Una interfaz vocal para la dirección de narración interactiva en entornos virtuales

Tesis de Máster

Álvaro Navarro Iborra

Directores:
Pablo Gervás Gómez-Navarro y
Federico Peinado Gil

Facultad de Informática
Universidad Complutense de Madrid
Madrid, 2008

Tabla de Contenidos

Resumen

En la mayoría de los sistemas de narración interactiva, así como en los videojuegos, la interacción con los usuarios se realiza a través de eventos generados mediante interfaces clásicas como el teclado, el ratón u otros periféricos similares. En este trabajo se plantea una alternativa para dirigir procesos de narración digital interactiva, complementando las interfaces clásicas con la posibilidad de dar órdenes por voz.

En primer lugar se realiza un análisis sobre el tipo de entornos virtuales sobre el que se desarrolla la narración, en segundo lugar se estudian las características de los distintos sistemas de dirección de narración interactiva y en tercer lugar se revisan las tecnologías que existen actualmente para desarrollar interfaces vocales. Se hace especial hincapié en los entornos virtuales que permiten representar el motor y las herramientas integradas en el videojuego Neverwinter Nights.

Tras este repaso al estado del arte, se especifica un lenguaje y un protocolo de comunicación para controlar entornos virtuales y se diseña un prototipo de interfaz vocal para dirigir narración interactiva en dichos entornos. Este prototipo se implementa mediante Java, XML y Sphinx, una librería de análisis de voz, lo que permite realizar pruebas de dirección real en un entorno específico en forma de módulo jugable para Neverwinter Nights.

La experimentación sobre este prototipo sirve para concluir que es posible la dirección de narración interactiva utilizando, al menos parcialmente, órdenes por voz. Esta experiencia nos permite además discutir las ventajas y los inconvenientes de introducir este nuevo medio de comunicación entre el director de juego y el entorno virtual, así como averiguar cual es el rendimiento de las tecnologías mencionadas al ser integradas en un mismo sistema de dirección de narración interactiva.

Palabras clave: Narración Interactiva, Reconocimiento de Voz, Entornos Virtuales, Juegos de Rol por Ordenador.

Agradecimientos

Agradezco a mis directores de tesis Pablo Gervás y Federico Peinado toda la ayuda que me han dado a lo largo del curso. Además quiero agradecer la paciencia que han tenido conmigo y su comprensión, especialmente en momentos de mayor dificultad. Asimismo agradezco a mi familia el apoyo que me han dado este año en la realización del Máster, especialmente a mis padres, Ángel y Joaquina y a mi tía Elena. También quiero agradecer a todos los amigos que me han animado a terminar la Tesis, así como a los compañeros que he tenido en el Master durante este año, que también me han animado a seguir adelante.

1. Introducción

La narración clásica abarca principalmente tres aspectos, una historia, una estructura y la acción de contar relatos. Estas tres dimensiones básicas de la narración constituyen un modelo de comunicación lineal con tendencia a ser unidireccional. Es decir, normalmente la narración se realiza por un narrador, el cual selecciona la información y la dispone en una única secuencia, manteniendo en todo momento el control del relato y provocando respuestas emotivas de su audiencia. La narración interactiva se basa en un modelo de comunicación en el cual el papel del emisor y el receptor son intercambiables. Ambos comparten el mismo espacio temporal, y pueden comunicarse en las mismas condiciones. A lo largo de la Historia han existido figuras de la sociedad que han sido narradores, como por ejemplo los trovadores y los Cuentacuentos.

El ejemplo clásico de la narración interactiva es el de los juegos de rol. Un juego de rol es un juego en el cual se interpreta un personaje en una realidad alternativa siguiendo una historia, a veces previamente diseñada y otras veces improvisada. La temática de las historias puede ser diversa, es decir, el juego puede estar basado en temas de romance, aventura, comedias, terror, o cualquier otro género.

En este tipo de juegos se construyen historias dinámicamente. Existe la figura de director de juego, también denominada Game Master o GM, que hace partícipes al resto de usuarios y mediante sus decisiones ayuda a construir la narración. Por tanto, el director tiene por misión organizar, relatar y preparar las aventuras para el jugador y los personajes del juego, describir los eventos y decidir cuales son los retos a los que se enfrentan los jugadores. Durante este proceso se usan modelos de reglas a los que están sujetos tanto el director como los jugadores, en los cuales la experiencia, las decisiones de juego, y la aleatoriedad contribuyen activamente para conformar la historia.

La narración digital interactiva es un término que nace con las nuevas herramientas de creación digital. En este proceso se adapta el arte clásico de la narración de historias enfocándolo hacia medios audiovisuales modernos, añadiendo además un componente de interactividad, que no siempre se da en la narración clásica. El ejemplo más característico de este tipo de narración se encuentra en los juegos denominados CRPG (Computer Rol Playing Game), que consisten básicamente en la adaptación de un juego de rol al ordenador.

La narración digital interactiva con un director humano es una experiencia que permite desarrollar una partida a un juego de rol por ordenador de manera análoga a la de un juego de rol de lápiz y papel.

Es decir, el papel que realizan todos los jugadores es el mismo en el mundo virtual. Mediante un buen director experimentado con dominio del entorno que esté controlando, se podrán introducir sorpresas y hacer el juego más atractivo que con un mecanismo de dirección previamente establecido, implementado con un director automático. Aunque un director de este tipo puede funcionar razonablemente bien, un humano siempre aportará elementos, sobre todo si conoce ya a los jugadores, que hagan la partida más amena y atractiva.

La dirección de una partida de rol de lápiz y papel no es sencilla, ya que se requiere un usuario experimentado que controle todos los aspectos del juego. Por tanto debe tener experiencia en este tipo de juegos y tener imaginación para plantear retos apasionantes y mantener un equilibrio, entreteniendo a todos los jugadores. Por tanto, dirigir una partida de rol en un ordenador es más complejo que sobre lápiz y papel, ya que además del dominio y experiencia de este tipo de juegos debe saber manejar todas las opciones del interfaz de director de juego.

Los entornos virtuales de representación que existen en los sistemas de narración interactiva, así como en los videojuegos, permiten al usuario navegar por un espacio virtual y experimentar el desarrollo de una historia que la aplicación le describe mediante gráficos 3D, según se mueve por él, y conversaciones con personajes interactivos. En la mayoría de estos sistemas, la interacción con los usuarios se realiza a través de eventos mediante periféricos, como el teclado, el ratón u otros similares. En este trabajo se plantea un estudio sobre otra alternativa para dirigir este tipo de sistemas, la dirección a través de la voz.

Como se verá en el estado del arte, hay sistemas como Neverwinter Nights, Neverwinter Nights 2 o Vampire The Masquerade: Redemption, que poseen interfaces de director de juego. La realización de una partida será por tanto menos fluida que sobre el papel, ya que se necesita un conocimiento amplio de estas interfaces, lo que a su vez requiere un tiempo de aprendizaje. Además, al jugar en red, las acciones no se pueden controlar de la misma manera que sobre el papel. Por ejemplo, si el director de juego debe controlar que un jugador no realice una acción, y está pendiente de los demás, el jugador en cuestión podría llegar a realizar esa acción. En relación con el proceso de jugar en red, puede requerir mucha más coordinación y preparación que la partida sobre el papel, ya que los jugadores pueden estar físicamente en sitios distintos. En una partida de lápiz y papel los jugadores juegan sobre un tablero, por lo que deben estar todos juntos, en cambio, en un juego por ordenador sólo necesitan estar conectados a un mismo servidor a una hora concreta, pero no tienen porque conocerse en el mundo real ni coincidir geográficamente en la

misma ubicación.

Una de las mejoras que se plantean al introducir un sistema de dirección vocal es la mejora en la accesibilidad a algunos menús, aumentar la velocidad en realizar ciertas acciones de control sobre el entorno y poder aprender a manejar la interfaz de director de juego más fácilmente.

La complejidad de algunos interfaces puede verse simplificada mediante reconocimiento de voz. Por ejemplo, para buscar un objeto o personaje del juego se debe buscar entre todos los escenarios y después buscar entre todos los elementos del escenario, lo cual podría hacer un usuario más fácilmente mediante una orden de voz.

Intuitivamente una interfaz vocal no puede pretender realizar absolutamente todas las acciones que realiza una interfaz gráfica, ya que esto implicaría un repertorio muy amplio de instrucciones con posibles variaciones en las mismas, con lo cual el objetivo de simplificar la dirección no se consigue. Por tanto una interfaz vocal podrá beneficiar a directores que no tengan un gran conocimiento del interfaz de director de juego del entorno que estén usando.

1.1. Objetivos

Los objetivos principales de este trabajo son:

- Averiguar cual es la utilidad real y los requisitos de una interfaz vocal que ayude a los directores de narración interactiva en el proceso de gestión en tiempo real del entorno virtual.

- Diseñar un sistema informático que incluya todos los componentes necesarios para desarrollar dicha dirección de narración interactiva con la ayuda de un reconocedor de voz.

- Probar que dicho sistema es tecnológicamente factible en el contexto del motor de un juego de rol por ordenador comercial.

1.2 Estructura

El trabajo consta de varios apartados: una revisión del estado del arte, el diseño del sistema que se ha implementado, el entorno virtual del sistema, una discusión sobre los aspectos de los interfaces vocales y unas conclusiones.

El capítulo 2 consiste en el estado del arte; en él se muestra en primer lugar un estudio de los motores de entornos virtuales más relevantes,

después una revisión de los principales sistemas de narración interactiva actuales y por último se hace una revisión de algunas de las tecnologías más importantes sobre reconocimiento de voz.

En el capítulo 3 se muestra una interfaz vocal para la dirección de narración interactiva donde se detalla la implementación que se ha realizado para poder llevar a cabo tanto la comunicación con el entorno como el reconocimiento de voz de los comandos.

El capítulo 4 presenta una aplicación del interfaz vocal sobre un entorno virtual narrativo, es decir, una descripción del prototipo llevado a cabo, que consiste en el guión de una aventura, su implementación en el entorno usado y muestra cómo se dirige esta aventura con el interfaz vocal.

La discusión se realiza en el capítulo 5, y en ella se hace un análisis sobre lo que aportan los interfaces vocales a estos entornos, en qué aspectos podrían beneficiar a videojuegos y en qué aspectos resultan menos útiles.

Por último, en el capítulo 6 se muestran las conclusiones destacando la experiencia obtenida en el uso de una interfaz vocal, viendo cuáles son sus aspectos positivos y cómo se podría mejorar en el futuro.

2. Estado del arte

En el estado del arte se detallan los sistemas que han sido objeto de estudio en este trabajo. Se engloban en tres tipos de sistemas distintos: entornos virtuales, sistemas de narración interactiva e interfaces vocales.

El primer tipo de sistemas son los entornos virtuales. En este apartado se citan varios sistemas centrados en la temática de los juegos de rol y se detalla más extensamente Neverwinter Nights, que es el que se ha usado en el trabajo.

En el apartado de los sistemas de narración interactiva se hace un estudio de los principales sistemas que se han desarrollado en este campo, analizando cuales son sus características más importantes.

Finalmente, los interfaces vocales son sistemas que analizan una señal de audio con una serie de palabras y son capaces de reconocer esa señal traduciéndola a un texto que representa el mensaje expresado en la voz de la señal de audio. Igualmente se detallan los sistemas más relevantes, centrándose especialmente en el sistema Sphinx, que se ha utilizado en el trabajo.

2.1. Entornos virtuales

Un entorno virtual es un entorno parcial o totalmente basado en las entradas generadas por un ordenador. Los entornos virtuales constituyen un campo muy amplio de investigación y se aplican a muchos ámbitos distintos dentro de campos como el de la enseñanza o el ocio.

En el ámbito de la enseñanza, los entornos virtuales se caracterizan por ampliar el acceso a la educación, promover el aprendizaje colaborativo y el trabajo en grupo, promover el aprendizaje activo, crear comunidades de aprendizaje, y hacer más fluidos los roles tradicionales del proceso de enseñanza y aprendizaje. En ocasiones, en el ámbito de la educación, se utilizan videojuegos como una herramienta atractiva para el aprendizaje (Sancho, 2007).

En otros ámbitos, estos sistemas suelen estar orientados al entretenimiento y el ocio, como en el caso de los juegos de ordenador, Los juegos de ordenador disponen de grandes comunidades de usuarios mediante las cuales pueden jugar en red muchos usuarios conectados desde diferentes puntos geográficos.

Esta característica es muy importante en cierto tipo de entornos denominados mundos virtuales, que son aplicaciones donde se simula el comportamiento del mundo real. Los mundos virtuales tienen mucho éxito hoy en día y cuentan con comunidades de miles usuarios. Actualmente, uno de los entornos de este tipo más conocido es el Second Life (Second, 2008), aunque hay varios otros: Sims Online (Sims, 2008), Active Worlds (Active, 2008), Forterra Systems (Forterra, 2008), Habbo Hotel (Haboo, 2008), Kaneva (Kaneva, 2008), There (There, 2008).

Los entornos virtuales poseen complejos motores de juego. En este apartado se analizan motores de entornos virtuales con las características que se pretenden estudiar en este trabajo. Es decir, motores de juegos de rol que incluyan opciones para poder usar un director de juego. Principalmente, nos centraremos en el motor del juego Neverwinter Nights, que es el que se ha utilizado para el desarrollo del prototipo mencionado en el capítulo 4.

2.1.1. Neverwinter Nights

Neverwinter Nights es un juego de rol por ordenador basado en las reglas del popular juego de lápiz y papel "Dungeons & Dragons", particularmente en la versión D20.

Cuando se lanzó el juego pretendió que recreara de manera bastante fiel la experiencia de los clásicos juegos rol de lápiz y papel sobre un tablero. Las principales dificultades en aplicar las reglas de "Dungeons & Dragons" a un juego de ordenador son la generación del escenario y su modificación dinámica.

Una de las ventajas más importantes que posee Neverwinter Nights es una herramienta que permite la creación de escenarios, que incluye la creación de mapas, un lenguaje de scripting para programar eventos y una librería de personajes predefinidos con Inteligencia Artificial.

Actualmente existen dos versiones del juego. Neverwinter Nights 2 se basa en la versión 3.5 de las reglas de Dungeons & Dragons e introduce mejoras notables con respecto a la primera versión, desde el punto de vista gráfico y desde el punto de vista de la herramienta de desarrollo. Los nuevos menús desplegables y contextuales sustituyen al menú radial permitiendo realizar cualquier acción con sólo dos clics. Una nueva barra de modos permite activar y desactivar fácilmente el modo sigilo, el ataque poderoso y los demás modos. En la herramienta de desarrollo los módulos son más fáciles de crear, son más personalizables y los scripts admiten nuevos tipos de datos.

En este apartado se analizarán ambas versiones, así como otras dos herramientas relacionadas con el juego: Neverwinter Nights Extender y Shadow Door.

Características

Neverwinter Nights 1 fue publicado en el año 2001 por la compañía Bioware para la plataforma Windows. Es un juego del género de aventuras y del subgénero de juego de rol, ambientado en un mundo de fantasía. El motor de juego es propietario y su lenguaje de programación es NWN Script, un lenguaje de scripting muy similar a C, que permite utilizar las instrucciones del núcleo del motor gráfico.

Neverwinter Nights 2 fue publicado en el año 2006 por la compañía Atari para las plataformas Windows y Mac OS X. Al igual que su predecesor es un juego del género de aventuras y del subgénero de juego de rol, ambientado en un mundo de fantasía. El motor de juego es una evolución de la versión anterior, aunque no es totalmente compatible con la versión anterior. El motivo que no sea totalmente compatible es que no podemos utilizar módulos de juego creados en Neverwinter Nights y cargarlos en Neverwinter Nights 2, ya que los scripts pueden no ser compatibles, aunque otros elementos como personajes del juego, diálogos o efectos sonoros si se pueden importar.

Neverwinter Nights tiene dos modos de juego, uno normal donde el jugador juega la aventura en primera persona y un modo llamado Dungeon Master donde el usuario puede participar como director de juego. El cliente Dungeon Master nos ofrece tanto para Neverwinter Nights 1 como para Neverwinter Nights 2 la posibilidad de dirigir una partida en red pudiendo realizar una gran variedad de acciones, como las que puede realizar un director en un juego de rol clásico. A continuación se detallan estas acciones.

- **Crear un solo objeto:** se escoge un objeto con el selector y se pulsa el botón “Crear”. El cursor se convertirá en un icono de objetivo. En este punto, se debe hacer clic en la zona del suelo donde se quiera crear el objeto.

- **Crear varios objetos a la vez:** se escoge un objeto con el selector y pulsar el botón “Crear” para crear un solo objeto. A continuación el cursor se convertirá en un icono de objetivo. En este punto, se debe pulsar la tecla MAYÚS y hacer clic en la zona del suelo donde se quiera crear el objeto. El cursor se quedará en el modo de objetivo y se podrán seguir creando más instancias del objeto.

- **Ir a:** esta acción moverá el avatar del Dungeon Master hasta el objeto seleccionado. Si se trata de un área, este avatar se moverá hasta el centro de la misma.

- **Matar:** esta acción terminará con criaturas y destruirá sus objetos. En el caso de las criaturas, el evento OnDeath se perderá. Las criaturas invulnerables no morirán.

- **Saltar:** esta acción hace saltar a la criatura o criaturas seleccionadas hasta el punto señalado. Después de pulsar el botón de saltar, el juego entrará en modo de objetivo. En este punto, hacer clic en el lugar del suelo donde se quiera que salten las criaturas.

- **Sanar:** cura a la criatura o criaturas seleccionadas hasta concederles todos los puntos de golpe que tuvieran.

- **Limbo:** mueve el objeto u objetos seleccionados hasta una zona "de suspensión". Mientras estén en el limbo, los objetos no tendrán Inteligencia Artificial. Se debe usar la acción de Saltar para recuperar un objeto del limbo.

- **Examinar:** abre la ventana de examinar para el objeto seleccionado.

- **Tomar el control:** se posee al objeto seleccionado. Poseer a una criatura es similar a poseer a un familiar en el juego normal. Las funciones de DM quedan anuladas mientras dure la posesión. Para salir de ésta, se debe hacer clic derecho en la criatura poseída y seleccionar "Finalizar posesión" en el menú radial.

- **Controlar con plenos poderes:** igual que poseer, pero no se pierden las funciones de DM.

- **Descansar:** devuelve todos los conjuros y puntos de golpe a la criatura o criaturas seleccionadas.

- **Invulnerable:** alterna la bandera de trama en el objeto, haciendo que éste sea inmune a la muerte y al daño.

- **Buscar por nombre:** busca un objeto específico por el nombre que tenga. Si lo localiza, el área donde éste se encuentre se abrirá con el objeto seleccionado.

- **Buscar siguiente:** busca el ejemplo siguiente de un objeto en particular usando la búsqueda actual.

- **Acciones con puertas:** estas acciones se llevan a cabo con el menú radial, y echan la llave a una puerta o la quitan.

- **Acciones con criaturas:** muchas de estas acciones son idénticas a las órdenes del panel del Selector y se realizan con el menú radial

- **Examinar:** lo mismo que la acción de examinar de un jugador.

- **Dar/coger:** seleccionar la acción que se desee y luego teclear la cantidad en la casilla de diálogo. Las acciones son: Dar oro y Coger oro, Dar PX y Llevarse PX, Dar nivel y Coger nivel.

- **Poseer:** permite controlar a la criatura existente, incluyendo sus aptitudes y habilidades. Se puede morir mientras se está poseyendo a otra criatura.

- **Suplantar:** permite ser el personaje o monstruo, pero sin renunciar a los poderes de DM. Se es invulnerable mientras se suplanta a otro personaje o criatura.

- **Acciones con un grupo:** cuando se tenga seleccionado a un grupo, se pueden realizar acciones con sus miembros y ordenarles que lleven a cabo acciones como grupo.

- **Seleccionar a un grupo:** mantener pulsada la tecla CTRL mientras se arrastra para seleccionar a un grupo. La selección quedará enmarcada por un recuadro amarillo. Cuando se deje de arrastrar, todas las criaturas seleccionadas tendrán círculos púrpuras alrededor de los pies.

- **Eliminar la selección de un grupo:** para dejar de seleccionar a un grupo, se arrastra el recuadro a un terreno vacío y se comprueba que no queda nada seleccionado.

- **Asignar una tecla de acceso rápido a un grupo:** pulsar CTRL y un número teniendo seleccionado a un grupo. Esto vinculará al grupo actual con esa tecla. Para volver a seleccionar al grupo basta con pulsar el número con el que se haya vinculado.

- **Llevar a cabo acciones de grupo:** por defecto, sólo el director de juego lleva a cabo las acciones estándar. Para hacer que en vez de él las realice el grupo seleccionado actualmente, se debe mantener pulsada la tecla MAYÚS mientras se da la orden.

- **Acciones que se puede permitir que realicen los grupos**
    - **Caminar:** el grupo seleccionado andará si se mantiene

pulsada la tecla *Mayús* y se hace clic en suelo vacío.
    - **Atacar:** el grupo seleccionado atacará si se mantiene pulsado *Mayús* y se hace clic en una criatura hostil; o si se mantiene pulsado *Mayús* y se usa la acción radial de atacar a partir de la criatura objetivo.

- **Lista de jugadores:** debajo del retrato del director de juego se dará una lista de todos los jugadores que estén en una partida en un momento dado. Al hacer clic en el retrato para acceder al menú radial, se puede llevar a cabo a través de él cualquier acción que normalmente se haría con la criatura.

- **Línea de comando:** para usar una acción dada con una línea de comando, se accede a la consola mediante la tecla º. También se pueden introducir comandos de la consola en la ventana de conversación si se preceden de una doble almohadilla "##".

- **Asignar una línea de comando a la barra de acceso rápido:** para conseguir esta asignación hay que hacer clic con el botón derecho en el espacio de la barra de acceso rápido para acceder al menú radial. Seleccionar la opción que se quiera asignar a la línea de comando. Introducir una etiqueta y luego el comando precedido por ##.

Hay dos partes dentro del juego, por un lado el motor del juego y por otro lado los módulos. El motor del juego es responsable de renderizar los objetos del juego y los efectos especiales, así como de mover los objetos, reproducir la banda sonora del juego y los sonidos y efectos ambientales, enviar los eventos a los scripts, que serán ejecutados por turno en una máquina virtual. Los módulos son archivos que contienen el contenido de la historia, esto incluye mapas de datos, objetos de la historia, scripts y ficheros de conversaciones.

Para jugar a Neverwinter Nights, un jugador empieza seleccionando el módulo al que va a jugar y después eligiendo el personaje con el que jugará en ese módulo. Posteriormente, el motor del juego cargará los scripts y los objetos del juego en memoria y el juego comenzará.

Interfaz de juego

Para que el usuario se mueva por el escenario puede hacerlo simplemente con un clic en el punto al cual desea desplazarse, y el personaje irá automáticamente al punto seleccionado, aunque en ese caso el punto no debe estar muy alejado o sino el sistema no sabrá calcular el camino para que el personaje se desplace. Otra manera es desplazarse con los cursores. Así mismo, para interactuar con los elementos del juego como objetos y personajes sólo es preciso un clic

en el objeto.

La pantalla principal de la interfaz tiene nueve secciones distintas, como podemos ver numeradas en la Figura 1, que proporcionan una información muy valiosa sobre el juego permitiendo el control sobre el personaje y la partida.

Figura 1. Escenario de Neverwinter Nights, donde aparece numerada cada una de las opciones del interfaz de juego.

1) El retrato del personaje: es la imagen que se escoge para representar a un personaje. Al lado del retrato hay una barra roja que muestra el estado de salud actual del personaje.

2) El recuadro de opciones: aquí se puede acceder en cada uno de los botones las pantallas de interfaz necesarias para manejar al personaje, como se puede ver con mayor detalle en la figura 2. Estas opciones son el mapa de juego, la pantalla del inventario, el diario, la hoja de personaje, la página de opciones, el libro de conjuros y el panel de opciones de jugador contra jugador.

3) La barra del grupo: aquí aparecen los miembros del grupo del jugador. Se pueden agregar más jugadores mediante la opción “socializar” del menú radial. También se puede usar la barra del grupo para obtener información rápida de los compañeros del jugador o llevar a cabo acciones con ellos.

Figura 2. Retrato del jugador y opciones para acceder a los distintas pantallas de usuario.

4) Las ventanas de conversación: muestran los mensajes de los demás jugadores y permite conversar con otro jugador, haciendo clic en su retrato.

5) La barra de entrada de conversación: aquí se escriben los mensajes de conversación. Para hacerlo se debe pulsar *Intro* o hacer clic en la barra.

6) La barra de acceso rápido: permite acceder rápidamente a muchas funciones del juego para agilizar la partida.

7) La brújula: sencillo indicador de la dirección. La "N" de la brújula siempre señala el norte, por lo que para hacer caminar a un personaje en una dirección debemos seguir la brújula.

8) La cola de acciones: una característica importante del motor de Neverwinter Nights es que todas las acciones que llevan a cabo los personajes no se ejecutan inmediatamente, sino que son almacenadas en una cola y el motor las va ejecutando en orden de llegada. Esto implica que determinadas acciones pueden no ejecutarse si sufren demora. En el panel de cola de acciones si muestran las acciones que el personaje de un jugador está llevando a cabo en el momento y todas las que se le ha ordenado realizar.

9) La barra de estado: muestra los efectos que estén activos en el personaje del jugador en un momento dado, como conjuros, venenos o pociones.

En Neverwinter Nights existe un menú radial que se puede obtener alrededor de los personajes y permite aplicar acciones pulsando en la opción deseada. Para desplegar este menú circular se debe hacer clic con el botón derecho del ratón sobre un personaje o sobre el propio

jugador. Una vez desplegado el menú, el jugador podrá elegir cualquiera de las opciones disponibles, como se ve en la figura 3. Estas acciones son ataques especiales, atacar, lanzar conjuro, hoja de personaje, sentimiento, aptitudes especiales y ordenar. Si la opción elegida tiene a su vez un conjunto de subopciones se desplegará un nuevo menú radial con dichas opciones.

Figura 3. Muestra de funcionamiento del menú radial de Neverwinter Nights.

La pantalla principal de la interfaz de Neverwinter Nights 2 tiene varias secciones distintas, que han variado en cuanto a la distribución de su predecesor, aunque las opciones son en general las mismas que Neverwinter Nights 1. En la figura 4 se puede apreciar que los menús que teníamos antes no se corresponden con los del interfaz de Neverwinter Nights 2. Además el menú radial ha sido sustituido por nuevos menús desplegables y contextuales que permiten realizar cualquier acción con dos clics.

En la parte superior izquierda de la pantalla aparecen los personajes que están interviniendo en la acción del jugador. En la parte inferior están los botones de acceso rápido para realizar conjuros, tomar pociones y acciones similares. En la parte inferior derecha se pueden cambiar algunos modos de juego, como modo en sigilo, modo de ataque o modo normal.

La pantalla principal de la interfaz tiene varias secciones distintas, como podemos ver numeradas en la Figura 4, que al igual que Neverwinter Nights permiten controlar todos los aspectos de la partida.

Figura 4. Captura del interfaz de juego de Neverwinter Nights 2.

1) Escenario de Juego: es la pantalla de acción que muestra la parte del área en la que se está jugando en ese momento, con los personajes, objetos y demás elementos del juego.

2) Personajes activos: son todos los que está controlando un jugador. Se puede cambiar el control entre el personaje de un jugador y el resto de personajes del grupo.

3) Mini-mapa: muestra una vista de arriba a abajo del área que rodea al personaje activo. La flecha en el centro muestra la dirección en la que el personaje activo se encuentra. El trapezoide del que emana de la flecha muestra la vista de cámara actual en el juego. Los botones + y – sirven para aumentar o alejar el zoom del mapa. Los puntos circulares en el mapa representan localizaciones especiales, personajes u objetos. Si se pasa el ratón por encima de estos puntos se obtiene información textual acerca del elemento.

4) Barra de modos de juego: se utiliza para alternar diferentes estados del jugador. Algunos de estos modos sólo están disponibles si el personaje activo posee ciertas habilidades. Los principales son: modo normal, modo sigilo, modo potencia de ataque y modo combate experto.

5) Ventana de Chat: muestra información del juego, como daños o tipos de ataque. En la opción multijugador todo lo que diga otro jugador aparece en esta ventana. Para usar el Chat y hablar con otros jugadores hay que pulsar *Enter* y escribir el mensaje.

6) Barra de menú: contiene botones para activar las principales interfaces del jugador. Estas interfaces son la hoja de personaje, el inventario, el diario, el libro de hechizos, la lista de jugadores (sólo para multijugador) y el menú de escape para salvar y cargar partidas o establecer opciones de generales del juego.

7) La barra de acceso rápido: permite acceder rápidamente a muchas funciones del juego, como hechizos, habilidades u objetos, para agilizar la partida. Cualquier acción se puede utilizar haciendo clic en ella o por pulsando su número en el teclado.

8) Barra de grupo: la barra de grupo es una lista con todos los miembros del grupo, en ella se muestra una imagen con el retrato de cada personaje. Haciendo clic en la imagen es análogo a hacer clic en el escenario sobre el personaje. Por lo que puede usarse para hace emitir un hechizo a un miembro del grupo.

9) Cola de acciones: como ya se ha comentado para Neverwinter Nights, las acciones no se ejecutan siempre inmediatamente sino que se van almacenando en una cola. Esta opción muestra la acción actual y las acciones que se han ordenado ejecutar. Cada acción se realiza en el orden en que aparece en la cola, y cuando se asigna una nueva orden se añade al final de la cola. Algunos comandos, como el movimiento, se liberan de la cola de acciones y son ejecutados inmediatamente. Para eliminar cualquier acción de la cola sólo hay que hacer clic en la acción pendiente.

10) Objetivo seleccionado: al seleccionar un objetivo la imagen con su retrato y su nivel de salud aparecerá a la derecha del mini-mapa. Este objetivo es el destinatario por defecto de cualquier habilidad que use el personaje, incluyendo cualquier hechizo.

Objetos de juego

En Neverwinter Nights existen una variedad de tipos de objetos que puede contener un módulo. Estos objetos son los siguientes:

- Áreas: un área es un espacio tridimensional donde se encuentran los personajes, objetos y demás entidades que ocupan un espacio físico en el entorno. Un módulo puede tener

varias áreas que se comunican por medio de transiciones, generalmente a través de puertas.

- Criaturas: las criaturas son las entidades que representan a los jugadores y a los personajes del juego que no son jugadores. Se pueden mover, participar en combate, conversar y almacenar objetos en su inventario.

- Puertas: pueden ser colocadas en ciertos sitios que tienen una ubicación específica para la puerta. Pueden ser usadas para pasar de un escenario a otro y pueden ser abiertas, cerradas, bloqueadas y golpeadas.

- Ubicables: son objetos inanimados como sillas, mesas o tesoros, que sirven como mera decoración o como contenedores de ítems que pueden ser recogidos por los jugadores.

- Triggers: son polígonos invisibles que ocupan un espacio de la superficie de un área. Cuando cualquier jugador pasa por esa superficie pueden dispararse eventos que cambian el curso actual del juego. Estos eventos pueden realizar una teleportación, activar una trampa o hacer una criatura.

- Encuentros: son un tipo especial de triggers para hacer aparecer un grupo de enemigos cuando el jugador lo dispara. Un encuentro hace aparecer un número de enemigos variable dependiendo del nivel del jugador.

- Mercaderes: son objetos invisibles que representan un almacén donde el jugador puede adquirir ítems y equipamiento. Normalmente un mercader suele estar asociado a un personaje que actúa como mercader propiamente dicho. El jugador puede comprar del almacén iniciando una conversación con el personaje.

- Sonidos: son objetos invisibles que tienen el propósito de difundir un sonido en una parte concreta del módulo. Cada sonido tiene un radio de efecto donde actúa.

- Ítems: son los objetos que existen en los inventarios de las criaturas, en contenedores o el suelo. Los ítems pueden ser armas, armaduras, pociones y libros. Muchos ítems pueden ser equipados por jugadores o personajes del juego aumentando sus habilidades. Pueden ser comprados o vendidos a mercaderes.

- Puntos de ruta: son objetos invisibles que el diseñador del módulo ubica en puntos del área. Son usados para controlar el

movimiento de algunas criaturas; por ejemplo, para guardias que continuamente estén patrullando de un punto a otro custodiado un tesoro.

Conversaciones

Las conversaciones con los personajes en Neverwinter Nights tienen una estructura de árbol, donde en cada nivel del árbol se suceden preguntas y respuestas entre jugador y otro personaje, como se observa en la figura 5. Cada nodo puede llevar un script asociado para ejecutar una determinada acción al llegar a un punto de la conversación.

Figura 5. Ejemplo de una conversación en Neverwinter Nights entre dos personajes.

Scripts

Los scripts son un recurso del motor de Neverwinter Nights que permite a usuarios escribir código para realizar sus propias acciones en el juego. Cada uno de los scripts se almacena en un archivo diferente, y podrán ser ejecutados por el motor de Neverwinter Nights desde cualquier elemento de juego que sea "scriptable". En la figura 6 se puede apreciar que cada script está escrito con el lenguaje NWNScript, que es básicamente lenguaje C con algunas particularidades propias del entorno.

El motor de juego proporciona una API que el autor puede utilizar en los scripts que programe para controlar la historia. Los scripts también permiten almacenar una variable de tipo entero, cadena de texto u objeto del juego en cualquier objeto del módulo.

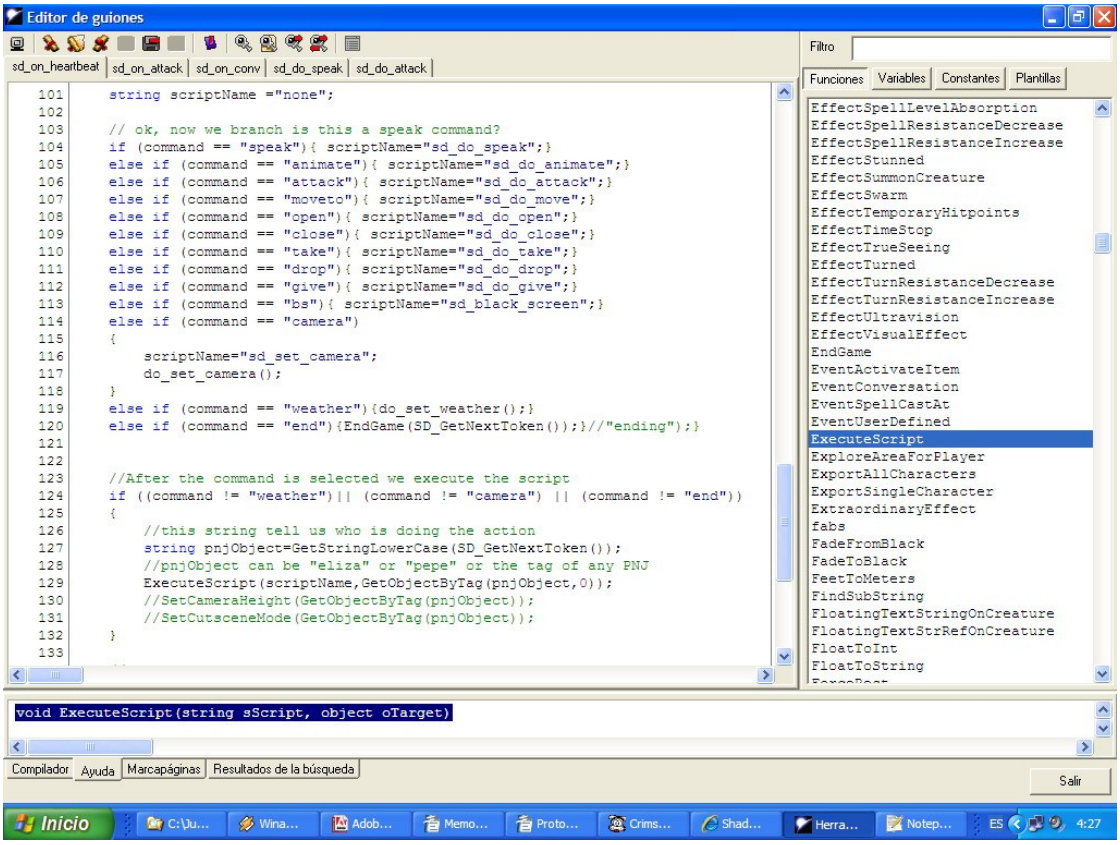


Figura 6. Edición de scripts en NWNScript con la herramienta Aurora de Neverwinter Nights.

Los scripts pueden ser adjuntados a los objetos de juego o a nodos de conversación. Estos objetos responden a ciertos eventos que pueden ser disparados durante la partida. Cuando se dispara un evento, el motor de juego busca el script correspondiente al objeto como respuesta al evento; después el motor ejecuta el script. Una vez que el script se ha terminado de ejecutar, el motor continúa ejecutando el juego para procesar nuevos eventos.

En la implementación se han utilizado algunas herramientas que no son exactamente motores de juego sino otras utilidades que han ayudado o han servido de orientación para realizar la conexión entre un lenguaje de alto nivel y el núcleo del motor de juego de Neverwinter Nights. A continuación se muestran Neverwinter Nights Extender, una librería para realizar comunicaciones externas con Neverwinter Nights y Shadow Door, una interfaz sencilla para el control remoto mediante sockets de las acciones de un personaje.

Neverwinter Nights Extender

Neverwinter Nights Extender o NWNX es un programa disponible para Windows (Avlis), que permite realizar determinadas operaciones que no están disponibles en el motor de NWN a través su lenguaje de scripting. Hay dos versiones de Neverwinter Nights Extender; NWNX2 sirve para realizar la comunicación con el juego NWN 1 y NWNX4 que sirve para realizar la comunicación con el juego Neverwinter Nights 2.

NWNX permite la conexión a bases de datos mediante ODBC, el uso de nuevas estructuras de datos en NWN como tablas hash, listas enlazadas o cualquier otra estructura.

La distribución de NWNX consiste en dos partes: el ejecutable, que despliega una ventana (figura 7) y hace correr el proceso, y gamespy watchdogs, que es una librería dinámica que corre dentro del servidor NWN. La librería actúa como un plugin, es decir, contiene funcionalidad adicional que se añadir al servidor en marcha.

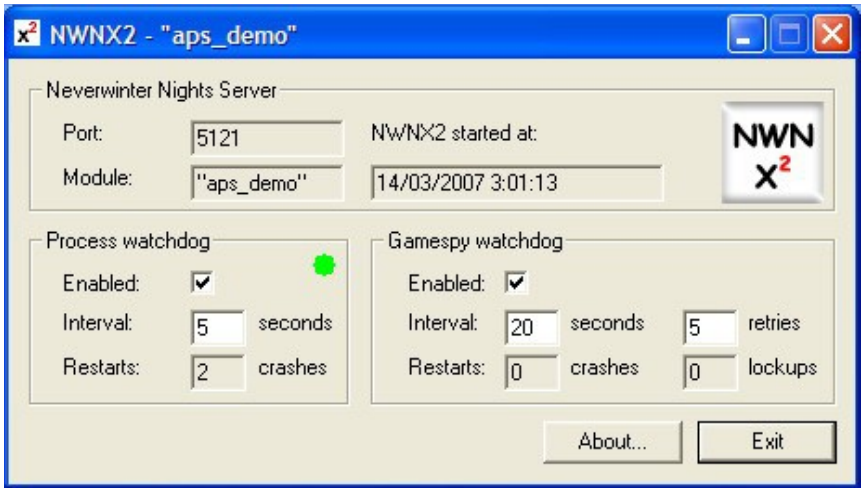


Figura 7. Ventana que aparece al ejecutar el archivo ejecutable NWNX2.exe. Después de esto aparece la ventana del servidor.

Para establecer una comunicación entre scripts y el mundo exterior, NWNX intercepta llamadas de la función de NWNScript *SetLocalString()*, y examina la variable name que se pasa a esta función. Si esta variable empieza con la cadena "NWNX!", NWNX parsea la respuesta y almacena cualquier dato resultante dentro del valor de esa variable, que en turno puede ser leída por un script con la función *GetLocalString()*.

Después de que NWNX2 haya cargado el servidor se encarga de reiniciarlo en caso de que este se caiga.

NWNX tiene disponibles un conjunto de plugins que implementan soluciones para situaciones que no se pueden realizar con Neverwinter

Nights. Estos plugins permiten realizar funciones como conectar a una base de datos en tiempo real, obtener estadísticas sobre la ejecución de scripts o implementar estructuras de datos complejas.

Shadow Door

Shadow Door (Zubek, 2003) es una API para el control de un personaje en Neverwinter Nights. Esta librería permite a procesos externos controlar a los agentes del juego que no sean el propio jugador. Usando Shadow Door, los desarrolladores pueden escribir programas externos usando lenguajes y plataformas arbitrarias.

La distribución de Shadow Door consiste en:

1) Un fichero DLL en forma de plugin para NWNX, que comunique procesos externos a través de sockets TCP.

2) Un grupo de scripts para controlar un solo agente que recibe comandos de procesos externos, los ejecuta en el entorno y devuelve ciertas observaciones.

3) Un API en Lisp con un ejemplo para conectarse al juego y controlar un agente.

La librería Shadow Door nos proporciona unas funciones básicas que podemos usar en cualquier script de nuestro módulo de Neverwinter Nights programado con NWNScript. Estas funciones básicamente nos servirán para obtener información enviada desde el exterior y para enviar información al exterior.

Shadow Door usa un protocolo sencillo para enviar comandos que comienzan por un nombre seguidos de varios argumentos. Los tokens estarán separados por el signo # y terminarán con fin de línea. Una de las principales limitaciones de Shadow Door es que sólo permite controlar externamente un único personaje.

2.1.2. Vampire The Masquerade: Redemption

Vampire The Masquerade (Nihilistic, 2000) es un juego de rol por ordenador que combina las reglas de un juego de rol con el entretenimiento de un juego de acción en primera persona, e incluso algunos elementos de una aventura gráfica.

El juego esta ambientado en el mundo de los vampiros, y trata temas como la humanidad, la cordura, la traición y ambiciones entre criaturas de su especie. En el juego cada vampiro tiene una

puntuación de la humanidad, que mide cuanto estrechamente están en contacto los vampiros con su naturaleza humana.

Uno de los aspectos más importante que tiene Vampire The Masquerade con respecto otros juegos de rol por ordenador similares es el sistema de narración que utiliza para que un jugador pueda ser narrador del juego. El modo multijugador se diseñó con la idea de que no fuese un simple juego en red con duelos y un sistema cooperativo, sino con la idea de crear algo que se aproximase a un director de juego o a un narrador. Esta idea estaba inspirada en los juegos del rol o juegos de texto multi-usuario o MUD (text-based multi-user dungeon).

Este sistema utiliza las normas generales de narración y una serie de mecanismos encaminados hacia la simulación de la existencia de vampiros. Un vampiro tiene un nivel de sangre que indica la cantidad de sangre humana actualmente en su cuerpo; está sangre puede ser proporcionar habilidades sobrenaturales, como la conversión en animales o en niebla, el hecho de dormir en el suelo, tener un carisma sobrenatural y poderes de sugerencia.

Características

Vampire The Masquerade: Redemption es un juego del género de aventuras y del subgénero de juego de rol, ambientado en un mundo de fantasía. El motor de juego es propietario y posee un modo narrador que permite a un director de juego dirigir partidas de uno a ocho jugadores. Además existe un software de desarrollo llamado Nod SDK, que permite generar nuevos escenarios donde realizar la partida.

Interfaz de juego

La interfaz del juego provee una serie de menús para interactuar con los personajes y objetos del mundo, conversar con los personajes, consultar las misiones, etc. Para luchar con los enemigos, igual que sucede en un juego de acción, el jugador debe hacer clic varias veces sobre los enemigos hasta que mueran. Según el modo de juego, ya sea modo narrador o modo normal, encontramos una serie de menús y opciones diferentes para cada uno de los casos, que se describen a continuación.

La pantalla principal de la interfaz tiene varias secciones distintas, como podemos ver numeradas en la Figura 8, que proporcionan una información muy valiosa sobre el juego permitiendo el control sobre el personaje y la partida.

1) El retrato del personaje: imagen con el rostro del personaje. Debajo del retrato hay una barra roja que muestra el nivel de

vida del personaje.

2) El recuadro de opciones: aquí se puede acceder en cada uno de los botones las pantallas de interfaz necesarias para manejar al personaje. Estas opciones son el panel de personajes, el panel de inventario, el panel de búsquedas, el panel de disciplinas y el mapa de juego.

Figura 8. Escenario de Vampire The Masquerade: Redemption, donde aparecen numeradas las opciones en el modo jugador.

3) El panel de conversación: se muestra el texto con los nombres de los personajes y objetos del juego, así como las acciones que se realizan en el juego.

4) La barra de objetos: aquí aparecen los objetos que se van cogiendo durante la partida, como bolsas de sangre.

5) La barra de estado del jugador: indica la disciplina que tiene el jugador en el momento, como puede ser alimentación o curación de sangre.

6) Cambio a modo narrador: en una partida en red donde un jugador puede ser narrador cambia las opciones para poder ser director de la partida.

Está opción incluye herramientas para la narración, el usuario puede seleccionar un escenario, personajes y objetos, invitar a nuevos jugadores a la partida. Además las herramientas para los narradores permiten crear escenarios, personajes y objetos, y controlar el desarrollo de una aventura, tomando posesión de personajes durante el juego hablando y actuando a través de ellos.

En este modo, como es lógico, no está activa la Inteligencia Artificial puesto que es el narrador o el moderador el que dirige la partida, exceptuando, el estado de los personajes al volverse agresivos.

En el modo narrador el director de juego determina la dificultad de las acciones, que pueden ser de nivel fácil, nivel normal o nivel muy difícil. La pantalla principal de la interfaz en el modo de narrador tiene varias secciones distintas, como podemos ver numeradas en la Figura 6. Igualmente se puede observar que el narrador puede tener forma de cabeza humana y es capaz de moverse por el escenario igual que si fuera un personaje.

1) Opciones del narrador: cada uno de estos botones corresponde a una de las opciones principales que puede realizar el narrador, estas son: poseer, revivir, cambiar de modo (hablador, neutral o enemigo), borra objeto, cambiar a modo de personaje.

2) El recuadro de opciones: aquí se puede acceder en cada uno de los botones las pantallas de interfaz necesarias para manejar al personaje, igual que en el modo usuario. Estas opciones son el panel de personajes, el panel de inventario, el panel de búsquedas, el panel de disciplinas y el mapa de juego.

3) El panel de conversación: se muestra el texto con los nombres de los personajes y objetos del juego, así como las acciones que se realizan en el juego, de la misma manera que el modo usuario.

4) Panel de elementos de juego: aquí aparecen todos los componentes de los escenarios de la partida. Los botones que aparecen corresponden a panel de escena, panel de los objetos, panel de participantes, panel de control para controlar los objetivos que se realizan, activar y desactivar la IA y por último el panel de posición.

5) Otras opciones del narrador: aquí hay tres botones, el primero sirve para pausar la partida, el segundo configurar el narrador para que entre otras cosas podamos visualizar o no al narrador en el juego, que como se puede observar en la figura 6 puede aparecer en forma de una gran cabeza sin cuerpo. El tercero

sirve para avanzar un turno, ya que el como juego de rol incorpora esta característica, en cada turno los diferentes personajes podrán avanzar de manera independiente.

Figura 9. Imágenes del modo Storyteller de Vampire Redemption.

2.1.3. Videojuegos controlados por voz

En este apartado se citan algunos juegos que se han considerados relevantes debido al uso de reconocimiento de voz para controlar algunos aspectos del juego.

Odama

Odama (Nintendo, 2006) es un videojuego para la plataforma GameCube. La idea del juego es una combinación de un juego de estrategia con un juego de pinball.

El argumento del juego consiste en que el jugador es un general feudal japonés, comandante de unos ejércitos enfrentados a otras grandes familias feudales niponas. Cada tablero, como vemos en la figura 10, es un terreno tridimensional donde se desarrollan combates entre las tropas del jugador y las tropas enemigas. El jugador debe derribar a los enemigos a través de una enorme bola, dándole impulso con las pale-

tas del pinball, al igual que para destruir ciertas partes del escenario para evitar el avance de las tropas enemigas.

Figura 10. Imagen del entorno de juego de Odama.

Odama tiene una opción en la que el usuario puede enviarle un órdenes mediante voz usando un micrófono que va conectado a la consola GameCube. Este micrófono es un dispositivo propio de Nintendo llamado GameCube Mic. A través del micrófono el juego es capaz de reconocer un pequeño repertorio de órdenes con las que el usuario es capaz de controlar las tropas. A través de estás ordenes se puede ordenar a las tropas que avancen hacia algún punto, se replieguen, ataquen o que defiendan más.

Mario Party 7

Mario Party 7 (Nintendo, 2006) es un videojuego para la plataforma GameCube. El juego es del género de puzzles.

La mecánica del juego consiste en elegir a un personaje y llevarlo a través de un tablero con casilla para conseguir monedas y estrellas. Existen 88 minijuegos diferentes en los que el jugador deberá resolver un pequeño puzzle para obtener más puntuación. En 10 de estos juegos es preciso utilizar el micrófono de la consola para realizar reconocimiento de voz y cumplir el objetivo de los juegos.

RainBow Six Vegas 2

RainBow Six Vegas (Ubisoft, 2008) es un juego del género de estrategia y acción. El argumento del juego consiste en que el jugador es un comandante al mando de un comando de élite de tácticas militares. El

jugador estará al mando de al menos dos soldados para llevar a cabo misiones de rescate, infiltración y demás operaciones similares. El personaje se puede personalizar eligiendo desde ropas, armadura y todo tipo de características físicas. Estas características se reflejaran tanto en el modo de juego individual como en el juego online. A medida que el jugador progrese se desbloquearan nuevos equipos y habilidades.

El juego soporta reconocimiento de voz para poder realizar un conjunto de instrucciones como desplazar tropas a un sector, asegurar una zona o realizar ataques con las tropas.

Tom Clancy´s HAWX

Tom Clancy´s HAWX (Ubisoft, 2008/09) es un videojuego muy reciente estará disponible a la venta previsiblemente para el primer trimestre del año 2009. La información que aparece tanto el la web comercial del juego como otras informaciones de Internet apuntan a que puede ser un juego que revolucione el mundo de los videojuegos debido a la introducción reconocimiento de voz como parte de la interacción.

Tom Clancy´s HAWX es un juego enmarcado en los géneros de simulación, vuelo y acción. El argumento del juego se basa en un mundo futurista en el que el usuario debe hacer frente al terrorismo y otras amenazas mediante el combate aéreo. Este juego a parte de unos gráficos espectaculares presenta una serie de comandos por voz con el que se pueden controlar ciertos aspectos como dar órdenes a los miembros del equipo para que ataquen, retrocedan, lancen una bomba y acciones similares.

2.1.4. Otros videojuegos

Según cuenta el artículo *The Game Master* (Tychsen, 2005) los juegos Dungeon Siege y Warcraft III admiten la posibilidad de usar una interfaz para un director de juego, sin embargo la investigación realizada no ha conseguido contrastar esta información, ya que no se han encontrado interfaces comparables a los que presentan juegos como Neverwinter Nights o Vampire The Masquerade:Redemption.

Dungeon Siege (Microsoft, 2002) es un juego de rol que combina elementos de un juego de rol con la intensidad de un juego de acción. El argumento se basa en que el jugador debe comenzar como un humilde agricultor, para luego viajar por el mundo y adquirir nuevas habilidades, para enfrentarse contra un ejército del mal.

Warcraft III (Blizzard, 2002) es un juego de estrategia que relata la historia de un mundo épico medieval con personajes de la literatura de

Tolkien. Se distingue de sus predecesores por incorporar s gráficos 3D y la aparición de dos nuevas *razas*. El juego consiste básicamente en administrar los recursos disponibles (oro, madera y alimentos) para producir unidades militares y desarrollar un ejército que dirigir en contra de los oponentes hasta destruir todos sus edificios y trabajadores. El juego provee varias estrategias de ataque o defensa, y se ejecutan las tácticas de combate y producción a partir de cuatro tipos diferentes de civilizaciones: humanos, orcos, elfos nocturnos y muertos vivientes. Cada una de estas razas es comandada a su vez por tres clases de héroes que encabezan y apoyan significativamente las batallas ante sus adversarios.

2.2. Sistemas de narración interactiva

En este apartado se describirán algunos de los sistemas existentes de narración interactiva más relevantes: Façade, I-Storytelling, Zocalo e IDA. Los sistemás de narración interactiva pueden ser automáticos, aunque también puede intervenir un ser humano. La narración en la que intervienen personas constituyen un campo bastante amplio de investigación en los que existen muchos proyectos.

2.2.1. Façade

Façade (Mateas, 2005) es un sistema de libre distribución que realiza una aproximación basada en agentes software con un razonador centralizado, aunque no incluye el código fuente. Ha sido programado en Java y JESS es el motor de inferencia que se encarga de ejecutar todas las reglas de comportamiento de los personajes y del director automático. Además posee un director automático implementado como un planificador que selecciona, ordena y ejecuta eventos que permiten que la historia avance.

El ejemplo que viene en la distribución del sistema consiste en una simulación sobre un escenario en tres dimensiones, en el que se encuentran dos personajes, una mujer llamada Grace y un hombre llamado Trip, los cuales están casados. El escenario es una casa donde el usuario puede moverse por el salón y las diferentes habitaciones.

Figura 11. Muestra donde se pueden apreciar personajes virtuales en 3D conversando en el sistema Façade.

Los comandos que un usuario realizar son básicamente tres: desplazarse por el entorno con las teclas de los cursores, hablar con los personajes introduciendo frases mediante el teclado y coger o usar objetos haciendo clic con el ratón. Las respuestas de las conversaciones con los personajes se producen mediante voz y opcionalmente con texto por pantalla mediante subtítulos.

A lo largo de la simulación el usuario puede interactuar con los dos personajes, pero además, los personajes pueden también realizar acciones entre ellos, como discutir o gesticular. Lo cierto es que a lo largo del juego se producen situaciones muy divertidas y sorprendentes en las que los personajes se ríen con tus comentarios o se enfadan y tienen discusiones entre ellos, llegando incluso hasta una fuerte discusión, en la que la pareja se separa.

Aunque el entorno gráfico no alcanza el nivel de calidad que se podría encontrar en un videojuego comercial, con elaborados personajes en tres dimensiones, los personajes a pesar de ser en dos dimensiones tienen una calidad aceptable, como se puede apreciar en los rasgos de la cara de la cara de la mujer en la figura 11, que tiene labios, nariz, ojos y cejas, lo cual permite expresar emociones. Estos personajes consiguen un alto grado de expresividad que puede superar al de algunos de estos videojuegos, por tanto, cumple el objetivo de hacer percibir al usuario una serie de rasgos emotivos en los personajes que imitan al comportamiento humano.

Desde el punto de vista de un desarrollador uno de los inconvenientes del sistema es que el autor debe aprender a programar en ABL, un len-

guaje específico para describir el comportamiento reactivo de los personajes. La ventaja es que este lenguaje es muy potente ya que permite expresar acciones concurrentes de varios personajes de manera que estos pueden coordinarse sin necesidad de monitorizarse mutuamente de forma permanente.

2.2.2. I-Storytelling

I-Storytelling (Cavazza, 2004) es un sistema de narración digital interactiva en el cual el autor edita el guión multiforme otorgando objetivos dramáticos a un reparto de actores virtuales cuya interacción en un entorno virtual, junto con la participación de un usuario, hace emerger nuevas historias. Estos actores persiguen sus objetivos mediante planes que elabora un planificador externo y centralizado que se basa en Redes Jerárquicas de Tareas o HTN (Hierarchical Task Networks).

El interés de los desarrolladores de I-Storytelling se centra en la recreación de historias con una línea argumental bien definida, en las cuales se puedan dar numerosas variantes como resultado de la interacción de personajes y de la intervención del usuario.

En la figura 12 se puede ver de manera intuitiva como se desarrolla la interacción del usuario en el sistema. Los receptores del sistema capturan sus movimientos y sus voces, produciéndose la interacción en el entorno virtual.

Si el usuario ha de desempeñar el papel de un personaje en la narración interactiva, tendrá que comunicarse con personajes virtuales reflejando cierta actuación. La actuación implica una serie de actitudes y gestos del cuerpo que son importantes para la comunicación virtual con los actores. Al mismo tiempo, la comunicación oral es un aspecto esencial de una narración interactiva realista. Tratar una comunicación de forma multimodal se enfrenta a varias dificultades en términos de tiempo real, rendimiento, cobertura y precisión.

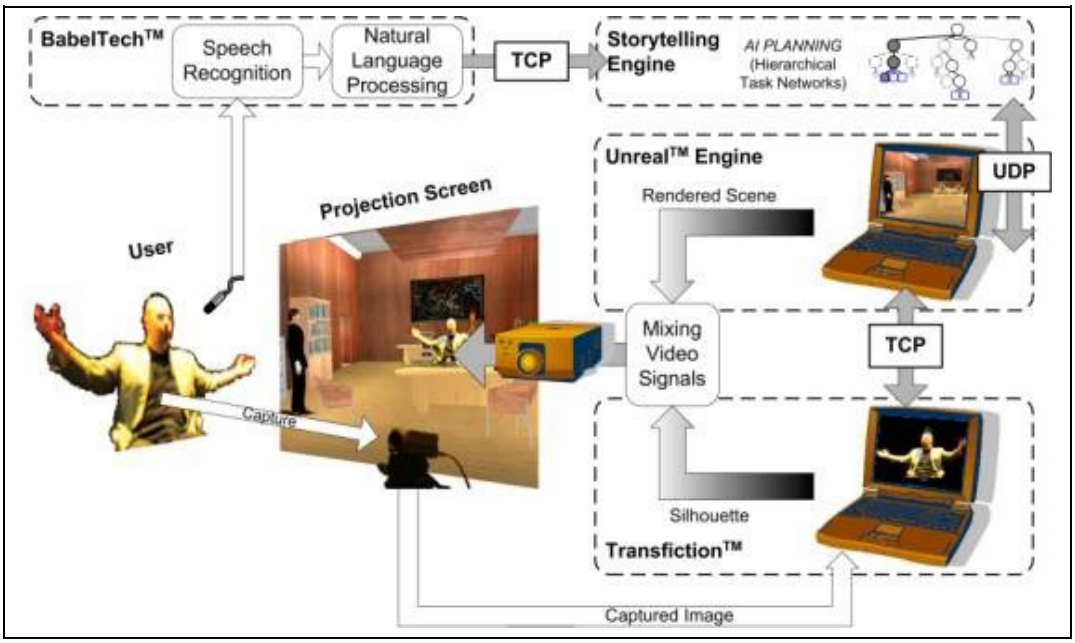


Figura 12. Arquitectura del sistema I-Storytelling obtenida de la web del grupo de Narración interactiva de la Universidad de Teesside.

I-Storytelling posee una interfaz vocal (Mead, 2003) compuesta por dos capas: una capa de reconocimiento del habla y otra capa de procesamiento del lenguaje natural. La capa de reconocimiento del habla se ha desarrollado utilizando un framework de desarrollo llamado EAR SDK, que proporciona un API escrito en C/C++, así como otras herramientas para el generación de voz sobre un sublenguaje reconocido.

I-Storytelling debe proporcionar un conjunto de declaraciones de cada posible situación narrativa, pero además también debe proporcionar la cantidad adecuada de expresiones semánticamente equivalentes para permitir al usuario hacer uso de la palabra con relativa libertad, sin necesidad de aprender un sublenguaje.

2.2.3. Zocalo

Zocalo (Young, 2008) es una arquitectura orientada a servicios web para controlar sistemas de narración interactiva. Este proyecto considera seriamente aspectos como escalabilidad, seguridad e interoperabilidad entre aplicaciones de narración digital e interactiva y está desarrollado en código abierto con tecnologías .NET (Microsoft).

Los desarrolladores de cada aplicación han de desarrollar clientes que se comuniquen con el gestor de ejecución de Zocalo. Actualmente se ha desarrollado un cliente que utiliza el motor Unreal Tournament 2004 y hay otro para Half-Life 2 en preparación, aunque ninguno de ellos está disponible al público.

Su arquitectura está organizada en tres capas: una capa de servicios

web, otra capa con el gestor de ejecución y una tercera capa con el entorno de ejecución correspondiente donde se representa cada paso de la ejecución y se gestiona el interfaz y la interacción con los usuarios.

El gestor de ejecución recibe los operadores conceptuales que devuelven los servicios web tras una petición HTTP y los transforma en operadores concretos para su envío y posterior ejecución en un entorno de ejecución concreto.

Para comunicar eficientemente cada componente de la red se necesitan extensiones para los distintos componentes de los diferentes sistemas operativos. Estas extensiones se pueden implementar mediante librerías dinámicas (DLL) o conexiones TCP mediante sockets. Los esquemas XML, para el envío y recepción de mensajes en formato XML entre los diferentes servicios están bien documentados, por lo que pueden ser reutilizados para desarrollar futuras extensiones a la arquitectura.

2.2.4. IDA

Interactive Drama Architecture (IDA) es una propuesta de un sistema de narración interactiva (Magerko, 2003) que incluye un director automático implementado como un sistema basado en reglas. Esta arquitectura usa un director de narración interactiva centralizado junto con personajes semiautónomos que colaboran en orden para desarrollar un guión expresivo, variable e interesante. La ventaja de utilizar esta arquitectura semi-distribuida, en la cual solo se da información parcial a un personaje particular en una situación concreta de la historia, es mucho menos compleja que una arquitectura con un director completamente distribuido.

El sistema esta integrado con un entorno creado con el motor Unreal usando algo similar a un “adaptador de software” para intercambiar mensajes con el entorno virtual. Hay algunos detalles técnicos documentados a cerca de la implementación de este adaptador, pero los autores explican que los eventos atómicos que ocurren en el modelo interno del mundo del Unreal son generalizados por predicados simples como MOVE-TO-ROOM(x, y), más fácilmente interpretables por el director.

El controlador de la historia envía mensajes que son comandos básicos para personajes no jugadores o NPCs (Non-Player Character) disponibles en el entorno por el jugador para interactuar con él, mediante mensajes como “explora el entorno”, “coge este objeto”, “vete a esta habitación”, “dile algo al jugador” y otros comandos similares.
IDA está compuesto por un jugador, un autor humano, un director y

un agente actor, y un mundo virtual donde tiene lugar la historia. El autor define un lugar para la historia usando la representación de nuestra historia, la cual es incluida en nuestro mecanismo de narración, el agente director. Los actores son agentes inteligentes semiautónomos. Ellos tienen la capacidad de ejecutar sus propios objetivos basados en el comportamiento o para basar su comportamiento en comandos del director. Las decisiones del director para dirigir la historia en cualquier momento dado dependen de las acciones del jugador, del guión especificado por el autor, el estado del mundo y las proyecciones del comportamiento futuro del jugador.

2.3. Interfaces vocales

En este apartado se analizarán los interfaces vocales más relevantes actualmente. Al referirnos a interfaces vocales nos estamos refiriendo a sistemas que realizan reconocimiento de voz para no sólo reconocer la voz, sino las órdenes que da el usuario para controlar una cierta aplicación. En la aplicación que se ha desarrollado, detallada en el capítulo 4, los cambios son solicitados por un director de narración interactiva para que se produzcan en un mundo virtual.

El campo de estudio de la voz es muy amplio, y se divide por un lado en la síntesis de voz y por otro en el análisis de voz. En el campo de la síntesis de voz se puede encontrar abundante información en Internet además de en otras publicaciones. Por el contrario en campo del análisis de voz hay menos información disponible y hay aspectos que no están totalmente resueltos o que pueden ser mejorados en el futuro.

Existen varios motores de software libre para el reconocimiento del discurso (speech recognition). En este trabajo se profundizará en Sphinx, el cual se ha utilizado en el trabajo. También se detallarán en menor grado otros sistemas.

2.3.1. Motores basados en modelos ocultos de Markov

En este apartado se muestran motores que se basan exclusivamente en modelos ocultos de Markov (Rabiner, 1989).

Un modelo oculto de Markov o HMM (Hidden Markov Model) es un modelo estadístico que suele aplicarse al reconocimiento de patrones y formas temporales como escritura manual, gestos o voz. El modelo está compuesto por un conjunto finito de estados, los cuales están asociados a una distribución de probabilidad. Las transiciones entre estados están dirigidas por un conjunto de probabilidades llamadas probabilidades de transición. En un estado puede ser generada una salida u observación de acuerdo a la distribución de probabilidad

asociada.

Los tres problemas fundamentales a resolver en el diseño de un modelo oculto de Markov son la evaluación de la probabilidad de una secuencia de observaciones, la determinación de la mejor secuencia de estados del modelo y el ajuste de los parámetros del modelo que mejor se ajusten a los valores observados.

Hidden Markov Model Toolkit

HTK (Hidden Markov Model Toolkit) es una herramienta para el manejo de modelos ocultos de Markov. Se destina principalmente para el reconocimiento de voz, pero se ha utilizado en muchas otras aplicaciones de reconocimiento de patrones que emplean modelos ocultos de Markov. HTK es propiedad de Microsoft, pero dispone de una licencia para el uso libre de la herramienta con fines no comerciales.

Hidden ISIP Toolkit

Este software consiste en es un conjunto funcional de herramientas para el desarrollo de aplicaciones de reconocimiento de voz. Es un proyecto del Instituto de Señales y Procesamiento de Información y la Universidad Estatal de Mississippi. La guía incluye un front-end, un back-end, un decodificador, y un módulo de capacitación. El front-end se refiere a la parte del sistema que convierte la entrada del texto en una representación simbólico-fonética y el back-end convierte la representación fonética y simbólica en el sonido. El motor de reconocimiento está especialmente pensado para su uso por desarrolladores de software.

The EMU Speech Database System

EMU es una colección de herramientas de software para la creación, manipulación y análisis de bases de datos de expresión. En el núcleo de la EMU está un motor de búsqueda de base de datos que permite al investigador encontrar diferentes segmentos de discurso basado en la secuencia y estructura jerárquica de las declaraciones en las que se producen. EMU incluye una “etiquetadora interactiva” que puede mostrar espectrogramas y otras formas de onda del discurso, y que permite la creación de etiquetas de manera jerárquica o secuencial, de un discurso. El nombre de EMU se debe a que es una reimplementación de un sistema anterior llamado MU+ desarrollado por la Universidad Macquarie.

2.3.1. Motores basados en N-gramas

En este apartado se muestran motores que se realizan un reconocimiento de palabras basado en N-gramas. Aunque también usan modelos ocultos de Markov, no lo hacen de forma exclusiva y por eso no se han incluido en la sección anterior. En la actualidad, los modelos ocultos de Markov, en conjunción con los n-gramas se estiman como los componentes esenciales para los sistemas de reconocimiento del habla.

Los *n*-gramas se emplean en varias áreas del procesamiento estadístico del lenguaje natural. Un n-grama es un modelo en el que la probabilidad de pronunciar una palabra sólo depende de las n-1 palabras pronunciadas. Un n-grama también es un modelo de estados finitos, cuyos estados son las secuencias de n-1 palabras y las transiciones entre estados vienen dadas por las secuencias de n palabras. Un *n*-grama de tamaño dos se denomina “bigrama” o “digrama”; de tamaño 3, “trigrama”; de tamaño 4 o más se denomina "*n*-grama" o "modelo de Markov de orden ($n - 1$)".

El éxito de los n-gramas en el reconocimiento del habla se debe a la existencia de potentes algoritmos para estimar los sucesos (n-gramas) que no han ocurrido en el conjunto de entrenamiento, pero que es factible que aparezcan en un conjunto de test.

Julius

Julius es un motor de reconocimiento de voz continua con gran vocabulario de alto rendimiento (Julius, 2008). El reconocimiento de las palabras esta basado en N-gramas y en modelos ocultos de Markov dependientes del contexto. Puede realizar decodificación sobre tareas de dictado en la mayoría de los PC actuales prácticamente en tiempo real.

La aplicación usa técnicas de búsqueda eficientes para recocer palabras, pero además de buscar la eficiencia, la aplicación está modularizada cuidadosamente para ser independiente de las estructuras de los modelos. Varios de modelos ocultos de Markov son soportados como “trifonos” de estado compartido y también se permite vinculado y mezcla de modelos, con cualquier número de mezclas, estados o fonemas. Se adoptan formatos estándar para ser compatible con otras herramientas de modelado como CMU-Cam SLM toolkit.

La principal plataforma es Linux y estaciones de trabajo Unix, aunque también está disponible para Windows. La versión más reciente se desarrolla sobre Linux y Windows (cygwin / MinGW), y también tiene versión Microsoft SAPI. Julius se distribuye con licencia abierta junto

con el código fuente.

Julius ha sido desarrollado como un software de investigación para el LVCSR japonés desde 1997, y el trabajo fue continuado por IPA Japanese dictation toolkit project (1997-2000), Continuous Speech Recognition Consortium, Japan (CSRC) (2000-2003) y actualmente por Interactive Speech Technology Consortium (ISTC).

Sphinx

Sphinx es un armazón para el reconocimiento de voz bastante flexible y puede usarse como generador o sintetizador de voz. Además tiene licencia de código abierto, por lo cual está disponible para toda la comunidad de desarrolladores.

Este armazón ha sido desarrollado entre el grupo Sphinx (Walker, 2008) de la Universidad Carnegie Mellon, los Laboratorios de Sun Microsystems y los Laboratorios de Investigación de Mitsubishi Electric (MERL), y Hewlett Packard (HP), con contribuciones de la Universidad de California de Santa Cruz (UCSC) y el Instituto Tecnológico de Massachusetts (MIT).

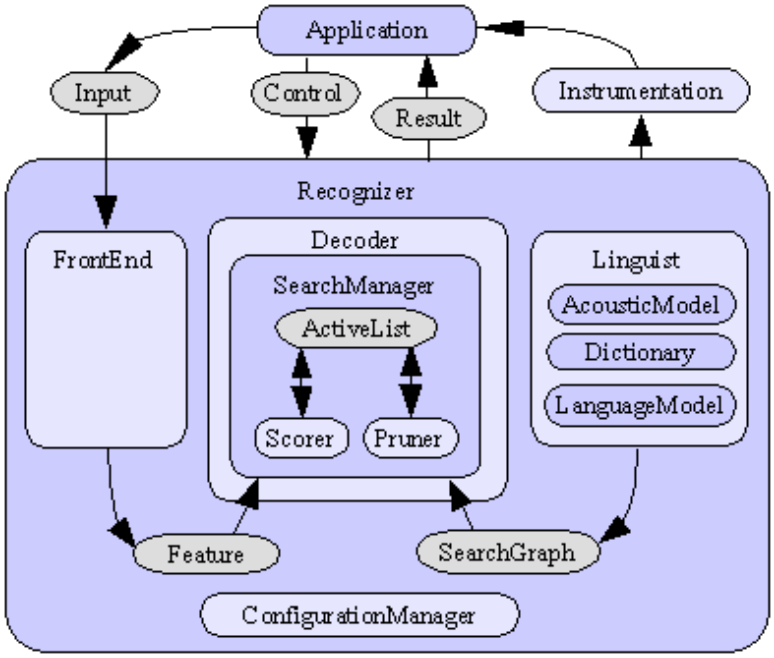


Figura 13. Diagrama de arquitectura de Sphinx4 obtenida de la documentación contenida en la distribución sphinx4-1.0beta.

Sphinx ha sufrido una evolución desde que comenzó el proyecto y cuenta con cuatro versiones distintas. A lo largo del trabajo siempre nos referiremos a Sphinx-4. La arquitectura de Sphinx es bastante compleja como se puede apreciar en la figura 13 y consta de muchos módulos que se utilizan para las diferentes fases en las que se

descodifica la señal.

Actualmente hay dos distribuciones de Sphinx, una con las librerías necesarias para usar el reconocedor de voz en aplicaciones propias, de la misma manera que viene expuesta en algunos ejemplos, y otra que incluye además todo el código fuente para desarrolladores que deseen realizar modificaciones.

Para utilizar Sphinx en un proyecto Java en el cual realicemos reconocimiento o generación de voz necesitamos un archivo de configuración XML donde se especificarán las propiedades de configuración de los diferentes componentes de Sphinx, los cuales son el reconocedor de palabras, el decodificador, la información lingüística, la gramática, el diccionario, el modelo acústico, la unidad manager, el front-end, los monitores y la miscelánea.

Además para que podamos usar todas las clases que nos ofrece debemos importar las librerías jsapi.jar, sphinx4.jar. Por otro lado debemos importar otras librerías que contienen un modelo acústico. Este modelo acústico debe ser definido el archivo de configuración XML mencionado anteriormente. Sphinx nos ofrece tres librerías jar con tres modelos acústicos diferentes:

- TIDIGITS_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar.
- WSJ_8gau_13dCep_8kHz_31mel_200Hz_3500Hz.jar
- WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar

La primera corresponde al modelo TI Digits, que reconoce 11 palabras distintas, los números del 0 al 9 y la palabra “oh!” (en inglés sinónimo de cero).

Los dos últimos son diferentes versiones del modelo más típico, el modelo Wall Street Journal (WSJ). WSJ tiene un rango de longitud de de onda suficiente amplio para su uso en teléfonos y asegurar que el reconocimiento de voz funcione bien.

Si quisiéramos entrenar nuestro propio modelo acústico, se puede hacer mediante una herramienta del grupo Sphinx llamada *SphinxTrain* que está disponible en su web. Con esta herramienta podemos crear nuestro propia librería jar con un modelo acústico creado por nosotros mismos. Para usar está herramienta está implementada en C++ y para utilizarla debemos disponer de Visual C++ 6.0.

Además del modelo acústico, el módulo reconocedor de Sphinx debe de tener una serie de reglas gramaticales definidas por el usuario. Estas reglas definen como se produce el lenguaje que estamos tratando de reconocer. Por ejemplo si estamos reconociendo el lenguaje de las

cifras (números naturales), debemos indicarle que una cifra puede producirse como la concatenación de cualquier número entre cero y nueve.

Este modelo del lenguaje se realiza a través de gramáticas, que serán definidas en un archivo del proyecto con extensión .gram. Este archivo además deberá de referenciarse también en el archivo de configuración XML. Para definir estas gramáticas, la versión actual de Sphinx usa una librería llamada Java Speech Grammar Format (JSGF). Esta librería pertenece a una plataforma independiente, que provee una representación textual de gramáticas para su uso en el reconocimiento del habla. JSGF adopta el estilo y las convenciones del lenguaje de programación Java, además de la utilización de la gramática tradicional notaciones. Sus principales características pueden encontrarse en la documentación de Sphinx o en la web de JSGF.

Lo cierto es que para problemas sencillos basta con definir gramáticas simples como en el ejemplo de las cifras, pero para manejar otros problemas más complejos, es preciso el uso de n-gramas. Utilizando gramáticas de n-gramas, deberemos configurar más de un reconocedor. En principio, deberíamos configurar una gramática por cada reconocedor.

2.3.3. Modelos acústicos

En este apartado se describen dos entrenadores para crear modelos acústicos para poder ser usados en otros sistemas, uno es SphinxTrain y el otro es el VoxForge.

SphinxTrain

La Universidad Carnegie Mellon proporciona un entrenador para crear un modelo acústico que puede ser utilizado para producir un modelo oculto de Markov continuo o semicontínuo. Produce modelos compatibles con PocketSphinx, Sphinx-2, Sphinx-3, y Sphinx-4.

Con el fin de utilizar los modelos de entrenados con SphinxTrain se empaqueta el modelo en un archivo JAR. La ventaja de tener en un archivo JAR es que puede ser fácilmente incluido en el classpath y ser referenciado en el archivo de configuración para poder ser utilizado en una aplicación basada en Sphinx-4.

VoxForge

VoxForge es un corpus del discurso y un repositorio del modelo acústico para motores de reconocimiento de voz (VoxForge, 2008). Se creó para recoger discursos transcritos para crear un corpus de discurso li-

bre GPL para su uso motores de reconocimiento de voz de código abierto. El archivo de audio de discurso será compilado en modelo acústico para su uso en motores de reconocimiento de voz como Julius, ISIP, Sphinx y HTK. VoxForge ha empezado recientemente a utilizar LibriVox como una buena fuente de datos de audio.

2.3.4. Motores de software propietario

Todos los sistemas relacionados con el reconocimiento de voces mostradas anteriormente son de software libre. En el ámbito comercial existen otras herramientas que no son de libre distribución y requieren licencia. Algunos de estos son los siguientes:

CSLU Toolkit

CSLU es un Instituto Universitario de Ciencia y Tecnología de Oregón y un Centro de investigación que se centra en las tecnologías del lenguaje hablado. El CSLU Toolkit se creó para proporcionar el marco básico y las herramientas para que los usuarios puedan crear, investigar y utilizar los sistemas interactivos de idiomas. Estos sistemas incorporan de vanguardia reconocimiento de voz, comprensión del lenguaje natural, síntesis de voz y tecnologías de animación facial. CSLU Toolkit proporciona un entorno flexible y potente para la creación de sistemas interactivos de idiomas que utilicen estas tecnologías, y para llevar a cabo la investigación al respecto.

La gente que está usando actualmente el kit de herramientas lo está haciendo para crear proyectos escolares para los profesores universitarios que estén experimentando con tecnologías del lenguaje básico y los sistemas de diálogo hablado. La aplicación esta desarrollada en lenguaje C, también usa el lenguaje Tcl/Tk para realizar scripting.

Dragon NaturallySpeaking

Dragon NaturallySpeaking es un software de la empresa Nuance Communications. La última version es Dragon NaturallySpeaking Professional 9, la cual permite a usuarios profesionales crear documentos y mensajes de correo electrónico, completar formularios y optimizar tareas de flujo de trabajo con sólo usar la voz.

Dragon NaturallySpeaking permite una velocidad aproximadamente tres veces superior a la que se puede llegar con el teclado y hasta un 99% más de precisión. La aplicación permite realizar un dictado de manera que las nuevas palabras aparezcan instantáneamente en el paquete de Microsoft Office, Internet Explorer, Corel WordPerfect, Lotus Notes y prácticamente todas las demás aplicaciones para Windows.

Existe también una aplicación llamada MacSpeech Dictate que usa el motor de Dragon NaturallySpeaking. MacSpeech Dictate es un programa de reconocimiento de voz para sistemas Mac. Trabaja con las aplicaciones Microsoft Word, Adobe Photoshop, QuarkXPress y varias otras.

Además Dragon NaturallySpeaking permite crear comandos de voz que ejecuten las tareas complejas del sistema operativo y también incluye herramientas opcionales para instalaciones en red, como compatibilidad con clientes ligeros Citrix.

IBM ViaVoice

ViaVoice es un software de reconocimiento de voz comercializado por IBM disponible en Linux y Mac OS X. La funcionalidad es similar a Dragon NaturallySpeaking. Proporciona capacidad de reconocimiento automático de voz y paso de texto a voz para dispositivos móviles, incluyendo sistemas telemáticos de automoción y sistemas de manos libres para teléfonos. Ayuda a reducir al mínimo el tiempo y las aptitudes necesarias para el desarrollo de aplicaciones de voz avanzadas para dispositivos y sistemas remotos. Reconoce listas de vocabulario con más de doscientas mil palabras, habladas en tiempo real y en una amplia gama de idiomas. Proporciona un kit para desarrolladores intuitivo y fácil de usar, desarrollado con tecnología Eclipse.

Microsoft Speech API

SAPI son las siglas de Speech API de Microsoft, es decir, es una interfaz de reconocimiento de habla y de traducción de texto a voz para la programación de aplicaciones. Permite incluir en cualquier aplicación basada en la Win32 API (Intel Win32s, Windows NT, Windows 95/98, MIPS Windows NT, DEC Alpha Windows NT, Power PC Windows NT) reconocimiento de habla y lectura hablada de texto. Actúa como una capa de abstracción entre las aplicaciones y los motores sobre tecnología del habla. También puede actuar como interfaz entre los motores y el hardware de reconocimiento del habla y de lectura hablada de texto.

La SAPI proporciona una interfaz relativamente cómoda para el reconocimiento de comandos, el dictado, el texto hablado y la telefonía.

El reconocimiento de comandos permite al usuario dar órdenes simples a la aplicación o al sistema operativo, el dictado permite reconocer textos arbitrarios para usarlos como entrada en procesadores de texto, el texto hablado permite transformar cualquier

texto escrito en texto hablado, y la telefonía permite realizar una comunicación de un usuario con una aplicación por medio del teléfono.

Philips SpeechMagic

SpeechMagic es una tecnología de reconocimiento de discursos para usuarios profesionales, que se integra en diversos sistemas de información y aplicaciones de dictado. SpeechMagic permite dictado, reconocimiento de voz y corrección en cualquier lugar dentro de un LAN, WAN, a través de Internet o en un servidor. El diseño de SpeechMagic responde a las necesidades específicas de grado industrial de creación de documentos. Las soluciones que SpeechMagic proporciona, aumentan la eficacia de los procesos de creación de documentos en un cincuenta por ciento de media. Este resultado permite crear informes a mayor velocidad y con menor coste. Con más de siete mil instalaciones en el mundo entero, SpeechMagic es líder en el mercado de reconocimiento de discursos profesionales.

2.4. Tecnologías de procesamiento de documentos XML

En este apartado se revisan las diferentes tecnologías y herramientas existentes para el procesamiento de documentos XML.

2.4.1. Data Object Model

Data Object Model o DOM es una API creada por el World Wide Web Consortium (W3C) que permite tratar documentos XML. DOM procesa documentos XML y crea una estructura de árbol, normalmente mantenida en memoria, como se puede observar en la figura 14. Una vez construido el árbol DOM las aplicaciones pueden procesarlo fácilmente: añadir y eliminar elementos, reposicionarlos, cambiar su contenido y escribir el árbol en un fichero XML.

La API de DOM es fácil de usar y bastante flexible. El consumo de memoria y la velocidad de procesamiento, debido a la creación del modelo del documento en memoria, dependen del tamaño del documento. En el caso de estar tratando documentos de gran tamaño se deberá crear un árbol DOM que ocupará gran cantidad de memoria, lo que influirá negativamente en el rendimiento global de la aplicación. Por tanto, la construcción de un árbol en memoria es útil para documentos pequeños que necesitan ser procesados en su práctica totalidad, pero con documentos más grandes puede no ser la mejor solución.

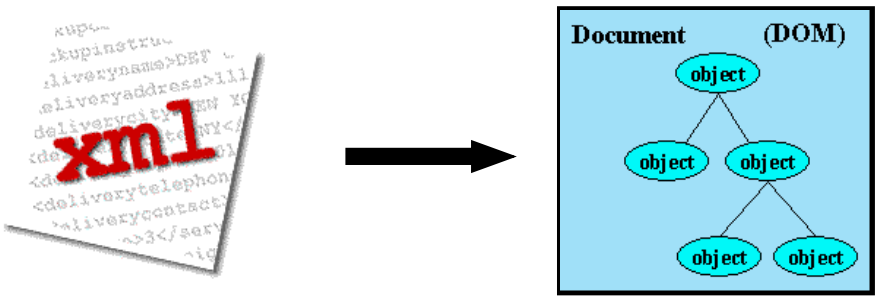


Figura 14. Ilustración de la idea básica de DOM: el documento XML se transforma a una estructura de árbol.

2.4.2. Simple API for XML

Simple API for XML o SAX es una API de bajo nivel basada en un mecanismo de eventos para parsear documentos XML. SAX reporta eventos de parsing tales como el comienzo y el fin de documento, el comienzo y el fin de elementos, y datos de tipo carácter, que hacen saltar métodos "callback" en una aplicación que implementa un manejador de eventos.

SAX realiza un acceso secuencial al documento sin crear un modelo de objetos del mismo en memoria, lo cual convierte a esta API en un método de procesamiento de documentos XML con bajo coste computacional. Sin embargo, la falta de una representación explícita del árbol del documento es un importante inconveniente en el caso de aplicaciones que necesitan mantener información de estado. Es decir, determinadas aplicaciones que necesiten conocer los elementos que se hayan procesado anteriormente para decir qué elemento procesar actualmente, pueden requerir una implementación compleja.

Generalmente esto se puede conseguir a través de la programación de máquinas de estados que modelen el comportamiento del lenguaje. Por este motivo, la programación de un manejador de eventos SAX puede ser una tarea tediosa y propensa a errores.

Por tanto, el enfoque orientado a eventos es útil para documentos grandes en los que sólo hay que procesar una pequeña parte de la información. Sin embargo, el mayor inconveniente resulta la complejidad estructural del lenguaje XML. Mientras éste sea más complejo, la complejidad del código programado será mayor.

2.4.3. XML Data Binding

Data Binding es un proceso de mapeo de los componentes de un determinado formato de datos a su representación específica para un lenguaje de programación concreto, que se ajusta al significado propio de los datos. Esto permite modelar la estructura lógica de los datos sin seguir forzosamente la estructura impuesta por el formato en el que se habían guardado.

Más específicamente, el Data Binding entre XML y JAVA se refiere al mapeo de los componentes estructurales de XML, como elementos y atributos, al modelo de objetos de JAVA. De esta manera se preserva la jerarquía lógica de los componentes y se expone su significado real representándolos en el formato nativo de JAVA. En la práctica esto conduce a interactuar con objetos, tipos primitivos y estructuras nativas de JAVA en lugar de trabajar con modelos de eventos o árboles genéricos resultantes del parsing.

El XML Data Binding presenta algunas ventajas en comparación con DOM y SAX, como son la provisión de un mecanismo eficiente de validación de datos XML, facilidad para crear un documento XML a partir de objetos JAVA y viceversa y legibilidad del código generado. Esto último implica un mantenimiento más fácil. Por tanto XML Data Binding permite una reducción del esfuerzo dedicado al desarrollo.

El Data Binding entre XML y JAVA se realiza por medio de un compilador de esquemas, en el cual se generan directamente clases JAVA específicas para un esquema XML origen. Dichas clases incluyen código necesario para la validación y comprobación de errores, junto con un completo conjunto de métodos accesores y mutadores para asegurar la consistencia de los documentos XML con el esquema XML origen. Estos objetos JAVA podrán parsear documentos XML para construir un modelo de objetos interno y validar el contenido de los datos.

Partiendo de estas clases se podrán instanciar objetos JAVA que podrán convertir un objeto a un flujo de datos y viceversa. Estos procesos se conocen como “marshalling” y “unmarshalling”.

Existen herramientas de Data Binding, en las que el proceso ya está automatizado para su uso por desarrolladores. Las herramientas facilitan el mantenimiento del código cuando se trabaja con esquemas XML de tamaño considerable gestionando el Data Binding de manera automática. El API oficial de Sun para binding de archivos XML es JAXB (JAVA Architecture for XML Binding). Existen otras herramientas como: Castor, XMLBeans, Zeus, Breeze y Xgen.

2.5. Herramientas de procesamiento del lenguaje

Un procesador de lenguaje es a grandes rasgos, es un sistema que procesa un lenguaje fuente y lo transforma en otro lenguaje objeto. Las herramientas de construcción de procesadores son principalmente de dos tipos: generadores de analizadores léxicos, y generadores de analizadores sintácticos. En este trabajo nos centramos en JavaCC, que se ha utilizado en la implementación e incluye ambos tipos de generadores. Aunque existen otras herramientas como LEX y JFlex, para el analizador léxico, o YACC y Cup para el analizador sintáctico.

2.5.1. Java Compiler Compiler

Java Compiler Compiler o JavaCC es una herramienta de Sun Microsystems para la generación de traductores. JavaCC lee la especificación de una gramática (en realidad, de un esquema de traducción) y la convierte en un programa Java que puede reconocer frases generadas por la gramática. JavaCC integra en una misma herramienta al analizador lexicográfico y al sintáctico, y el código que genera es independiente de cualquier biblioteca externa, lo que le confiere una interesante propiedad de independencia respecto al entorno.

En la implementación se usa el método recursivo descendente, generándose una clase de Java la cual contendrá un método por cada no-terminal definido en la gramática. Las acciones semánticas pueden escribirse mediante código Java "encerrado" entre llaves dentro de la especificación de las producciones de la gramática. No hay necesidad de separar la especificación del analizador lexicográfico del sintáctico, aunque de estar presentes en el fichero, puede generarse código sólo para una de ellas si así se desea. Las características más importantes de JavaCC son las siguientes:

- Genera analizadores descendentes, permitiendo el uso de gramáticas de propósito general y la utilización de atributos tanto sintetizados como heredados durante la construcción del árbol sintáctico.

- Las especificaciones léxicas y sintácticas se ubican en un solo archivo. De esta manera la gramática puede ser leída y mantenida más fácilmente. No obstante, cuando se introducen acciones semánticas, se recomienda el uso de ciertos comentarios para mejorar la legibilidad.

- Adopta una notación BNF propia mediante la utilización de símbolos propios de expresiones regulares, tales como (A)*, (A)+.

- Admite el uso de estados léxicos y la capacidad de agregar acciones léxicas incluyendo un bloque de código Java tras el identificador de un token.

- Incorpora distintos tipos de tokens: normales (TOKEN), especiales (SPECIAL_TOKEN), espaciadores (SKIP) y de continuación (MORE). Ello permite trabajar con especificaciones más claras, a la vez que permite una mejor gestión de los mensajes de error y advertencia por parte de JavaCC en tiempo de metacompilación.

- La especificación léxica puede definir tokens de manera tal que no se diferencien las mayúsculas de las minúsculas bien a nivel global, bien en un patrón concreto.

- Permite depurar tanto el analizador sintáctico generado como el lexicográfico, mediante las opciones DEBUG_PARSER, DEBUG_LOOKAHEAD, y DEBUG_TOKEN_MANAGER.

- Incluye la herramienta JJTree, un preprocesador para el desarrollo de árboles con características muy potentes.

- Incluye la herramienta JJDoc que convierte los archivos de la gramática en archivos de documentación.

- Posee una buena gestión de errores. De entre los generadores de parsers, JavaCC se halla entre los que tienen mejor gestión de errores, ya que son capaces de localizar exactamente la ubicación de los errores, proporcionando una información diagnóstica completa.

- Permite entradas codificadas en Unicode, de forma que las especificaciones léxicas también pueden incluir cualquier carácter Unicode. Esto facilita la descripción de los elementos del lenguaje, tales como los identificadores Java que permiten ciertos caracteres Unicode que no son ASCII.

- Tiene una amplia comunidad de usuarios. JavaCC es quizá el generador de parsers usado con aplicaciones Java más popular. Es altamente eficiente, lo que lo hace apto para entornos profesionales y lo ha convertido en uno de los metacompiladores más extendidos.

- Ofrece muchas opciones diferentes para personalizar su comportamiento y el comportamiento de los parsers generados.

- Funciona en una plataforma completamente Java. Ello permite que pueda ser usado en diferentes máquinas sin problemas de

portabilidad.

Cuando se genera el procesador con JavaCC se crean varias clases Java con las diferentes partes del procesador. Partiendo de un fichero con extensión .jj se producen tres ficheros dependientes y cuatro que son siempre idénticos.

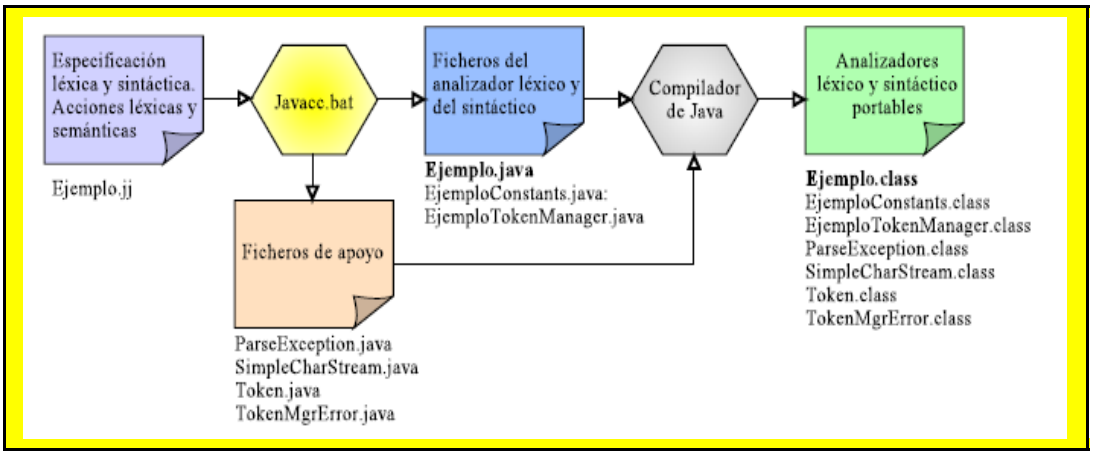


Figura 15. Diagrama de flujo de JavaCC. Muestra como generar un parser Java a partir de un archivo de especificación con extensión jj.

Las clases dependientes tendrán nombres creados a partir de un nombre que se dé en la especificación; si dicho nombre es XXX, los archivos se llamarán XXX.java, XXXTokenManager.java y XXXConstants.java. En la figura 15 vemos cómo se produce todo este proceso a partir de la especificación contenida en el archivo Ejemplo.jj.

Las principales clases generadas son la clase principal del parser, la clase TokenManager y la clase Constants, que en la figura 13 aparecen como Ejemplo.java, EjemploTokenManager.java y EjemploConstants.java respectivamente. El resto de clases de apoyo son ParseException.java, SimpleCharStream.java, Token.java y TokenMgrError.java.

La clase principal del parser implementa el traductor en sí, la clase TokenManager es la que implementa el analizador léxico y la clase Constants tiene una representación de todos los tokens que posee el lenguaje.

3. Una interfaz vocal para la dirección de narración interactiva

Después de tener información sobre la utilidad de una interfaz vocal, como se verá en el punto 3.1, se realizó el diseño de la arquitectura del sistema. El sistema está compuesto por varios módulos, los cuales realizan una función distinta en la aplicación. En primer lugar tenemos un módulo para procesar el lenguaje de comunicación con el entorno virtual que se ha definido, el lenguaje RCEI. El módulo implementa un parser que es capaz de procesar mensajes RCEI y también es capaz de componerlos, en el caso de que el usuario quiera hacer crearlos de manera manual para después enviarlos al sistema.

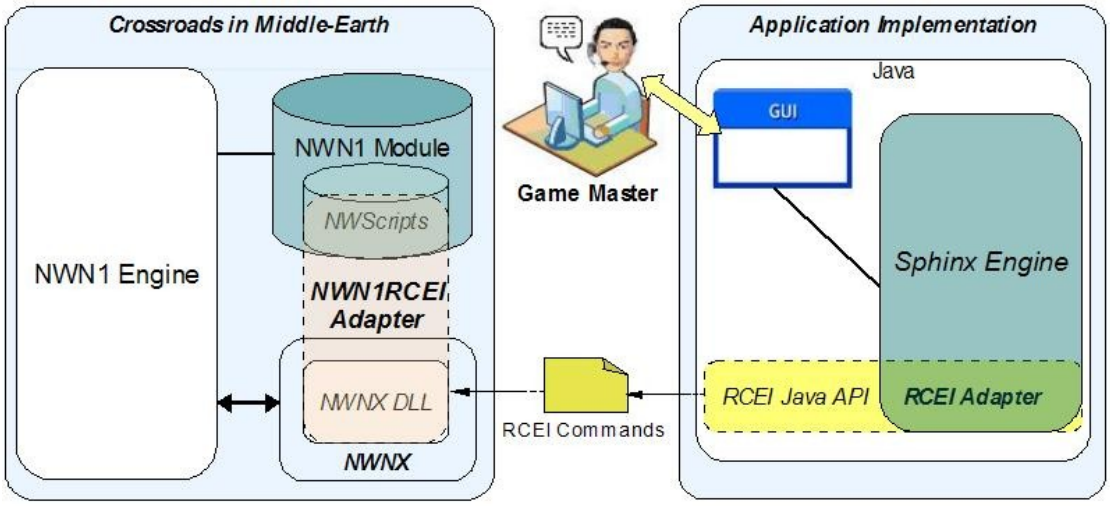


Figura 16. Esquema de la arquitectura del sistema.

Después tenemos un módulo de comunicación con el entorno, que se encarga de establecer una conexión entre la interfaz de java y el motor del juego. Mediante este módulo se envían cadenas de texto, en las cuales se enviará información formateada con el lenguaje XML definido para los comandos y hechos que procesa el sistema. El motor de Neverwinter Nights recibirá estos mensajes gracias a una librería dinámica que se ha implementado en el sistema (nwnx_rcei.dll). Para implementar está librería se ha creado un proyecto con Visual C++ 7.0, en el cual se han usado algunos componentes de NWN Extender.
Para que todos los comandos se puedan procesar en el entorno de Neverwinter Nights se ha definido un conjunto de scripts con el lenguaje NWNScript, que permiten su identificación y procesamiento en el sistema. En estos scripts se incluye también un parser del lenguaje RCEI, que permite distinguir los comandos e irlos almacenando en una cola. Cada cierto tiempo, aproximadamente tres segundos, se comprueba el contenido de la cola y si no está vacía, los comandos son ejecutados por el motor.
Por otro lado tenemos el módulo de reconocimiento de voz, que se

encarga de reconocer órdenes vocales. Estás ordenes vocales serán recogidas y serán enviadas al módulo de la lógica de la aplicación, que será el que se encargue de interpretarlas y convertirlas en comandos de RCEI. El repertorio de órdenes vocales que se pueden reconocer se ha seleccionado con dos criterios, primero que fueran órdenes fonéticamente distintas para que no haya muchos errores en el reconocimiento de órdenes que tengan una pronunciación parecida. En segundo lugar se ha buscado que los nombres de los personajes u objetos se correspondan con los de la aventura que se ha diseñado.

Por último, la lógica de la aplicación se encarga de ejecutar un tipo de comportamiento sobre las entradas que reciba el sistema por medio de órdenes vocales. Principalmente se realizará un mapeo de las órdenes vocales a comandos definidos que reconocerá el entorno. Una vez reconocida la orden vocal, la lógica debe componer un comando sabiendo según el tipo de orden reconocida el tipo de proceso que representa esa orden y los argumentos que le acompañan.

3.1. Metodología

En primer lugar se realizó un análisis sobre algunos de los sistemas de narración interactiva más representativos para ver cuales eran sus características y si usaban o no interfaces vocales para la interactividad. En la mayoría de los casos se comprobó que no hay aplicaciones de este tipo que usen interfaces vocales, auque si que existen algunos juegos comerciales que usan interfaces vocales para reconocer un pequeños repertorio de comandos, para que el usuario ejecute algunas acciones mediante voz.

Por otro lado se analizaron diferentes motores de juego de juegos comerciales, de los cuales nos centramos en Neverwinter Nights y Neverwinter Nights 2. Neverwinter Nights 2 es más moderno y tiene una interface más potente, pero también presenta ciertos inconvenientes como la incompatibilidad parcial con su predecesor y unos requisitos técnicos muy elevados. Estas razones, además de cierta experiencia previa, nos hicieron decantarnos por Neverwinter Nights.

Además del análisis de motores de juego, se quiso hacer un estudio del objeto del trabajo entre la comunidad de jugadores y directores de juegos de rol, para investigar sobre el interés real que podría tener en videojuegos y en que partes del juego podría ser más útil. Para ello se realizaron dos encuestas en Internet. La manera de poder realizar las encuestas fue publicando mensajes en diferentes foros de Internet relacionados con juegos de rol, invitando a los usuarios a rellenar la encuesta. También se enviaron las encuestas a gente conocida aficionada a juegos de rol.

Después se realizó un estudio sobre las tecnologías existentes para el reconocimiento de voz e interfaces vocales. Principalmente se buscaba una tecnología que pudiera ser aplicable al proyecto, que fuera de código abierto, y a ser posible que fuera compatible con el lenguaje Java. De todas las estudiadas se llegó a la conclusión que Sphinx era la más adecuada para usar en la implementación de este trabajo.

Para realizar el prototipo se han buscado las herramientas que pudieran ser más idóneas, que fueran de libre distribución y que estuvieran escritas en Java. La elección del entorno virtual pretendía que se pudiera realizar una comunicación otra aplicación y que además que el entorno fuera de tipo narrativo, con personajes y escenarios con cierto grado de realismo.

3.2. Análisis de requisitos

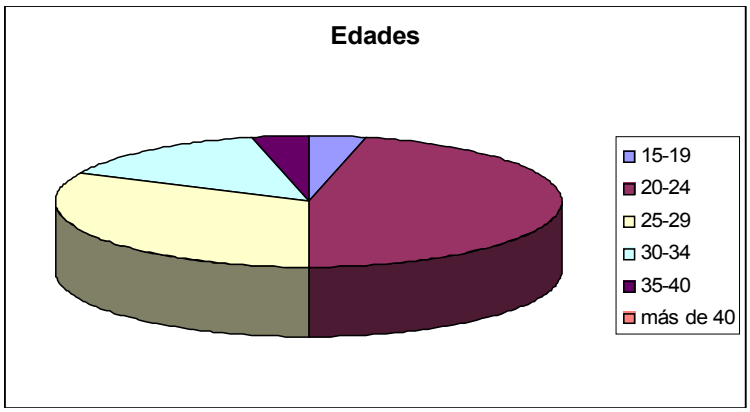
Antes de realizar el diseño de la interfaz vocal se realizaron dos encuestas, en un portal de Internet (SurveyMonkey), entre personas aficionadas a juegos de rol. La primera estaba dirigida a jugadores de rol y la segunda a directores de partidas de juegos de rol. La primera encuesta fue realizada a veintiocho personas, mientras que la segunda se realizó a dieciséis personas. Es lógico que haya más jugadores, ya que no todo jugador de rol sabe dirigir una partida, y por lo general suele haber muchos menos directores.

A continuación se muestran las preguntas que se han realizado en cada una de las encuestas y los resultados obtenidos.

Encuesta 1

1. ¿Cuántos años tienes?

15-19	1
20-24	13
25-29	9
30-34	4
35-40	1
Más de 40	0



2. ¿De qué sexo eres?

	Response Percent	Response Count
Masculino	96.4%	27
Femenino	3.6%	1
	answered question	28
	skipped question	0

3. ¿Qué nivel de estudios tienes?

	Response Percent	Response Count
Sin estudios	0.0%	0
Primarios	0.0%	0
Secundarios	14.3%	4
Universitarios	85.7%	24
	answered question	28
	skipped question	0

4. ¿Qué grado de experiencia tienes jugando partidas en juegos de rol de lápiz y papel?

	Response Percent	Response Count
Ninguna experiencia	3.6%	1
He jugado alguna vez	39.3%	11
He jugado durante algún tiempo	35.7%	10
Suelo jugar a menudo	7.1%	2
Estoy jugando casi siempre	14.3%	4
	answered question	28
	skipped question	0

5. ¿Qué nivel de experiencia tienes como jugador en juegos de rol por ordenador?

	Response Percent	Response Count
Ninguna experiencia	0.0%	0
He jugado alguna vez	10.7%	3
He jugado durante algún tiempo	21.4%	6
Suelo jugar a menudo	42.9%	12
Estoy jugando casi siempre	25.0%	7
answered question		28
skipped question		0

6. ¿Qué nivel de experiencia tienes en los estos juegos de rol por ordenador?

	Ninguna experiencia	He jugado alguna vez	He jugado durante algún tiempo	He jugado a menudo	He jugado muchísimo tiempo	Response Count
Neverwinter Nights 1	18.5% (5)	11.1% (3)	11.1% (3)	14.8% (4)	**44.4% (12)**	27
Neverwinter Nights 2	**35.7% (10)**	10.7% (3)	17.9% (5)	10.7% (3)	25.0% (7)	28
Vampire The Masquerade: Redemption	**46.2% (12)**	19.2% (5)	11.5% (3)	7.7% (2)	15.4% (4)	26
Dungeon Siege	**50.0% (13)**	19.2% (5)	23.1% (6)	3.8% (1)	3.8% (1)	26
Otros	4.0% (1)	28.0% (7)	8.0% (2)	**48.0% (12)**	12.0% (3)	25
answered question						28
skipped question						0

7. ¿Cuál es tu e-mail? Esta pregunta es opcional y únicamente se debe rellenar para recibir los resultados de este estudio. Si quieres dejar algún comentario personal también puedes hacerlo.

En esta pregunta algunas personas dejaron su e-mail y algunas pidieron que se publicaran los datos en los foros.

Encuesta 2

1. ¿Qué grado de experiencia tienes como director de juego en juegos de rol de lápiz y papel?

	Response Percent	Response Count
Ninguna experiencia	25.0%	4
He jugado alguna vez	25.0%	4
He jugado durante algún tiempo	18.8%	3
Suelo jugar a menudo	12.5%	2
Estoy jugando casi siempre	18.8%	3
answered question		16
skipped question		0

2. ¿Qué nivel de experiencia tienes como director de juego en juegos de rol por ordenador?

	Response Percent	Response Count
Ninguna experiencia	37.5%	6
He jugado alguna vez	12.5%	2
He jugado durante algún tiempo	25.0%	4
Suelo jugar a menudo	0.0%	0
Estoy jugando casi siempre	25.0%	4
answered question		16
skipped question		0

3. ¿En qué juegos de rol por ordenador has dirigido partidas?

	Response Percent	Response Count
Neverwinter Nights 1	80.0%	8
Neverwinter Nights 2	60.0%	6
Vampire The Masquerade: Redemption	40.0%	4
Dungeon Siege	0.0%	0
Otros	40.0%	4
answered question		10
skipped question		6

4. ¿Conoces algún otro juego de rol por ordenador con la opción de director de juego?

De las diez personas que contestaron la pregunta, nueve pusieron no y una puso el comentario que se reproduce a continuación: “Sí, pero a nivel masivo (MMORPG); también algunos MUDs antiguos. También la serie baldur's gate (los predecesores de los NWN).”

5. En los juegos de rol por ordenador que hayas jugado, ¿Cuanto tiempo de aprendizaje has necesitado para poder realizar una partida en red como director?

En esta pregunta se dejo la opción a los usuarios que comentasen su experiencia. A continuación se muestran las respuestas.

- 1 Mes.
- Aproximadamente 1 mes.
- Bastante (mínimo uno o varios meses).
- Una semana.
- Poco.
- Nada, es parecido a una partida de rol de mesa. Ojo parecido, nunca igual.
- Unas pocas horas, es tan parecido a los juegos de mesa que uno por intuición aprende a usar las herramientas.
- Un par de días para acostumbrarme a la interfaz y las normas del servidor, después de un par de años como jugador viendo como lo hacían los directores desde abajo.
- Mucho tiempo... demasiado. Al final no tuvo mucho éxito y lo dejé porque requería mucho esfuerzo.
- No aplicable.

6. ¿Qué acciones son más complejas de realizar en estos interfaces?

	nada compleja	poco compleja	medianamente compleja	muy compleja	Response Count
Buscar objeto en un escenario	18.2% (2)	**45.5% (5)**	36.4% (4)	0.0% (0)	11
Buscar por nombre	**63.6% (7)**	27.3% (3)	9.1% (1)	0.0% (0)	11
Crear objetos/criaturas	**45.5% (5)**	27.3% (3)	18.2% (2)	9.1% (1)	11
Ir a	**54.5% (6)**	36.4% (4)	9.1% (1)	0.0% (0)	11
Matar	**81.8% (9)**	9.1% (1)	9.1% (1)	0.0% (0)	11
Abrir/Cerrar puerta	**72.7% (8)**	18.2% (2)	9.1% (1)	0.0% (0)	11
Examinar	**81.8% (9)**	18.2% (2)	0.0% (0)	0.0% (0)	11
Dar/Coger	**45.5% (5)**	36.4% (4)	18.2% (2)	0.0% (0)	11
Poseer	**54.5% (6)**	45.5% (5)	0.0% (0)	0.0% (0)	11
Atacar	**72.7% (8)**	27.3% (3)	0.0% (0)	0.0% (0)	11
Descansar	**54.5% (6)**	45.5% (5)	0.0% (0)	0.0% (0)	11
Sanar	**63.6% (7)**	36.4% (4)	0.0% (0)	0.0% (0)	11
Mover	**63.6% (7)**	18.2% (2)	18.2% (2)	0.0% (0)	11
Revivir	**45.5% (5)**	**45.5% (5)**	9.1% (1)	0.0% (0)	11
				answered question	11
				skipped question	5

7. ¿Qué utilidad le ves a usar órdenes vocales para estas acciones?

	ninguna utilidad	poca utilidad	medianamente util	mucha utilidad	Response Count
Buscar objeto en un escenario	**41.7% (5)**	8.3% (1)	**41.7% (5)**	8.3% (1)	12
Buscar por nombre	**33.3% (4)**	25.0% (3)	25.0% (3)	16.7% (2)	12
Crear objetos/criaturas	**50.0% (6)**	16.7% (2)	16.7% (2)	16.7% (2)	12
Ir a	**33.3% (4)**	16.7% (2)	16.7% (2)	**33.3% (4)**	12
Matar	33.3% (4)	**50.0% (6)**	16.7% (2)	0.0% (0)	12
Abrir/Cerrar puerta	**41.7% (5)**	25.0% (3)	25.0% (3)	8.3% (1)	12
Examinar	**33.3% (4)**	**33.3% (4)**	16.7% (2)	16.7% (2)	12
Dar/Coger	33.3% (4)	**41.7% (5)**	16.7% (2)	8.3% (1)	12
Poseer	33.3% (4)	**50.0% (6)**	8.3% (1)	8.3% (1)	12
Atacar	33.3% (4)	**50.0% (6)**	0.0% (0)	16.7% (2)	12
Descansar	**41.7% (5)**	33.3% (4)	16.7% (2)	8.3% (1)	12
Sanar	25.0% (3)	**41.7% (5)**	0.0% (0)	33.3% (4)	12
Mover	**41.7% (5)**	33.3% (4)	8.3% (1)	16.7% (2)	12
Revivir	**41.7% (5)**	33.3% (4)	25.0% (3)	0.0% (0)	12
				answered question	12
				skipped question	4

8. ¿Cómo de acuerdo está en que combinar voz con teclado y ratón sea beneficioso para dirigir partidas en juegos de rol por ordenador?

	Response Percent	Response Count
no estoy nada de acuerdo	13.3%	2
poco de acuerdo	20.0%	3
bastante de acuerdo	13.3%	2
muy de acuerdo	40.0%	6
totalmente de acuerdo	13.3%	2
answered question		15
skipped question		1

9. ¿Cuál es tu e-mail? Esta pregunta es opcional y únicamente se debe rellenar para recibir los resultados de este estudio. Si quieres dejar algún comentario personal también puedes hacerlo.

Algunos usuarios dejaron su e-mail y uno de ellos hizo un comentario, que se muestra a continuación: "Sobre la pregunta 8, si se refiere al uso de la voz para comunicarse con los jugadores, además del teclado y el ratón, es muy beneficioso para dirigir partidas. Si se refiere a la posibilidad de dar comandos técnicos a la máquina, me parece poco beneficioso. Para comandos sencillos puede tener alguna utilidad. Para comandos complicados (crear objetos/criaturas con todas sus propiedades, por ejemplo), podría ser un engorro, ya que habría que recordar de memoria un montón de propiedades y cómo se introducen sus valores de forma verbal, cosa que con un interfaz gráfico se podría simplificar, eso sí (pero eso ya lo tenemos al alcance del ratón/teclado)."

3.3. Módulo de comunicación con el entorno

En este apartado se detallan los diferentes aspectos que involucran la comunicación de la aplicación en Java con el entorno virtual. Esta comunicación es compleja, ya que involucra diferentes sistemas como son el motor del juego y la aplicación en Java, incluyendo una librería dinámica para el envío de paquetes mediante sockets.

3.3.1. Protocolo de comunicación

Debemos definir un protocolo de comunicación para delimitar claramente como se va a relacionar el sistema con el entorno concreto y poder concretar que tipo de respuestas va a recibir el sistema. Para ello se ha definido un lenguaje y un protocolo de comunicación RCEI

(Remote Controlled Environments Interface).

En un principio, para poder realizar está comunicación se ha creado una librería dinámica para Windows que a su vez hace uso Neverwinter Nights Extender para crear un servidor de Neverwinter Nights. Para realizar el diseño de esta librería nos inspiramos inicialmente en Shadow Door. La librería se llama nwnx_rcei.dll, y sirve para que una aplicación acceda al motor de Neverwinter Nights para enviar y recibir paquetes, con un formato propio.

Este protocolo y este lenguaje de comunicación entre aplicaciones están pensados para conectar bidireccionalmente un entorno virtual con una aplicación de control remoto de manera genérica, sin entrar en detalles relativos a la presentación de los elementos en el entorno virtual ni la representación interna de los mismos en la aplicación de control remoto.

El protocolo RCEI tiene las siguientes características:

- La comunicación es síncrona bloqueante y el envío de mensajes es estrictamente alterno (siempre primero un mensaje de la aplicación, después otro del entorno, uno de la aplicación, y así sucesivamente).

- Generalmente las respuestas del entorno son hechos (facts) que confirman las órdenes del sistema (commands).

- La aplicación de control remoto es quien da comienzo a las comunicaciones. El primer mensaje del entorno virtual es un mensaje de identificación.

- La aplicación de control remoto debe ser quien finalice las comunicaciones con un mensaje final al entorno, aunque este también puede mandar mensaje de error a la aplicación (como el de “finalización inesperada”).

- Si el bloqueo de la aplicación de control remoto en espera de recibir mensaje se alarga demasiado, automáticamente se produce una “finalización inesperada por exceso de tiempo en responder”.

- La aplicación envía comandos y recibe hechos a través de una interfaz de sockets. El protocolo se implementa con sockets en ASCII extendido de 8 bits, con un máximo de 1024 bytes de longitud.

3.3.2. Lenguaje de comunicación

El lenguaje de comunicación es un sublenguaje de XML orientado a indicar acciones individuales a determinados elementos del entorno para que estos efectúen un cambio de comportamiento o realicen un acto determinado.

El lenguaje define la estructura de un mensaje RCEI, el cual está compuesto por un número de mensaje y una lista de instrucciones. Las instrucciones serán comandos si el sentido del mensaje es desde el emisor hacia en entorno o hechos si el sentido es desde el entorno hasta el emisor. Cada una de las instrucciones estará compuesta por un sujeto, un proceso y en función del tipo de proceso pueden llevar varios complementos para precisar el comportamiento de la acción. Los diferentes complementos son el complemento directo, el complemento indirecto, el complemento de modo y el complemento de lugar.

El proceso es una acción que modifica el estado de una entidad haciendo que cambie. En RCEI han divido los procesos en varias categorías según en la teoría de Schank. Los tipos de procesos pueden ser transiciones físicas (ir, equipar, desequipar, coger, soltar, atacar, destruir), transiciones mentales (cambiar estado mental), construcciones físicas (crear objeto, crear personaje) o transiciones abstractas (cambio de clima, cambio de cámara, empezar, terminar).

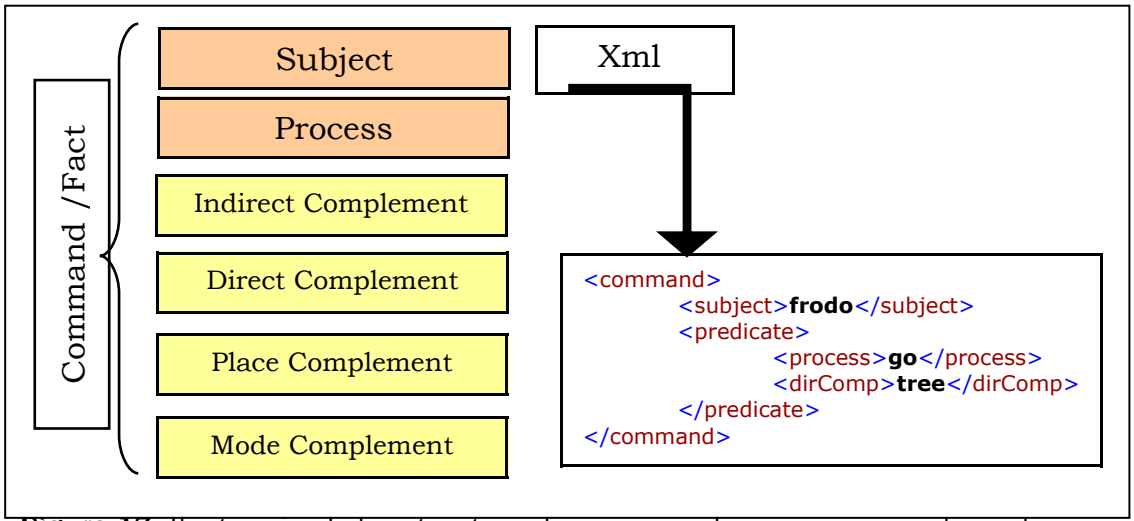


Figura 17. Ilustración de la estructura de un comando y como se escribe en lenguaje RCEI.

Los procesos siempre se aplican sobre un sujeto. El tipo de sujeto podrá ser un agente, que se corresponde con un personaje del entorno, o un sujeto especial denominado *environment*. Éste se usa cuando el proceso que se está realizando es una transición abstracta que se aplica a todo el entorno, como llover.

Por tanto, el sujeto y el proceso son las partes imprescindibles en un comando o hecho RCEI. El resto de complementos pueden ser opcionales según el tipo de proceso, como se puede apreciar en la figura 17. El complemento directo se usa para indicar sobre que elemento se ejecuta un proceso. Por ejemplo, si un agente quiere ir hasta un árbol, el complemento directo será el árbol. El complemento directo puede ser un agente, un objeto, un enlace o una frase que sea pronunciada por un agente. Los objetos se corresponden con los objetos físicos del entorno, como armas, monedas o comida, y los enlaces se corresponden con las transiciones entre diferentes áreas, que generalmente son puertas.

El complemento indirecto sólo se usa en el proceso give. Mediante este complemente se indica a qué agente se le debe de entregar el objeto. El complemento de lugar se utiliza para indicar una ubicación. Por ejemplo si se está ejecutando un proceso del entorno como llover, es necesario indicar sobre que localización se aplica el proceso. Una localización se corresponde con un área del entorno. En el entorno existen generalmente varias áreas conectadas por transiciones que, como se ha comentado suelen ser puertas.

El complemento de modo, se usa con el proceso change, para indicar el estado en que deben quedar las puertas (cerrado, abierto), o en el proceso create para indicar el punto sobre el cual se quiere crear un agente. El proceso change también puede cambiar el estado de un objeto o agente, que depende de cómo estén definidos en el entorno; por ejemplo, un agente se podría cambiar de contento a enfadado.

El proceso synchronize es distinto a los anteriores, posee dos complementos diferentes: conceptos y relaciones. Los conceptos son todos los elementos que componen el entorno y las relaciones son las que tienen entre si los componentes, como relaciones de pertenencia con objetos, de amistad o alianza con personajes. Por ejemplo un agente A puede ser enemigo de un agente B, o un cofre puede tener una bolsa de monedas.

Especificación de la gramática de RCEI

La descripción formal de todas las instrucciones se detalla en la gramática que define el lenguaje RCEI. La gramática se utiliza para un construir un parser que procesa documentos con formato RCEI. A continuación se muestra la gramática completa (figura 18).

```
//Cabecera principal del mensaje RCEI.
Message::= <RCEI>
           <messageNumber> numInt </messageNumber>
           LInstruction
           </RCEI>
```

```
LInstruction::= LCommand

LInstruction::= LFact

//Cada mensaje puede ser una lista de comandos
LCommand::= Command LCommand | Command

Command::= <command>
           <subject> Subject </subject>
           <predicate> Predicate </predicate>
           </command>

//Cada mensaje puede ser una lista de hechos
LFact::= Fact LFact | Fact

Fact::= <fact>
        <subject> Subject </subject>
        <predicate> Predicate </predicate>
        </fact>
```

```
//El sujeto será una lista de agentes o el propio sistema (environ-
ment)
Subject::= environment | LAgent
```

```
//La gramática permitirá que el proceso se descomponga posteriormente
en otros procesos a partir de los procesos primitivos
Predicate::= <process> PSpeak </process>
             <dirComp> string </dirComp>

Predicate::= <process> PSpeak </process>

Predicate::= <process> PBegin </process>

//Los procesos para comenzar y terminar pueden ir sin argumentos
Predicate::= <process> PBegin </process>
             <dirComp> string</dirComp>

//Move realiza la animación de un agente
Predicate::= <process> PMove </process>
             <dirComp> Movement </dirComp>

//El proceso de dar un objeto lleva 2 argumentos
Predicate::= <process> PGive </process>
             <dirComp> Object </dirComp>
             <indComp> Agent </indComp>

//El proceso de moverse puede ir sin argumentos (movimiento aleatorio)
Predicate::= <process> PGo </process>

//PTrans indica que es una trasición física
Predicate::= <process> PTrans </process>
             <dirComp> Object </dirComp>


//El proceso Physical Build indica una construcción de algo material,
//por ello es necesario indicarle la ubicación y las coordenadas.
Predicate::= <process> PBuild </process>
```

```
            <dirComp> Agent </dirComp>
            <modeComp> Waypoint </modeComp>
            <placeComp> Location </placeComp>

//El proceso change servirá para cambiar tanto situaciones del propio
//sistema como propiedades de objetos o personajes.
Predicate::= <process> PChange </process>
            <dirComp> Atrans </dirComp>
            <modeComp> Mode </modeComp>
            <placeComp> Location </placeComp>

Predicate::= <process> PChange </process>
            <dirComp> Agent </dirComp>
            <modeComp> Mode </modeComp>

Predicate::= <process> PChange </process>
            <dirComp> Object </dirComp>
            <modeComp> Mode </modeComp>

Predicate::= <process> PChange </process>
            <dirComp> Link </dirComp>
            <modeComp> Mode </modeComp>

// synchronize es una Abstract transaction (Schank)
Predicate::= <process> PSync </process>
            <concepts> Concepts </concepts>
            <relations> Relations </relations>
```

```
//Dentro del predicado de la instrucción begin puede haber conceptos y
//relaciones
Concepts::= Concept Concepts | λ

Relations::= Relation Relations | λ

//Se permiten dos notaciones para enumerar localizaciones, agentes,
//objetos o links, una más completa y otra simplificada
Concept::= <locations> LLocation </locations>

Concept::= <locations> LLocation2 </locations>

Concept::= <agents> LAgent </agents>

Concept::= <agents> LAgent2 </agents>

Concept::= <objects> LObject </objects>

Concept::= <objects> LObject2 </objects>

Concept::= <links> LLink </links>

Concept::= <links> LLink2 </links>

//Las relaciones siempre son entre dos elementos. Pueden existir //va-
rios tipos de relaciones. La relación de un elemento con respecto //a
otro es unidireccional
Relation::= <is> Domain Range </is>

Relation::= <has> Domain Range </has>
```

```
Relation::= <ally> Domain Range </ally>

Relation::= <enemy> Domain Range </enemy>

//En una relación el dominio puede ser una o varias localizaciones,
//agentes, objetos o links
Domain::= LLocation

Domain::= LLocation2

Domain::= LAgent

Domain::= LAgent2

Domain::= LObject

Domain::= LObject2

Domain::= LLink

Domain::= LLink2

//En una relación el rango puede ser una o varias localizaciones,
//agentes, objetos o links
Range::= LLocation

Range::= LLocation2

Range::= LAgent

Range::= LAgent2

Range::= LObject

Range::= LObject2

Range::= LLink

Range::= LLink2

//Una lista de localizaciones puede tener una o varias
LLocation::= <location> Location </location> LLocation | λ
LLocation2::= Location LLocation | λ

//Una lista de agentes puede tener varios agentes o ninguno
LAgent::= <agent> Agent </agent> LAgent | λ
LAgent2::= Agent LAgent | λ


//Una lista de objetos puede tener varios objetos o ninguno
LObject::= <object> Object </object> LObject | λ
LObject2::= Object LObject | λ

//Una lista de puertas puede tener varias puertas o ninguna
LLink::= <link> Link </link> LLink | λ
LLink2::= Link LLink | λ
```

```
PSpeak::= speak

//Esta producción es para procesos que realizan una animación concreta
de un personaje, como pueden ser beber, reir, enfadarse, etc...
PMove::= move

//Tipos de animación que puede representar un agente
Movement::= none | drink | reed | greeting | listen | talk | implore |
furious | laugh | victory| adore | harass | reverence |
steal

//PTrans (physhical transaction) está tomado de Schank
PTrans::= PGo | equip | unequip | take | drop | attack | destroy | use

PGo::= go
//give es physhical transaction(Schank) con dos argumentos
PGive::= give

//PBuild es physhical building (Schank) y se usa para crear personajes
o trasladarlos de un punto a otro.
PBuild::= create | locatedAt

//Corresponde a las coordenadas x y z del espacio en 3 dimensiones
Waypoint::= num, num, num

PChange::= change

Atrans::= weather | sky | fog | light | camera

//Para el caso del change open y close cambiaran el estado de una
puerta
Mode::= id

PBegin::= begin | end

PSync::= syncronize

//Estas producciones sólo tienen sentido de cara a las propiedades del
sujeto, y los complementos, para fijar como restricción que el id sea
de un tipo determinado
Location::= id

Agent::= id
Object::= id
Link::= id
```

Figura 18. Gramática que define el lenguaje RCEI.

El parser de RCEI se encargará de comprobar que todos los mensajes que se envían y reciben del entorno son sintácticamente correctos. Además, en caso de haberse procesado correctamente, el parser se encarga de crear los objetos java con el valor de los atributos.

La formalización del lenguaje está basada en XML, por lo tanto para procesar el lenguaje desde fuera del entorno necesitaremos un parser de XML adaptado al lenguaje de comunicación con el entorno RCEI.

Repertorio de instrucciones

A continuación se muestra el repertorio de instrucciones del lenguaje, que se dividen en tres categorías: simulación, narración y metaprocesos. Para cada proceso se detalla el valor puede tomar cada uno de los complementos que lleva, en caso se que el complemento sea opcional, se indicará.

Simulación

En todos los casos se da por supuesto que todos los procesos tienen un sujeto que se corresponde con un agente, en caso de que el proceso pueda tener el sujeto *environment*, se indicará en la explicación

- **Attack**: este proceso hace referencia a la acción que un agente realiza para atacar a otro agente.
    - dirComp: agente u objeto sobre el que se realiza la acción.

- **Change**: este proceso hace referencia a la acción un agente realizan para cambiar un estado en un enlace, o bien a la acción que realiza el entorno para cambiar el estado de una localización, como por ejemplo el clima.
    - dirComp: en caso de que el sujeto sea una agente, el complemento directo será un enlace (puerta). En caso de que el sujeto sea el entorno (environment), el sujeto podrá ser weather, sky, light. Con weather se cambiará el clima, con sky el tipo de cielo, con Light el tipo de luz, y con camera el tiro de cámara.
    - placeComp: ubicación donde se realiza el cambio. Puede ser en todo el módulo o en una localización concreta.
    - modeComp: modo de ejecutarse el cambio. Determina el valor que puede tomar el tipo de cambio realizado. El en caso de un enlace puede tomar los valores open y close (abierto y cerrado). En el caso de weather puede tomar los valores snow, rain o clear (nieve, lluvia, despejado). En el caso de light puede tomar los valores night y day (noche y día).

- **Create**: proceso hace referencia a la acción de crear un agente u objeto en una localización. En este proceso el sujeto siempre es *environment*.
    - dirComp: tipo de agente u objeto creado.
    - placeComp: localización donde se realiza el cambio.
    - modeComp: coordenadas cartesianas de la localización donde se desea crear el agente.

- **Destroy**: proceso hace referencia a la acción de eliminar un

agente u objeto de una localización. En este proceso el sujeto siempre es *environment*.
    - dirComp: agente u objeto que se va a destruir.

- **Drop**: proceso hace referencia a la acción que un agente realiza para desprenderse de un objeto que posee.
    - dirComp: objeto sobre el que se realiza la acción.

- **Equip**: este proceso hace referencia a la acción que un agente realiza para equiparse con un objeto que ya posee.
    - dirComp: objeto sobre el que se realiza la acción.

- **Give**: este proceso hace referencia a la acción que un agente realiza para dar un objeto a otro agente.
    - dirComp: objeto que es entregado al agente.
    - indComp: agente al cual se le da el objeto.

- **Go**: proceso hace referencia a la acción que un agente realiza para desplazarse de un punto a otro punto.
    - dirComp: objeto hasta el cual el agente quiere desplazarse. Este complemento es opcional sino se pone se producirá un movimiento aleatorio.

- **Kill**: proceso hace referencia a la acción de matar a un agente. Es similar a destroy pero conlleva que el personaje debe realizar la acción de morirse antes de eliminarse. En este proceso el sujeto siempre es *environment*.
    - dirComp: agente sobre el que se realiza la acción.

- **Move**: este proceso activa una pequeña animación en un agente, como un saludo, una reverencia, una sonrisa u otros similares.
    - dirComp: tipo de animación realizada por el agente. Los tipos de animación son: none (ninguna), drink (beber), reed (leer), greeting (saludo), listen (escuchar), talk (hablar), furious (hablar furioso), laugh (hablar riendo), reverence (reverencia) y steal (robar).

- **Speak**: proceso hace referencia a la acción que un agente realiza para comunicarse con otro agente.
    - dirComp: mensaje de texto que envía el agente.

- **Take**: proceso hace referencia a la acción que un agente realiza para coger un objeto.
    - dirComp: objeto sobre el que se realiza la acción.

- **Unequip**: proceso hace referencia a la acción que un agente realiza para quitarse un objeto con el que está equipado y

guardarlo.
    - dirComp: objeto sobre el que se realiza la acción.

Narración

En todos estos procesos el sujeto siempre es el entorno, por lo que se omite en la explicación de los comandos.

- **Change**: este proceso hace referencia a la acción que realiza el entorno para cambiar el estado de un personaje u objeto.
    - dirComp: el complemento directo será un agente que tenga un estado mental asociado que pueda ser modificado.
    - placeComp: localización donde se realiza el cambio.
    - modeComp: modo de ejecutarse el cambio. Determina el valor que puede tomar el tipo de cambio realizado. Un dependiendo de cómo se haya definido el comportamiento del agente puede pasar de feliz a triste.

- **Search**: proceso hace referencia a la acción de buscar un agente u objeto en una localización.
    - dirComp: objeto o agente sobre el que se realiza la acción.
- **Control**: proceso hace referencia a la acción de controlar un agente que forma parte del entorno. Una vez se tome control de ese agente se podrá actuar a través de él.
    - dirComp: agente sobre el que se realiza la acción.

Metaprocesos o procesos generales de RCEI

En este caso, al igual que el anterior, se supone que el sujeto que lleva cada proceso es el entorno. El proceso synchronize es diferente a todos los demás, ya que no tiene ninguno de los complementos mencionados, sino dos complementos diferentes: concepts y relations.

- **Begin**: proceso inicializa una nueva sesión en el entorno.

- **End**: proceso que finaliza una sesión en el entorno.
    - dirComp: nombre de un fichero con una animación que se reproducirá al terminar la sesión en el entorno. Es opcional, si no se incluye la sesión terminará sin más.

- **Synchronize**: proceso que obtiene todos los elementos y relaciones del entorno en un momento concreto.
    - concepts: lista de todos los elementos de juego en un momento dado. Es decir, todos los agentes, objetos, enlaces y localizaciones que existen en un instante concreto.
    - relations: todas las relaciones que existen entre los

conceptos. Entre objetos pueden darse relaciones de pertenencia, entre agentes pueden darse relaciones de amistad, alianza o enemistad.

3.4. Módulo de gestión de voz

El procesamiento de voz es un área muy extensa y compleja que abarca muchas disciplinas. Como se ha visto en el punto 2.3 existen varias herramientas para el reconocimiento de voz. Se ha elegido Sphinx para realizar implementación porque era la más completa e idónea de todas las que se han investigado.

La complejidad de la interfaz vocal pude llegar a ser muy elevada, por eso en este trabajo solo se reconocerá un conjunto reducido de frases para poder realizar el prototipo que se pretende mostrar. El reconocimiento de voz se aplica al reconocimiento de pequeñas órdenes que se convertirán en órdenes en formato RCEI, y se enviarán para que se ejecuten acciones en el entorno. El tipo de órdenes que se admiten se detallarán más adelante. Por tanto se usará Sphinx para reconocer pequeñas órdenes, que pueden sufrir pequeñas variaciones.

Para realizar el módulo reconocedor de voz con Sphinx se han tenido que configurar una serie de parámetros que se establecen en un archivo de configuración llamado dialog.config.xml. Este archivo contiene toda la información necesaria para configurar el motor, como se ha explicado en el punto 2.3.1.

La información más relevante para nuestra implementación es el modelo acústico y la gramática, ya que el modelo acústico define el conjunto de palabras que se pueden reconocer y la gramática índica las frases que se pueden reconocer. Se puede ver como están configurados ambos en la figura 19, donde se muestra la parte del archivo de configuración de Sphinx en la que aparecen. El resto de parámetros aunque son igualmente muy importantes apenas varían para realizar otros proyectos similares. Si se quieren ver todos los componentes del archivo de configuración se puede ver en el apéndice A.

```
<?xml version="1.0" encoding="UTF-8"?>

<config>

    ...
    <!-- ******************************************************* -->
    <!-- The Grammar  configuration                              -->
    <!-- ******************************************************* -->

    <component name="jsgfGrammar" type="edu.cmu.sphinx.jsapi.JSGFGrammar">
        <property name="dictionary" value="dictionary"/>
```

```
        <property name="grammarLocation"
             value="resource:/dialog.Dialog!/dialog/"/>
        <property name="grammarName" value="menu"/>
       <property name="logMath" value="logMath"/>
    </component>

    <!-- ******************************************************** -->
    <!-- The Dictionary configuration                             -->
    <!-- ******************************************************** -->

    <component name="dictionary"
        type="edu.cmu.sphinx.linguist.dictionary.FastDictionary">
        <property name="dictionaryPath"  value="resource:/edu.cmu.sphinx.mo-
del.acoustic.WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.Model!/edu/cmu/sphinx/mo-
del/acoustic/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz/dict/cmudict.0.6d"/>
        <property name="fillerPath"      value="resource:/edu.cmu.sphinx.mo-
del.acoustic.WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.Model!/edu/cmu/sphinx/mo-
del/acoustic/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz/dict/fillerdict"/>
        <property name="addSilEndingPronunciation" value="false"/>
        <property name="allowMissingWords" value="false"/>
        <property name="unitManager" value="unitManager"/>
    </component>

    ...
</config>
```

Figura 19. Parte del archivo de configuración de Sphinx donde se aprecia como se configura el modelo acústico y la gramática.

El diseño para reconocer órdenes vocales y transformarlos en comandos RCEI, está compuesto por varios menús de diálogo. Es decir, existirán menús que se encargarán de reconocer cada una de las partes de un comando. El menú principal contendrá una lista con todos los procesos que puede tener un comando RCEI, los cuales se podrán reconocer mediante voz. Lo mismo ocurrirá con el resto de submenús.

En la definición del repertorio de instrucciones se explicó que no todos los procesos llevan todos los complementos, sino que según el tipo de proceso, un complemento puede estar o no. Para el reconocimiento vocal se realizará de manera similar, es decir, según el proceso reconocido en el menú principal, se mostrarán los demás menús sucesivamente. Por ejemplo, si se desea que un agente se mueva hasta un objeto, primero se reconocería el comando *go* en el menú principal, después se pasaría al menú de sujeto, donde se mostrarían todos los posibles agentes que podrían representar a un sujeto. Una vez reconocido el sujeto se pasaría al menú complemento directo, en el cual se reconocería el objeto al cual se va a desplazar el agente. Una vez finalizado el reconocimiento de cada una de las partes de la orden vocal ya se puede componer un comando RCEI. Si una orden vocal se encuentra en alguno de los submenús y se desea abortar la orden actual, existe una opción para volver al menú principal.

Tanto el menú principal como cada uno de los submenús se reconocen mediante una gramática JSGF contenidos en archivos con extensión

.gram. Hay un archivo con una gramática para cada menú. El cambio de gramática a reconocer se realiza utilizando la clase DialogManager de Sphinx, en el cual se crean un conjunto de nodos, a los cuales se les asocia un comportamiento. Este comportamiento se corresponde con la gramática de cada uno de los submenús.

4. Aplicación del interfaz vocal sobre un entorno virtual narrativo

En el entorno virtual se detallarán principalmente tres aspectos. Por un lado todo el guión y la historia que se ha elaborado para jugar la aventura, por otro como se ha implementado en Neverwinter Nights y por otro la implementación y finalmente se detalla como es la dirección mediante el interfaz vocal, es decir, qué tipo de órdenes se dan mediante voz y con qué comandos.

4.1. Escenario de ejemplo

La aventura se llama Encrucijada en la Tierra Media (figura 31) y está basada en un arquetipo de historia conocido como *"Rags to Ritches"* según el libro "*The seven basic plots*" (Broker, 2004). Trata de un elfo que tiene que descubrir datos sobre el pasado de su pueblo por la ciudad en la que vive, posteriormente adentrarse en la tierra media para realizar sus objetivos.

Inicialmente se estudiaron distintos arquetipos de historias para el escenario de ejemplo y se eligió *"Rags to Ritches"* debido a que era el más atractivo para una aventura.

A continuación se muestra un resumen del argumento: *"El protagonista vive en la ciudad donde viven los humanos y es el único elfo que sobrevivió a la batalla de Burbag, por tanto se siente sólo porque es diferente de los humanos y es el único habitante de su raza. Nadie le ha explicado porque es el único elfo que vive en Reuel y porque no hay más elfos.*
Un buen día Calin empieza a indagar sobre su pasado pero todo el mundo permanece en silencio por orden del rey, por esto Calin decide ir a ver al rey. El rey le cuenta la verdad de su pueblo y la profecía que dice que el es el elegido para devolver la paz a la Tierra Media, debe cumplir su destino e ir al templo de Isilmang para arrebatar al mago su poder y así vengar a su pueblo."

El personaje debe resolver una serie de puzzles para conseguir sus objetivos. A continuación se detallan a grandes rasgos cuales son las diferentes tramas de los escenarios.

Acto 1. Descubriendo la verdad

- *Hablando con la gente de la ciudad.* El personaje busca información sobre sus orígenes y descubre que necesita hablar con el rey Reuel para conseguir dicha información.

- *Intentando hablar con el rey*. Al intentar hablar con el rey Héldur, éste le ignora al estar muy ocupado. Por tanto hasta que el protagonista no consigue que sus consejeros se vayan no puede hablar con él. Celendel ayudará al personaje para que pueda hablar con el rey.

- *Revelación de la historia y nuevo objetivo*. El rey le contará al protagonista la historia de su pasado y le dirá que tiene que hacer para vengarse. El rey le entregará una llave para salir de la ciudad e ir al bosque negro a buscar al brujo Vuurk para que ayude al personaje a vengarse.

Figura 31. Captura de escenario de Encrucijada en la Tierra Media.

Acto 2. El camino a la venganza

- *Buscando a Vuurk*. El protagonista busca la casa del brujo para que le ayude pero no la encuentra, solo encuentra a su aprendiz Elvanriel. Tas hablar con ella, indicará el camino para ir a casa del brujo.

- *En casa del brujo*. El protagonista va a casa del brujo a través de un portal secreto y el brujo le da una llave para llegar al templo de Isilmang y un arma para luchar contra él.

Acto 3. El retorno del elegido

- *Hacia el templo*. El protagonista tiene todo lo necesario para enfrentarse a Isilmang, y se dirige hacia su templo. Por el camino

se encuentra a un elemental de fuego convertido así por un hechizo del brujo.

- *En casa del brujo*. El protagonista va a casa del brujo, le derrota y le roba su cetro de poder. Se deshace el hechizo del elemental de fuego que se transforma en humano. El protagonista va a ver al brujo, se casa con Elvanriel y la historia tiene un final feliz.

4.2. Implementación del entorno

El entorno virtual elegido es Neverwinter Nights, que como se ha citado en el aparatado 2.1, es un juego comercial de temática de aventuras y acción en 3D con un sistema de reglas de estilo rol basadas en turnos.

Para diseñar y modelar el entorno de juego se ha utilizado el editor Aurora, de Neverwinter Nights. Con Aurora se han desarrollado todos los elementos necesarios para recrear la aventura que hemos diseñado. La implementación del entorno virtual se divide en tres partes: implementación de escenarios, implementación de elementos de interacción e implementación de la interacción de todas las entidades.

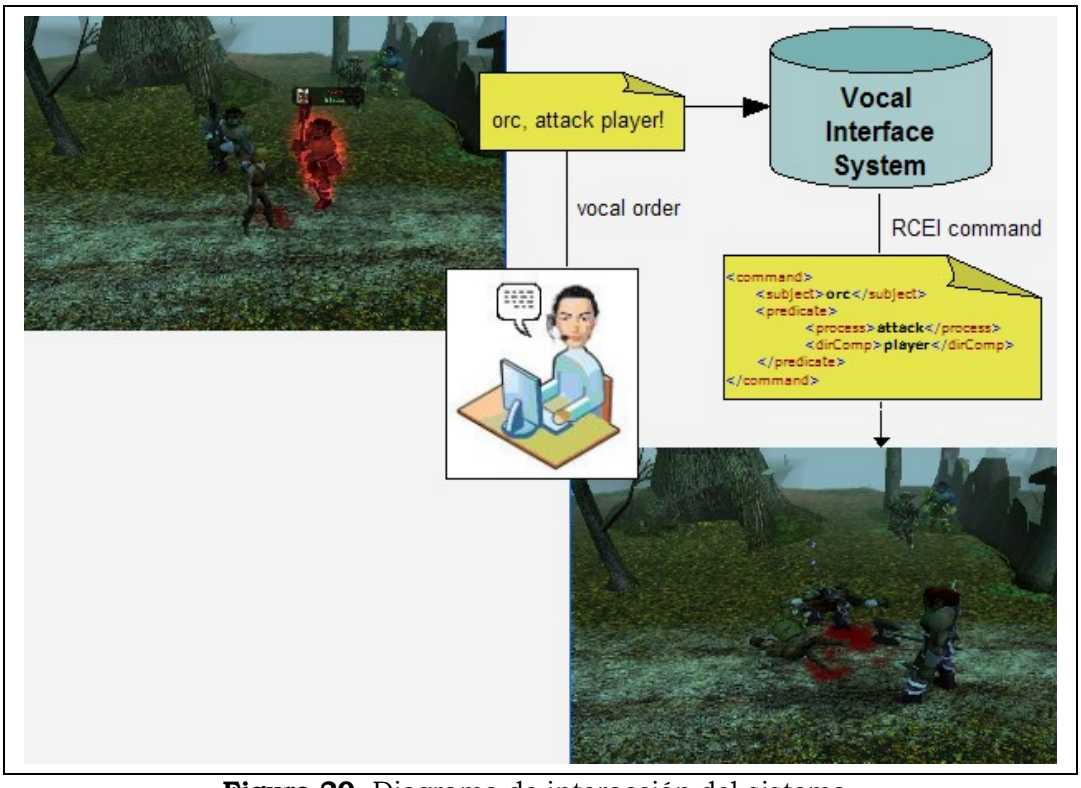


Figura 20. Diagrama de interacción del sistema.

La implementación de escenarios y demás elementos de juego es la parte donde se desarrollará la aventura, es decir, el trasfondo. Para realizar los escenarios basta con tener un boceto previo de un área, y

elegir los personajes y objetos de la aventura. En esta parte se diseña los elementos que ambientan el juego pero que no realizan funciones en el juego. En Neverwinter Nights se diseñan diversas áreas independientes de un cierto tamaño, conectadas a través de transiciones de área, generalmente caracterizadas por puertas. Un área está compuesta por un conjunto de celdas. Cada una de estas celdas se corresponde físicamente con un espacio de 10x10 metros.

Los elementos de interacción son los elementos que intervienen en la trama a la hora de diseñar el juego. Esto es personajes, objetos, el diario, los encuentros con enemigos y los puntos de ruta. La interacción se llevará a cabo mediante las acciones entre personajes como diálogos, ataques o uso de objetos y hechizos. Los diálogos deberán estar previamente definidos, aunque el director de juego puede elegir en que momento se lanzan. Otros elementos, como encuentros con los enemigos no son necesarios, ya que un director puede crear encuentros enemigos en cualquier momento.

4.2.1 Comunicación con Neverwinter Nights

Para poder llevar a cabo la comunicación con Neverwinter Nights, se ha implementado una librería dinámica que conlleva el uso de 3 procesos diferentes que trabajan de manera conjunta:

- Un proceso externo que envía y recibe cadenas de texto a través de sockets.
- Una extensión del servidor que realiza una traducción entre la información del socket y la información del juego en curso.
- Un conjunto de scripts de NWN que ejecutan comandos y devuelven hechos.

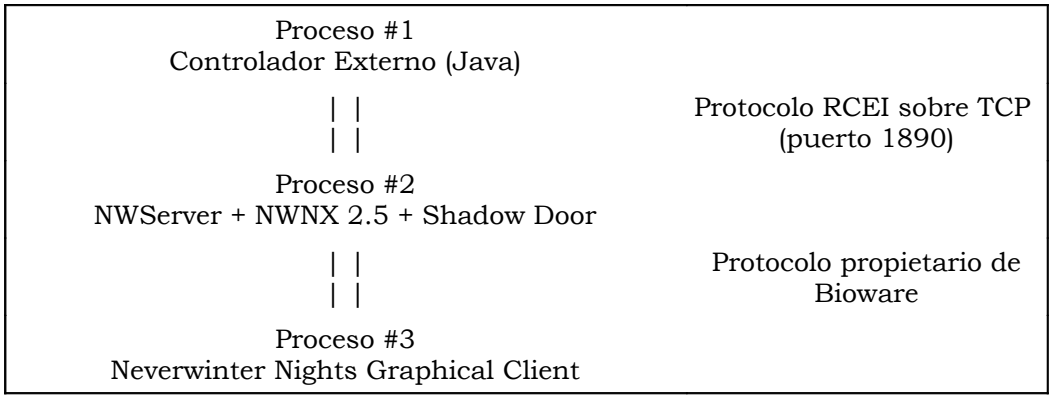


Figura 21. Esquema de interacción del motor de Neverwinter Nights con una aplicación externa.

Cuando un usuario inicia Neverwinter Nights Extender (NWNX2) con

la librería *nwnx_rcei.dll*, comienza a ejecutarse un nuevo servidor del juego, pero también abre un socket en el puerto 1890 y se pone a escuchar comandos entrantes. En la figura 21 se puede observar un esquema con las tres capas con las que se establece una conexión con el entorno.

4.2.2 Scripting en Neverwinter Nights

La interacción de todos los elementos de juego es compleja, ya que requiere programar scripts en lenguaje C, con funciones propias del motor de juego. Para mostrar las diferencias entre directores con interfaces digitales y con interfaces vocales no será necesario tener programados muchos scripts porque se supone la mayoría de las acciones deben ser realizadas por el director humano, y no de manera automática.

El módulo de juego de Neverwinter Nights debe tener un conjunto de scripts que han sido programados para poder ejecutar en el entorno todas las acciones que se han definido en el lenguaje RCEI, y que son enviadas al entorno mediante cadenas de texto a través de la librería nwnx_rcei.dll.

Este conjunto de scripts debe realizar dos funciones, por un lado ser capaces de decodificar cada uno de los comandos enviados y por otro ejecutar cada uno de ellos. Los scripts que ejecutan procesos en el entorno usan funciones propias del motor de Neverwinter Nights. A continuación se muestran los scripts más importantes.

Scripts para la comunicación con el entorno

El script *rcei_on_load*, que se muestra en la figura 22, se encarga de invocar a las funciones que inicializan la librería dinámica nwnx_rcei.dll para que pueda empezar a transmisor cadenas de caracteres mediante sockets.

```
#include "rcei_include"

void main()
{
    // Init placeholder for RCEI gateway
    RCEI_Init ();
    RCEI_SetVerbose ();
}
```

Figura 22. Script rcei_on_load, que inicializa la conexión usando la librería nwnx_rcei.dll.

El script *rcei_on_hearbeat* recibe los mensajes desde fuera del motor de Neverwinter Nights mediante la función *RCEI_GetNextToken* de la librería, y se encarga de ejecutarlos. Este script tiene dos funciones,

receiveCommands, que se encarga de recibir la cadena de texto con los mensajes y de ejecutarlos, y *sendFacts*, que se encarga de enviar se envían al exterior todos los hechos que se hayan producido en el entorno.

En NeverWinter Nights todos los elementos de un módulo de poseen un evento que se ejecuta cada seis segundos aproximadamente, aunque si este evento no lleva ningún script asociado, no se realiza ningúna acción. Para lograr una comunicación con todo el entorno se debe asociar *rcei_on_hearbeat* al evento *OnHeartBeat* de las propiedades del módulo. En el interior del script se distingue que elemento del entorno ejecuta la acción, así como la propia acción. Después de ello invocará un al comando del sistema *ExecuteScript* que ejecutará la acción concreta.

```
#include "rcei_include"
#include "rcei_parser"
#include "rcei_serializer"
#include "rcei_scripts"
#include "rcei_do_change"

// Action dispatch
void receiveCommands()
{
    // See if there is a message to process
    string msgCheck = RCEI_RequestMsg ();

    // Abort if nothing was returned.
    if (GetStringLength(msgCheck) == 0) { return; }
    // if no message, abort. 0 means no message. -1 means already processing a
message
    if (msgCheck == "0" || msgCheck == "-1" ) { return; }

    struct MessageList listMsgRCEI; struct RCEIMessage msg; string scriptName;

    string str=RCEI_GetNextToken();

    //Parse the string received by de DLL
    listMsgRCEI=parse(str);

    //store the list of commands in local variables of the module
    storeRCEIMsg(listMsgRCEI);

    ExecuteScript("rcei_log2",GetObjectByTag("beggar"));

    //Execute all the commands in the RCEI list
    int size=listMsgRCEI.size;
    while (size>0)
    {
        msg=getMsgFromList(listMsgRCEI);
        scriptName=getScriptName(msg.process);

        if (scriptName!="error")
        {
            ExecuteScript(scriptName,GetObjectByTag(msg.subject));
        }
        else
        {
            ExecuteSystem(rceiMsgList);
        }
```

```
        listMsgRCEI=removeMsgFromList(listMsgRCEI);
        storeRCEIMsg(listMsgRCEI);
        size-=1;
    }//end while */

    //clear all data structures
    listMsgRCEI=initMsgList();
    msg=initRCEIMessage();
    SetLocalString(GetModule(),"msg","");

    //Tell the DLL we are done with this message
    RCEI_ReleaseMsg();
}

// Send a RCEI message from NWN
void sendFacts()
{
    string facts=getFactList();

    if (facts!="")
    {
        RCEI_Send(facts);

        //We increase the message counter and clear de Message
        SetLocalString(GetModule(),"factList","");
        int factNumber=GetLocalInt(GetModule(),"factNumber");
        SetLocalInt(GetModule(),"factNumber",factNumber+1);
    }
}

/** OnHeartbeat appears to be called every 6 seconds or so,
  * so we need to increase the frequency manually.*/
void main()
{
    float delay=0.0;
    while (delay < 6.0)
    {
        DelayCommand (delay, receiveCommands());
        //receiveCommands();
        delay+=3.0;
    }

    //Frecuency sendig facts 1 time every 6 seconds
    DelayCommand (0.0, sendFacts());
    DelayCommand (2.0, sendFacts());
    DelayCommand (4.0, sendFacts());
}
```

Figura 23. Script *on_hearbeat*, con las funciones principales para recibir comandos y enviar hechos.

Scripts para procesar el lenguaje

Una vez que el motor de Neverwinter Nights reciba la cadena de caracteres con el mensaje en formato RCEI, se en se deben componer los comandos para que se ejecuten. Para ello es preciso tener un script con de un parser de RCEI y otro con un serializer para que componga los hechos a partir las estructuras de datos que se usan para almacenar internamente los hechos. Estos scripts son rcei_parser y rcei_serializer.

Script para ejecutar acciones

Estos scripts se usan para ejecutar procesos sobre elementos del entorno. Todos los posibles procesos se detallan en la gramática de RCEI. Los scripts son: *rcei_do_animate*, *rcei_do_attack*, *rcei_do_change*, *rcei_do_create*, *rcei_do_destroy*, *rcei_do_drop*, *rcei_do_equip*, *rcei_do_give*, *rcei_do_go*, *rcei_do_move*, *rcei_do_search*, *rcei_do_speak*, *rcei_do_sychronize*, *rcei_do_take*, *rcei_do_unequip*. En la figura 24 se muestra la implementación del script *rcei_do_attack*, que realiza un ataque sobre un personaje. La implementación del resto de este conjunto de scripts es similar.

```
#include "rcei_message"

// This method executes the action do_attack ()
void main()
{
    struct MessageList rceiMsg=loadRCEIMsg();
    struct RCEIMessage msg=getMsgFromList(rceiMsg);

    string sPC = msg.dirComp;
    object oPC = GetObjectByTag(sPC,0);
    int bPassive = TRUE;

    // and attack
    if (GetIsObjectValid(oPC) == TRUE)
    {
        ActionAttack (oPC, bPassive);
    }
}
```

Figura 24. Script *rcei_do_attack*, que realiza el ataque a un personaje.

Script para iniciar o terminar una sesión en el entorno

Para iniciar o reiniciar una sesión en el entorno se utiliza el comando begin, que está implementado en el script *rcei_do_begin*. Para terminar la sesión y acabar el juego se utiliza el script *rcei_do_end*. En la figura 25 se muestra el código del script *reci_do_begin*; el código para *rcei_do_end* es similar.

```
#include "rcei_synchronize"

void main()
{
    struct RCEIMessage msg=initRCEIMessage();
    msg.number = "0";
    msg.subject = "environment";
    msg.process = "speak";
    msg.dirComp = "Wellcome NeverWinter Nights. The connection with jRCEI has
been established";

    string area="rceiarea1";

    string factMessage=GetLocalString(GetModule(),"factList");
    factMessage+=sinchronize("begin",area);
    factMessage+="<fact> "+ getFact(msg)+ " </fact> ";
```

```
    //Store the fact
    SetLocalInt(GetModule(),"factNumber",0);
    SetLocalString(GetModule(),"factList",factMessage);
}
```

Figura 25. Script *reci_do_begin*, que inicia una sesión en el entorno.

4.3. Implementación del procesador del lenguaje

El lenguaje de comunicación con el entorno, como se ha comentado, está basado en un subconjunto de XML. Se implementará una solución genérica para construir procesadores de documentos XML utilizando uno de los estándares de procesamiento actuales más eficientes, SAX, combinado con una herramienta de procesadores de lenguajes, JavaCC.

La idea de esta propuesta surge a partir de trabajos relacionados con el procesamiento de documentos XML, como por ejemplo ANTXR (ANother Tool for XML Recognition), una herramienta construida sobre el entorno de desarrollo de procesadores de lenguaje ANTLR. Por tanto el desarrollo de este módulo podría reutilizarse posteriormente para otras aplicaciones.

El procesamiento dirigido por la gramática de un lenguaje basado en XML ofrece una serie de ventajas importantes. La principal es que el hecho de que sea la sintaxis la que dirija el procesamiento hace que la implementación se realice a alto nivel y sea muy sencilla de programar. Esto se contrapone con la programación de una tecnología como SAX. Aunque SAX tiene aspectos muy positivos en cuanto a eficiencia, puede resultar muy difícil de programar un procesador con esta tecnología si la gramática es muy compleja.

Un procesador de un lenguaje típico en una pasada consta principalmente de dos partes: un analizador léxico, y un traductor. El traductor acopla el resto de fases del procesador (análisis sintáctico, comprobación de restricciones contextuales y generación). La misión del analizador es suministrar los tokens al traductor en el momento que éste se lo pida. La misión del traductor es procesar el lenguaje mediante una gramática.

JavaCC es una herramienta que genera procesadores de lenguajes a partir de la definición de sus categorías léxicas y de la definición de la gramática del lenguaje. Por otro lado, SAX es una interfaz para el procesamiento de documentos XML mediante eventos.

En esencia lo que se propone en esta parte del trabajo es construir un parser sustituyendo la parte del scanner generado con JavaCC por uno propio construido con SAX. Por tanto será SAX el que proporcione los tokens al traductor.

En la programación guiada por eventos se dispone un procesador al cual deben añadirse uno o varios manejadores de eventos. Para que el desarrollador programe estos manejadores sólo dispone de interfaces con las funciones para manejar eventos de cada una de las partes del documento (inicio del documento, fin del documento, caracteres, espacios en blanco, elemento de inicio, elemento de final, etc.). Por tanto, para realizar la construcción del traductor con estos recursos hay que tener en cuenta los diferentes estados que puedan darse en el proceso de traducción e implementar una máquina de estados o algún mecanismo similar.

4.3.1. Conexión JavaCC con XML

Existen diferentes metodologías para procesar archivos XML, citadas en el apartado 2.4. Cada una de ellas presenta ciertas ventajas e inconvenientes para el procesamiento de documentos XML. SAX ofrece algunas características que pueden resultar útiles.

En primer lugar no requiere la creación de un modelo de datos, es la más eficiente en cuanto al procesamiento y tiene un uso de memoria muy bajo, es decir, potencialmente tiene un muy buen rendimiento. El modelo basado en eventos presenta el problema de tener una implementación compleja, ya que normalmente requiere el uso máquinas de estados para preservar el estado de procesamiento de los eventos.

La implementación que se ha realizado en este trabajo conecta SAX con JavaCC. La implementación se basa en las ideas usadas en ANTXR, una extensión construida sobre ANTLR (otro entorno de generación de traductores), y muestra que dichas ideas son fácilmente trasladables a cualquier entorno de construcción de procesadores de lenguaje. En la implementación propuesta, a través de JavaCC se construye un traductor que sirve para procesar un lenguaje que se corresponde con la estructura lógica de un lenguaje basado en XML.

Este módulo puede funcionar por si sólo, pero sólo para la estructura lógica de XML. La implementación de otras características, como la definición y uso de entidades o el procesamiento de comentarios serán manejadas por el marco SAX. Nótese que si se quisiera implementar estas funcionalidades programándolas directamente mediante la especificación de la gramática en JavaCC, se tendría que programar un código muy complejo.
La principal ventaja que tiene conectar JavaCC con una herramienta de procesamiento de XML es la capacidad de procesamiento de lenguajes con un modelo dirigido por la sintaxis. Este modelo permite definir procesadores con un alto nivel de abstracción. Esto es posible

gracias a la especificación en JavaCC de los archivos con extensión jj. Otra ventaja es que las características de SAX convierten a nuestro procesador en un procesador de XML completo.

Para poder obtener el comportamiento deseado implementaremos una arquitectura según el modelo cliente servidor, de manera que el traductor sea el cliente y el scanner el servidor. De esta manera será el traductor el que "tire" y el scanner el que "empuje".

Por tanto el scanner debe trabajar de modo independiente al traductor que se encuentra en el hilo principal del programa. Cada vez que el traductor requiera consumir un token, deberá solicitarlo al scanner. En este proceso el traductor debe de quedar bloqueado hasta que el scanner le devuelva el correspondiente token. La parte del servidor, es decir, el scanner, debe de permanecer bloqueada hasta que le llegue una petición. Posteriormente debe leer un token, ponerlo disponible y volver a bloquearse.

En conclusión, para poder implementar el procesador necesitamos que el scanner esté corriendo en un hilo distinto del programa principal. Además, para lograr la sincronización usaremos un buffer para que el scanner lo deposite ahí y el parser lo lea después.

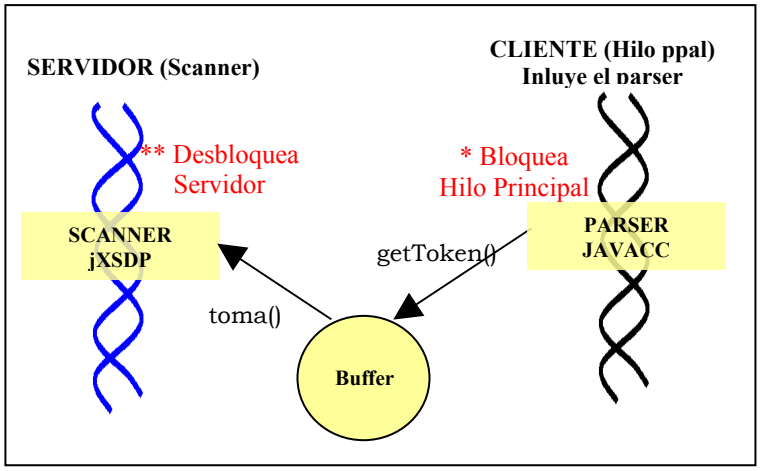


Figura 26. El parser solicita un token en un momento del procesamiento.

El comportamiento del procesador dirigido por la sintaxis es el siguiente. Primero se crean el traductor y el scanner, el traductor tiene una referencia del scanner. Después se ejecuta el hilo con el traductor, que inicialmente está bloqueado a la espera de una petición para procesar el siguiente token. Más tarde el traductor solicita un token cuando llega a un terminal. Al instante el scanner recibe la petición y se desbloquea el XMLReader. Cuando en el manejador de eventos se procesa un elemento, se coloca el elemento en el buffer y se le manda al hilo que se quede bloqueado a la espera de la siguiente petición. Por

último, el traductor lee el token del buffer y sigue el procesamiento. Si se solicita de nuevo un token se repite el mismo proceso.

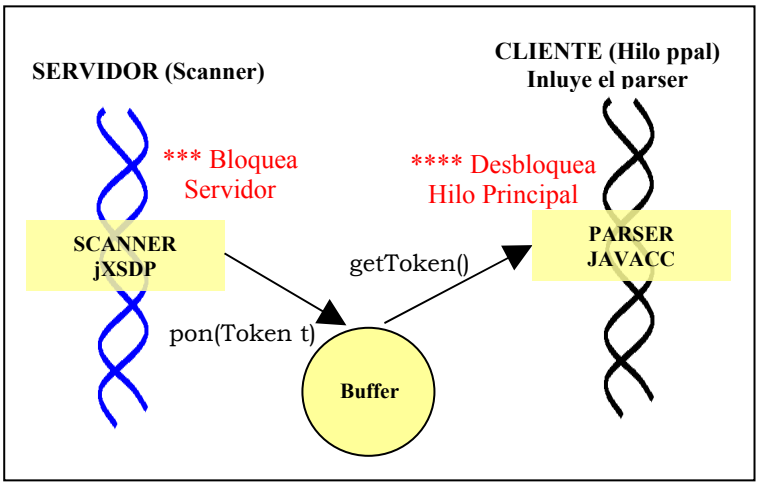


Figura 27. El scanner atiende la solicitud y devuelve el token.

Para lograr el objetivo de reemplazar el scanner generado con JavaCC por nuestra implementación particular, crearemos una clase con el mismo nombre que el TokenManager que implemente su interfaz. Nuestro módulo necesita sincronizarse con el traductor, pero la clase de SAX XMLReader posee un método parse() que procesa todo el documento y no por partes. Los tokens debemos obtenerlos desde el manejador de eventos que está dentro de XMLReader.

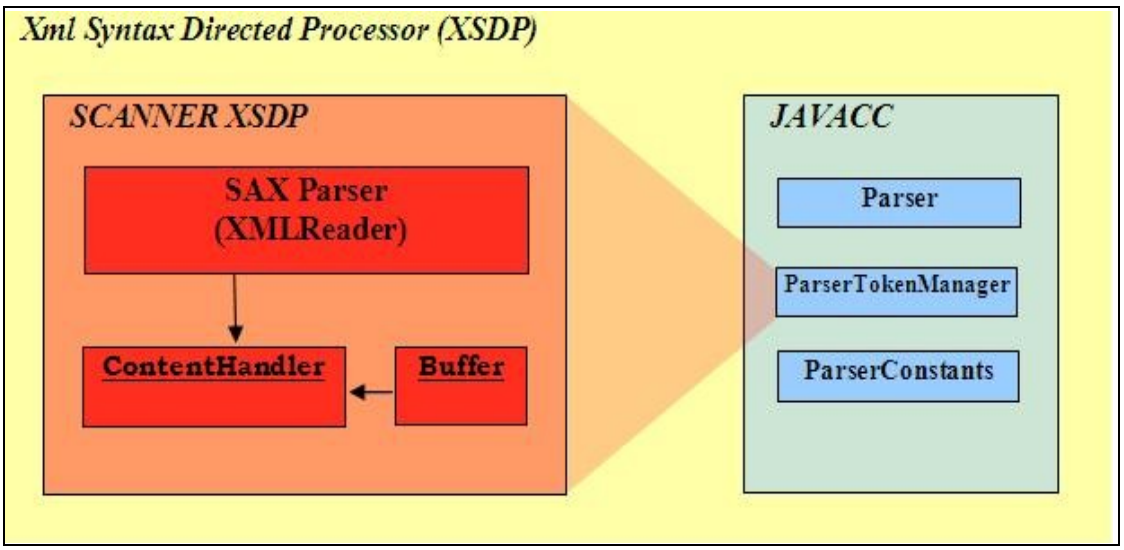


Figura 28. Diagrama del módulo de procesamiento de ficheros XML.

En conclusión se realiza un procesamiento dirigido por la gramática del lenguaje basado en XML. Por ello a la implementación del modulo del sistema se le ha llamado jXSDP, abreviatura de Java XML Syntax Directed Processor.

4.4. Control remoto de entornos virtuales mediante un interfaz manual

La interfaz manual de control remoto es una herramienta que permite el control remoto de un entorno, introduciendo mediante ratón y teclado los comandos que se quieren enviar al entorno (Neverwinter Nights) para que se ejecuten.

Esta interfaz de usuario hace uso del parser, del serializer y del módulo de conexión con entornos. Los que es estrictamente la interfaz es independiente y se podría usar con otra conexión a otro entorno y con otro parser, con lo cual se podría utilizar para implementar controladores para otros entornos distintos a Neverwinter Nights. La interfaz gráfica al igual que otras partes la aplicación está implementado en Java y ha sido desarrollada con Eclipse.

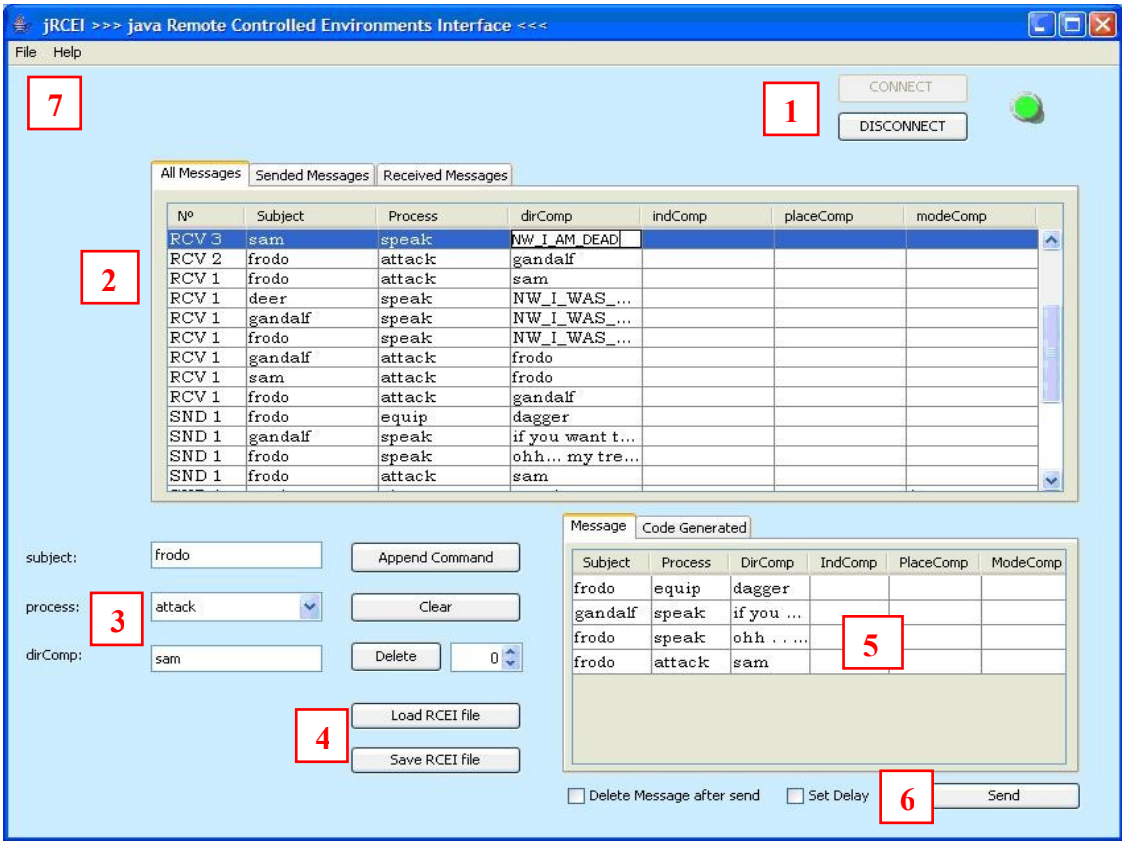


Figura. 29. Interfaz manual para el control remoto del entorno. Podemos crear comandos, enviarlos a NWN y ver la respuesta que nos devuelve.

La interfaz de usuario consta de una ventana con varios botones y paneles con los que establece una conexión, se componen comandos, se envían al entorno y se visualiza la información que proporciona el entorno. En la figura 29 se puede ver el aspecto de la interfaz manual. Cada una de las diferentes partes se ha numerado para a continuación proceder a su explicación.

En la parte superior tiene dos botones para conectar y desconectar la aplicación con el entorno. Para que se realice la conexión el entorno debe estar funcionando. En los paneles centrales se pueden ver los mensajes enviados y recibidos.

Con el resto de botones de la parte inferior se componen y envían los mensajes, y también para salvar y cargar mensajes guardados previamente en el formato XML de los mensajes RCEI.

1) Botones de conexión y desconexión con el entorno. Si pulsamos el botón *Connect*, se establecerá una conexión con un servidor de Neverwinter Nights. Si la conexión se establece correctamente, a la derecha del botón aparece una luz verde, sino permanece una luz roja. La causa más posible de un fallo de conexión es que no se haya arrancado Neverwinter Nights Extender. De la misma manera si hay una conexión establecida y se pulsa el botón *Disconnect* se cierra la conexión con el entorno y aparece una luz roja.

2) Panel de mensajes. En este panel hay 3 pestañas. La primera pestaña muestra todos los mensajes, tanto los que se han enviado al entorno, como los que se han recibido desde el entorno. La segunda pestaña sólo muestra los mensajes que se han enviado al entorno. La tercera pestaña sólo muestra los mensajes que han sido recibidos desde el entorno.

3) Paneles para componer el mensaje. Hay varias cajas de texto donde se puede rellenar el valor de los argumentos de un comando, es decir el sujeto, complemento directo, complemento indirecto, complemento de modo y complemento de lugar. El proceso se podrá seleccionar con un listBox. Según el proceso seleccionado sólo aparearán cajas de texto para los argumentos que realmente pueda llevar un proceso.

4) Botones para actuar sobre un comando. El primer botón es *Append* y sirve para añadir un nuevo comando al mensaje que se está componiendo, ya que un mensaje puede tener varios comandos. El segundo botón es *Clear*, y sirve para eliminar todos comandos que se han añadido a un mensaje. El siguiente botón es *Delete*, que sirve para borrar un solo comando de un mensaje para no tener que eliminarlo entero, sino sólo una parte. Al lado tiene un textBox para indicarle el índice dentro del mensaje. Por último tenemos los botones *Save RCEI file* y *Load RCEI file*, que sirven para guardar un mensaje que acabamos de componer en un fichero XML o para cargar un fichero XML.

5) Panel de mensaje actual. En este panel sirve para representar un mensaje que se está componiendo o que se ha cargado. Posee dos pestañas distintas. En la primera pestaña se muestra una tabla, en la que en cada fila representa un comando. Cada una de las columnas se corresponde con un parámetro de un comando, es decir, sujeto, proceso y complementos. En la segunda pestaña aparece la representación en lenguaje RCEI del mensaje que aparece visualmente en la otra pestaña.

6) Botón de envío. Este botón se encarga de enviar al entorno el mensaje que haya actualmente en el panel de mensaje actual. Una vez que se envía, se añade el mensaje al panel de mensajes de arriba, además de enviarse al entorno. Hay 2 checkbox que se pueden seleccionar antes de enviar un mensaje. Uno es para establecer un retraso de un segundo antes de enviarlo, por si el cambio en el entorno se produce demasiado rápido para que el usuario lo vea, y el otro es para eliminar el mensaje del panel del mensaje actual una vez que se ha enviado.

7) Barra de menú. En la barra de menú se puede cargar o cerrar un entorno. Cuando se cierra un entorno se elimina todos los botones y paneles de la pantalla. Cuando se carga se crean todos. Esta opción sirve para en el caso de que se creen conexiones a otros entornos distintos, se pueda cargar una configuración de pantalla distinta. Además hay un botón con información general de la aplicación.

4.5. Dirección mediante el interfaz vocal

El escenario propuesto para el prototipo será una pequeña aventura en Neverwinter Nights basada en el argumento que se menciona en el punto 4.4, donde se puede comparar las diferencias de una aventura con un director humano mediante una interfaz manual en el cliente Dungeon Master, y una aventura dirigida con una interfaz vocal.

En la implementación se ha realizado un diálogo automático a través de varios menús de diálogo. En el menú principal se han establecido cinco tipos de procesos: buscar objetivos, matar personajes, crear objetos y personajes, abrir y cerrar puertas, y controlar personajes del juego.

En la figura 30 se puede observar la gramática del menú principal, contenida en el archivo menu.gram. Cada tipo de proceso tiene asociada información que el DialogManager utilizará para saber a que menú debe realizar la transición y que gramática asociada lleva ese menú. El resto de gramáticas de los demás submenús reconocerán a personajes y elementos del juego.

```
#JSGF V1.0;

/**
 * JSGF Grammar for dialog manager example create enemy
 */

grammar menu;

public <command> = search target          { goto_search } |
                   kill people            { goto_kill } |
                   opening door           { goto_open } |
                   make entity            { goto_entity } |
                   control people         { goto_control } |
                   show statistics        { stats } |
                   exit the program       { exit } |
                   get help               { help } ;
```

Figura. 30. Gramática JSGF del menú principal.

En las pruebas que se han realizado al realizar la implementación nos hemos dado cuenta que una de las causas de que exista una tasa elevada de fallos en el reconocimiento de palabras se debe a la similitud fonética entre diferentes palabras. Por ejemplo si se quiere buscar al rey, se deberá pronunciar “search king”, pero si por ejemplo queremos buscar a un niño y pronunciamos “search kid”, en un gran número de ocasiones el reconocedor reconoce “search king” incorrectamente.

Por tanto, en todos los menús de acciones se han elegido conjuntos de palabras fonéticamente diferentes para que el menor número de similitudes fonéticas, para que el conjunto de palabras de cada menú sea bastante heterogéneo, desde el punto de vista fonético, y evitar así problemas en el reconocimiento.

A continuación se muestran las frases que se han probado con éxito en cada los diferentes submenús de diálogo:

- Search
  - search children
  - search beggar
  - search Evandro
  - search gypsy woman
  - search Rita
  - search pretty wife
  - search girl
  - search councilor
  - search first guard
  - search second guard
  - search courtship man
  - search first soldier
  - search second soldier

- search king
- search treasurer
- search Mondale Hallenbeck

- Open
  - opening first door
  - opening second door
  - opening third door
  - opening castle door

- Close
  closing first door
  closing second door
  closing third door
  closing castle door

- Create
  - make money
  - make dagger
  - make creature
  - make monster

- Change wather
  - make rain
  - make snow
  - make cloudy

- Kill
    - death children
    - death beggar
    - death gypsy woman
    - death Rita
    - death pretty wife
    - death girl
    - death councilor
    - death first guard
    - death second guard
    - death courtship man
    - death first soldier
    - death second soldier
    - death king
    - death treasurer
    - death Mondale Hallenbeck

- Control
    - control children
    - control beggar

- control Evandro
- control gypsy woman
- control Rita
- control pretty wife
- control girl
- control councilor
- control first guard
- control second guard
- control courtship man
- control first soldier
- control second soldier
- control king
- control treasurer
- control Mondale Hallenbeck

La interfaz vocal obviamente no tendrá las mismas posibilidades que la interfaz manual de Neverwinter Nights, ya que algunas de las opciones que incorpora el cliente Dungeon Master. Si se llegaran a implementar con voz muchas de las opciones que tiene la interfaz de Neverwinter Nights, serían demasiado complicadas para utilizarse con soltura en la dirección de una partida.

Intervenciones del Director de juego sin el interfaz vocal

En el interfaz manual debemos poder realizar varias acciones para esta aventura. A continuación se muestran las acciones que se pueden realizar.

- Crear nuevos objetos y enemigos para darle emoción a la aventura.

- Dormir a un jugador, para por ejemplo evitar que la IA haga que un personaje ataque al jugador.

- Matar a un jugador para poder evitar que un enemigo con una puntuación muy alta tarde mucho en morir y pueda cansar al jugador.

- Suplantar a un personaje para poder actuar y hablar por él.

- Sanar a un personaje para el caso de que el jugador estuviera hiriendo a un personaje necesario para resolver la trama, buscar por nombre, para localizar a un jugador o un personaje.

- Buscar por nombre un personaje escribiendo la etiqueta que lo identifica.

- Manipular el estado de las puertas, pudiendo abrirlas, cerrarlas, o incluso bloquearlas para que nadie pueda usarlas y también desbloquearlas.

Intervenciones de Director de juego con el interfaz vocal

- Buscar un jugador por nombre: pronunciando el comando search junto con el nombre del jugador para localizarlo a través de todas las pantallas.

- Buscar un objeto por nombre: pronunciando el comando search junto con el nombre del objeto para localizarlo a través de todas las pantallas.

- Matar a un personaje: para poder matar a un personaje con sólo pronunciar el comando kill y el personaje que se quiere matar.

- Abrir puerta: para abrir una puerta usando el comando open.

- Cerrar puerta: para cerrar una puerta usando el comando close.

- Controlar un personaje: para poseer un personaje y poder actuar a través de él.

- Pause: para parar un juego sin tener que buscar el comando en los menús.

5. Discusión

En este apartado se realizará un análisis de la metodología aplicada en el desarrollo de la aplicación y los resultados que se han obtenido.

5.1 Metodología

En lo que se refiere al motor de juego, se ha elegido Neverwinter Nights por varias razones, la principal es que Neverwinter Nights permite a los usuarios crear y editar sus propias aventuras de manera muy sencilla. Se puede crear una aventura en muy poco tiempo sin preocuparnos de muchos de los aspectos gráficos de los cuales deberíamos ocuparnos si usásemos un motor gráfico a más bajo nivel, y no un motor de juego. Además programar eventos en escenarios es también sencillo debido a la potencia que nos proporciona su lenguaje de scripting. En lo que se refiere al aspecto gráfico el resultado es muy vistoso, los efectos y la jugabilidad hacen de Neverwinter Nights un juego muy atractivo. Por otro lado debido a la gran popularidad del juego, existe una amplia comunidad de usuarios en Internet, de donde se puede extraer información que se puede aprovechar para crear nuestros propios módulos, como escenarios, personajes, scripts complejos, otros módulos, librerías o cualquier otro componente.

Desde este punto de vista nos podríamos haber planteado usar Neverwinter Nights 2 que es más reciente y los efectos gráficos son mucho mejores. Lo negativo es que Neverwinter Nights 2 tiene unos requisitos técnicos muy grandes, además aunque es mucho más complejo y tiene varias mejoras respecto a Neverwinter Nights, determinadas partes del editor no son compatibles con las de su antecesor, como por ejemplo el scripting. Muchos de los scripts de Neverwinter Nights no funcionan en Neverwinter Nights 2, por lo que no se pueden reutilizar íntegramente módulos ya creados en Neverwinter Nights, aunque sí algunas partes.

Con otros motores de juego que se podrían haber considerado para realizar una aventura de las características que se han planteado, no se podrían editar módulos de juego de manera tan sencilla. Además tanto Neverwinter Nights como Neverwinter Nights 2 tienen el módulo Dungeons Master, que permite jugar partidas en red con un director de juego, lo cual es permite comparar está opción con la opción de la interfaz vocal.

En lo referente al parser que se ha implementado, primero se implementó con una herramienta de construcción de compiladores, JavaCC. Posteriormente, después de investigar tecnologías XML, se

realizó modificó la implementación para aplicar estas tecnologías y conseguir una manera más genérica y eficiente de poder procesar sublenguajes de XML.

Este enfoque ofrece dos grandes ventajas. La primera es que facilita la construcción y mantenimiento de procesadores basados en SAX. Esto es debido a que la implementación se especifica mediante esquemas de traducción formulados sobre la gramática. Para realizar el procesamiento de los datos solo debemos almacenar los valores de los terminales de las producciones, y mediante gramáticas de atributos, propagar y si fuera preciso transformar estos valores. La segunda es la herencia de todas las características comunes de cualquier lenguaje XML, como la cabecera del documento XML, los comentarios o la gestión de entidades.

Por tanto, al pasar de una programación guiada por eventos a una especificación de alto nivel mediante un esquema de traducción orientado a la traducción predictivo-recursiva se consigue un doble objetivo. Por un lado se simplifica bastante la definición del parser que se pretende generar y por otro proporcionamos indirectamente toda la funcionalidad de XML a nuestro lenguaje.

En lo que respecta al reconocedor de voz, se puede decir que Sphinx presenta algunas ventajas, pero también algunos inconvenientes. El principal inconveniente en el reconocimiento de voz es que se produce una tasa de fallos alta en el reconocimiento de voz, no solamente al no reconocer palabras porque no detecte bien la señal de audio, sino porque muchas veces se confunden palabras fonéticamente similares. No se puede decir que esto sea un fallo se Sphinx, sino que es una de las complejidades que presenta el reconocimiento de voz en general.

En la implementación se ha trabajado con los dos modelos acústicos que proporciona la distribución de Sphinx. El modelo TIDIGITS está preparado para reconocer números y es más simple, el modelo Wall Street Journal es mucho más amplio y cuenta con un diccionario de cinco mil palabras, suficiente para poder reconocer una gran combinación de palabras, es decir muchas frases diferentes.

Sin embargo el trabajar con un conjunto tan grande de palabras es uno de los factores que hace que se puedan producir conflictos entre palabras fonéticamente similares. Una posibilidad para reducir este problema sería crear un modelo acústico propio con un conjunto de palabras más reducido acorde con nuestras necesidades. Con SphinxTrain se podría llegar a realizar un modelo acústico apropiado a las necesidades del sistema creando un repertorio de órdenes reducido, lo que probablemente acotaría el problema de reconocer palabras, al ser significativamente más pequeño el conjunto de

palabras. Por otro lado, el uso de SphinxTrain es complejo y puede llevar cierto tiempo, ya que requiere una instalación en Linux, y además requiere la realización de un entrenamiento con todas palabras del modelo acústico.

Cuando se empezaron a probar gramáticas con Sphinx se intentaron reconocer en primer lugar palabras sueltas, pero presentaban muchos más problemas de reconocimiento que frases con varias palabras. Esto se debe a que mientras más palabras tenga la frase, se hace más identificable en el conjunto de frases de la gramática. Por las pruebas realizadas, las frases con dos y tres palabras dan mejores resultados que frases con una sola palabra. Aunque si aumentamos más el número de palabras, sobre todo si la gramática tiene un número elevado de frases, la tasa de reconocimiento es menor, ya que el usuario tiene más posibilidad de pronunciar mal la frase y el sistema puede tener un mayor conflicto para identificar unas frases con otras. En la distribución de Sphinx se incluye un ejemplo para reconocer un conjunto de setenta y dos frases de diferente longitud, que van desde frases con tres palabras a frases con quince palabras. Los resultados no son demasiado buenos debido a estos motivos y en una gran mayoría de casos se reconoce otra frase.

Una de las ventajas de usar la voz para dirigir un proceso de narración interactiva es que puede facilitar ciertas acciones, además de mejorar la interacción por el entorno, y por tanto hacerlo más atractivo para el usuario. Existe también un motivo muy importante para considerar de cara al futuro, cuando este tipo de sistemas estén más perfeccionados, y es que velocidad con la que se puede llegar a enviar información mediante voz es mucho mayor que la que se puede enviar mediante teclado. En el sistema Dragon NaturallySpeaking, la velocidad para realizar algunas tareas mediante voz es tres veces superior a la que se puede llegar a realizar con el teclado. Por lo tanto, el reconocimiento de voz puede llegar a incrementar mucho la productividad de un director de juego.

Lo cierto es que actualmente muy pocos sistemas utilizan la voz para interactuar con el sistema, de los que se han visto en el apartado 2, sólo I-Storytelling (Cavazza, 2004) utiliza la voz como parte de la interacción con el entorno virtual, pero no tiene una distribución libre disponible para los usuarios. El sistema Façade (Mateas, 2005) es un muy útil para comprobar la reacción emocional de un jugador con personajes simulados, y en él se usa un sistema para convertir el texto en voz. La comunicación del usuario con los personajes se realiza mediante teclado, aunque sería muy interesante que existiera una interfaz vocal para que el usuario se pudiera comunicar, aunque incrementaría la complejidad del sistema.

En la industria de los videojuegos todavía no se le ha sacado mucho partido al reconocimiento de voz, aunque existen algunos juegos que lo han incorporado, como Odama y Mario Party (Nintendo, 2006), apenas reconocen un conjunto pequeño de instrucciones para indicar movimientos. Actualmente hay propuestas de juegos que saldrán próximamente al mercado, como Tom Clancy´s HAWX, que prometen explotar más la interacción mediante voz que juegos anteriores. Esto puede deberse a que al ser un juego de simulación de aviones con alto grado de realismo, por un lado los complejos controles de un avión necesitan más componentes de interacción para no tener que hacerlo todo mediante ratón y teclado. Por otro lado, el intento constante que hace la industria de los videojuegos de imitar o incluso superar la realidad.

5.2. Resultados

Los resultados del trabajo que se pueden discutir son principalmente dos. El primero es el de la encuesta que se ha realizado entre usuarios de juegos de rol y entre directores de juegos de rol. Aunque las encuestas no tienen un número muy elevado de encuestados, es un número suficiente y representativo para poder sacar conclusiones.

En primer lugar, el perfil de un jugador de rol es el de un hombre de entre 18 y 35 años con un nivel de estudios alto. Esto lo podemos deducir de las tres primeras preguntas de la encuesta de jugadores, donde el 96,4 % son hombres, el 87,7% tiene un nivel de estudios universitarios y no hay nadie menor de 18 años ni mayor de 35.

Del grupo de jugadores prácticamente todos tienen un cierto nivel de experiencia, y un 14,3 % son jugadores muy experimentados. Los juegos más utilizados son el Neverwinter Nights y el Neverwinter Nights 2, que cuentan con una amplia comunidad y son más recientes. Sin embargo el Vampire The Masquerade: Redemption y el Dungeon Siege son relativamente antiguos y aunque más de la mitad han jugado alguna vez o durante un tiempo no lo hacen muy a menudo.

En lo referente a los directores de juego, se puede decir, que el 75% de los encuestados había dirigido partidas de rol de lápiz y papel antes de dirigirlas en juegos de rol por ordenador, aunque con distinto grado de experiencia. Este porcentaje disminuye un poco en directores de juegos de rol por ordenador; algo más del 60% tiene experiencia en la dirección. Además un 25% del total se declara jugador asiduo dirigiendo partidas de rol por ordenador. Al igual que ocurre en la encuesta de jugadores, Neverwinter Nights y Neverwinter Nights 2 son los juegos más populares. Aunque llama la atención que un 40% de los encuestados ha dirigido partidas en Vampire The Masquerade:

Redemption, esto es debido a que como se indicaba en el apartado 2.1.2 este juego tiene una opción de narrador para dirigir partidas bastante buena, el inconveniente principal como se ha indicado anteriormente es que el juego no goza de mucha popularidad al tener cierta antigüedad.

En cuanto al tiempo de aprendizaje para dirigir una partida, la mayoría de los usuarios encuestados apunta a que es un tiempo alto, la mayoría indica cifras alrededor de un mes, aunque algunos pocos indican un tiempo de aproximadamente una semana.

Por otro lado, los usuarios consideran que la mayoría de acciones que se realizan con la interfaz manual no son complejas, de hecho de las acciones preguntadas la mayoría considera que buscar un objeto en un escenario, crear objetos y criaturas, dar, coger, y revivir presentan un poco de complejidad con los interfaces manuales, mientras que buscar por nombre, abrir, cerrar, examinar, atacar, sanar y mover no presentan apenas ninguna dificultad para una gran parte de los encuestados.

La parte más interesante de la encuesta es la utilidad que los directores piensan que pueden tener órdenes vocales para realizar acciones. En todas las acciones más de la mitad de los usuarios piensa que las órdenes vocales pueden tener utilidad aunque sea poca. Sólo en un caso de una de las acciones el 50 % piensa que no tiene ninguna utilidad, esta es la de crear objetos y criaturas. Esto tiene bastante sentido ya que para poder indicar la posición exacta donde se quiere crear un objeto siempre será mucho más cómodo y preciso el uso del ratón. También las acciones descansar, mover y revivir poseen una porcentaje de personas muy elevado (41,7%) que creen que no tiene ninguna utilidad.

En el extremo opuesto se encuentran las acciones buscar un objeto en un escenario, e ir a, de las cuales un 50% considera que son bastante o muy útiles. Otras acciones tienen porcentajes menos elevados, pero significativamente altos, de usuarios que piensan que las acciones son bastante o muy útiles, como son buscar por nombre con un 41,7%, examinar con un 33,4% y abrir y cerrar, con un 33,3 %.

Hay otro grupo de acciones las cuales cerca del 50% piensan que son poco o medianamente útiles, estas son: matar, dar, coger, poseer, atacar, sanar. Una de las conclusiones que se puede sacar es que como los usuarios consideran que estas acciones son fáciles de realizar con una interfaz manual, algunos no consideran que pueda ser muy importante realizar esta acción mediante voz, pero sin embargo otros pueden pensar que quizás la realización de estas acciones con voz pueda ser igual de sencillo, pero con la comodidad de

no tener que usar el ratón.

En conclusión se puede decir que la mayoría de usuarios piensa que sería positivo poder combinar voz con teclado y ratón como se refleja en la última pregunta de la encuesta de directores, donde un 66,6% está bastante o muy de acuerdo, un 20% poco de acuerdo y sólo un 13,3% no está nada de acuerdo. Del análisis sobre la utilidad de los diferentes comandos se podría deducir que todas las acciones que impliquen búsquedas sobre listas de objetos y personajes o sobre diferentes escenarios son más útiles porque son más complejas y llevan más tiempo a los directores. En cambio todas aquellas acciones que son muy directas con un simple clic, como descansar y mover o acciones que necesitan parámetros que no son fijos ni calculables, como crear objeto, no son útiles. En crear objeto o personaje se necesitan de las coordenadas de posición, que no se pueden indicar fácilmente a porque no se conocen, sin embargo con un clic de ratón se soluciona muy fácilmente.

El segundo resultado es que se ha logrado realizar una la implementación de una interfaz vocal para dirigir un proceso de narración. Se podrían haber utilizado otras con la misma funcionalidad. Por ejemplo, para el lenguaje de comunicación se podría haber definido un lenguaje que no fuera un subconjunto de XML, o se podrían haber utilizado otras herramientas en vez de JavaCC, como LEX o YACC. Para el analizador de voz se podrían haber usado otras aplicaciones como Julius o Dragon NaturallySpeaking. Incluso para el entorno se podrían haber usado otros juegos como, como Vampire The Masquerade: Redemption, Neverwinter Nights 2 o Unreal Tournament. La parte que podría ser más complicada para la implementación del sistema en otros entornos es precisamente la conexión con el entorno, para el envío y recepción de información y la programación de los eventos del entorno.

Las herramientas que se han utilizado son de software libre y han conseguido el objetivo de poder reconocer una orden vocal y enviarla al entorno para que se ejecute. Aunque funciona en tiempo real, debido a la carga de tener funcionando en un mismo equipo el juego más el servidor de del juego, más el entorno de desarrollo, en ocasiones el sistema puede ralentizarse y no funcionar de manera óptima. Por tanto, el funcionamiento del reconocedor podría refinarse para que funcione de manera más optima y ocupando menos memoria. Para ello se podría estudiar como dividir las diferentes partes del reconocimiento en varios hilos que distribuyan la carga de manera más eficiente.

6. Conclusiones

Los interfaces vocales aportan a los entornos virtuales un nuevo medio para interactuar con el entorno, la voz. Este medio no debe pretender sustituir a los elementos clásicos de interacción, como el reconocimiento de eventos mediante ratón o teclado o el procesamiento de textos en el caso de aventuras textuales. En el futuro los interfaces de voz pueden dar lugar a entornos multimedia donde la voz juegue un papel relevante. Este es el caso de los entornos virtuales de nueva generación, donde el usuario tendrá que acceder a información y en los que los interfaces de voz podrían complementar a los interfaces de gestión de movimiento. O incluso en videojuegos donde los usuarios además de usar solamente comandos mediante teclado puedan participar en una parte del juego mediante órdenes de voz.

El uso de voz para dirigir un proceso de narración interactiva puede ser un componente que facilite la dirección, pero no por si sólo. El reconocimiento de voz es un campo de investigación muy amplio, con muchos problemas por resolver y muchas cosas que mejorar, principalmente debido a que es muy complejo. Por el contrario, la generación de voz a parir de texto es un campo de menor complejidad, en el cual la mayoría de los problemas ya han sido resueltos. El reconocimiento de voz tiene una tasa de fallos elevada ya que transformar una onda de voz, que representa una palabra, en texto admite un gran número de variaciones dependiendo de muchos factores, como el tono, la intensidad, la velocidad o la pronunciación. Por tanto, sistemas interactivos basados exclusivamente en interfaces vocales no serían muy útiles, ya que aunque algunas órdenes vocales facilitan el uso de ciertas acciones, otras hacen todo lo contrario. Sin el resto de elementos de interacción manual el sistema quedaría demasiado pobre, por lo que la interacción en el mismo sería compleja y ciertamente incómoda e ineficiente para los usuarios.

En este trabajo se ha diseñado un sistema con todos los componentes necesarios para poder realizar una narración dirigida por voz. En la implementación que se ha realizado se han particularizado los diferentes componentes con el uso de tecnologías o herramientas concretas, además de un entorno virtual particular. Lo importante es que en el diseño se definen los tres componentes necesarios para tener una arquitectura que permita llevar a cabo un proceso de narración dirigida por voz. Estos componentes son un analizador de voz, un procesador del lenguaje de comunicación y un módulo de conexión con el entorno, además del propio entorno.

Se ha demostrado que se puede implementar un sistema de reconocimiento de voz aplicado a un juego real. Se ha logrado realizar

una conexión de Neverwinter Nights con una aplicación en Java, que es capaz de enviarle información de manera estructurada, con un lenguaje propio, que el motor de Neverwinter Nights es capaz de recibir, interpretar y ejecutar. La aplicación Java es capaz de reconocer secuencias de palabras mediante voz, utilizando el armazón Sphinx. Por tanto, se ha desarrollado un prototipo que es capaz de recoger órdenes vocales y ejecutarlas en el entorno Neverwinter Nights. Además de ser factible, excepto el entorno y la librería dinámica que se ha creado para conectar con el entorno, todo el código ha sido desarrollado en Java con herramientas de libre distribución, por lo que el prototipo que se ha desarrollado es de código libre, lo que permite que pueda ser continuado y ampliado por otros usuarios o investigadores que estén interesados en el proyecto. Debido a que la gestión de voz es un proceso muy complejo en cual intervienen muchos factores, actualmente el reconocimiento presenta una tasa de fallos elevada, sobre todo si se quieren reconocer muchas palabras seguidas. Para que se fueran añadiendo estas interfaces de una manera activa sería necesario mejorar las tecnologías de reconocimiento de voz. Por tanto el mayor problema es la precisión de las órdenes realizadas mediante la voz, que dificultan una interacción fluida.

En este trabajo se ha logrado demostrar que es tecnológicamente posible realizar una implementación de una interfaz vocal para la dirección de la narración. Como trabajo futuro quedaría probar la dirección de partidas en red con varios jugadores reales. En esta experiencia se podría realizar un estudio que contrastase las diferencias entre una partida en red con la interfaz manual con otra usando la interfaz vocal. De esta manera se podría medir de manera global los beneficios del uso de la interfaz vocal.

Además se debería ampliar el repertorio de órdenes vocales, para que se pudieran ejecutar más comandos en el entorno. También se podría investigar la viabilidad de usar otros entornos que no fueran de rol. En el estado del arte se cita el juego de aviones Tom Clancy´s HAWX, donde se comenta que mediante el uso de voz se permiten controlar grupos de aviones para que ataquen, lancen bombas, se replieguen, etc. Sería interesante comprobar como afecta a la jugabilidad en juegos complejos, no necesariamente desde el punto de vista del director de juego, sino del jugador.

Referencias

Active Worlds (2008, agosto). Home of the 3D Chat, Virtual Worlds Building Platform.
http://www.activeworlds.com/

ANTXR: Easy XML Parsing, based on the ANTLR parser generator (2008).
http://javadude.com/tools/antxr/index.html

Apache XMLBeans. The Apache XML Project.
http://xml.apache.org/xmlbeans

Avlis Team. Neverwinter Nights Extender 2.6.1.
http://www.nwnx.org

Bioware (2001). Neverwinter Nights Diamond.
http://nwn.bioware.com

Birbeck M., Diamond J., Duckett J., Gudmundsson O.G., Kobak P., Lenz E.: Professional XML 2nd Edition. Programmer to Programmer. Wrox Press Ltd, Arden House, 1102 Warwick Road, Acocks Green, Birmingham, B27 6BH, UK.

Blizzard Entertainment (2002). Warcraft III.
http://www.blizzard.com/us/war3/

Booker, C. (2004). The Seven Basic Plots: Why we tell stories. Ed Continuum.

Carbonaro M., Cutumisu M., Duff H., Gillis H., Onuczko C., Siegel J., Schaeffer J., Schumacher A., Szafron D., Waugh K. (2008). Interactive Story Authoring: A Viable Form of Creative Expression for the Classroom. Computers and Education. Supplementary materials.
http://www.cs.ualberta.ca/~script/docs/2007CompEd.pdf

Cavazza M., Martin O., Charles F., Mead S. J, Marichal X. (2003). Interacting with Virtual Agents in Mixed Reality Interactive Storytelling. School of Computing and Mathematics, University of Teesside, United Kingdom. Laboratoire de Télécommunications et Télédétection, Université catholique de Louvain, Belgium.
http://www-scm.tees.ac.uk/f.charles/publications/conferences/2003/IVA03_Cavazza.pdf

Cavazza M., Lugrin J.L., Pizzi D. (2004). Madame Bovary on the Holodeck: Immersive Interactive Storytelling. University of Teesside, Middlesbrough.
http://www-scm.tees.ac.uk/users/f.charles/publications/conferences/2007/mm07_cavazza.pdf

Cavazza M., Charles F., Pizzi D. (2008). I-Storytelling. Intelligent Virtual Environments Research Group.
http://www-scm.tees.ac.uk/users/f.charles/

Charles F., Cavazza M., Steven J. M., Martin O., Nandi A., Marichal X. (2004). Mixed Reality Interactive Storytelling. University of Teesside, Middlesbrough.
http://www-scm.tees.ac.uk/users/f.charles/publications/conferences/2004/Charles_ACE2004.pdf

Cutumisu M., Onuczko C., McNaughton M., Roy T., Schaeffer J., Schumacher A., Siegel J., Szafron D., Waugh K., Carbonaro M., Duff H., Gillis S. (2007). ScriptEase: A Generative/Adaptive Programming Paradigm for Game Scripting. Science of Computer Programming.
http://www.cs.ualberta.ca/~script/docs/2007SCP.pdf

Definiciones sobre entornos virtuales I (2008, agosto).
www.glasgowschoolofart.ac.uk/gsa.cfm

Definiciones sobre entornos virtuales II (2008, agosto).
www.fas.org/spp/military/docops/usaf/2020/app-v.htm

Enhydra Zeus. Open source Java-to-XML Data Binding tool.
http://forge.objectweb.org/projects/zeus/

Forterra Systems Inc (2008, agosto). Transforming Enterprise Processes through Virtual Worlds.
http://www.forterrainc.com/

Gálvez Rojas S., Mora Mata M.A.: Java a tope: Traductores y Compiladores con LEX/YACC, JFLEX/CUP y JavaCC. Universidad de Málaga. Edición electrónica.

Gygax G., Arneson D. (2008). Dungeons and Dragons: Players Handbook: Vol 3.5. Ed Stark.

Habbo Hotel (2008, agosto).
http://www.habbo.com/

Hidden Markov Model Toolkit (2008).
http://htk.eng.cam.ac.uk/

Hidden ISIP Toolkit (2008).
http://www.isip.msstate.edu/project/speech/

Huebner R. (2000, august). Nihilistic Software's Vampire: The Masquerade -- Redemption. Gamasutra.
http://www.gamasutra.com/view/feature/3147/nihilistic_softwares_vampire_the_.php?

Java Architecture for XML Binding (JAXB)
http://jaxb.dev.java.net

Sun (2008). Java Compiler Compiler (JAVACC): The JAVA Parser Generator.
http://javacc.dev.java.net

Java Speech Grammar Format (2008, agosto).
http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/

Julius (2008).
http://julius.sourceforge.jp/en_index.php

Loiseaux J. (2004, march). Vocal interface to a computer animation System. BSc (Honours) Computer Studies. School of Computing, Middlesbrough.
http://loiseaux.free.fr/report.pdf

Manual Aurora en español (2008, julio).
http://www.terra.es/personal5/oupmfebh/ocio/nwn/nwn_00_indice.htm

Magerko B. (2002). A Proposal for an Interactive Drama Architecture. University of Michigan.
http://gel.msu.edu/magerko/papers/magerko.AAAI.2002.pdf

Magerko B., Laird J.E. (2003). Building an Interactive Drama Architecture. International Conference on Technologies for Interactive Digital Storytelling and Entertainment. Darmstadt, Germany.

Mateas M., Stern A. (2005). Structuring Content in the Façade Interactive Drama Architecture. Artificial Intelligence and Interactive Digital Entertainment Conference.
http://www.interactivestory.net/papers/MateasSternAIIDE05.pdf

Mead S., Cavazza M., Charles F (2003). Influential Words: Natural Language in Interactive Storytelling. University of Teesside, Middlesbrough.
http://ive.scm.tees.ac.uk/beta/data/media/HCI03_Mead.pdf

Microsoft Game Studios (2002). Dungeon Siege.
http://www.microsoft.com/games/DungeonSiege/

Microsoft .NET Framework (2008).
www.microsoft.com/NET

Modelos ocultos de Markov (2008, agosto). Teoría y definiciones.
http://books.google.es/books?id=hNPMEnqfc44C&pg=PA126&lpg=PA126&dq=n+gramas+y+modelos+ocultos+de+Markov&source=web&ots=EmwXOl7Peq&sig=4Lus_C2-Q_64PR6rOehE3lEH8Mo&hl=es&sa=X&oi=book_result&resnum=1&ct=result#PPA127,M1

NWN Community Expansion Pack (CEP 2.0).
http://nwvault.ign.com

Nihilistic-Software (2000). Vampire The Masquerade: Redemption.
http://www.planetvampire.com/redemptions/

Nintendo (2006). Mario Party 7, GameCube.
www.marioparty.com/

Nintendo (2006). Odama, GameCube.
http://www.nintendo.es/NOE/es_ES/games/gc/odama.html

Pajares Tosca, S. (2001). Role-playing in multiplayer environments. Vampire: The masquerade. redemption. In *Computer Games & Digital Textualities*, pages 10–18, IT University of Copenhagen, Köbenhavns.

Pajares Tosca S. (2001, march). Role-playing in multiplayer environments. Vampire: The Masquerade Redemption. Computer Games & Digital Textualities. IT-University of Copenhaguen.
http://www.itu.dk/people/tosca/multiplayer.html

Peinado, F. (2007). Knowledge-Intensive Interactive Digital Storytelling system.
https://sourceforge.net/projects/kiids/

Peinado, F. (2006). Narración Digital e Interactiva: Dirección automática de entornos virtuales
http://www.fdi.ucm.es/profesor/fpeinado/publications/2006-peinado-narracion-PRE.pdf

Peinado, F. (2004). Mediación Inteligente entre Autores e Interactores para Sistemas de Narración Digital Interactiva
http://www.fdi.ucm.es/profesor/fpeinado/publications/2004-peinado-mediacion-MPhil.pdf

Peinado, F., Gervás, P. (2004, june). "Transferring Game Mastering Laws to Interactive Digital Storytelling". In Göbel, S., Spierling, U., Hoffmann, A., Iurgel, I., Schneider, O., Dechau, J., Feix, A. (Eds.): Technologies for Interactive Digital Storytelling and Entertainment. Darmstadt, Germany.
http://www.fdi.ucm.es/profesor/fpeinado/publications/2004-peinado-transferring.pdf

Rabiner R. (1989). A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.
http://www.cs.ubc.ca/~murphyk/Software/HMM/rabiner.pdf

Sancho P., Gomez P.P., Fernández-Manjón B. (2008). Multiplayer Role Games Applied to Problem Based Learning. Complutense University of Madrid.
http://www.e-ucm.es/drafts/e-UCM_draft_101.pdf

Sandia National Laboratories. JESS: the Rule Engine for the Java Platform.
http://herzberg.ca.sandia.gov/

Simple API for XML: SAX (2008, agosto).
www.saxproject.org

Second Life (2008, agosto). Virtual worlds, avatars, 3D chat, online meetings.
http://secondlife.com/

Siegel, J.D. (2007). Dialogue Patterns in Computer Role-Playing Games. Master of Science. University of Alberta.
www.cs.ualberta.ca/~siegel/jeff-thesis-05-01-07.pdf

SpeechMagic (2008).
http://www.speechrecognition.philips.com/

Stanchfield, S. ANTXR: Easy XML Parsing, based on the ANTLR parser generator. JavaDude.com, Hillcrest Communications & FGM, Inc.
http://javadude.com/tools/antxr/index.html

Sun Microsystems, Inc. (1994-2008). Java.
http://java.sun.com/

SurveyMonkey (2008). Encuestas en Internet.
http://www.surveymonkey.com/

There (2008, agosto). The online virtual world that is your everyday hangout.
http://www.there.com/

The Breeze Factor: XML and JAVA Data Binding.
http://www.breezefactor.com

The Castor Project.
http://www.castor.org

The Sims Online (2008, agosto).
http://www.thesimsonline.com.br/handcrafted2.asp

The Virtual World of Kaneva (2008, agosto).
http://www.kaneva.com

Toolset Tutorials for Neverwinter Nights (2008, julio).
http://nwvault.ign.com/modules/tutorials/viewlets/

Tutoriales de Aurora Neverwinter Nights (2008, julio).
http://nwn.bioware.com/builders/

Tychsen A., Hitchens M., Brolund T., Kavakli. (2005). The Game Master. Macquarie University. Building E6A, 2109 North Ryde, Sydney, NSW.
http://www.ics.mq.edu.au/~atychsen/html2/images/IE_2005.pdf.

Ubisoft Entertainment (2008/09). Tom Clancy´s HAWX.
http://hawxgame.es.ubi.com/

Ubisoft Entertainment (2008). Tom Clancy´s RainBow Six Vegas 2.
http://rainbowsixgame.es.ubi.com/home.php

Virtual Environments, Virtual Worlds, Social MMOGs, MUVEs, CVEs, MMOs (2008, agosto).
http://www.virtualenvironments.info/

VoxForge (2008).
http://www.voxforge.org/

Walker W., Lamere P., Kwok P., Raj B., Singh R., Gouvea E., Wolf P., Woelfel J. (2008). Sphinx-4: A Flexible Open Source Framework for Speech Recognition.
http://cmusphinx.sourceforge.net/sphinx4/

Wide Ruled (2008). Automatically generated stories, using an author goal-based planning system.
http://www.soe.ucsc.edu/~jskorups/wiki/wide_ruled/wide_ruled

XML and Java technologies: Data binding, Part 2: Performance.
http://www.ibm.com/developerworks/xml/library/x-databdopt2/

XML Pull Parsing.
http://www.xmlpull.org/

Young, M. (2007). Zocalo.
http://zocalo.csc.ncsu.edu/

Zubek, R. (2003). Shadow Door: Neverwinter Nights NPC Control Interface.
http://www.zubek.net/robert/software/shadow-door/

Apéndice A. Configuración del reconocedor de voz

Archivo de configuración utilizado para generar un proyecto con Sphinx. Todos los parámetros configuran el motor de Sphinx, y le indican entre otras cosas el archivo donde se encuentra la gramática que se va a reconocer y el modelo acústico con el que se está trabajando.

```
<?xml version="1.0" encoding="UTF-8"?>

<!--  Sphinx-4 Configuration file -->

<!-- ******************************************************** -->
<!--  an4 configuration file                                  -->
<!-- ******************************************************** -->

<config>

    <!-- ******************************************************** -->
    <!-- frequently tuned properties                              -->
    <!-- ******************************************************** -->

    <property name="logLevel" value="WARNING"/>
    <property name="absoluteBeamWidth"  value="-1"/>
    <property name="relativeBeamWidth"  value="1E-80"/>
    <property name="wordInsertionProbability" value="1E-36"/>
    <property name="languageWeight"     value="8"/>
    <property name="frontend" value="epFrontEnd"/>
    <property name="recognizer" value="recognizer"/>
    <property name="showCreations" value="false"/>

    <!-- ******************************************************** -->
    <!-- dialog manager                                           -->
    <!-- ******************************************************** -->

    <component name="dialogManager" type="dialog.DialogManager">
        <property name="microphone" value="microphone"/>
        <property name="jsgfGrammar" value="jsgfGrammar"/>
        <property name="recognizer" value="recognizer"/>
    </component>

    <!-- ******************************************************** -->
    <!-- recognizer configuration                                 -->
    <!-- ******************************************************** -->

    <component name="recognizer" type="edu.cmu.sphinx.recognizer.Recogni-
zer">
        <property name="decoder" value="decoder"/>
        <propertylist name="monitors">
            <item>accuracyTracker </item>
            <item>speedTracker </item>
            <item>memoryTracker </item>
        </propertylist>
    </component>

    <!-- ******************************************************** -->
    <!-- The Decoder   configuration                              -->
    <!-- ******************************************************** -->

    <component name="decoder" type="edu.cmu.sphinx.decoder.Decoder">
        <property name="searchManager" value="searchManager"/>
    </component>
```

```
    <component name="searchManager"
        type="edu.cmu.sphinx.decoder.search.SimpleBreadthFirstSearchManager"
>
        <property name="logMath" value="logMath"/>
        <property name="linguist" value="flatLinguist"/>
        <property name="pruner" value="trivialPruner"/>
        <property name="scorer" value="threadedScorer"/>
        <property name="activeListFactory" value="activeList"/>
    </component>


    <component name="activeList"
             type="edu.cmu.sphinx.decoder.search.PartitionActiveListFactory"
>
        <property name="logMath" value="logMath"/>
        <property name="absoluteBeamWidth" value="${absoluteBeamWidth}"/>
        <property name="relativeBeamWidth" value="${relativeBeamWidth}"/>
    </component>

    <component name="trivialPruner"
                type="edu.cmu.sphinx.decoder.pruner.SimplePruner"/>

    <component name="threadedScorer"
                type="edu.cmu.sphinx.decoder.scorer.ThreadedAcousticScorer">
        <property name="frontend" value="${frontend}"/>
        <property name="isCpuRelative" value="true"/>
        <property name="numThreads" value="0"/>
        <property name="minScoreablesPerThread" value="10"/>
        <property name="scoreablesKeepFeature" value="true"/>
    </component>

    <!-- ******************************************************** -->
    <!-- The linguist  configuration                              -->
    <!-- ******************************************************** -->

    <component name="flatLinguist"
                type="edu.cmu.sphinx.linguist.flat.FlatLinguist">
        <property name="logMath" value="logMath"/>
        <property name="grammar" value="jsgfGrammar"/>
        <property name="acousticModel" value="wsj"/>
        <property name="wordInsertionProbability"
                value="${wordInsertionProbability}"/>
        <property name="languageWeight" value="${languageWeight}"/>
        <property name="unitManager" value="unitManager"/>
    </component>


    <!-- ******************************************************** -->
    <!-- The Grammar  configuration                               -->
    <!-- ******************************************************** -->

    <component name="jsgfGrammar" type="edu.cmu.sphinx.jsapi.JSGFGrammar">
        <property name="dictionary" value="dictionary"/>
        <property name="grammarLocation"
             value="resource:/dialog.Dialog!/dialog/"/>
        <property name="grammarName" value="menu"/>
       <property name="logMath" value="logMath"/>
    </component>


    <!-- ******************************************************** -->
    <!-- The Dictionary configuration                            -->
    <!-- ******************************************************** -->

    <component name="dictionary"
```

```
        type="edu.cmu.sphinx.linguist.dictionary.FastDictionary">
        <property name="dictionaryPath"
        va-
lue="resource:/edu.cmu.sphinx.model.acoustic.WSJ_8gau_13dCep_16k_40mel_130Hz
_6800Hz.Model!/edu/cmu/sphinx/model/acoustic/WSJ_8gau_13dCep_16k_40mel_130Hz
_6800Hz/dict/cmudict.0.6d"/>
        <property name="fillerPath"
        va-
lue="resource:/edu.cmu.sphinx.model.acoustic.WSJ_8gau_13dCep_16k_40mel_130Hz
_6800Hz.Model!/edu/cmu/sphinx/model/acoustic/WSJ_8gau_13dCep_16k_40mel_130Hz
_6800Hz/dict/fillerdict"/>
        <property name="addSilEndingPronunciation" value="false"/>
        <property name="allowMissingWords" value="false"/>
        <property name="unitManager" value="unitManager"/>
    </component>

    <!-- ******************************************************* -->
    <!-- The acoustic model configuration                        -->
    <!-- ******************************************************* -->
    <component name="wsj"
      type="edu.cmu.sphinx.model.acoustic.WSJ_8gau_13dCep_16k_40mel_130Hz_68
00Hz.Model">
        <property name="loader" value="wsjLoader"/>
        <property name="unitManager" value="unitManager"/>
    </component>

    <component name="wsjLoader"
               type="edu.cmu.sphinx.model.acoustic.WSJ_8gau_13dCep_16k_40-
mel_130Hz_6800Hz.ModelLoader">
        <property name="logMath" value="logMath"/>
        <property name="unitManager" value="unitManager"/>
    </component>


    <!-- ******************************************************* -->
    <!-- The unit manager configuration                          -->
    <!-- ******************************************************* -->

    <component name="unitManager"
        type="edu.cmu.sphinx.linguist.acoustic.UnitManager"/>

    <!-- ******************************************************* -->
    <!-- The frontend configuration                              -->
    <!-- ******************************************************* -->

    <component name="frontEnd" type="edu.cmu.sphinx.frontend.FrontEnd">
        <propertylist name="pipeline">
            <item>microphone </item>
            <item>premphasizer </item>
            <item>windower </item>
            <item>fft </item>
            <item>melFilterBank </item>
            <item>dct </item>
            <item>liveCMN </item>
            <item>featureExtraction </item>
        </propertylist>
    </component>

    <!-- ******************************************************* -->
    <!-- The live frontend configuration                         -->
    <!-- ******************************************************* -->
    <component name="epFrontEnd" type="edu.cmu.sphinx.frontend.FrontEnd">
        <propertylist name="pipeline">
            <item>microphone </item>
            <item>speechClassifier </item>
            <item>speechMarker </item>
```

```
            <item>nonSpeechDataFilter </item>
            <item>premphasizer </item>
            <item>windower </item>
            <item>fft </item>
            <item>melFilterBank </item>
            <item>dct </item>
            <item>liveCMN </item>
            <item>featureExtraction </item>
        </propertylist>
    </component>

    <!-- ******************************************************* -->
    <!-- The frontend pipelines                                  -->
    <!-- ******************************************************* -->

    <component name="speechClassifier"
               type="edu.cmu.sphinx.frontend.endpoint.SpeechClassifier">
        <property name="threshold" value="20"/>
    </component>

    <component name="nonSpeechDataFilter"
               type="edu.cmu.sphinx.frontend.endpoint.NonSpeechDataFilter"/>

    <component name="speechMarker"
               type="edu.cmu.sphinx.frontend.endpoint.SpeechMarker" >
        <property name="speechTrailer" value="50"/>
    </component>


    <component name="premphasizer"
               type="edu.cmu.sphinx.frontend.filter.Preemphasizer"/>

    <component name="windower"
               type="edu.cmu.sphinx.frontend.window.RaisedCosineWindower">
    </component>

    <component name="fft"
            type="edu.cmu.sphinx.frontend.transform.DiscreteFourierTransform
">
    </component>

    <component name="melFilterBank"
        type="edu.cmu.sphinx.frontend.frequencywarp.MelFrequencyFilterBank">
    </component>

    <component name="dct"
            type="edu.cmu.sphinx.frontend.transform.DiscreteCosineTransform"
/>

    <component name="liveCMN"
               type="edu.cmu.sphinx.frontend.feature.LiveCMN"/>

    <component name="featureExtraction"
               type="edu.cmu.sphinx.frontend.feature.DeltasFeatureExtractor"
/>

    <component name="microphone"
               type="edu.cmu.sphinx.frontend.util.Microphone">
        <property name="closeBetweenUtterances" value="false"/>
    </component>


    <!-- ******************************************************* -->
    <!--  monitors                                               -->
    <!-- ******************************************************* -->
```

```
    <component name="accuracyTracker"
                type="edu.cmu.sphinx.instrumentation.AccuracyTracker">
        <property name="recognizer" value="${recognizer}"/>
        <property name="showAlignedResults" value="false"/>
        <property name="showRawResults" value="false"/>
    </component>

    <component name="memoryTracker"
                type="edu.cmu.sphinx.instrumentation.MemoryTracker">
        <property name="recognizer" value="${recognizer}"/>
       <property name="showSummary" value="false"/>
       <property name="showDetails" value="false"/>
    </component>

    <component name="speedTracker"
                type="edu.cmu.sphinx.instrumentation.SpeedTracker">
        <property name="recognizer" value="${recognizer}"/>
        <property name="frontend" value="${frontend}"/>
       <property name="showSummary" value="true"/>
       <property name="showDetails" value="false"/>
    </component>


    <!-- ******************************************************* -->
    <!--  Miscellaneous components                               -->
    <!-- ******************************************************* -->

    <component name="logMath" type="edu.cmu.sphinx.util.LogMath">
        <property name="logBase" value="1.0001"/>
        <property name="useAddTable" value="true"/>
    </component>


    <!-- ******************************************************* -->
    <!-- The Search Manager                                      -->
    <!-- ******************************************************* -->

    <component name="wordPruningSearchManager"
    type="edu.cmu.sphinx.decoder.search.WordPruningBreadthFirstSearchManager
">
        <property name="logMath" value="logMath"/>
        <property name="linguist" value="lexTreeLinguist"/>
        <property name="pruner" value="trivialPruner"/>
        <property name="scorer" value="threadedScorer"/>
        <property name="activeListManager" value="activeListManager"/>
        <property name="growSkipInterval" value="0"/>
        <property name="checkStateOrder" value="false"/>
        <property name="buildWordLattice" value="false"/>
        <property name="acousticLookaheadFrames" value="1.7"/>
        <property name="relativeBeamWidth" value="${relativeBeamWidth}"/>
    </component>


    <!-- ******************************************************* -->
    <!-- The Active Lists                                        -->
    <!-- ******************************************************* -->

    <component name="activeListManager"
             type="edu.cmu.sphinx.decoder.search.SimpleActiveListManager">
        <propertylist name="activeListFactories">
          <item>standardActiveListFactory</item>
          <item>wordActiveListFactory</item>
          <item>wordActiveListFactory</item>
          <item>standardActiveListFactory</item>
          <item>standardActiveListFactory</item>
          <item>standardActiveListFactory</item>
```

```
      </propertylist>
    </component>

    <component name="standardActiveListFactory"
             type="edu.cmu.sphinx.decoder.search.PartitionActiveListFactory"
>
        <property name="logMath" value="logMath"/>
        <property name="absoluteBeamWidth" value="500"/>
        <property name="relativeBeamWidth" value="1E-80"/>
    </component>

    <component name="wordActiveListFactory"
             type="edu.cmu.sphinx.decoder.search.PartitionActiveListFactory"
>
        <property name="logMath" value="logMath"/>
        <property name="absoluteBeamWidth" value="20"/>
        <property name="relativeBeamWidth" value="1E-60"/>
    </component>


    <!-- ******************************************************* -->
    <!-- The linguist  configuration                             -->
    <!-- ******************************************************* -->

    <component name="lexTreeLinguist"
                type="edu.cmu.sphinx.linguist.lextree.LexTreeLinguist">
        <property name="logMath" value="logMath"/>
        <property name="acousticModel" value="wsj"/>
        <property name="languageModel" value="trigramModel"/>
        <property name="dictionary" value="dictionary"/>
        <property name="addFillerWords" value="false"/>
        <property name="fillerInsertionProbability" value="1E-10"/>
        <property name="generateUnitStates" value="false"/>
        <property name="wantUnigramSmear" value="true"/>
        <property name="unigramSmearWeight" value="1"/>
        <property name="wordInsertionProbability" value="1E-16"/>
        <property name="silenceInsertionProbability" value=".1"/>
        <property name="languageWeight" value="7.0"/>
        <property name="unitManager" value="unitManager"/>
    </component>
</config>
```

Apéndice B. Documentación complementaria

En este apéndice se presenta una lista de toda la documentación complementaria de esta tesis de Master que se encuentra disponible en soporte electrónico para todos aquellos lectores que deseen conocer más detalles sobre esta investigación.

1. Versión en formato PDF de esta memoria de tesis de master.
2. Versión en formato PDF de la memoria de la Beca de Colaboración con el Departamento de Inteligencia Artificial e Ingeniería del Software 2006/07.
   http://www.fdi.ucm.es/profesor/fpeinado/projects/rcei/rcei-report_es.pdf

3. Versión en formato PDF de todas las publicaciones relacionadas.

- León, C., Peinado, F., Navarro, A. (2008). An Intelligent Plot-Centric Interface for Mastering Computer Role-Playing Games. ICIDS08, 1st International Conference on Interactive Digital Storytelling.

- Peinado, F., Navarro, A., Gervás, P. (2008). A Testbed Environment for Interactive Storytellers. INTETAIN, Workshop on Integrating Technologies for Interactive Stories, 8-10/1.

- Peinado, F., Navarro, A. (2007). RCEI: An API for Remote Control of Narrative Environments. Virtual Storytelling: Using Virtual Reality Technologies for Storytelling 4871:181-186, Saint-Malo, France.

4. Última versión estable de jRCEI, la implementación Java del API, ejemplos y documentación JavaDoc.
   http://federicopeinado.com/projects/rcei
   http://code.google.com/p/rcei/