grupo MMI de la W3C.

Front-end/Back-end: En diseño de software el front-end es la parte del software que interactúa con el o los usuarios y el back-end es la parte que procesa la entrada desde el front-end. La separación del sistema en "front ends" y "back ends" es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas. La idea general es que el front-end sea el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas, y procesarlas de una manera conforme a la especificación que el back-end pueda usar. La conexión del front-end y el back-end es un tipo de interfaz.

Head Mounted Displays: Dispositivo de visualización similar a un casco, que permite reproducir imágenes creadas por ordenador sobre un "display" muy cercano a los ojos.

Lenguaje de marcas básico: Lenguaje de marcas sencillo, con una sintaxis propia.

Lenguaje de marcas dedicado: Lenguaje de marcas dedicado a especificar una tarea concreta.

Lenguaje de marcas híbrido: Lenguaje de marcas basado en otros lenguajes de marcas básicos.

Línea de tiempo: Secuencia de escenas a través de la cual los elementos que intervienen en la animación o escena 3D sufren un cambio en alguna de sus propiedades.

Metáfora: Entidades que se diseñan deliberadamente para ser fácilmente manipulables y así conseguir un control más intuitivo de una cierta tarea.

Modalidad: Proceso para analizar y producir porciones de información. Existen modalidades usadas por los humanos (diálogo, gestos, ...) y modalidades usadas por los ordenadores (gráficos, síntesis de diálogo, ...).

Realidad virtual: Sistema tecnológico, basado en el empleo de ordenadores y otros dispositivos, cuyo fin es producir una apariencia de realidad que permita al usuario tener la sensación de estar presente en ella.

Usabilidad: Facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto. En los entornos web, Jakob Nielsen definió la usabilidad en el 2003 como un atributo de calidad que mide lo fáciles de usar que son las interfaces web. En diseño web (o desarrollo web) hace referencia a la visualización del usuario navegante por un lado (front-end), y del administrador del sitio con sus respectivos sistemas por el otro (back-end).

Cast Library (Adobe Director): La lista de miembros del Cast usados en una película. Todas las películas tienen al menos un Internal Cast pero las películas pueden tener varios Internal Cast y External Cast.

Internal cast (Adobe Director): Es el elemento del cast por defecto en Adobe Director. Esto significa que se almacena internamente con la película. Son fáciles de manejar porque no crean ficheros separados.

External cast (Adobe Director): Se crean y se usan del mismo modo que los Internal casts, sin embargo Adobe Director los guarda como ficheros separados (usualmente con una extensión ".cst").

List (Adobe Director): Es un tipo de variable que puede contener varias unidades de datos a la vez. Cada unidad (conocidas como elementos o entradas) pueden manipularse sin afectar al resto.

Stage (Adobe Director): Es la pantalla principal de Adobe Director donde tiene lugar toda la acción.

Cast member (Adobe Director): Es un miembro del reparto o cast almacenado en un Cast Library. Un elemento tal como un mapa de bits, un texto, un sonido, un gráfico (una forma, un dibujo vectorial, etc.), una pieza de video digital o un script.

3D cast member (Adobe Director): Un Cast member que define un elemento 3D.

Score (Adobe Director): Un diagrama que muestra qué members aparecen en el Stage en determinados momentos.

Sprite (Adobe Director): La descripción de qué member se está mostrando, dónde está en el Score, dónde aparece en el Stage y otras propiedades. Permite controlar un canal de sprite (Sprite channel que representa un canal de sprite en la ventana Score) y al miembro del reparto que se encuentre en él.

Xtra (Adobe Director): Un módulo software que añade capacidades enlazadas a Adobe Director. Algunos vienen con Adobe Director y otros pueden adquirirse por separado para luego integrarlos en nuestra instalación y utilizarlos en nuestros proyectos.

Property (Adobe Director): Son las palabras que determinan los atributos o parámetros de un elemento.

Global (Adobe Director): Variable compartida por más de un handler.

Handler (Adobe Director): Son fragmentos de código en los scripts que se ocupan de definir qué es lo que hace la película ca da vez que se produce un suceso (event).

Key (Adobe Director): Objeto para controlar el teclado del sistema desde la película.

Mouse (Adobe Director): Objeto para controlar la actividad del ratón desde la película.

Movie (Adobe Director): Representa la película que se está reproduciendo por medio de la aplicación Player.

Player (Adobe Director): Representa el motor de reproducción de la película que ejecuta la película en el sistema operativo.

Sound (Adobe Director): Controla la reproducción del sonido en cualquiera de los ocho canales de audio disponibles (objetos Sound channel).

System (Adobe Director): Proporciona acceso al sistema operativo en el que se reproduce la películaObjeto para controlar el teclado del sistema desde la película.

Window (Adobe Director): Representa una ventana en la que se está reproduciendo una película.

Índice alfabético

Referencias

Adobe (2010a). Adobe Director. http://www.adobe.com/products/director/. 59, 110

Adobe (2010b). Adobe Flash. http://www.adobe.com/products/flash/. 15, 59

Adobe (2010c). Adobe Shockwave. http://www.adobe.com/products/shockwaveplayer/. 60

Adobe (2010d). Adobe SVG. http://www.adobe.com/es/svg/viewer/install/mainframed.html. 15

Adobe (2010e). Comparing Adobe Director to Adobe Flash. http://www.adobe.com/products/director/compare/. xi, 36

Adobe (2010f). FLEX. http://www.adobe.com/es/products/flex.html. 33

Adobe (2010g). MXML. https://learn.adobe.com/wiki/display/Flex/MXML. 33

Adobe (2010h). PostNetText. http://help.adobe.com/en_US/Director/11.5/UsingScripting/WSc3ff6d0ea77859461172e0811d64c1a1b3-7c63.html. 112

AITOOLS (2011). AITOOLS. http://aitools.org/. 11

ALICE (2011a). AIML. http://www.alicebot.org/aiml.html. 11

ALICE (2011b). Alice superbot. http://alicebot.org/superbot.html. 11

Amyot, D. & Simoes, R. (2006). Combining VoiceXML with CCXML. http://www.site.uottawa.ca/~damyot/pub/CCNC07-Amyot.pdf. 12

Apache Foundation (2012). Apache Tomcat. http://tomcat.apache.org/. 109, 113

Araki, M. & Tachibana, K. (2006). Multimodal dialog description language for rapid system development. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, SigDIAL '06, 109–116. 14

Aubourg, Julian et al. (2010). XMLHttpRequest. http://www.w3.org/TR/XMLHttpRequest/. 34

Autodesk (2010). Autodesk 3D Studio MAX. http://www.autodesk.es/. 59, 111

Avatare, A., Frécon, E., Hagsand, O., Jää-Aro, K.M., Simsarian, K. & Ståhl, O. (1997). Dive - the distributed interactive virtual environment. Tech. rep., Swedish Institute of Computer Science, Box 1263, 164 28 Kista. 7, 8

Away3D (2010). Away3D. http://away3d.com/. 36

Axelsson, Jonny et al. (2010). XHTML+Voice Profile. http://www.w3.org/TR/xhtml+voice/. 11

Baggia, P. & Scott, M. (2010). Voice Browser Call Control: CCXML Version 1.0. http://www.w3.org/TR/ccxml/. 12, 15

Barnett, J., Dahl, D., Kliche, I., Tumuluri, R., Yudkowsky, M., Bodell, M., Porter, B., Raggett, D., Raman, T. & Wahbe, A. (2008). Multimodal architecture and interfaces. http://www.w3.org/TR/mmi-arch/. 4, 15, 24, 106

Barnett, Jim et al. (2010). State Chart XML (SCXML): State Machine Notation for Control Abstraction. http://www.w3.org/TR/scxml/. 15, 25

Beringer, N., Kartal, U., Louka, K., Schiel, F., Tuerk, U. & Trk, U. (2002). Promise - a procedure for multimodal interactive system evaluation. In *Proceedings of the LREC Workshop on Multimodal Resources and Multimodal Systems Evaluation, Las Palmas*, 77–80. 37, 88

Bernsen, N.O., Dybkjær, L. & Dybkjær, L. (2000). A methodology for evaluating spoken language dialogue systems and their components. In *In Luperfoy, S. (Ed.): Automatic Spoken Dialogue Systems*, 183–188, MIT Press. 31

Billinghurst, M. (1998). Put that where? voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.*, **32**, 60–63. 7

Bolt, R.A. (1980). "put-that-there": Voice and gesture at the graphics interface. In *SIGGRAPH'80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, 262–270, ACM, New York, NY, USA. 91

Booch, G., Maksimchuk, R., Engle, M., Young, B., Conallen, J. & Houston, K. (2007). *Object-oriented analysis and design with applications, third edition*. Addison-Wesley Professional, 3rd edn. 44

Bouchet, J. (2004). Icare software components for rapidly developing multimodal interfaces. 251–258, ACM Press. 8, 22

Bouchet, J. & Nigay, L. (2004). Icare: a component-based approach for the design and development of multimodal interfaces. In E. Dykstra-Erickson & M. Tscheligi, eds., *CHI Extended Abstracts*, 1325–1328, ACM. 8

Bouzá, G.B. (1997). *El guión multimedia*. Anaya Multimedia. 43

Boyer, Linda et al. (2000). Voice eXtensible Markup Language (VoiceXML) Version 1.0. `http://www.w3.org/TR/2000/NOTE-voicexml-20000505/`. 10

Brooke, J. (1996). Sus: A quick and dirty usability scale. 40

Brutzman, D. (2010). The Virtual Reality Modeling Language and Java. `http://web.nps.navy.mil/~brutzman/vrml/vrmljava.pdf`. 34

Brutzman, D. & Daly, L. (2007). *X3D: Extensible 3D Graphics for Web*. Elsevier. 3

Budd, T.A. (2001). *An Introduction to Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Inc., 3rd edn. 44

Bufill, J.J.P. (2003). *Programación en 3D con Java3D*. Ra-Ma. 34

Bulterman, D.C.A. & Hardman, L. (2005). Structured multimedia authoring. *ACM Trans. Multimedia Comput. Commun. Appl.*, **1**, 89–109. 44

Burkhardt, F. & Schröder, M. (2010). Emotion Markup Language (EmotionML) . `http://www.w3.org/TR/emotionml/`. 14

Burnett, D.C., Walker, M.R. & Hunt, A. (2010). SSML. `http://www.w3.org/TR/speech-systhesis/`. 11

Carrión, Z.C. (2008). Desarrollo de sistemas de diálogo oral adaptativos y portables. reconocimiento de emociones, adaptación al idioma y evaluación de campo. `http://inf.ucv.ro/~ntand/courses/MMIA222/carrion.pdf`. 29

Cerekovic, A., Pejsa, T. & Pandzic, I.S. (2009). Realactor: Character animation and multimodal behavior realization system. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, IVA '09, 486–487, Springer-Verlag, Berlin, Heidelberg. 8, 23

CERN (2013). Twenty years of a free, open web. `http://info.cern.ch/`. 17

Charte Ojeda, F. (2008). Programa desde el navegador: Adobe Flash vs Microsoft Silverlight. *PCACTUAL*, **202**, 147–150. xi, 35

Corradini, A., Mehta, M., ole Bernsen, N. & Charfuelan, M. (2005). Animating an interactive conversational character for an educational game system. In *In Proceedings of the 10th Int. Conf. on Intelligent User Interfaces*, 183–190. 8

Cortona (2010). CORTONA 3D. `http://www.cortona3d.com/`. 108

Crocker, D. (2010). ABNF. `http://www.ietf.org/rfc/rfc2234.txt`. 11

Dachselt, R. (2000). Action spaces - a metaphorical concept to support navigation and interaction in 3d interfaces. In *In Workshop Usability Centred Design and Evaluation of Virtual 3D Environments*. xi, 17, 18

Dachselt, R. & Rukzio, E. (2003). BEHAVIOR3D: An XML-Based Framework for 3D Graphics Behavior. In *In Web3D '03: Proceeding of the eighth international conference on 3D Web technology, ACM*, 101, Press. 13

Dachselt, R., Hinz, M. & Meiner, K. (2002). CONTIGRA: An XML-Based Architecture for Component-Oriented 3D Applications. In *Web3D'02: Proceeding of the seventh international conference on 3D Web technology, ACM*, 155–163. 13, 17

Dahl, D. (2004). *Practical Spoken Dialog Systems*. Springer. 3

Diehl, S. (1997). VRML++: A Language for Object-Oriented Virtual-Reality Models. In *TOOLS '97: Proceedings of the Technology of Object-Oriented Languages and Systems-Tools - 24*, 141, IEEE Computer Society, Washington, DC, USA. 13

Dybkjær, L., Bernsen, N.O. & Minker, W. (2004). Evaluation and usability of multimodal spoken language dialogue systems. *Speech Communication, 43(1-2)*, 33–54. 37

ECMA International (2010). ECMAScript. `http://www.ecma-international.org/publications/standards/Ecma-262.htm`. 55

Erickson, T.D. (1995). Human-computer interaction. chap. Working with Interface Metaphors, 147–151, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 17

Escudero-Mancebo, D., Olmedo-Rodríguez, H. & Cardeñoso-Payo, V. (2008). Arquitectura multimodal controlada por voz: revisión de metáforas de interacción. *V Jornadas en Tecnologías del Habla*, 175–178. 5, 180

Feuerstack, S. (2013). MINT Framework. `http://www.multi-access.de/`. 8, 24

Feuerstack, S. & Pizzolato, E. (2011). Building multimodal interfaces out of executable, model-based interactors and mappings. In J. Jacko, ed., *Proceedings of the HCI International 2011; 14th International Conference on Human-Computer Interaction; Human-Computer Interaction, Part I*, LNCS 6761, pp. 221228, Springer, Heidelberg (2011), Hilton Orlando Bonnet Creek, Orlando, Florida, USA. 8

Feuerstack, S., Oliveira, A. & Araujo, R. (2011). Model-based design of interactions that can bridge realities the augmented drag-and-drop. In *Proceedings of the 13th Symposium on Virtual and Augmented Reality (SVR 2011)*, Uberlândia, Minas Gerais, Brazil, iSSN 2177-676. 8

Figueroa, P., Green, M. & Hoover, H.J. (2001). 3DML. `http://www.cs.ualberta.ca/~pfiguero/3dml/`. 12

Foley, J.D., van Dam, A., Feiner, S.K. & Hughes, J.F. (1990). *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc. 103

Freitas, S.d., Rebolledo-Mendez, G., Liarokapis, F., Magoulas, G. & Poulovassilis, A. (2009). Developing an Evaluation Methodology for Immersive Learning Experiences in a Virtual World. In *Proceedings of the 2009 Conference in Games and Virtual Worlds for Serious Applications*, VS-GAMES '09, 43–50, IEEE Computer Society, Washington, DC, USA. 39

Gabbard, J.L., Hix, D. & Swan, J.E. (1999). User-centered design and evaluation of virtual environments. *IEEE Computer Graphics and Applications*, **19**, 51–59. 39, 115, 126

Garshol, L.M. (2010). BNF and EBNF: What are they and how do they work? `http://www.garshol.priv.no/download/text/bnf.html`. 11

Gibbon, D., Moore, R. & Winski, R. (1997). *Handbook of Standards and Resources for Spoken Language Systems*. Mouton de Gruyter. 31

González-Ferreras, C. (2009). *Estrategias para el acceso a contenidos Web mediante habla*. Tesis doctoral en informática, Departamento de Informática, Universidad de Valladolid. 30

GRAH (2010). *DialSDK: Desarrollo de Sistemas de Diálogo*. Documento de estudio, UPV-EHU. 27

Griol Barres, D. (2008). *Desarrollo y evaluación de diferentes metodologías para la gestión automática del diálogo*. Tesis doctoral en informática, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia. 30

Gustafson, J., Bell, L., Beskow, J., Boye, J., Carlson, R., Edlund, J., Granström, B., House, D. & Wirn, M. (2000). AdApt - a multimodal conversational dialogue system in an apartment domain. In *Proceedings of ICSLP 2000*, 134–137. 3, 8

Hague, J.V. & Jackson, C. (2003). *Flash 3D. Animation, Interactivity, and Games*. Macromedia Inc. 33

Halbert, D. & O'Brien, P. (1987). Using types and inheritance in object-oriented programming. *Software, IEEE*, **4**, 71–79. 44

Hare, A.B.I., M.; Doubleday & Ryan, M. (1995). Intelligent presentation of information retrieved from heterogeneous multimedia databases. In *IMMI 95*. 22

Harel, D. (1987). Statecharts: A visual formalism for complex systems. `http://www.wisdom.weizmann.ac.il/~harel/papers/Statecharts.pdf`. 15

Harrison, W. & Ossher, H. (1993). Subject-oriented programming: a critique of pure objects. *SIGPLAN Not.*, **28**, 411–428. 44

Hartman, J. & Wernecke, J. (1996). *The VRML 2.0 handbook: building moving worlds on the web*. Addison Wesley Longman Publishing Co., Inc. 3, 12, 34

Hassenzahl, M. (2008). The interplay of beauty, goodness, and usability in interactive products. *Hum.-Comput. Interact.*, **19**, 319–349. 40

Hix, D., Swan, J.E., II, Gabbard, J.L., McGee, M., Durbin, J. & King, T. (1999). User-Centered Design and Evaluation of a Real-Time Battlefield Visualization Virtual Environment. In *VR'99: Proceedings of the IEEE Virtual Reality*, 96, IEEE Computer Society, Washington, DC, USA. 39

Holzapfel, H. (2011). JSGF Editor. `http://sourceforge.net/projects/jsgfeditor/`. 12

Hone, K.S. & Graham, R. (2000). Towards a tool for the subjective assessment of speech system interfaces (sassi). *Nat. Lang. Eng.*, **6**, 287–303. 39

Hudson, A. (2010). SAI. `http://www.xj3d.org/tutorials/general_sai.html`. 12, 34

Hunt, A. (2010). JSGF. `http://www.w3.org/TR/jsgf/`. 12

Hunt, A. & McGlashan, S. (2010). SRGS. `http://www.w3.org/TR/speech-grammar/`. 11, 33

IBM (2012). WebSphere Voice Toolkit. `http://www.ibm.com/developerworks/websphere/downloads/voicetoolkit.html`. 10

IMDB (1966). Star Trek, la conquista del espacio. `http://www.imdb.com/title/tt0060028/`. 75

IMDB (2005). Charlie y la fábrica de chocolate. `http://www.imdb.com/title/tt0367594/`. 18

Ishizuka Laboratory (2010). MPML-VR 2.0. `http://www.miv.t.u-tokyo.ac.jp/mpmlvr/`. 14

Johnston, M. (2010). W3C, EMMA: Extensible MultiModal Annotation markup language. `http://www.w3.org/TR/emma/`. 14

Kaiser, E., Olwal, A., McGee, D., Benko, H., Corradini, A., Li, X., Cohen, P. & Feiner, S. (2003). Mutual disambiguation of 3D multimodal interaction in augmented and virtual reality. In *Proceedings of the 5th international conference on Multimodal interfaces*, ICMI '03, 12–19, ACM, New York, NY, USA. 8

Kalawsky, R.S. (1999). VRUSE-a computerised diagnostic tool: for usability evaluation of virtual/synthetic environment systems. *Applied Ergonomics*, **30**, 11 – 25. 39

Kaplan, R.M. (1975). On process models for sentence comprehension. *Explorations in congnition, San Francisco*. 7

Keeney, R. & Raiffa, H. (1976). *Decisions with multiple objectives: Preferences and value tradeoffs*. John Wiley and Sons. 38

Khnel, C. (2012). *Quantifying Quality Aspects of Multimodal Interactive Systems*. Springer Publishing Company, Incorporated. 40

Kipp, M., Heloir, A., Schröder, M. & Gebhard, P. (2010). Realizing multimodal behavior: closing the gap between behavior planning and embodied agent presentation. In *Proceedings of the 10th international conference on Intelligent virtual agents*, IVA'10, 57–63, Springer-Verlag, Berlin, Heidelberg. 8, 23

Kirakowski, J. & Corbett, M. (1993). SUMI: the Software Usability Measurement Inventory. *British Journal of Educational Technology*, **24**, 210–212. 40

Kuppevelt, J.V., Dybkjær, L., Dybkjær, L., Bernsen, N.O., Wilks, Y., Webb, N., Setzer, A. & Catizone, R. (2005). Advances in natural multimodal dialogue systems. 29

Larson, J.A. (2010). W3C SIF. `http://www.w3.org/TR/voice-intro/`. 10

Latoschik, M.E. (2002). Designing Transition Networks for Multimodal VR-Interactions Using a Markup Language. *Multimodal Interfaces, IEEE International Conference on*, **0**, 411. 14

López Cózar, R. (2011). Sistemas de diálogo hablado y multimodal. `http://www.ugr.es/~rlopezc/sistemas_dialogo.htm`. IX, 27, 28

Lund, A.M. (2001). Measuring usability with the use questionnaire. `http://www.stcsig.org/usability/newsletter/0110_measuring_with_use.html`. XI, 40, 116, 117

MacGillivray, C. & Head, A. (2005). *3D for the web*. Elsevier. 34, 43

Marriott, A., Beard, S., Stallo, J. & Huynh, Q. (2001). Vhml - directing a talking head. In *Proceedings of the 6th International Computer Science Conference on Active Media Technology*, AMT '01, 90–100, Springer-Verlag, London, UK, UK. 13

Martin, J.C. (1998). Tycoon: theoretical and software tools for multimodal interfaces. 19

Mauro Dos Santos Anjo, E.P. & Feuerstack, S. (2012). A real-time system to recognize static hand gestures of brazilian sign language (libras) alphabet using kinect. In *Proceedings of IHC 2012, the 6th Latin American Conference on Human-Computer Interaction*, Cuiabá/Mato Grosso, Brazil. 8

Mcglashan, S. & Axling, T. (1996a). A speech interface to virtual environments. 7, 19

Mcglashan, S. & Axling, T. (1996b). Talking to agents in virtual worlds. In *Proc of 3 rd UK VR-SIG Conference*. 7

McGlashan, Scott et al. (2004). Voice Extensible Markup Language (VoiceXML) Version 2.0. `http://www.w3.org/TR/2004/REC-voicexml20-20040316/`. 10

McTear, M.F. (2002). Spoken dialogue technology: enabling the conversational user interface. *ACM Comput. Surv*, **34**, 90–169. 30

Megginson, D. (2010). SAX. `http://www.saxproject.org/`. 109

Michel, T. (2010a). SMIL. `http://www.w3.org/AudioVideo/`. 15

Michel, T. (2010b). "Synchronized Multimedia" Working Group (SYMM WG) Charter. `http://www.w3.org/AudioVideo/2004/symm-wg-charter20060601.html`. 15

Microsoft (2003). Speech Application Language Tags (SALT). `http://msdn.microsoft.com/en-us/library/ms994629.aspx`. 10

Microsoft (2010). WPF 3D. `http://blogs.msdn.com/b/wpf3d/archive/2006/12.aspx`. 33

Microsoft (2011a). TELLME. `http://www.microsoft.com/speech/`. 11

Microsoft (2011b). Tellme Studio. `https://studio.tellme.com/downloads/voicestudio/`. 10

Microsoft (2012). Máquina Virtual Java de Microsoft. `http://support.microsoft.com/gp/lifean12`. 108

Möller, S., Smeele, P., Boland, H. & Krebber, J. (2007). Evaluating spoken dialogue systems according to de-facto standards: A case study. *Computer Speech and Language*, 26–53. 39

Moon, D.A. (1986). Object-oriented programming with flavors. *SIGPLAN Not.*, **21**, 1–8. 44

Nexaweb (2010). Nexaweb. `http://www.nexaweb.com/home/us/index.html`. 35

Nielsen, J. (1995). Usability inspection methods. In *Conference companion on Human factors in computing systems*, CHI '95, 377–378, ACM, New York, NY, USA. 39

Noda, T. & Helwig, S. (2005). Rich internet applications technical comparison and case studies of ajax, flash, and java based ria executive summary web applications: New trends. `http://www.uwebc.org/opinionpapers/docs/RIA.pdf`. xi, 35

Nugues, P. (2000). Verbal interactions in virtual worlds. `http://www.cs.utep.edu/novick/nlchi/papers/nugues/chi2000.html`. 8

Object Management Group (2012). UML 2.3. `http://www.omg.org/spec/UML/2.3/`. 15

Okazaki, N., Saeyor, S., Dohi, H. & Ishizuka, M. (2005). An extension of the multimodal presentation markup language (MPML) to a three-dimensional VRML space. *Syst. Comput. Japan*, **36**, 69–80. 14

Olmedo-Rodríguez, H. (2010a). Aplicación "Robot room" basada en CORTONA. `http://www.youtube.com/watch?v=rIW-UBmYxMQ`. 114

Olmedo-Rodríguez, H. (2010b). Metáfora de interacción vocal de "agente interfaz". `http://www.youtube.com/watch?v=o6GiO3o4_ro`. 114

Olmedo-Rodríguez, H. (2010c). Museo Virtual 3D VIDEO. `http://www.youtube.com/watch?v=httHBnGOYcU`. 60

Olmedo-Rodríguez, H. (2013). Virtuality Continuum's State of the art. *VARE2013 "Virtual and Augmented Reality in Education"*, –. 5, 127, 180

Olmedo-Rodríguez, H. & Augusto, J. (2012). De la realidad aumentada a la realidad mixta: opciones tecnológicas. *INTERACCIÓN*, 293–296. 5, 127, 180

Olmedo-Rodríguez, H. & Augusto, J. (2013). Towards the commodification of augmented reality: Tools and Platforms. 5, 127, 180

Olmedo-Rodríguez, H., Escudero-Mancebo, D., González-Escribano, A., González-Ferreras, C. & Cardeñoso-Payo, V. (2006). Primeros Pasos Hacia la Especificación Formal de Interacción Multimodal en Escenarios 3D. *INTERACCION*, 509–512. 5, 180

Olmedo-Rodríguez, H., Escudero, D., González, A., González, C. & Cardeñoso, V. (2007a). Propuesta de Especificación y Arquitectura para aplicaciones de Interacción Multimodal en Escenarios 3D. *CEIG*, 287–290. 5, 180

Olmedo-Rodríguez, H., Escudero-Mancebo, D., González-Escribano, A., González-Ferreras, C. & Cardeñoso-Payo, V. (2007b). XMMVR: Especificación y Arquitectura para el desarrollo de aplicaciones de Interacción Multimodal en Escenarios 3D. *INTERACCIÓN*, 221–229. 5, 180

Olmedo-Rodríguez, H., Cardeñoso-Payo, V. & Escudero-Mancebo, D. (2008a). A framework for the development of applications allowing multimodal interaction with virtual reality worlds. *WSCG*, 79–85. 5, 180

Olmedo-Rodríguez, H., Escudero-Mancebo, D. & Cardeñoso-Payo, V. (2008b). 3D en las Rich Internet Applications: comparativa de opciones tecnológicas. *JOREVIR*, 145–152. 5, 180

Olmedo-Rodríguez, H., Escudero-Mancebo, D. & Cardeñoso-Payo, V. (2009a). Evaluation Proposal of a Framework for the Integration of Multimodal Interaction in 3D Worlds. In *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, 84–92, Springer-Verlag, Berlin, Heidelberg. 5, 180

Olmedo-Rodríguez, H., Escudero-Mancebo, D. & Cardeñoso-Payo, V. (2009b). Interacción multimodal con espacios virtuales. *INTERACCIÓN*. 5, 180

Olmedo-Rodríguez, H., Escudero-Mancebo, D., Cardeñoso-Payo, V., González-Ferreras, C. & González-Escribano, A. (2009c). Conceptual and practical framework for the integration of multimodal interaction in 3D worlds. 87–95. 5, 180

Olmedo-Rodríguez, H., Sanz Prieto, A., Escudero-Mancebo, D. & Cardeñoso-Payo, V. (2010). Interacción Multimodal con Espacios Virtuales, un caso de estudio: Museo Virtual 3D MultiModal. *INTERACCIÓN*, 201–204. 5, 180

OpenLaszlo (2010). OpenLaszlo. `http://www.openlaszlo.org/`. 35

Opera (2010). Opera. `http://www.opera.com/`. 19

Oviatt, S. & Cohen, P. (2000). Multimodal interfaces that process what comes naturally. *Communications of the ACM*, **43**, 45–53. 7, 19

Paillard, S. (2011). Chatterbox. `http://www.sylvainpaillard.com/wordpress/?p=216`. 11, 27

Papervision3D (2010). Papervision3D. `http://blog.papervision3d.org/`. 36

Parisi, T. (2010). Ajax 3D. `http://www.ajax3d.org/`. 34

Parra Márquez, J.C., García Alvarado, R. & Santelices Malfanti, I. (2001). *Introducción práctica a la realidad virtual*. Eds. Universidad del Bío-Bío. 2

Pascual-Gaspar, J.M., Olmedo-Rodríguez, H., Exposito, A.I., Exposito, A. & Finat, J. (2008). A simple and effective system for computer-assisted semen analysis. In *Advances in Medical, Signal and Information Processing, 2008. MEDSIP 2008. 4th IET International Conference on*, 1–4. 180

Phelps, A.M. (2010). Introduction to the External Authoring Interface, EAI. `http://andysgi.rit.edu/andyworld10/gallery/archives/vrml/media/eaiclass.doc`. 12, 108, 109

Pihkala, K., Honkala, M. & Vuorimaa, P. (2002). A Browser Framework for Hybrid XML Documents. In *IMSA*, 164–169. 10

Powers, M. (2005). Flatland Rover. `http://www.flatland.com/`. 12

Quint, V. & Carcone, L. (2010). W3C Amaya. `http://www.w3.org/Amaya/`. 15

Reenskaug, T.M.H. (1979). MVC. `http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html`. 24, 25

Salas, J. (2010). WPF 3D Framework. `http://www.codeplex.com/wpf3dFramework`. 36

Salisbury, M.W., Hendrickson, J.H., Lammers, T.L., Fu, C. & Moody, S.A. (1990). Talk and draw: Bundling speech and graphics. *Computer*, **23**, 59–65. 7

Sanz Prieto, A. (2009). *Museo virtual web 3D de la historia de la informática*. Proyecto fin de carrera, Universidad de Valladolid. 60

Sanz Prieto, A. (2010). Museo Virtual 3D. `http://www.fi.uva.es/museo3d`. 60

Seneff, S., Hurley, E., Lau, R., Pao, C., Schmid, P. & Zue, V. (1998). Galaxy-II: a reference architecture for conversational system development. In *ICSLP'98: Proceedings of the 5th International Conference on Spoken Language Processing, Sydney, Australia*, 931–934. 29

Serrano, M., Nigay, L., Lawson, J.Y.L., Ramsay, A., Murray-Smith, R. & Denef, S. (2008). The openinterface framework: a tool for multimodal interaction. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, 3501–3506, ACM, New York, NY, USA. 8, 23

Siroux, J., Guyomard, M., Multon, F. & Rémondeau, C. (1997). Multimodal references in GEORAL Tactile. In *Referring Phenomena in a Multimedia Context and their Computational Treatment*, ReferringPhenomena '97, 39–43, Association for Computational Linguistics, Stroudsburg, PA, USA. 21

Spatialview (2010). Spatialview. `http://www.spatialview.com/`. 36

Sutcliffe, A. & Gault, B. (2004). Heuristic evaluation of virtual reality applications. *Interacting with Computers*, **16**, 831 – 849. 39

Szabó, K. (1995). Metaphors and the user interface. `http://www.katalinszabo.com/metaphor.htm`. 16

Szyperski, C. (2002). *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Inc., 2nd edn. 44

The Moving Picture Experts Group (2013). MPEG4. `http://mpeg.chiariglione.org/standards/mpeg-4`. 13

Traum, D. & Rickel, J. (2001). Embodied agents for multi-party dialogue in immersive virtual worlds. 766–773. 8

Turunen, M. (2004). *Jaspis - A spoken dialogue architecture and its applications*. Academic dissertation, University of Tampere. 28

Turunen, M., Hakulinen, J., Melto, A., Heimonen, T., Laivo, T. & Hella, J. (2009). Suxes - user experience evaluation method for spoken and multimodal interaction. In *INTERSPEECH*, 2567–2570, ISCA. 40

UsabilityNet (2012). ISO 9241. `http://www.usabilitynet.org/tools/r_international.htm#9241-1x`. 38

van Ballegooij, A. & Eliéns, A. (2001). Navigation by query in virtual worlds. In *Web3D'01: Proceedings of the sixth international conference on 3D Web technology*, 77–83, ACM, New York, NY, USA. 8

van Lun, E. (2011). Chatbots. `http://www.chatbots.org/es`. 27

van Welbergen, H., Reidsma, D., Ruttkay, Z.M. & Zwiers, J. (2010). Elckerlyc - a bml realizer for continuous, multimodal interaction with a virtual human. *Journal on Multimodal User Interfaces*, **3**, 271–284. 8, 23

Verbio (2012). Verbio. `http://www.verbio.com/`. 108, 114

Vince, J. (1995). *Virtual reality systems*. ACM Press/Addison-Wesley Publishing Co. 101

Vince, J. (2004). *Introduction to Virtual Reality*. SpringerVerlag. 9

VoiceXML Forum (2010). VoiceXML Forum: "Voice eXtensible Markup Language". `http://www.voicexml.org/`. 3, 10

W3C (2010a). SVG. `http://www.w3.org/Graphics/SVG/`. 15

W3C (2010b). W3C Multimodal Interaction Activity. `http://www.w3.org/2002/mmi/`. 14, 24

W3C (2010c). W3C VBWG. `http://www.w3.org/Voice/`. 10

W3C (2011). World Wide Web Consortium (W3C). `http://www.w3.org/Consortium/`. 24

Walker, M., Litman, D.J., Kamm, C.A. & Abella, A. (1997). Paradise: A framework for evaluating spoken dialogue agents. *Annual Meeting of the Association for Computational Linguistics (ACL)*, 271–280. 37

Wallace, R.S. (2011). The Anatomy of A.L.I.C.E. `http://nic-nac-project.de/~chali/programowanie/Anatomy_of_ALICE.pdf`. 11

Web3D Consortium (2010). X3D. `http//www.web3d.org/x3d`. 12, 34

Williams, J.D. & Young, S. (2007). Partially observable markov decision processes for spoken dialog systems. *Comput. Speech Lang.*, **21**, 393–422. 31

Young, D. (1994). *The X Window System Programming and Applications with XT*. PTR. 17