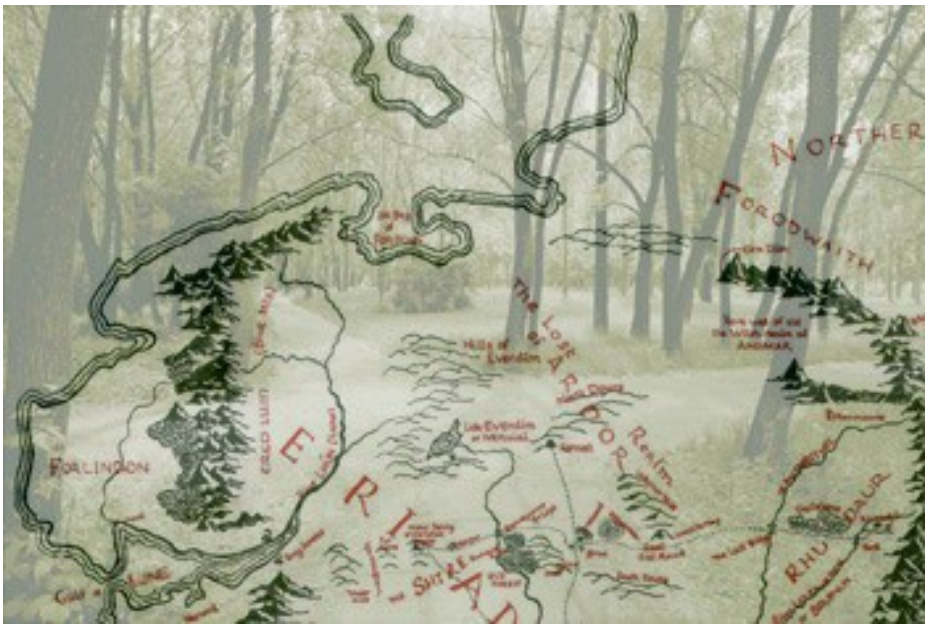
Memoria del proyecto de la beca de colaboración
Narración interactiva y conocimiento OWL

Encrucijada en la Tierra Media



Centro: Facultad de Informática
Departamento: Ingeniería del Software e Inteligencia Artificial
Director del proyecto: Pablo Gervás Gómez-Navarro
Coordinador: Federico Peinado Gil
Becario: Álvaro Navarro Iborra

Índice

Resumen

El objetivo del proyecto ha sido el de desarrollar una primera aplicación para el sistema *Knowledge-Intensive Interactive Digital Storytelling system (KIIDS)* [7] llamada *Encrucijada en la Tierra Media*. Está aplicación estará inspirada en el mundo de *J.R.R Tolkien*, es decir, de género de aventuras o rol ambientada en la Tierra Media.

La beca de colaboración se ha divido en tres fases, en primer lugar la investigación y documentación del motor de juego de *Neverwinter Nights* [1], un juego comercial de última generación, con un gran potencial de aplicación en distintas líneas de investigación del departamento. Además se ha desarrollado un módulo implementado en *Neverwinter Nights* para *Encrucijada en la Tierra Media* y se han elaborado de siete historias detalladas en documentos, con los que se deben formalizar las ontologías de la futura base de casos.

En segundo lugar, se ha diseñado e implementado un protocolo y un lenguaje de comunicación llamado *Remote Controlled Environments Interface (RCEI)* [8] para el control remoto de entornos virtuales. Este proyecto, implementado en JAVA, ha sido realizado para la posterior comunicación del sistema *KIIDS* con el entorno concreto, pero podría utilizarse para otras aplicaciones distintas relacionadas con agentes, sistemas inteligentes u otras aplicaciones de narración interactiva. Además se ha creado un adaptador de *Neverwinter Nights* a *RCEI*, llamado *NWN1RCEIAdapter.*

En tercer lugar se han elaborado documentos con especificaciones de las ontologías necesarias para la base de conocimiento de *Encrucijada en la Tierra Media*. Además se han hecho algunas pruebas de integración de RCEI con KIIDS, pero la implementación en OWL [9] [10] de todas las ontologías se ha dejado para una futura implementación por limitaciones de tiempo.

Queda pendiente para más adelante terminar la integración con *KIIDS* y realizar extensiones sobre RCEI, así como implementar conectores a otros entornos de diferente tipo como pueden ser juegos de estrategia, shooters, aventuras conversacionales, etc. (*Knights of the Old Republic*, *Unreal Tournment 2007*, *Warcraft 2*, *Call to Power 2*, *Grand Theft Auto 3*).

Agradecimientos

Quiero agradecer toda la ayuda y el apoyo prestado a todas las personas que me han ayudado durante la realización del proyecto. En primer lugar mi agradecimiento a *Pablo Gervás Gómez-Navarro* por su ayuda a la hora de solicitar la beca de colaboración, su apoyo y ayuda con cualquier problema, su seguimiento del proyecto y la gran libertad que nos ha dado a la hora de trabajar.

En segundo lugar mi sincero agradecimiento a *Federico Peinado Gil* por toda su estrecha colaboración a lo largo de todo el proyecto, ha sido un placer poder trabajar con él en este proyecto, con él he aprendido muchas cosas y la colaboración ha sido siempre muy agradable.

También quiero dar las gracias al departamento de *Inteligencia Artificial e Ingeniería del Software* por las facilidades que me han dado para desarrollar la la beca de colaboración proporcionando un laboratorio para poder llevar a cabo el trabajo, concretamente *Luis Hernández Yáñez* y también a *Javier Muñoz* su ayuda en el laboratorio.

Además agradezco a mis compañeros *Samer Hassan* y *Mónica González* sus consejos y sus ideas a la hora de avanzar en el proyecto. Por último, agradezco a mis padres *Ángel* y *Joaquina* y a mi hermano *Alex* que han hecho de betatesters en alguna ocasión y me han aportado también su punto de vista a la hora de diseñar y probar el juego.

1. INTRODUCCIÓN

El objetivo general de este proyecto es implementar un prototipo simple de juego de rol por ordenador, apoyado en ontologías en formato OWL [9][10], que sea capaz de proporcionar una interacción fluida y una estructura narrativa interesante para el usuario a base de aplicar prototipos de herramientas ya existentes de dirección automática de narración digital e interactiva.

Los juegos de rol por ordenador permiten a un 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. Los primeros juegos de mesa temática fantástica, como *Dungeons & Dragons*, generaron mucha afición, y son el origen de los juegos de rol de PC, que hoy en día son un género de mucho éxito.

Fig. 1. Captura de Encrucijada en la Tierra Media en el momento de un enfrentamiento con unos orcos.

Uno de los problemas que tienen este tipo de juegos es lo que se ha llamado el Dilema de Narración Interactiva: si la historia ha sido predefinida por el diseñador del juego, el jugador puede frustrarse al ver que las decisiones no cuentan para nada, pero por otro lado, si es el jugador el que tiene todo el control, no hay garantías de que la historia tenga sentido ni que posea una estructura narrativa que cumpla unos mínimos de calidad.

Se propone un trabajo de integración, utilizando soluciones ya implementadas en el departamento para la mediación automática en aplicaciones de narración digital e interactiva en espacios virtuales, para implementar un prototipo simple de juego de rol por ordenador en el que sea un sistema inteligente el que se haga cargo del Dilema de la Narración Interactiva. Se utilizarán ontologías en OWL para representar conceptualmente los ingredientes que deba manejar el sistema, y el lenguaje de guiones de propósito específico NeverwinterScript para implementar los detalles de bajo nivel de la representación multimedia del sistema.

Se espera que el trabajo propuesto requiera del alumno una dedicación de 15 horas semanales a lo largo de la duración de la beca. Se establecerá un plan de reuniones periódicas con el director del proyecto que garantice tanto un asesoramiento más detallado al principio del proyecto como un seguimiento de la actividad realizada según el alumno vaya obteniendo resultados de forma autónoma.

El trabajo requerirá diseñar un pequeño entorno de juego, representarlo conceptualmente e integrarlo con el módulo de gestión de interacciones.

Para el desarrollo del trabajo será necesario llevar a cabo las siguientes tareas:

1. Familiarización con las herramientas de modelado del conocimiento a utilizar.

2. Familiarización con el editor de Neverwinter Nights (NWN), Aurora [24] [25] y el lenguaje de guiones NeverwinterScript [28].

3. Desarrollo de un módulo simple de Neverwinter Nights que de soporte a las aventuras propuestas.

4. Construcción de la ontología OWL necesaria para el juego concebido, así como las poblaciones correspondientes a cada aventura.

5. Integración de la ontología con el módulo de dirección automática de narración interactiva para asegurar un funcionamiento adecuado del juego.

6. Documentación del proceso completo de desarrollo e integración.

El desarrollo propuesto tiene interés tanto desde el punto de vista de la docencia como desde el punto de vista de la investigación. Por lo que respecta a la docencia, el trabajo propuesto supone la puesta en práctica de manera conjunta de una serie de herramientas para llevar a cabo tareas que se han estudiado en la asignatura de Inteligencia Artificial, y que empiezan a ser aplicadas en el mundo de la educación y el entretenimiento. El alumno participante tendrá la ocasión de ver como una serie de conceptos que ha estudiado a nivel teórico se ponen en práctica en la implementación de herramientas reales, y como esas herramientas se utilizan para obtener resultados prácticos concretos.

La cantidad de historias que debía usar la base de conocimiento se limitó desde un primer momento a 7-9 historias debido al coste que su pondría realizar una base de casos más extensa. La parte de interacción del juego, es decir, el módulo de Neverwinter Nights aglutinará todas las aventuras, en vez de realizar varios módulos para cada aventura. La aplicación *Knowledge-Intensive Interactive Digital Storytelling system* (KIIDS) [7] será la que se encargue de gestionar todos los componentes del módulo Neverwinter Nights [1] y manejar los elementos de simulación y narración de cada aventura.

Las historias estaban pensadas para ser jugadas en poco tiempo, aproximadamente entre 15 y 45 minutos, dependiendo de la experiencia del jugador. La idea inicial era que hubiera varios personajes predefinidos diferentes y según el tipo de jugador se comenzara jugando una de las historias. Uno de los objetivos que se pretendía poder realizar en el proyecto era ver la reacción que produce en los usuarios que utilizasen el juego. Por eso está pensado que los jugadores prueben el juego y rellenen un pequeño cuestionario con preguntas sobre la experiencia, para posteriormente poder analizar los resultados y evaluar la experiencia.

El motivo de hacer una cantidad pequeña pero variada de casos, era que los experimentos ofrecieran resultados interesantes y que hubiera una variedad temática entre las historias. Para ello nos hemos basado en el libro "The Seven Basic Plots"[19]. Este libro engloba todos los tipos de historias de novelas o películas en siete tipos de categorías. Por ello cada una de las historias de correspondientes a cada caso pertenecen a los distintos tipos de categorías que menciona el libro. Finalmente la duración de la beca no ha permitido implementarlas todas, sola una de ellas, pero extensa de duración. Se espera que en el futuro la beca pueda ser continuada o retomar el proyecto.

Por lo que se respecta a la investigación, el trabajo propuesto supone la integración en un único prototipo de aplicación un conjunto de herramientas de libre distribución, más un motor de juego comercial de última generación, con un gran potencial de aplicación en distintas líneas de investigación del departamento. Este trabajo de familiarización y uso es fundamental para cualquier posterior trabajo de investigación, y, dado que se incluye una tarea de documentación cuidadosa de las experiencias realizadas, con un editor de código abierto[17], facilitará futuros esfuerzos en los que estas herramientas jueguen un papel auxiliar, como puede ser el desarrollo de interfaces inteligentes del tipo que se contemplan para aplicaciones de domótica, robótica, inteligencia ambiental, y otras líneas similares consideradas de prioridad en recientes declaraciones institucionales de objetivos , tanto a nivel nacional como a nivel europeo.

2. PLANIFICACIÓN

La planificación está estructurada por semanas, de manera que al término de cada semana el becario envía un correo electrónico informando sobre la evolución del proyecto y entregando los resultados obtenidos. Si el informe o los resultados no son los esperados, se toman medidas para resolver la situación y se modifica la planificación convenientemente.

2.1. Primera Iteración

Esta primera etapa se ha dedicado principalmente al estudio del editor de Neverwinter Nights para poder editar módulos que representen una historia o una aventura. Además se han desarrollado y documentado 7 historias para tener una base de casos amplia, aunque finalmente por cuestiones de tiempo, sólo se ha implementado en Neverwinter Nights una de ellas.

Octubre

Semana	*Trabajo Realizado*
2-6	Instalación, prueba y familiarización con la plataforma NWN. ● Experimentación con alguna de las campañas oficiales NWN. ● Herramienta Aurora, tutorial básico de construcción (áreas, objetos, encuentros, diálogos, puertas y sus propiedades, facciones...) http://nwn.bioware.com/builders/modtutorial.html
9-13	Especificación de uno de los casos que componen la base de conocimiento de Crossroads in Middle-Earth, documentados en español. ● Desarrollo de la historia y del documento que detalla los datos de la historia, es decir, escenarios, personajes, diálogos, trasfondo, desarrollo de la aventura, etc. ● Experimentación con la aplicación de generación automática de nombres para crear nombres propios de la Tierra Media.
16-20	Implementación de una primera versión demo de una de las historias de Crossroads in Middle-Earth, diseño de escenarios. ● Diseño de escenarios de la primera y segunda aventura con la Herramienta Aurora. Se diseñan varios escenarios donde transcurren la primera y segunda historia, pensando teniendo en cuenta que puedan ser reutilizables en otras historias. ● Búsqueda de información en Internet sobre herramientas de NWN que se puedan usar, concretamente NwnMultiLang, que podría utilizarse en el futuro como herramienta de traducción al inglés para acelerar el proceso de traducción ya que permite extraer toda la información del módulo en vez tener que ir a cada escenario, personaje, etc.
23-27	Implementación de una primera versión demo de una de las historias de Crossroads in Middle-Earth, diseño de personajes y diálogos redactando diálogos y textos en español.. ● Se ha realizado investigación del funcionamiento de la herramienta de diálogos y de las posibles interacciones entre personajes a través de scripts del NWN. ● Diseño de diálogos e interacción del jugador con varios personajes de la primera parte de la misión. En los diálogos se realizan algunas acciones sencillas entre personajes como intercambiar objetos o dar puntos de experiencia que pueden crearse automáticamente con el Wizard.

Noviembre

Semana	*Trabajo Realizado*
30-3	● Especificación de caso 3 de Crossroads in Middle-Earth, caso la base de conocimiento de Crossroads in Middle-Earth, documentado en español. ● Especificación de una primera estructura de la memoria y se comienza a documentar. ● Federico realiza prueba de juego sobre la parte de la demo que se ha realizado.
6-10	● Especificación de casos 1,3,4 y 5 de Crossroads in Middle-Earth, que componen la base de conocimiento de Crossroads in Middle-Earth, documentados en español. Para diseñar las aventuras se hacen más pruebas sobre Aurora para estudiar que personajes, escenarios y objetos son aptos para incluir en las historias y se hacen algunos retoques puntuales y pruebas en el módulo de NWN. ● Se deciden cambios en la estructura de la memoria, para darle más importancia a la documentación y no perder información relevante. También se decide realizar la documentar semanalmente y no al final del proyecto para no perder información.
13-17	● Especificación de casos 5,6 de Crossroads in Middle-Earth, que componen la base de conocimiento de Crossroads in Middle-Earth, documentados en español. ● Modificación de la estructura de la memoria y se empieza a documentar parte de lo realizado anteriormente en el proyecto.
20-24	● Especificación de caso 7 de Crossroads in Middle-Earth, caso la base de conocimiento de Crossroads in Middle-Earth, documentado en español. ● Investigación sobre la herramienta Aurora, lectura de manuales de la herramienta. En particular se ha investigado en el asistente de tramas, una herramienta que permitirá diseñar las misiones de las aventuras de manera más sencilla. El diseño de tramas con el asistente encaja con los diagramas de actividad de la parte de interacción de cada una de las historias. ● Se añade más documentación a la memoria. ● Modificación del módulo CIME para hacer pruebas con el asistente de tramas y para quitar algunos bugs de versiones anteriores del CASO 2.
27-1	● Revisión de la especificación de todos los casos, modificación de algunos detalles del documento, corrección de fallos, compactar el documento para que no sea demasiado extenso. ● Se revisan fallos de la memoria y se añade más documentación ● Implementación de algunas tramas con el asistente en el CASO 2.

Diciembre

Semana	*Trabajo Realizado*
4-8	● ***Federico presenta este proyecto en TIDSE'06, Alemania*** ● Implementación de caso 2 de Crossroads in Middle-Earth, etiquetando en inglés todos los elementos que después serán accedidos por scripting. ● Estudio del lenguaje NeverwinterScript. ● Ampliación de la memoria.
11-15	● Implementación de caso 2 de Crossroads in Middle-Earth, etiquetando en inglés todos los elementos que después serán accedidos por scripting. ● Estudio de RAD Video Tools y creación de pequeño vídeo para la DEMO usando PowerPoint + Camtasia Studio. ● Ampliación y modificación de la memoria, traducción de partes de la memoria que estaban en inglés.
20-24	● Ampliación de la memoria, añadir bibliografía necesaria y otras cosas para la presentación. ● Primera demo jugable de la historia 2.
27-1	VACACIONES DE NAVIDAD

2.2. Segunda Iteración

Esta segunda etapa se nos hemos centrado en conseguir la conectividad del motor de juego de Neverwinter Nights con una aplicación externa. Hemos partido de una aplicación llamada Shadow Door, que usa comunicación por sockets y un pequeño interfaz de comunicación. El módulo de ejemplo que Shadow Door proporciona permite controlar solamente un agente y permitirle que realize 4 acciones básicas: hablar, atacar, moverse y realizar una animación.

Nosotros hemos logrado algunas ampliaciones notables, creando un protocolo de comunicación propio, que ha permitido el control de muchos agentes, con una gran variedad de acciones. Todo ello mediante un lenguaje estructurado y consistente a la vez que sencillo, lo cual permite a los usuarios editar archivos de nuestro lenguaje.

Enero

Semana	*Trabajo Realizado*
1-5	● VACACIONES DE NAVIDAD
8-12	● Instalación de NWN Extender + ShadowDoor + Allegro Common Lisp ● Investigación del entorno ShadowDoor
15-21	● Diseño del adaptador KIIDS4NWN (parte de lado de NWN). ● Diseño del protocolo de comunicación entre KIIDS y NWN a la que llamaremos **RCEI** (Remote Controlled Environments Interface).
22-28	● Primera versión de la aplicación en java para el conector RCEI. Por ahora se realiza una comunicación de manera manual, sin uso ninguno de ontologías, para probar el funcionamiento del conector que posteriormente conectaremos con KIIDS.
29-2	● EXÁMENES

Febrero

Semana	*Trabajo Realizado*
5-10	● EXÁMENES
12-16	● EXÁMENES (día 12 último examen) ● Implementación del adaptador en NeverwinterScript, usando ShadowDoor. ● Conseguida interacción entre varios agentes y no sólo uno como en el ejemplo original que viene en ShadowDoor.
19-23	● Desarrollo de la memoria, documentación sobre la parte de RCEI. ● Nuestra aplicación en java la llamaremos desde este momento jRCEI.
26-2	● Investigación de algunas herramientas de Neverwinter Nights, como un módulo de ajedrez en Neverwinter, visor de mdl para NWN y alguna otra utilidad como la de compilar el módulo de juego en un ejecutable. ● Desarrollo de la parte de java del conector (aplicación jRCEI). ● Registro de usuario en la web de bioware para tener un usuario disponible para el departamento para hacer pruebas en red.

Marzo

Semana	*Trabajo Realizado*
5-9	● Revisión de la gramática de RCEI y adaptación de la misma para hacerla más genérica de manera que no sea muy dependiente de Neverwinter Nights, con vista a poder ser usada en otros entornos. ● Investigación del código fuente de NWN Extender 2 ● Investigación del código fuente de Shadow Door
12-16	● Desarrollo aplicación jRCEI (versión 0.0.5) ● Ampliación de la memoria, se añade más documentación de ShadowDoor y jRCEI
19-22	● Implementación del código en NWNScript de los comandos que tiene por ahora el lenguaje RCEI para ser ejecutados en el entorno. ● Implementación del envío y recepción de mensajes de desde el motor de juego de NWN hacia el exterior y viceversa. ● Pruebas del uso de la DLL que nos proporciona Shadow Door para la comunicación con el entorno.
26-30	● Desarrollo de la siguiente versión de la aplicación RCEI. ● Presentación de jRCEI a los profesores del departamento. ● Modificación del código fuente de Shadow Door para crear un proyecto propio en Visual C++ 2005 y crear un DLL propia para no depender de Shadow Door.

Abril

Semana	*Trabajo Realizado*
2-6	● *Vacaciones de Semana Santa*
9-13	● Se añaden los comandos begin y se modifica el comando “speak” del lenguaje RCEI (el más importante para poder desarrollar la narración de la historia. Implementación del código en NWNScript y pruebas de funcionamiento con la aplicación jRCEI.
16-20	● Desarrollo de la implementación de Encrucijada en la Tierra Media preparándola para su posterior uso en KIIDS. Se parte del módulo ya diseñado y se eliminan todo “el cableado” automático de la historia a través scripts.
23-27	● Cambios en la gramática del lenguaje RCEI, ampliación para mejorar el lenguaje simplificando algunos comandos y añadiendo otros. ● Siguiente versión de jRCEI, mejoras en el interfaz y en la arquitectura del sistema. ● Pruebas alpha de Crossroads in Middle-Earth

2.3. Tercera Iteración

Esta última etapa se ha centrado principalmente seguir desarrollando RCEI y en empezar a sentar las bases para la parte de la base de conocimiento del sistema. Hemos hecho cambios en el módulo original creado para funcionar de manera autonoma con Neverwinter Nights para adaptarlo a nuestro interfaz de comunicaciones (jRCEI) que en un futuro se integrará con KIIDS [7] (Knowledge-Intensive Interactive Digital Storytelling system) en tiempo real. Para que KIIDS opere sobre nuestro juego debemos definir las ontologías para Encrucijada en la Tierra Media.

Mayo

Semana	*Trabajo Realizado*
30-4	● Ampliaciones en jRCEI (versión 0.3). Se incluye un interfaz para el su manejo desde otras aplicaciones. Se añade un manejador para posibles usos de RCEI con el manejo de agentes. ● Se modifican la gramática, y parte de la documentación para hacer pequeños cambios en el lenguaje.
7-11	● Población de la ontología de KIIDS con extensiones NWN para Crossroads in Middle-Earth. ● Pruebas de comunicación entre KIIDS y NWNX. ● Desarrollo de documento sobre la ontología que formaliza y detalla aún más lo que es un "entorno de control remoto": RCEOnto. Uso de googleCode y googleDocs (Ej: http://code.google.com/p/dlmodel/)
14-19	● Continuación del desarrollo de la ontología. ● Integración KIIDS>Crossroads con Encrucijada de la Tierra Media B.
21-25	● Continuación del desarrollo de la ontología. ● Continuación de la integración KIIDS>Crossroads con Encrucijada de la Tierra Media B. ● Manual de usuario resumido para Crossroads in Middle-Earth como preparación de las pruebas beta. Pruebas beta de Crossroads in Middle-Earth ● Día 25, día del Orgullo Friki (primer test a un jugador con el juego controlado por KIIDS) (Fede va a estar todos los días en la Facultad. Reuniones rápidas para planificar rápidamente futuros cambios)

Junio

Semana	*Trabajo Realizado*
28-1	● Escribir artículo sobre jRCEI partiendo de lo que ya tenemos elaborado en inglés. ● Fede tiene que escribir un artículo sobre KIIDS para enviar a ICVS. ● Revisar y pulir la memoria y dejarla lo mejor escrita posible.
4-8	Fede se va a Austria
11-15	
18-22	EXAMEN. Últimos retoques de la memoria y de la actual versión de jRCEI y entrega final. Recolección de todo el material para entregarlo en CDs.

3. REVISIÓN DEL ESTADO DEL ARTE

A continuación vamos a hacer una breve descripción de las herramientas que se han utilizado en la elaboración del proyecto.

3.1. Aurora

Aurora [23] [24] es el motor de juego que viene con Neverwinter Nights. Con Aurora podremos diseñar nuestras propias campañas. Aurora es un motor de juego donde los elementos que se utilizan en el juego están prediseñados, así que no será necesario ponerse a implementar personajes o escenarios en 3D con programas como 3DStudio, Maya o aplicaciones de este tipo, aunque se podrían importar para crear personajes personalizados en formato MDL [15].

Aurora es una herramienta muy completa, que permite desde el desarrollo de escenarios, edición de diálogos, personajes, sonidos, eventos, etc., además permite el uso de scripts en C [28] para realizar diversas acciones. Desarrollaremos las diferentes partes de Aurora en mayor profundidad en el apartado 6.

3.2. WordGenerator y NameGenerator

WordGenerator es una aplicación que podemos para generar palabras aleatorias que podemos encontrar en la Web www.kessels.com [12]. Además de esta, podemos encontrar algunas otras aplicaciones interesantes como por ejemplo Name Generator, PasswordGenerator, Random Sentence Generator, The Oracle Says, Shakespearian insult generator y Other random things.

Este programa generará palabras aleatorias artificiales. Esto está pensado para situaciones creativas en las que se necesita un nombre para un password o un nuevo nombre para un producto o para una compañía.

El programa usa un texto de ejemplo como entrada, cuenta la ocurrencia de una secuencia de caracteres, y genera palabras basadas en esas estadísticas. Por ejemplo: 'n' es más probable que sea seguida por 'e' que por 'q'. Si el ejemplo de texto es francés, entonces las palabras generadas sonaran como las palabras francesas. Tendremos palabras "agradables" de ser pronunciadas, usualmente más fáciles de recordar que las palabras completamente aleatorias. Algunas veces el generador generará palabras ya existentes.

Un modo diferente de generar palabras es usar NameGenerator, que genera nombres aleatorios seleccionando 3 sílabas aleatorias de una lista. Por ejemplo, "Gal", "a", y "mond" son combinados para formar "Galamond". Los nombres generados son realmente buenos y se les puede dar mucho uso en juegos de rol , de fantasía y otros de ese estilo.

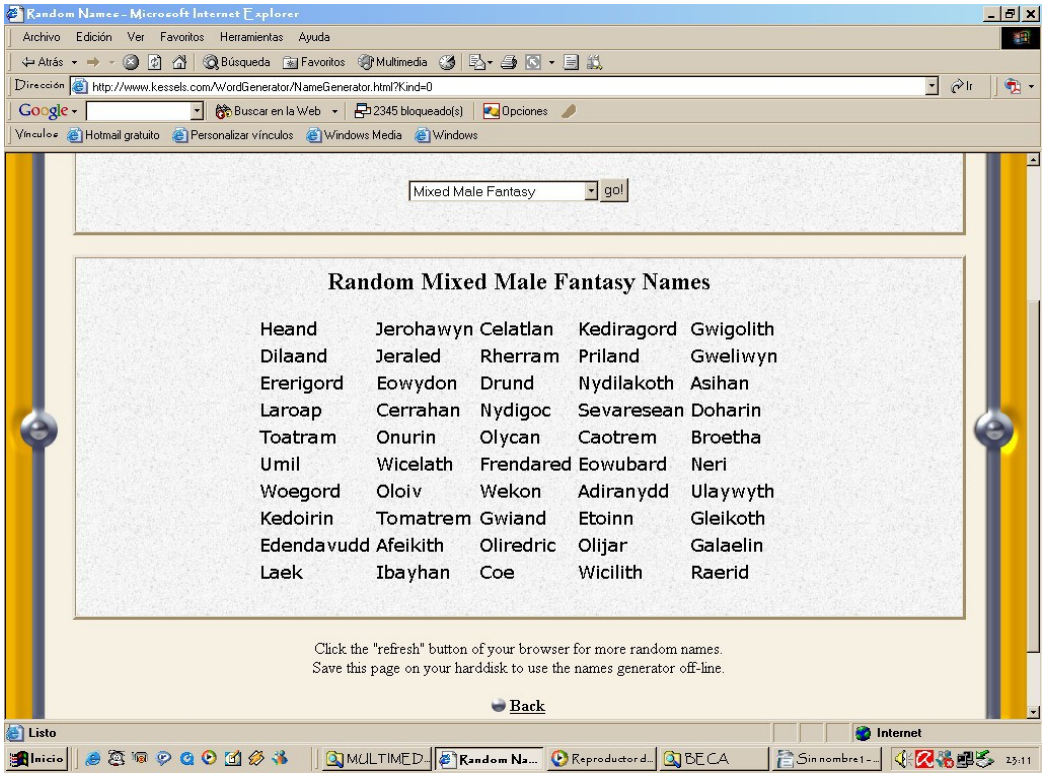


Fig. 2. Captura de la web de WordGenerator. Como se puede ver basta con seleccionar en la pestaña el tipo de raza y darle al botón Go. Si se quieren ver más nombres aleatorios basta con refrescar en el explorador.

Para el proyecto se ha usado solamente la aplicación NameGenerator para generar los nombres de los personajes. Esta sencilla aplicación consta de una pestaña donde podemos seleccionar unas cuantas opciones sobre el tipo de personaje para el nombre que vamos a generar. Entre estás opciones se encuentran si el personaje es masculino o femenino o el tipo de raza según el universo de Tolkien, como pueden ser elfos, orcos o humanos.

3.3. NwnMultiLang

Esta herramienta se utiliza para seleccionar todos los componentes (ficheros) que componen un módulo y volcarlos a un fichero de texto. NwnMultiLang [13] facilita la labor de la traducción, ya que en un módulo de Neverwinter Nights se puede traducir a varios idiomas cada elemento del módulo, ya sean personajes, objetos, diálogos, etc., pero para hacerlo de todo el módulo habría que ir accediendo manualmente uno a uno a todos los elementos, hecho que haría perder tiempo a una persona que se está dedicando solamente a traducir.

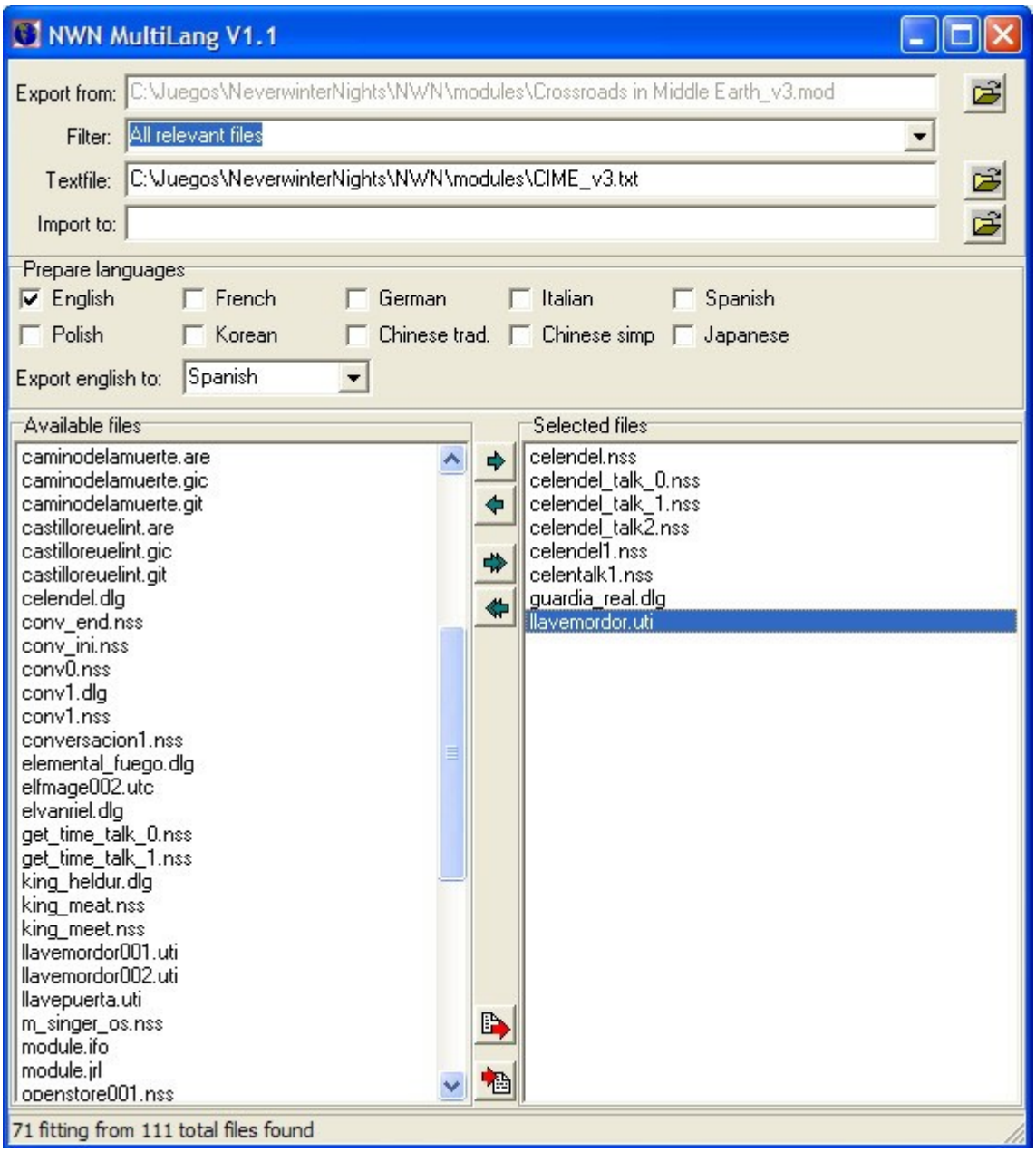


Fig. 3. Captura del programa NwnMultiLang. Como se puede ver basta con seleccionar los archivos deseados para extraerlos en un fichero de texto.

Para realizar la traducción hay que indicar en el apartado export from la ruta donde está el archivo con extensión .mod con el nombre del módulo. Internamente el archivo .mod está compuesto por varios archivos distintos. Debemos seleccionarlos y posteriormente exportarlos como texto. Una vez que tenemos este fichero hay que cambiar todos los nombres y conversaciones de un idioma al otro idioma.

Una vez que se haya traducido todo el texto en el propio fichero, debemos realizar el proceso inverso. Primero cargamos el fichero de texto ya traducido y seleccionamos el nombre del módulo al cual lo vamos a importar. De esta manera la traducción se realizará de manera más rápida.

3.4. RAD Video Tools

Esta herramienta permite transformar vídeos en formato avi, mov o cualquier otro formato de vídeo a formato bink video que es el que usa Neverwinter Nights para mostrar vídeos a modo de introducción de una campaña, cuando terminamos alguna misión importante, en los créditos, etc.

Podemos bajarnos está herramienta gratuitamente de la web www.radgametools.com [14], después hay que instalarla y ejecutarla. Esta herramienta puede ser útil no solo para Neverwinter Nights sino para otros juegos, ya que muchos otros juegos usan este formato para mostrar vídeos y animaciones.

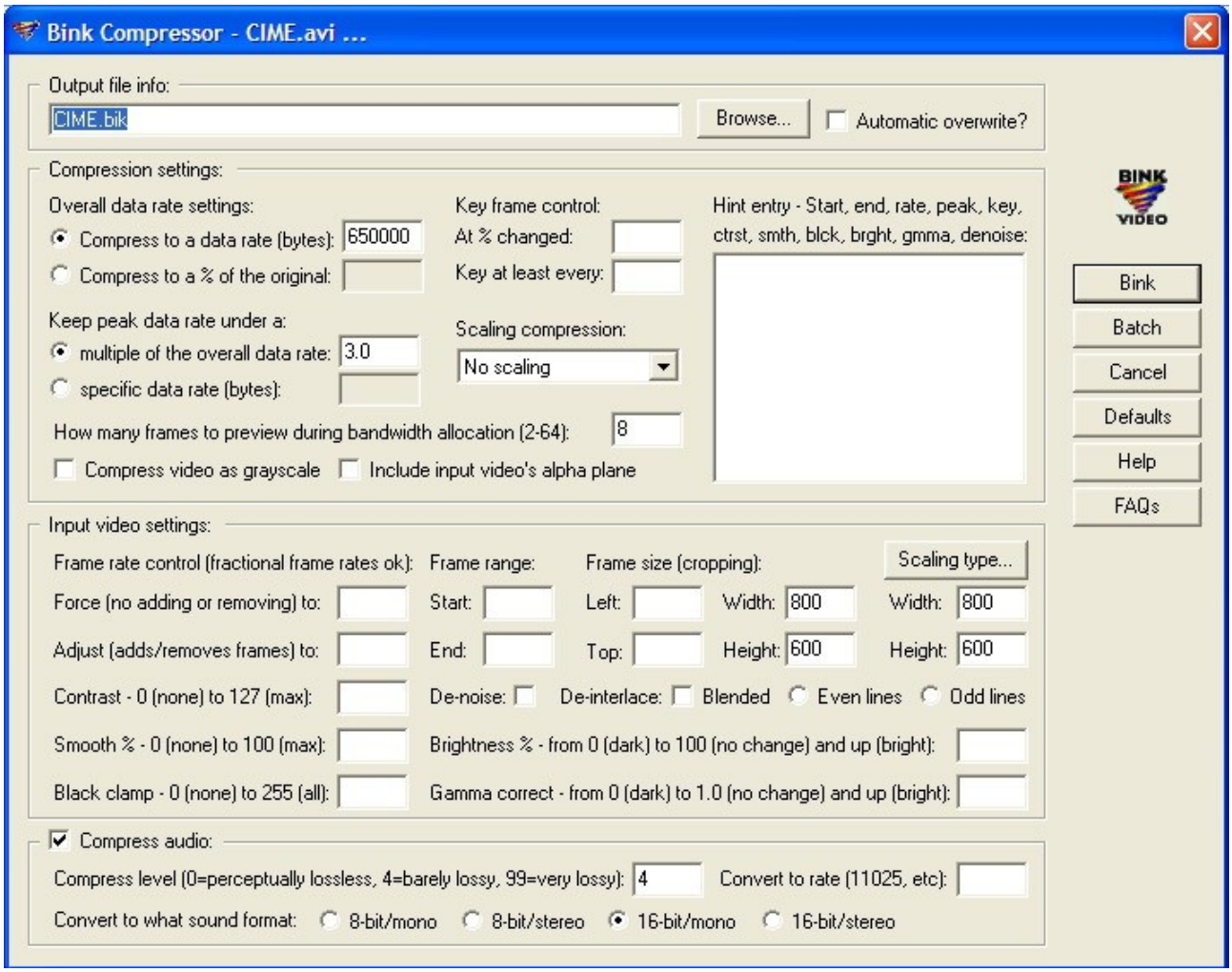


Fig. 4. Captura de RAD Video Tools. En primer lugar seleccionamos el vídeo que queramos transformar. Posteriormente aparecerá está pantalla donde aparecen varios parámetros de configuración del vídeo.

En primer lugar nos aparecerá una ventana desde la cual debemos seleccionar el vídeo que queramos transformar a formato bik. El vídeo que queramos transformar puede ser un vídeo que nos den o uno que hagamos nosotros mismos. En la web de NWN hay un tutorial de como realizar vídeos [30] con PowerPoint y luego pasarlos a formato avi.

Para pasar a formato avi unas transparencias de PowerPoint se usa la aplicación llamada Camtasia Studio [15], disponible en gratuitamente por periodo de prueba de 30 días.

Para crear un módulo con un vídeo de introducción basta con guardar el vídeo en formato bik en la carpeta movies donde tengamos instalado Neverwinter Nights, después en Aurora debemos ir a las propiedades del módulo, y a la pestaña avanzado. Aquí hay una opción que pone película de inicio, donde podremos seleccionar de una lista todos los vídeos que están en la carpeta movies.

3.5. Neverwinter Nights Extender

Neverwinter Nights Extender [5] (NWNX) es un programa que tiene el poder de sobrepasar los límites iniciales de NWN y de su lenguaje integrado de scripting. Hay dos versiones de Neverwinter Nights Extender; NWNX2 que sirve para realizar la comunicación con el juego Neverwinter Nights y NWNX4 que sirve para realizar la comunicación con el juego Neverwinter Nights 2.

Mientras NWN y sus herramientas son complejas y potentes, hay ciertas cosas que están limitadas, NWNX permite la conexión a bases de datos mediante ODBC, el uso de nuevas estructuras de datos en NWN como tablas hash, listas enlazadas, etc.

La distribución de NWNX consiste en dos partes: el ejecutable, que despliega una ventana y hace correr el proceso watchdog y la aplicación gamespy watchdogs , y una librería dinámica (DLL en windows, SO en linux), que corre dentro del servidor NWN.

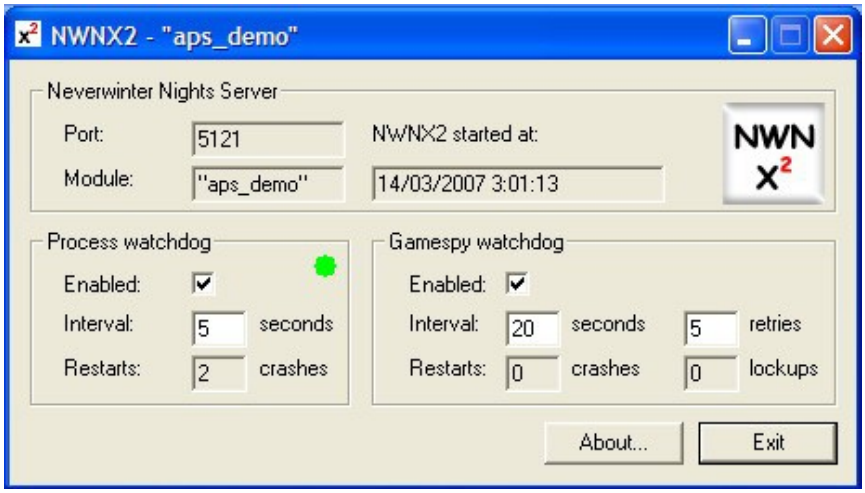


Fig. 5. Ventana que aparece al ejecutar el archivo ejecutable NWNX2.exe. Después de esto aparece la ventana del servidor.

Para establecer una comunicación entre scripts y el mundo exterior, NWNX intercepta llamadas de la función de NWNScript [28] ***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. Además NWNX soporta todas las versiones de Microsoft Windows.

NWNX tiene disponibles una serie de plugins que pueden resultar de mucha utilidad, las últimas versiones con toda la documentación están disponibles en su web (nwnx.org). Hay 5 plugins disponibles: odbc, profiler, hashset, functions, time.

A. ODBC. Es el más importante, hace poco tiempo, no existía una solución para tener una base de datos en tiempo real para NWN. Cuando el equipo de Avlis sacó la primera versión de NWNX se abrió un mundo a los propietarios para crear mundos persistentes.

B. PROFILER. El Profiler es un plugin que reveló las estadísticas de los scripts en ejecución. Muestra la frecuencia en que un script es invocado, cuanto tiempo le lleva para ejecutarse, y da unas estadísticas globales. También muestra el tiempo de CPU necesario para encontrar el path.

C. HASHSET. Este plugin implementa una serie de estructuras de datos complejas combinando una tabla hash y un conjunto. Esto permite usar estructuras de datos en NWNScript que no posee originalmente con lo que podremos realizar scripts con algoritmos más complejos.

D. FUNCTIONS. Este es un plugin experimental que da acceso a nuevas funciones para usarse con los items. Su propósito principal es como modificar la memoria del servidor para cambiar varias propiedades de los items.

E. TIME. Este plugin implementa un temporizador simple pero altamente preciso. Perfecto para saber cuanto tardan realmente tus funciones en ejecutarse.

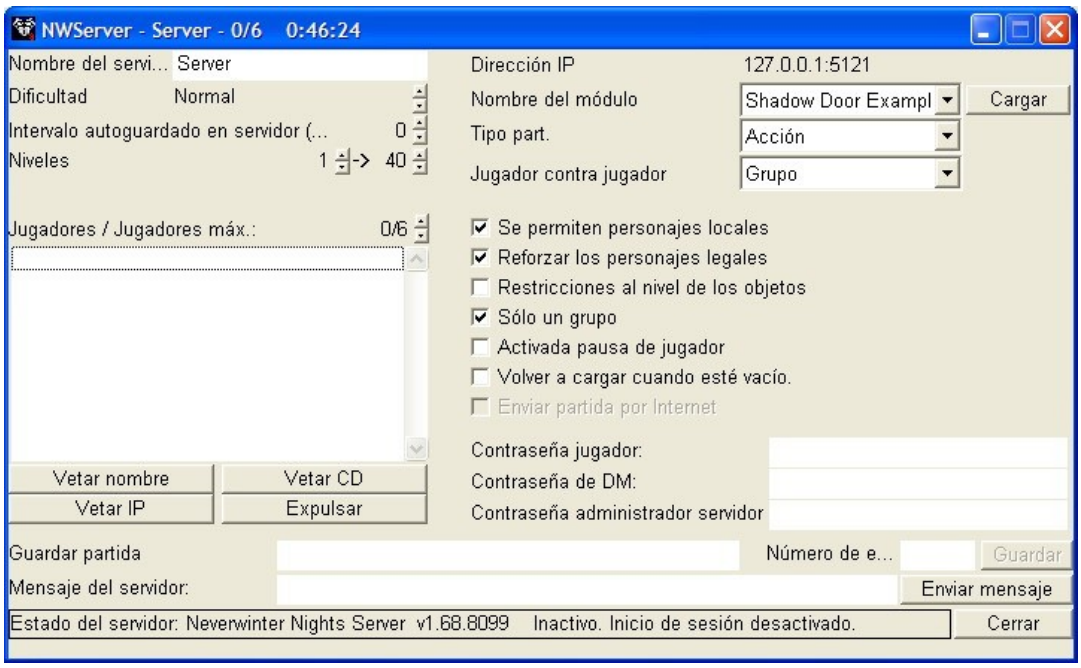


Fig. 6. Ventana del servidor NWServer, donde cargaremos un mod para jugar en red y configuraremos las opciones del mismo a nuestra conveniencia.

3.6. Shadow Door

Shadow Door [6] es un interfaz de control de agentes para el juego Neverwinter Nights desarrollada por ***Phillip Saltzman*** y ***Robert Zubek*** en la universidad ***Northwestern University***. 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 que hagan uso de Inteligencia Artificial, usando lenguajes arbitrarios y plataformas arbitrarias, para jugar al juego con jugadores humanos.

La distribución de Shadow Door consiste en:

1. Una DLL para la extensión del servidor, 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.

El uso de Shadow Door [6] conlleva el uso de 3 procesos diferentes que trabajan de manera conjunta:

I) Un proceso externo que envía comandos y recibe observaciones a través de sockets usando el protocolo definido por ShadowDoor.
II) Una extensión del servidor que realiza una traducción entre el la información del socket y la información que del juego en curso.
III) Un conjunto de scripts de NWN que ejecuta comandos y devuelve observaciones.

Cuando un usuario corre *NWNX2* con la librería Shadow Door, comienza a ejecutarse un nuevo servidor del juego, pero también abre un socket en el puerto 1890 y se pone a escuchar comandos entrantes. El diagrama de su funcionamiento es el siguiente:

Process #1 **External AI controller (Lisp, etc.)**	
| | | |	Shadow Door Protocol over TCP on port 1890
Process #2 **NWServer + NWNX 2.5 + Shadow Door**	
| | | |	BioWare'sBioWare's Propertiary Protocol
Process #3 **Neverwinter Nights Graphical Client**	

La librería ShadowDoor 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. Las funciones son las siguientes:

void SD_Init ()	Inicializa los scripts de Shadow Door
string SD_RequestMsg () => <*result*>	Si el resultado es 0 significa que no hay mensajes disponibles. Si el resultado es -1 que actualmente todavía se está procesando un mensaje.
string SD_GetNextToken () => <token>	Devuelve el siguiente token en el comando actual que se está procesando. El primer token siempre es el nombre o el tipo de comando y el resto son los argumentos propios de cada comando. Devuelve un string vacío si no hay más token disponibles o si el token reconocido no está especificado en el lenguaje.
void SD_ReleaseMsg ()	Indica el fin del procesamiento de un mensaje. La función se invocará después de finalizar todos los procesos relativos al mensaje.
void SD_Send (string <*message*>**)**	Envía un mensaje en un string a través de Shadow Door.

En Neverwinter Nights todos los personajes y objetos tienen una serie de eventos a los que puede asociárseles un script para ejecutar ciertas acciones. Por defecto en todos los personajes vienen unos scripts ya implementados. Si no se asocia ningún script el evento no se produce.

Los eventos más importantes son OnConversation, OnHeartbeat, OnPhysicalAttacked, OnSpawn. A través de los scripts que se ejecutan se realizan las acciones de un NPC, como atacar, hablar, moverse, etc.

Estos cuatro eventos serán los que cambiemos con los nuevos scripts que usarán las funciones de Shadow Door que se han mencionado antes. Todos los scripts que usen funciones de Shadow Door comenzaran por **sd_** , además internamente deberán incluir en la cabecera el script donde se encuentran estas funciones (#include "sd_include").

El script asociado a OnHeartbeat será el más importante de todos. En este script, se van leyendo los comandos enviados externamente a Neverwinter Nights y procesados para ejecutar una determinada acción.

Por defecto este script se ejecuta una vez cada seis segundos, por eso en el código se utiliza un comando de NWNScript para que cada vez que se ejecute el script el método principal se ejecute 6 veces cada vez con un retraso de un segundo.

El método principal de sd_on_heartbeat lee el primer token del comando para distinguir que acción se va ejecutar. Según la acción que se ejecute, como mover, atacar, etc., llamaremos a la función correspondiente de NWNScript con los parámetros necesarios, que obtendremos al solicitar los siguientes tokens.

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 (LF). La sintaxis por defecto de Shadow Door es la siguiente:

```
<command-or-observation>[#[<parameter>]]+<LF>
```

Con la siguiente gramática:

```
command :=head body terminator
head := token
body := parameter body | parameter
parameter := separator token | separator
separator := '#'
token := tokenchar+
tokenchar := any ASCII character - { CR , LF , # }
terminator :=LF
```

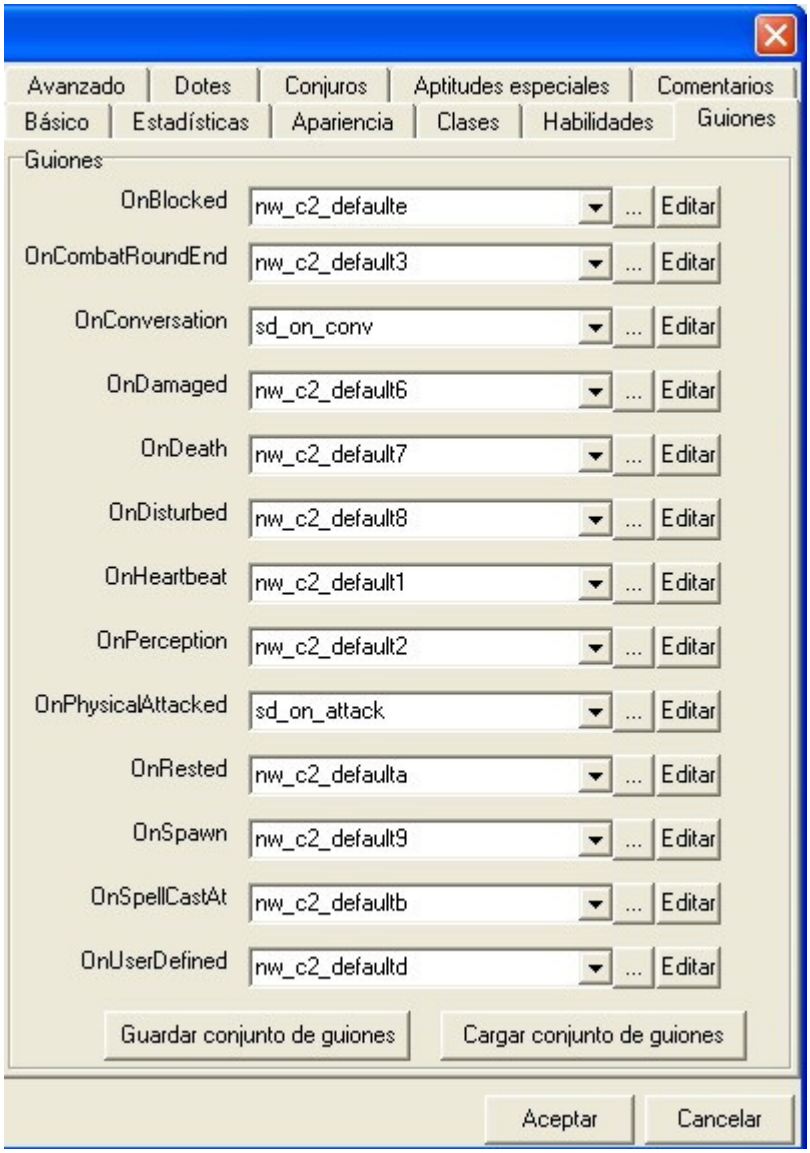


Fig. 7. Captura de los scripts asociados a un NPC (Non Player Character).

Una de los principales limitaciones de Shadow Door que nos afectaba directamente era que en el ejemplo que viene con Shadow Door sólo se puede controlar externamente un único personaje.

Esto es debido a que la funcionalidad del control de cada personaje se implementa en OnHeartbeat, y si asociamos el mismo script a más de un personaje, las acciones se bloquean porque se está ejecutando concurrentemente y mismo script, y el sistema no sabe cual de los personajes debe realizar la acción.

Este problema se ha solucionado usando OnHeartbeat solamente en el módulo y modificando el script. En la mayoría de instrucciones se añade un token para identificar que NPC u objeto está realizando la acción. Las funciones que antes estaban en OnHeartbeat, las llevamos cada una a un script a parte con un método main() donde se realiza la acción concreta, atacar, hablar, etc.

Desde OnHeartbeat usaremos la siguiente función para que un NPC cualquiera o un objeto ejecute un script concreto: ***void ExecuteScript(string sScript, object oTarget)***

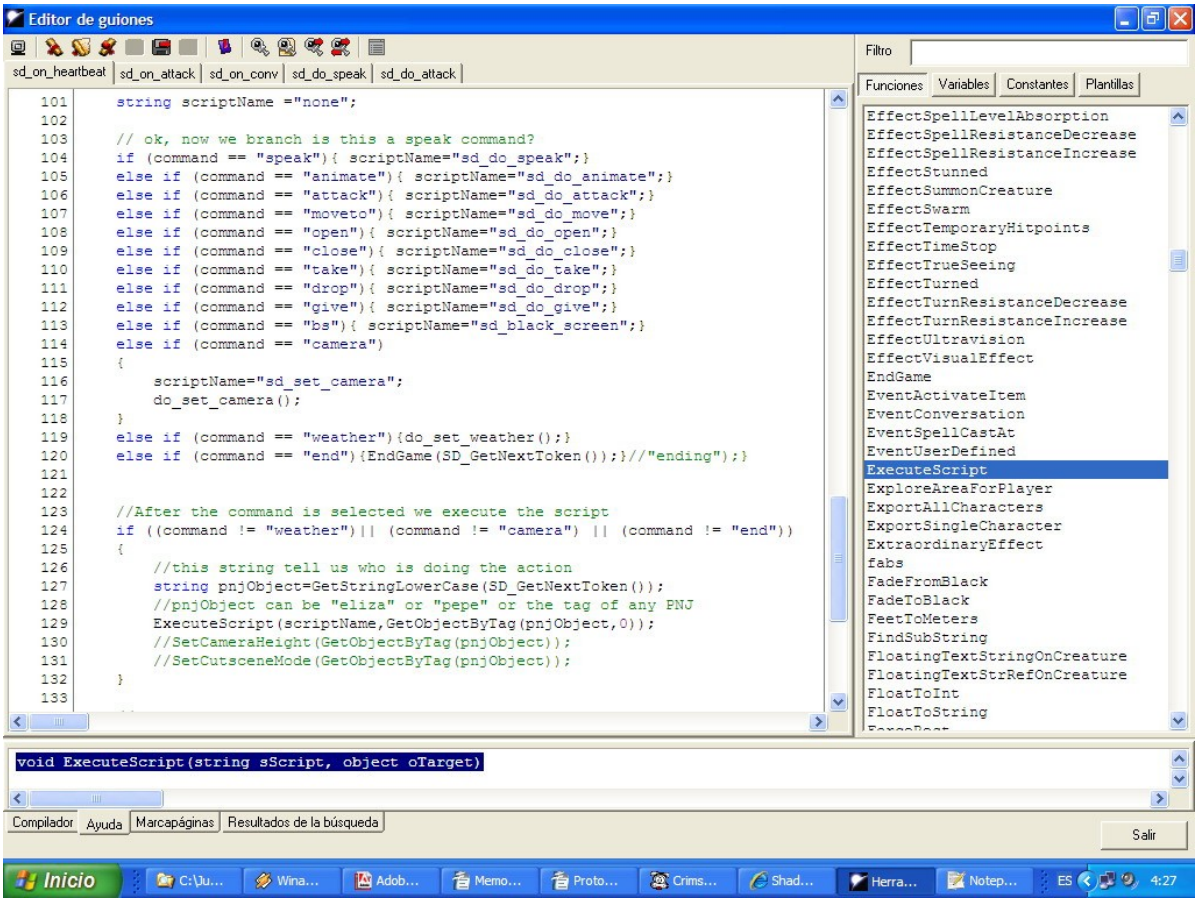


Fig. 8. Edición de los scripts con NeverwinterScript. Este es el aspecto del editor de scripts con el que debemos programar los scripts del juego, y en particular los asociados a ShadowDoor para realizar la comunicación hacia el exterior.

4. DISEÑO DE LA BASE DE CONOCIMIENTO

El planteamiento inicial de Encrucijada en la Tierra Media funcionando integrado con el sistema KIIDS [7] era el de realizar una base de conocimiento con 7 casos distintos. Cada uno de estos casos se adaptaría a uno de los 7 modelos de historia diferentes que existen según el libro "The Seven Basic Plots" [18]. La idea es que teniendo varios casos representativos se podrán obtener resultados interesantes desde el punto de vista de la interactividad de la narración.

Para la creación de dichas historias se han utilizado referencias bibliográficas sobre el mundo de Tolkien [21], también se han usado documentos que detallan todos los perfiles de jugadores de rol [20] para adecuarlos a los protagonistas de cada una de las historias y otros materiales de donde sacar ideas bien para la creación de la historia o bien para detallar elementos de las ontologías como puede ser el libro de rol del Señor de los Anillos.

Fig. 9. Captura de Encrucijada en la Tierra Media en el que el protagonista se enfrenta a una criatura. Como se puede ver, la temática del juego es como el universo de Tolkien, el protagonista es un caballero que se enfrenta a todo tipo de criaturas.

Inicialmente se decidió diseñar estos 7 prototipos de historias, pero posteriormente debido a la limitación de tiempo de la duración de la beca de colaboración y a las pruebas realizadas sobre la implementación del primero de los casos, se decidió usar una sola historia, con vistas a ampliar la base de casos cuando en el futuro en sistema KIIDS esté integrado con RCEI y preparado para su visualización en Neverwinter Nights [1] o quizás más adelante en otro entorno. De todas formas se han elaborado 7 documentos con la especificación de cada una de las historias, incluyendo los personajes, ubicaciones, la trama, el trasfondo y todo lo relativo a la historia.

El módulo que se implementó en Neverwinter Nights para la primera historia, tenía una duración de una hora aproximadamente, por lo cual se decidió invertir tiempo en otras partes del proyecto porque se puede subdividir en varias historias o como ya se ha dicho incluir más casos posteriormente.

Para la implementación de la base de conocimiento consideraremos el uso de estas herramientas para crear las ontologías en formato OWL:

- **Protégé-OWL [9]:** Utilizamos la última beta disponible porque incorpora más funcionalidad relacionada con la importación de ontologías.

- **SWOOP [10]:** Consideramos el uso de Swoop como alternativa a Protégé debido a que para los archivos en formato OWL genera menos código y lo organiza de una manera más ordenada.

- **Pellet [11]:** La justificación para usar Pellet en vez de Racer es básicamente económica. La funcionalidad es similar, pero Pellet se diferencia por ser un sistema gratuito y de código libre. Parece ser que el servidor DIG de Pellet escucha en el puerto 8081 en vez del 8080 que es el que tiene configurado por defecto Protégé.

Se ha realizado la documentación detallada de una ontología que usará KIIDS para Encrucijada en la Tierra Media. Esta ontología, llamada *RCEOnto*, describe todos los procesos que se pueden realizar en el entorno, asi como propiedades, conceptos y relaciones del entorno.

El sistema KIIDS hará uso de otras ontologías propias como *KnowledgeOnto* para modelar el conocimiento, *TimeOnto* para modelar el tiempo, *WorldOnto* para modelar el mundo, *SimulationOnto* para modelar la simulación del entorno y algunas otras.

La implementación de la base de conocimiento ha quedado pendiente; aunque se han comenzado a definir detalladamente las ontologías necesarias para Encrucijada en la Tierra Media, corriendo sobre KIIDS [7], pero no ha dado tiempo por la duración de la beca a probar las ontologías integradas en KIIDS y hacer pruebas. Esta labor esperamos completarla en un futuro próximo.

5. IMPLEMENTACIÓN DEL ENTORNO VIRTUAL

La implementación del entorno virtual se realizará íntegramente con la herramienta Aurora [24] [25] incluida en Neverwinter Nights [1]. Aunque también usaremos una expansión llamada CEP [3] (Community Expansion Pack) que se debe instalar conjuntamente con Neverwinter Nights para poder ampliar el repertorio de objetos y personajes disponibles por defecto en el editor Aurora.

La implementación del entorno virtual estará dividida en tres partes: implementación de escenarios, implementación de los elementos de interacción e implementación de la interacción de todos los elementos de cada aventura. Aurora es el motor de juego del Neverwinter Nights y proporciona un entorno adecuado para que los usuarios puedan editar sus propias aventuras y expansiones a las campañas del Neverwinter Nights.

La inspiración inicial del juego surge de la Tierra Media, aunque evitaremos usar nombres reales de personajes o lugares que aparezcan en los libros para evitar problemas de copyright. El juego se realizará inicialmente en español, dejando para el final la posibilidad de traducirlo al inglés.

Fig. 10. Captura de escenario del caso 2 de Crossroads in Middle-Earth. Interior del castillo de Reuel donde se encuentra el rey con la gente de palacio , el personaje debe de ir a conversar con ellos.

La implementación de escenarios y demás elementos de juego es la parte de la implementación menos complicada. Por ejemplo, teniendo un boceto previo de un área, es un relativamente sencillo llevarla a cabo, o elegir los personajes y objetos de la aventura. Sin embargo la interacción de todos los elementos de juego es compleja, ya que requiere hacer uso de código para implementar estás relaciones.

El código que hay que implementar se realiza con NeverwinterScript [28], que es una herramienta con la cual se realizan acciones sencillas a través de scripts programados en C. Estos scripts describen el comportamiento de los personajes, como controlar las conversaciones, etc. Además para programar estos scripts se dispone de *wizards* que ayudan a crearlos de manera sencilla, incluso para quien no sepa programar.

Aunque la mayoría de las acciones que se realizan son básicas y no requieren un código excesivamente complejo, según aumenta el módulo del juego hay que manejar gran cantidad de scripts, variables locales, diálogos, etc. A esto hay que añadir que NeverwinterScript posee gran cantidad de funciones predefinidas que realizan multitud de operaciones en el juego en las que podemos perdernos, pero esto hace posible que se puedan ir complicando los scripts básicos para llevar a cabo comportamientos complejos.

En Internet podemos encontrar varios manuales sobre la herramienta Aurora y sobre como manejar sus componentes. Hay manuales bastante completos, como el manual que viene en el CD del juego [22], aunque también hay otros más prácticos [23] [24] [25] y recomendables para aquellos que no hayan manejado nunca la herramienta, que explican como hacer un módulo sencillo, de manera que vayamos aprendiendo a usar cada una de las partes paso a paso.

5.1. Implementación de escenarios

La implementación de escenarios se refiere a la ubicación donde se desarrollará la aventura, al trasfondo del juego. En esta parte veremos los elementos que ambientarán el juego pero que no realizan funciones en el desarrollo del juego. En Neverwinter Nights se diseñan diversas áreas independientes de un cierto tamaño, todas las áreas se conectarán a través de transiciones de área generalmente caracterizadas por puertas.

El hecho de que un área este bien diseñada es importante por varios motivos, principalmente debe ser atractiva para el jugador que la vaya a usar. Además el tamaño debe ser adecuado en función de lo que se quiere que dure el juego y de lo que pueda tardar el jugador en terminar la aventura. También hay que tener otros aspectos menos importantes como que la transición entre diversas áreas tenga apariencia de continuidad para no desconcertar al jugador y que los objetos y personajes que intervengan en la trama sean fácilmente reconocibles para el jugador. Por ejemplo, si entramos en una casa no tiene sentido que el área siguiente sea un bosque, o si la apariencia exterior de la casa es pequeña el escenario deberíamos encontrarnos unas habitaciones pequeñas y no una mansión descomunal.

5.1.1. Edición de áreas

En Aurora tenemos un escenario en 3D que consta de una serie de cuadrículas, como una matriz nxn, donde cada cuadrícula representa un trozo del escenario de 10x10 metros.
Al comenzar a realizar un área debemos configurar algunos parámetros para seleccionar el tipo de área que deseamos realizar así como el tamaño de área. El tipo de área sirve para definir el tipo de escenario en que el jugador se va a desenvolver, un bosque, el exterior de una ciudad, una cueva, el interior de una casa, etc. Los tamaños de área en número de cuadrículas que se pueden seleccionar por defecto en Aurora son menuda (2x2) pequeña (4x4), mediana(8x8) y grande (16x16), aunque el usuario puede fijar cualquier otro tamaño (nxm).

Una vez que tenemos creada el área, podemos visualizarla en la pantalla y desplazarnos por ella con la ayuda del ratón o de botones de desplazamiento que hay debajo del escenario, como se puede ver en la figura 3.

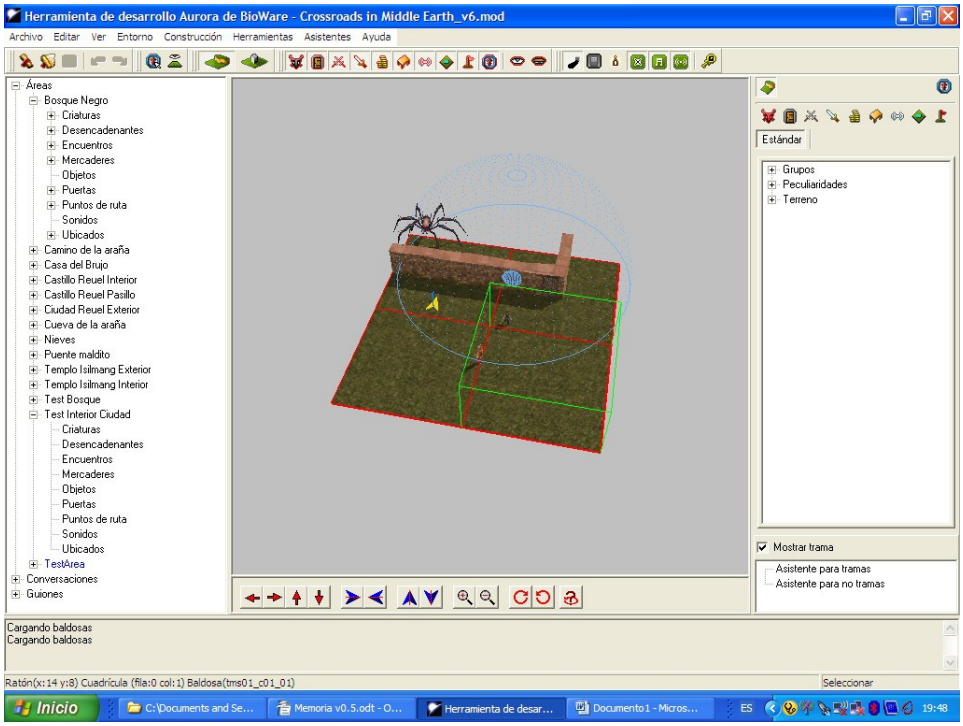


Fig. 11. Captura del editor Aurora para la edición de escenarios. En la parte de la izquierda se tienen todos los áreas que tiene el módulo, cada una con todos sus componentes. A la derecha tenemos una barra de herramientas donde se incluyen los elementos propios de un área (personajes, edificios, etc). Con las flechas que hay debajo nos podemos mover por el área (alejar, acercar, etc).

Para diseñar el área con más elementos el editor contiene una serie de elementos prediseñados que podemos ir añadiendo para modificar el terreno. A nivel de cuadrícula podemos encontrar ríos, paredes, árboles, caminos, subir el nivel, etc. Además de esto hay diversos tipos de edificaciones que pueden ocupar varias cuadrículas.

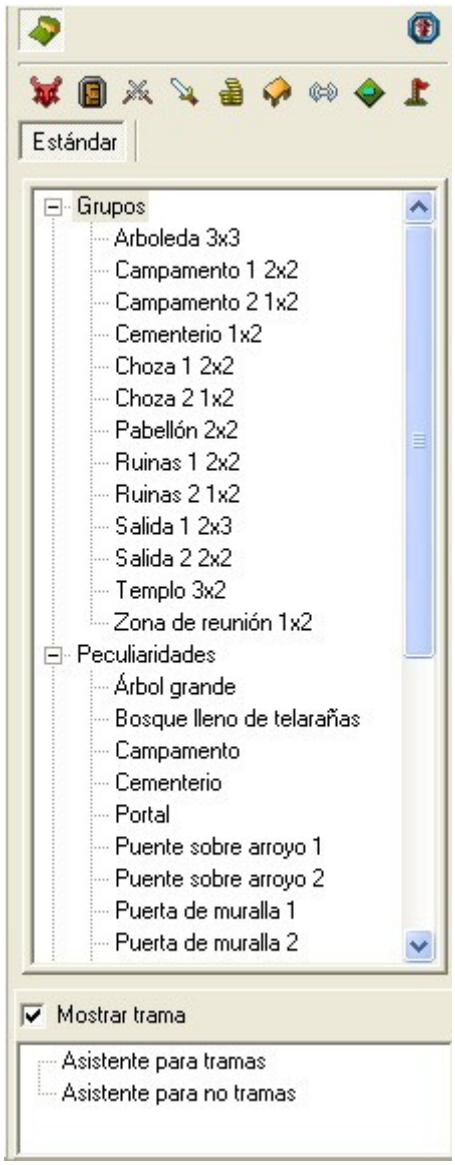


Fig. 12. Captura del panel donde se encuentran los elementos para pintar terreno.

5.1.2. Transición entre áreas

Como ya hemos comentado, en Neverwinter Nights los escenarios se diseñan individualmente y luego se crean transiciones entre los diversos escenarios para dar la sensación de uniformidad en el mundo virtual en que nos movemos. Estas transiciones de área se realizan generalmente a través de puertas, aunque en ocasiones podemos encontrarnos portales mágicos o zonas de transición en una zona del terreno.

Cada elemento de Neverwinter Nights está etiquetado con un nombre, de esta manera cuando se quiere realizar una transición se le indica a la puerta que al usar el usuario la puerta debe aparecer en la puerta que le indique la etiqueta que le indiquemos; de esta manera nos desplazaremos a otro área o a otro lugar dentro del mismo área. Lo mismo ocurrirá con los personajes y objetos, cada personaje tendrá su propia etiqueta, de esta manera se reconocerá a quién estamos hablando o qué objeto estamos usando.

Para seleccionar la transición de una puerta a otra debemos ir a las propiedades del objeto (pinchando con el botón derecho del ratón) y pinchar la pestaña de Transición de Área. En esta pestaña debemos escribir la etiqueta de destino de la puerta a la que queremos transitar. La transición entre distintas puertas puede ser en los dos sentidos o también de la puerta origen a la puerta destino o viceversa.

Al implementar el módulo debemos tener cuidado con no nombrar a dos puertas con el mismo nombre, aunque estén en áreas distintas, porque las transiciones pueden no funcionar bien. También debemos tener cuidado cuando renombremos las etiquetas de las puertas porque en ese momentos las transiciones que había establecidas dejarán de funcionar y debemos volver a indicarlas.

5.2. Implementación de los elementos de interacción

En esta parte comentaremos los elementos intervienen en la trama a la hora de diseñar el juego y como manejarlos, esto es personajes, objetos, el diario, los encuentros con enemigos, los puntos de ruta, etc.

5.2.1. Personajes

Los personajes de juego están predefinidos y basta con seleccionarlos de la paleta y ubicarlos en el terreno. En Neverwinter Nights hay dos categorías principales de personajes Monstruos y PNJ´s (Personajes), o en inglés NPC´s (Non Player Characters).

A su vez cada uno se divide en varias categorías y subcategorías para los tipos de razas de personajes, enanos, hombres, elfos, orcos, etc., o los distintos tipos de monstruos animales, gigantes, dragones, insectos, etc.

Una vez que tenemos el personaje en el territorio podemos seleccionarlo y configurar todos sus parámetros como seleccionar el diálogo que hemos diseñado previamente, ponerle un nombre, cambiarle la etiqueta, el nivel del personaje, su inventario, los comentarios que ve el jugador cuando le observa, características como inmortalidad, aspecto gráfico, etc.

Cuando tengamos el personaje con las características deseadas basta con darle aceptar y ya lo tendremos configurado a nuestro gusto.

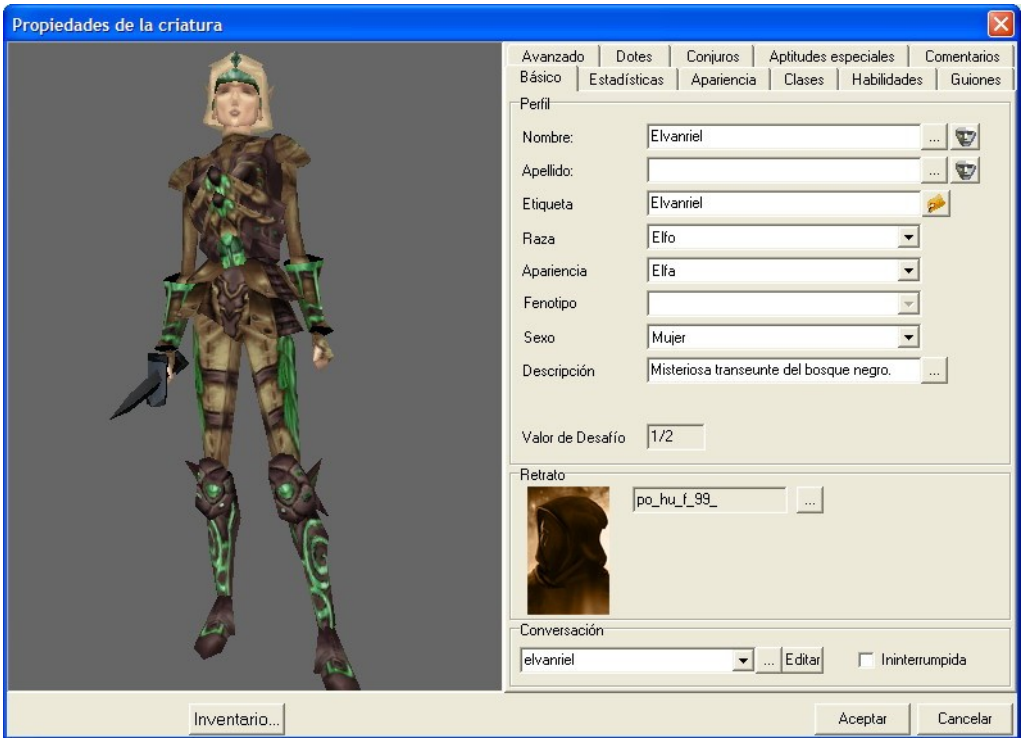


Fig. 13. Captura del editor Aurora. Podemos ver como se configura un personaje.

5.2.2. Objetos

En Neverwinter Nights hay dos tipos de objetos: objetos ubicables y objetos "usables". Los objetos ubicables son los que forman parte del paisaje del escenario, pueden ser meramente decorativos o pueden ser objetos que contengan en su inventario otros objetos que el personaje puede coger o usar, como por ejemplo, cofres, cajas, armeros, etc.

Los objetos que el jugador puede coger los puede llevar al inventario y usarlos, darlos, dejarlos, intercambiarlos, equiparse con ellos, etc, dependiendo del tipo que sean. Los hay de diversos tipos como pueden ser: hechizos, armas, capas, pociones, monedas, llaves, etc.

Hay ciertos objetos que son indispensables para poder terminar la aventura, ya que estos objetos están involucrados directamente con una acción que está esperando su uso para poder pasar a realizar otra acción, es decir, pasar a otro estado; estos objetos se llaman objetos de trama.

5.2.3. Diario

El diario es una herramienta para ayudar al jugador a avanzar en la aventura, se abre con un botón del menú de la derecha en el juego y en él se encuentran las misiones que estamos llevando a cabo y las misiones que hemos completado. En las misiones completadas nos aparecerá que hemos realizado el objetivo final y lo que hemos hechizo.
En las misiones en curso irán apareciendo pequeñas pistas de lo que debemos realizar a continuación para que avancemos en la misión y finalmente completarla.

Las diferentes entradas de una misión se van cambiando en un nodo del árbol de diálogo de algún personaje cuando cumplimos un objetivo. Es decir, cuando ya hemos hecho algo que nos estaba indicando el diario que había que hacer eso desaparece y se nos indica que es lo próximo que hay que hacer.

Cada misión del diario se puede crear desde el editor de diarios y añadirle todas las entradas que queramos y desde los diálogos con personajes irle indicando cuando se muestra cada entrada, hasta la última entrada en la que la misión pasa a la lista de misiones completadas.

Si usamos el editor de tramas, crearemos el diario y las entradas del mismo cada vez que creemos un nodo de la trama, además se configurará automática que los nodos de todos los diálogos que activan la entrada de un diario.

5.3. Implementación de la interacción

En este apartado detallaremos los elementos que diseñamos en Aurora para que el jugador se relacione con el entorno, principalmente son los diálogos, los ataques a personajes, uso de objetos, etc.

5.3.1. Edición de conversaciones

Las conversaciones en Aurora se crean mediante un editor que permite una estructura de árbol, donde en cada nivel del árbol se suceden preguntas y respuestas entre jugador y otro personaje.

En cada nodo se pueden asociar scripts para establecer una condición sobre un nodo, es decir, que sólo se visualice una frase si se ha cumplido alguna condición o también para ejecutar una determinada acción al llegar a un punto de la conversación.

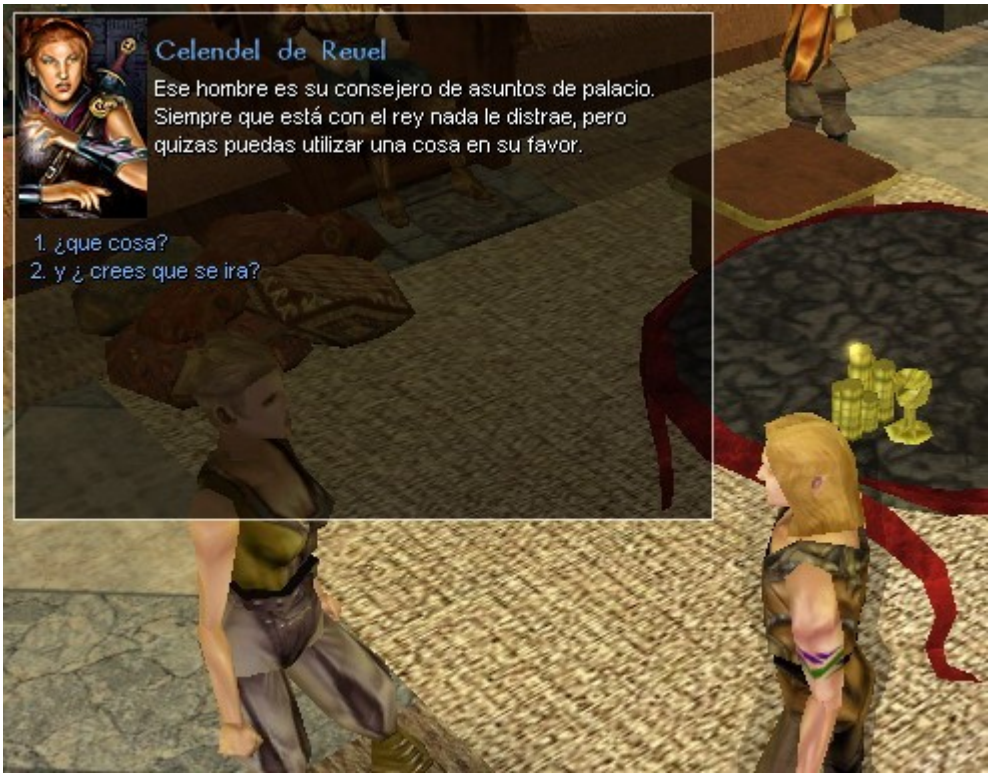


Fig. 14. Captura de escenario de Encrucijada en la Tierra Media. Conversación del personaje con la hija del rey para que le ayude a llegar hasta el rey.

Cualquier conversación medianamente extensa requiere de el uso de variables locales al personaje para marcar los tramos de conversación que ya ha leído el jugador y no repetirlos.

Posteriormente para que la conversación arranque al hablar con un personaje determinado debemos ir a las propiedades del personaje concreto y asociar en la pestaña de conversación del personaje el nombre de la conversación que hemos creado.

5.3.2. Asistente para tramas

Existe una herramienta en Aurora NWN que sirve para diseñar tramas [26] de forma genérica. En vez de tener que crear los personajes y los diálogos de manera independiente y luego adaptar los diálogos para realizar determinadas ejecutadas mediante scripts [27], este asistente permite diseñar misiones o búsquedas mediante un guía paso a paso.

El esquema de cualquier búsqueda es muy simple:

1) El PNJ encarga la misión al jugador.
2) El jugador resuelve esa misión (buscar algo, matar a alguien).
3) El PNJ entrega recompensa al jugador por el trabajo realizado.

Se puede enredar el asunto de muchas maneras. Por ejemplo, que no sepamos donde está el malo al que debemos matar, y que debamos pedir ayuda a otro PNJ, que a su vez nos encargue otra misión a cambio del soplo. Pero son simples repeticiones o modificaciones del esquema básico.

Al iniciar una nueva trama (o misión) en el asistente de tramas debemos configurar los siguientes apartados.

1) **<u>Opciones básicas.</u>**
   - Nombre de la trama.
   - Nombre del diario, en el que iremos añadiendo los avances del personaje.
   - Tipo diseño de trama
     - Asesinato con prueba
     - Asesinato sin prueba
     - Buscar y enviar
     - Buscar y recuperar

2) **<u>Definir el reparto.</u>**
   En este apartado se define el rol de los personajes que intervienen en la trama.
   - Iniciador de trama
   - Villano
   - Extras (objetos ubicados)

3) **Definir los soportes.**
   Se incluyen los objetos de trama indispensables en el desarrollo de una misión. Estos objetos son portables en el inventario del jugador o de otros personajes.
   Crear nodos de trama.

4) Crear nodos de trama.
   Esta parte es la más importante y se corresponde con el diagrama de actividad de la parte de interacción de las misiones. Se crean varios nodos para implementar cada una de las submisiones que componen una misión completa en el diario. Cada uno de estos nodos tiene varias opciones para configurar una serie de misiones distintas con el mismo patrón.

   Este apartado permite realizar una jerarquía en la ejecución de los nodos a modo de grafo para que otro nodo no se ejecute hasta que se haya completado el objetivo de cualquier otro nodo.

   Estructura de un nodo de trama:

   Información básica.
   ¿Qué pasa?
   Tipo de conversación
   Conversación
   Prerequisito
   Otra conversación
   Diario

   5) Finalizar
   Se guardan todos los cambios realizados en la trama.

Siguiendo este asistente y configurando todos los datos en cada una de estas etapas se pueden crear misiones de manera general. Es mucho más sencillo que hacerlo editando cada uno de los personajes y creando scripts en algunas de las entradas de los diálogos.

De esta manera se crean diálogos de manera automática siguiendo un patrón, lo malo de hacerlo así es que los diálogos creados son muy simples. Para que las conversaciones sean más realistas, es decir, con muchas frases distintas y varias posibilidades de realizar la conversación hay que crear muchos más nodos sobre el árbol de diálogo creado.

La estructura de diálogo que nos da el asistente para tramas es la siguiente:

1) Saludo
2) Aceptación
3) Acción
4) Rechazo

Ejemplo:

1) Hola, aventurero, quizá puedas ayudarme. Un trasgo me ha robado mi alianza de oro. La recuperaría yo mismo, pero...

Si el jugador acepta, podría decir lo siguiente:

2) Ya, pero no te atreves. Y quieres que lo haga yo.

Si en cambio, lo rechaza:

3) Pues que tengas suerte. Yo tengo algo de prisa.

Si ha aceptado, el PNJ contestaría algo similar a:

4) ¡Sí, claro! Te pagaría cien monedas de oro por esa alianza. Fue un regalo de mi mujer y me matará si descubre que me la han robado.

Diseños de Trama Predefinidos

I) Asesinato sin prueba	***I) Asesinato***
1. Encuentro inicial con [desencadenante de trama]	1. Encuentro inicial con [desencadenante de trama]
2. Charla con [desencadenante de trama] antes de matar al [Villano]	2. Conflicto con el [Villano]
3. Conflicto con el [Villano]	3. Hablar con [desencadenante de trama] después de matar al [Villano]
4. Hablar con [desencadenante de trama] después de matar al [Villano]	4. Epílogo de [desencadenante de trama]
5. Epílogo de [desencadenante de trama]	
III) Buscar y enviar	***IV) Buscar y recuperar***
1. Hablar con [remitente] para obtener el objeto a enviar.	1. Encuentro inicial con [desencadenante de trama].
2. Hablar con [remitente] después de conseguir el objeto.	2. Encontrar el objeto.
3. Dale objeto al [receptor]	3. Llevar el objeto al [desencadenante de trama]
4. Hablar con [remitente] después de llevar el objeto	4. Charla con [desencadenante de trama] después de devolver el objeto
5. Hablar con [receptor] después de llevar el objeto.	

5.3.3. Problemas y aspectos a tener en cuenta

Debemos tener cuidado con la organización de los scripts que realizan la interacción entre personajes. Los diálogos de los personajes tienen la opción de establecer condiciones para ver en que momento el personaje se va por una rama del árbol de diálogo o por otra. Para decidir esto se usan variables locales a través de scripts.

Las variables locales pueden ser accedidas y modificadas por scripts que se activen dentro de otros personajes, así que, en realidad son variables globales. Por eso es importante ver que variables se usan como las vamos a nombrar, al igual que los scripts, para no confundirse cuando haya muchos scripts definidos. Por ejemplo, si creamos una variable para ver cuantas veces ha hablado el rey con el personaje, debemos poner en cada nodo del árbol de diálogo que queramos la condición de que la variable sea igual a un cierto valor, y en otros nodos del árbol modificar su valor para que el diálogo cambie.

Con muchos personajes y con muchas acciones se pueden confundir los scripts y realizar diálogos no deseados saltándonos conversaciones o no llegar a parte de una conversación cuando es necesario. Por ahora la elección para nombrar las variables se pone primero el nombre del personaje empezando con minúscula y se pone después alguna palabra o abreviatura que describa lo que hace la variable.

Teniendo en cuenta que todos los casos van a ir en un mismo módulo de Neverwinter conveniente que todos los identificadores de variables, etiquetas, scripts, etc., tengan un prefijo común, como puede se c1 seguido del nombre que sea para saber que ese elemento pertenece al caso 1.

Al usar el asistente para tramas hemos de tener cuidado, el asistente de tramas nos ayuda a configurar automáticamente varios diálogos de varios personajes con varias acciones en dichos diálogos en vez de tener que crear cada acción en un script. Debemos tener cuidado de que esté bien definida la trama y no volver que editar la trama, ya que esto puede dar problemas como crear varias entradas iguales en el diario del personaje, sobrescribir scripts y sobrescribir diálogos. Estas cosas pueden hacernos perder bastante tiempo, ya que si hemos hecho un diálogo a través de un editor, con dos o tres nodos de preguntas y respuestas y posteriormente lo hemos ampliado para que el diálogo sea más realista al volver a editar y salvar la trama nos cargamos todo lo que ya habíamos hecho y deberemos volver a crear los nodos y asociarles los scripts correspondientes en caso de que los tuvieran.

También hay que tener cuidado con las facciones, ya que en algún momento pueden suceder que lleven a que ciertos scripts no funcionen bien. Por ejemplo al crear nodos de enfrentamiento con el villano, si las facción se cambia a hostil el personaje debería atacarnos. Al realizar esto, hay momentos en que no ha funcionado y no se ha averiguado el motivo hasta ir a las facciones y volver a guardarlas, al refrescar todo después ha funcionado.

Una utilidad que nos puede facilitar el trabajo a la hora de hacer pruebas en NWN es cambiar el modo de la pantalla de modo de pantalla completa, el habitual en NWN a modo ventana. NWN puede ejecutarse en modo ventana modificando el fichero de configuración nwn.ini, que está en la raíz de la carpeta donde se ha realizado la instalación de NWN.

Dentro del apartado [Display Options] hay que poner FullScreen=0 AllowWindowedMode=1 y asegurarse de tener Width y Height a unas dimensiones razonables, en cualquier caso menores que como tengamos configurado el escritorio.

Para un escritorio de 1280x1024 se recomienda ejecutar NWN a 1024x768 o 800x600, para un escritorio de 1024x768 se recomienda ejecutar NWN a 800x600. Parece ser que el modo ventana no admite tamaños menores de 800x600.

La única diferencia de control es que para desplazar la cámara hay que mantener pulsada la rueda del ratón o usar directamente los cursores.

Pueden existir problemas con el controlador de pantalla al ejecutar varias veces el juego después de hacer cambios en el módulo.

6. IMPLEMENTACIÓN DEL ADAPTADOR PARA EL SISTEMA KIIDS

El adaptador de KIIDS [7] a Neverwinter Nights debe conectar el sistema de ontologías con nuestro módulo del juego. La comunicación se llevará a cabo mediante sockets. Para realizar esta tarea lo primero que se va a realizar es un conector para poder realizar la comunicación entre KIIDS y Neverwinter Nights. Aunque usaremos el conector para enviar y recibir mensajes a NWN, la idea es que podría usarse para realizar la comunicación entre KIIDS y cualquier otro juego o entorno de agentes que fuera similar.

Antes que nada, debemos definir un lenguaje y un protocolo de comunicación para delimitar claramente como se va a relacionar KIIDS con el entorno concreto y poder concretar que tipo de respuestas va a recibir el sistema. Para ello crearemos el lenguaje y el protocolo de comunicación RCEI [8] (Remote Controlled Environments Interface), que comentaremos más adelante.

En un principio, para poder realizar está comunicación hemos usado una librería llamada ShadowDoor [6] conjuntamente con otra aplicación para crear un servidor de Neverwinter Nights llamada Neverwinter Nights Extender [5] (NWNX). La librería DLL de Shadow Door sirve para que una aplicación acceda al motor de Neverwinter Nights para enviar y recibir paquetes, con un formato propio. Posteriormente hemos creado nuestra propia DLL, basándonos en Shadow Door, para no tener que depender de otra librería.

6.1. Protocolo RCEI

Para llevar a cabo esta interacción definiremos un lenguaje y un protocolo con el que comunicar Neverwinter Nights y KIIDS [7] en ambos sentidos.

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 entorno virtual será de tipo narrativo, con personajes y escenarios con cierto grado de realismo. Este protocolo y este lenguaje se utilizarán para conectar el motor de juego Neverwinter Nights con una aplicación Java que extiende el sistema KIIDS para dirigir automáticamente un proceso de narración digital e interactiva.

El protocolo RCEI [8] tiene las siguientes características:

(1) La comunicación será 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, etc.).
(2) Generalmente las respuestas del entorno son hechos (*facts*) que confirman las órdenes del sistema (*commands*).
(3) La aplicación de control remoto es quien da comienzo a las comunicaciones.
(4) El primer mensaje del entorno virtual es un mensaje de identificación, listando los dominios soportados por el módulo concreto que se está ejecutando y listando todas las observaciones referentes al estado inicial que debe conocer la aplicación.
(5) 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"). Por otra parte, 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 enviará comandos y recibirá hechos a través de un interface de sockets. El protocolo se implementará con sockets en ASCII extendido de 8 bits, con un máximo de 1024 bytes de longitud.

En la aplicación RCEI usaremos una gramática diferente a la que viene por defecto con Shadow Door, algo más compleja, para añadir más comandos estructurados en un lenguaje claro y sencillo.

6.2. Lenguaje RCEI

El código de la librería Shadow Door [6] está implementado en Visual C++ 7.0, y utiliza algunas clases de NWN Extender y otras propias de ShadowDoor. Para modificar su protocolo debemos modificar en la clase principal (Shadow Door.cpp) la función **void ShadowDoor::RunListener ()**.

En está función podríamos alterar lo que sería el analizador léxico, por ejemplo podríamos añadir nuevos tokens para reconocer en la gramática, añadir otros separadores, etc.
En nuestro lenguaje usaremos el siguiente repertorio de instrucciones con los parámetros correspondientes. En algunos casos las funciones de scripting de Neverwinter Nights permitirán realizar más opciones que las especificadas en nuestro lenguaje, pero la idea es hacer un lenguaje general que permita realizar todas aquellas relaciones que estarán en nuestra base de conocimiento.

Proceso	Parámetros
SIMULACIÓN	
speak	Para que un agente hable con otro. 1) subject – agente que realiza la acción. 2) dirComp – mensaje que envía el agente.
move	Activa pequeña animación de un personaje, como un saludo, una reverencia, una sonrisa, etc. 1) subject – agente u objeto que realiza la acción. 2) dirComp – tipo de animación realizada. Este argumento puede tomar alguno de los siguientes valores: none :: ninguna drink :: beber reed :: leer greeting :: saludo listen :: escuchar talk :: hablar implore :: suplicar furious :: hablar furioso laugh :: hablar riendo victory:: victoria adore :: adorar harass :: hostigar reverence :: reverencia steal :: robar
attack	Para que un agente ataque a otro o algún objeto. 1) subject – agente que realiza la acción. 2) dirComp -- agente u objeto sobre el que se realiza la acción.
go	Movimiento desde un sitio a otro. 1) subject – agente u objeto que realiza la acción. 2) dirComp (opcional) -- agente u objeto hasta el que se mueve el agente. Si no se usa este parámetro el movimiento es aleatorio.

take	Para coger un objeto del escenario. 1) subject – agente que realiza la acción. 2) dirComp -- Objeto sobre el que se realiza la acción.
drop	Para soltar un objeto y depositarlo en el suelo. 1) subject – agente que realiza la acción. 2) dirComp -- Objeto sobre el que se realiza la acción.
give	Un agente le da un objeto a otro. 1) subject – agente que realiza la acción. 2) dirComp -- Objeto dado a otro agente. 3) indComp -- agente al que se le da el objeto.
equip	Un personaje se equipa con un objeto de su propio inventario. 1) subject – agente que realiza la acción. 2) dirComp – Objeto del inventario del agente.
unequip	Un personaje se quita un objeto con el que está equipado y lo guarda en su propio inventario. 1) subject – agente que realiza la acción. 2) dirComp – Objeto del agente.
create	Para crear un agente o un objeto en la posición determinada de un escenario. En NWN el objeto o personaje debe estar predefinido por el usuario para indicarle la etiqueta concreta cuando invocamos a create. 1) subject – Sólo system realiza la acción. 2) dirComp -- Objeto o agente creado. 3) placeComp **–** escenario concreto donde se crea el objeto o NPC y las coordenadas exactas X,Y,Z donde se crea.
destroy	Para destruir un objeto de un escenario. 1) subject – NPC que realiza la acción. 2) dirComp -- Objeto sobre el que se realiza la acción.
locatedAt	Para teletransportar un agente o un objeto a una posición determinada de un escenario. 1) subject – Sólo system realiza la acción. 2) dirComp – Objeto o agente creado. 3) placeComp **–** escenario concreto donde se crea el objeto o agente y las coordenadas exactas X,Y,Z donde se traslada.
change	Para cambiar propiedades del entorno: el clima, la luz, el cielo,la cámara, etc; o de un personaje apariencia física, propiedades, etc. 1) subject – Sólo system realiza la acción. 2) dirComp – Tipo de cambio realizado en el sistema: weather, sky, fog, light, camera; o en un agente: apariencia, propiedades, etc. Si es un agente, objeto o link, este campo contendrá su etiqueta (tag). 3) placeComp (opcional) – lugar donde se realiza el cambio: puede ser en todo el módulo o en un escenario concreto. 4) modeComp – modo de ejecutarse el cambio. Determina el tipo de clima (nieve, lluvia, nada) o de niebla (espesa, clara) o de luz (día, noche). En caso de que sea un agente, objeto o link, determina su estado. Por ejemplo, abierto o cerrado para una puerta o para cambiar un diálogo, etc. open :: abrir objeto abrible close :: cerrar objeto abrible state :: estado para los diálogos de los agentes
INTERACCIÓN	

	relativos a goals de los players = quest, a información del player como nombre y perfil, ya sea buttkicker o lo que sea, la relación que tiene el player con el player character (agent)
NARRACIÓN	
GENERALES DE RCEI	
begin	Para realizar la carga de un módulo del juego. 1) subject – Sólo system realiza la acción. 2) concepts (opcional)– conceptos del entorno. 3) relations (opcional)– relaciones existentes en el entorno.
end	Para terminar el juego. Puede visualizarse una animación antes de terminar. 1) subject – Sólo system realiza la acción. 2) dirComp (opcional)– el nombre de la animación dentro del directorio movies.

Estas son los comandos y hechos que posee el lenguaje RCEI. La gramática detallada y el protocolo están contenidos en un documento donde se explican estos aspectos en mayor profundidad. Podemos ver la gramática de RCEI en su versión reducida en el apéndice D de la memoria.

6.3. Implementación de RCEI

La aplicación desarrollada se llama jRCEI, está implementada en JAVA y ha sido desarrollada con ECLIPSE. Posee unas pocas clases donde se implementa la comunicación mediante sockets y también está dotado de un sencillo interface donde se pueden crear comandos y enviarlos a Neverwinter Nights para que se ejecuten.

Internamente jRCEI esta estructurada en varios paquetes, principalmente para separar el parser y el serializer de RCEI, el wizard y algunos otros componentes como un API para su uso desde otros proyectos. El parser está implementado con la herramienta JAVACC [17]. La interfaz de usuario, tiene varios botones para realizar la conexión y preparar los mensajes que se van a enviar. 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 formato XML.

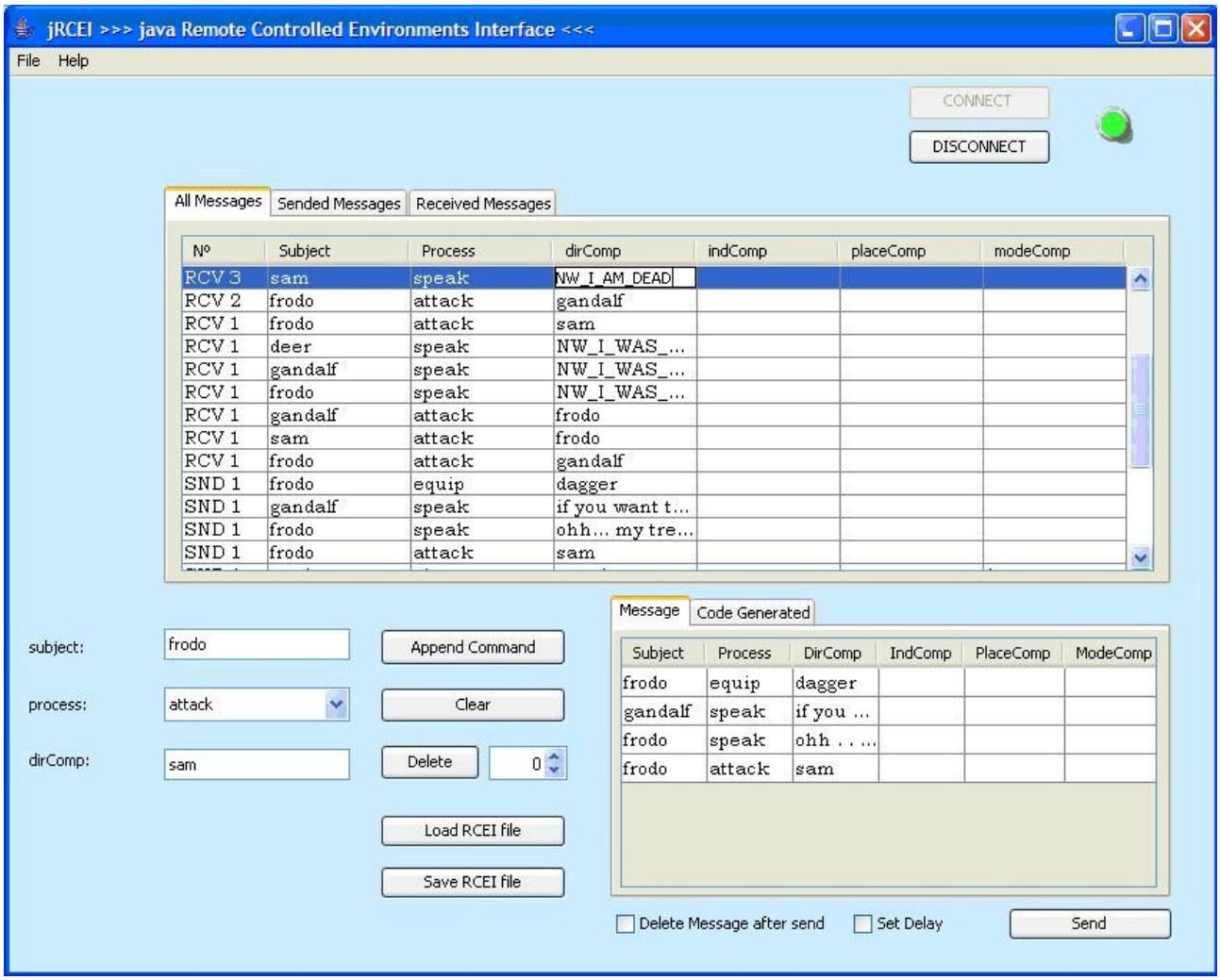


Fig. 15. Interface de la aplicación jRCEI. Podemos crear comandos, enviarlos a NWN y ver la respuesta que nos devuelve.

7. RESULTADOS DE PRUEBAS

Una vez que tengamos implementado módulos jugables sobre los que hagamos pruebas para corregir bugs comprobar jugabilidad o finalmente para estudiar el resultado del proyecto. En primer lugar incluiremos pruebas sencillas sobre los primeros experimentos.

Resultados del primer experimento preliminar (sobre la versión v0_0_04 del módulo):

Usando este sujeto: mujer, 27 años, conocimiento básico del PC pero sin experiencia en videojuegos.

- Tardó 25 minutos en conseguir las llaves que te da el rey. A los 32 minutos se dio por vencida, aunque /con una pequeña ayuda/ pudo continuar hasta los 43 minutos en los que llegó a encontrarse con el mago maligno.

(Comentario curioso: le daba la sensación de que el personaje estaba continuamente corriendo).

Resultados de experimentos preliminares (sobre la versión 0_1_02del módulo):

En esta versión se han ampliado las tramas, entradas en el diario, se han acotado escenarios, se han añadido más personajes y objetos.

Usando este sujeto: hombre, 27 años, conocimiento alto a nivel usuario del PC y con mucha sin experiencia en videojuegos y ha jugado varias campañas del Neverwinter Nights.

- Tardó 20 minutos en conseguir las llaves que te da el rey y salir de la ciudad. Al tener experiencia, se entretuvo en coger todos los objetos que pudo, vender y comprar a un comerciante y armarse debidamente. Uno de los encuentros en un lateral del castillo le mató la primera vez.
- Tardó 10 minutos en llegar hasta el brujo Vuurk y otros 10 minutos en llegar al brujo malo y matarlo. En total unos 40-45 minutos.
- Su prueba ayudo a corregir varios bugs y aportar algunos conocimientos sobre el juego. Por ejemplo entre los fallos se han detectado están ver que una caja no se abre, en un armero no se puede coger un anillo mágico. La trama de la mujer del consejero le confundió un poco que se hablara de un anillo que sólo formaba parte de la conversación (en la prueba de la otra versión si había un anillo)

Usando este sujeto: mujer, 50 años, conocimiento básico del PC pero sin absolutamente ninguna experiencia en ningún videojuego.

- Jugo durante 35 minutos hasta que no pudo avanzar más en el juego "por voluntad propia". Resolvió las primeras tramas, pero en la trama de matar al amante decidió que no quería matar a nadir, con lo cual no podía continuar. Le gustaba mucho el ambiente en 3D y le llamaba la atención que en la foto de la derecha el personaje fuera calvo. En cuanto aprendió a usar alguna de las acciones del inventario y de opciones del jugador solo quería vestir al personaje con la capa y comprarle accesorios. También quería ir a descansar encima de la alfombra de oso.

(Comentarios curiosos: no quería matar a nadie cuando aparecían encuentros con enemigos. No le gustaba el aspecto de algunos personajes, el mendigo ´le parecía gay, la princesa le parecía un marimacho, el rey le parecía un pirata).

Resultados de experimentos sobre la aplicación jRCEI

Una de las partes más satisfactorias del proyecto ha sido la del protocolo y el lenguaje de comunicaciones RCEI y su posterior implementación en JAVA (jRCEI), del cual finalmente hemos escrito un artículo detallando el proyecto y remarcando sus posibles utilidades en el campo de representación de entornos virtuales en sistemas inteligentes. Vamos a mostrar una secuencia de imágenes donde se ve el wizard de la aplicación en funcionamiento de manera que enviamos mensajes al entorno, éste los ejecuta y devuelve una respuesta.

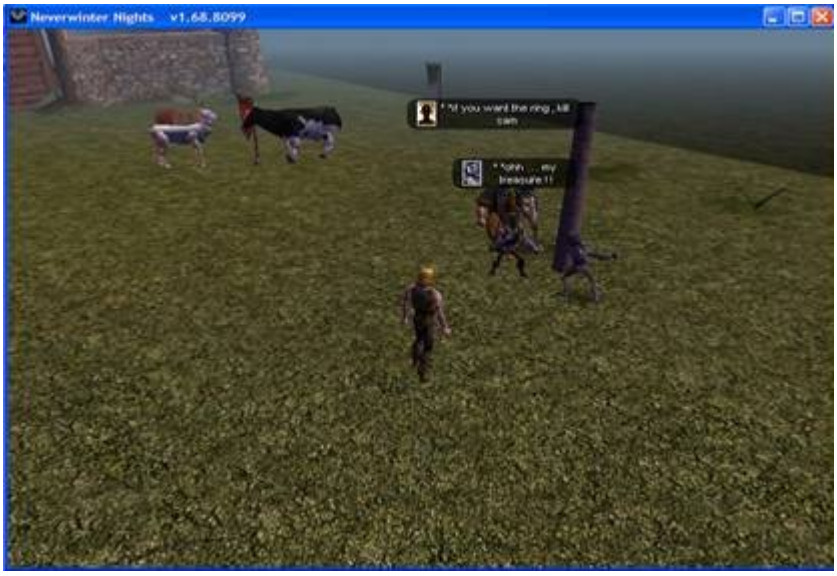


Fig. 16. Ejemplo de entorno basado en Neverwinter Nights. Localización con algunos agentes y objetos.

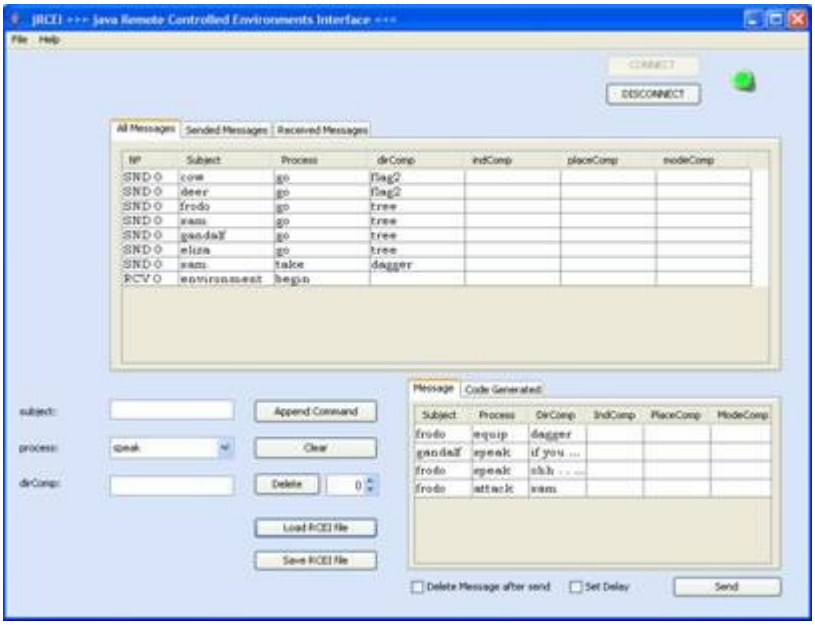


Fig. 17. RCEI Wizard conectado al entorno y enviando mensajes con varios comandos.

Fig. 18. Los comandos enviados desde RCEI Wizard son ejecutados en paralelo dentro del entorno.

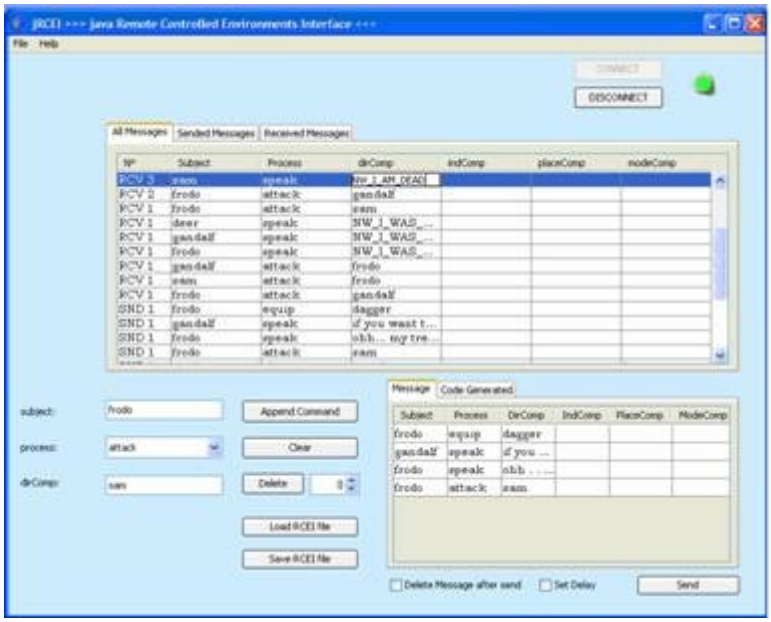


Fig. 19. Los hechos que se producen en el entorno son recibidos y mostrados al usuario en RCEI Wizard.

8. CONCLUSIONES Y TRABAJO FUTURO

Como trabajo futuro debemos terminar los trabajos de integración con KIIDS y realizar experimentos con Encrucijada en la Tierra Media para evaluar los resultados. A pesar de no haber dado tiempo a terminar toda la integración con el sistema KIIDS, el resultado del proyecto ha sido muy satisfactorio. El principal logro del proyecto ha sido poder conectar el entorno Neverwinter Nights con una aplicación externa y lograr, por medio de esta aplicación, el control remoto de los agentes del entorno y sus componentes, algo que cabe recordar no existía hasta el momento. Lo más aproximado a nuestro proyecto era Shadow Door, pero su mayor problema es que tiene muchas limitaciones, ya que sólo permite el control remoto de un agente y su repertorio sólo tiene 4 commandos y no tiene una organización de lenguaje. Con lo cual jRCEI ha conseguido cumplir nuestras expectativas iniciales.

Para la aplicación de control remoto jRCEI, deberían de hacerse algunos cambios y algunas ampliaciones. En primer lugar falta completar la implementación del comando begin y ampliar la funcionalidad del hecho begin. Hay otros comandos que no se han terminado de implementar completamente como son locatedAt, create. En estos comandos hay que precisar que parámetros son necesarios para crear un agente y trasladarlo a una ubicación. Además hay otras instrucciones que podrían mejorar su implementación, como es el caso de change para los diálogos. Hasta ahora los diálogos que son cambiados por RCEI y no están cableados, se manejan por una máquina de estados de cada personaje. Por el momento estos estados simplemente están numerados, aunque se podrían mejorar particularizándolos usando un vocabulario específico. Por ejemplo estados como "happy", "sad", "wantingToKillMyWife", etc.

En lo que se refiere al código JAVA de RCEI, se podría mejorar el código para hacerlo más completo, más extensible y más modular utilizando patrones software para implementar algunas de sus partes. Por ejemplo, se podría utilizar para el wizard el patron Abstract Factory y tal vez Singleton o Adapter; también podríamos utilizar el patron Command para las instrucciones de la gramática o tal vez el patrón Composite.

Además para poder incluir en el futuro varios conectores y varias extensiones para las particularidades en el lenguaje de cada entorno, debemos hacer extensible el código de jRCEI a extensiones de vocabulario de RCEI. Una manera de hacerlo sería tener un interfaz global con métodos para todas las instrucciones básicas de la gramática de RCEI, que se pueda extender facilmente creando un nuevo interfaz para cada el nuevo vocabulario del entorno, añadiendo nuevos métodos más concretos. Para ello, este interfaz global debe contar con parámetros que incluyan los elementos de la base de conocimiento, es decir, las etiquetas de los componentes del módulo de juego.

Por otro lado, en lo que se refiere a los mensajes que se envían al entorno, que están compuestos por varios comandos atómicos, podríamos plantearnos la posibilidad, en caso de ser útil, de poder componer varios comandos. De esta manera, se podría fusionar en un mismo comando 2 acciones que deban ser ejecutadas en un mismo instante. Aunque habría que estudiar todos los pormenores, ya que en el caso de que se pueda obtener un mismo resultado con mensajes compuestos por varios comandos, no sería realmente necesario.

En otro orden de cosas, podríamos enriquecer algunas instrucciones básicas de Neverwinter Nights con animaciones que resulten apropiadas. Por ejemplo, cuando un personaje de un objeto a otro, sería más vistoso que ejecutara una animación de movimiento para que el usuario sepa que la acción de dar un objeto se ha realizado.

Al haber logrado la compatibilidad con NWNX y ShadowDoor, además de haber creado nuestra propia DLL y la tecnología necesaria para conectar RCEI con KIIDS, nos planteamos para un futuro próximo la implementación de un adaptador para el nuevo motor de juego de Neverwinter Nights 2. Las ventajas no sólo serían estéticas sino que además el juego incorpora un conjunto de herramientas mejoradas que permiten mayor versatilidad en los contenidos de las historias.

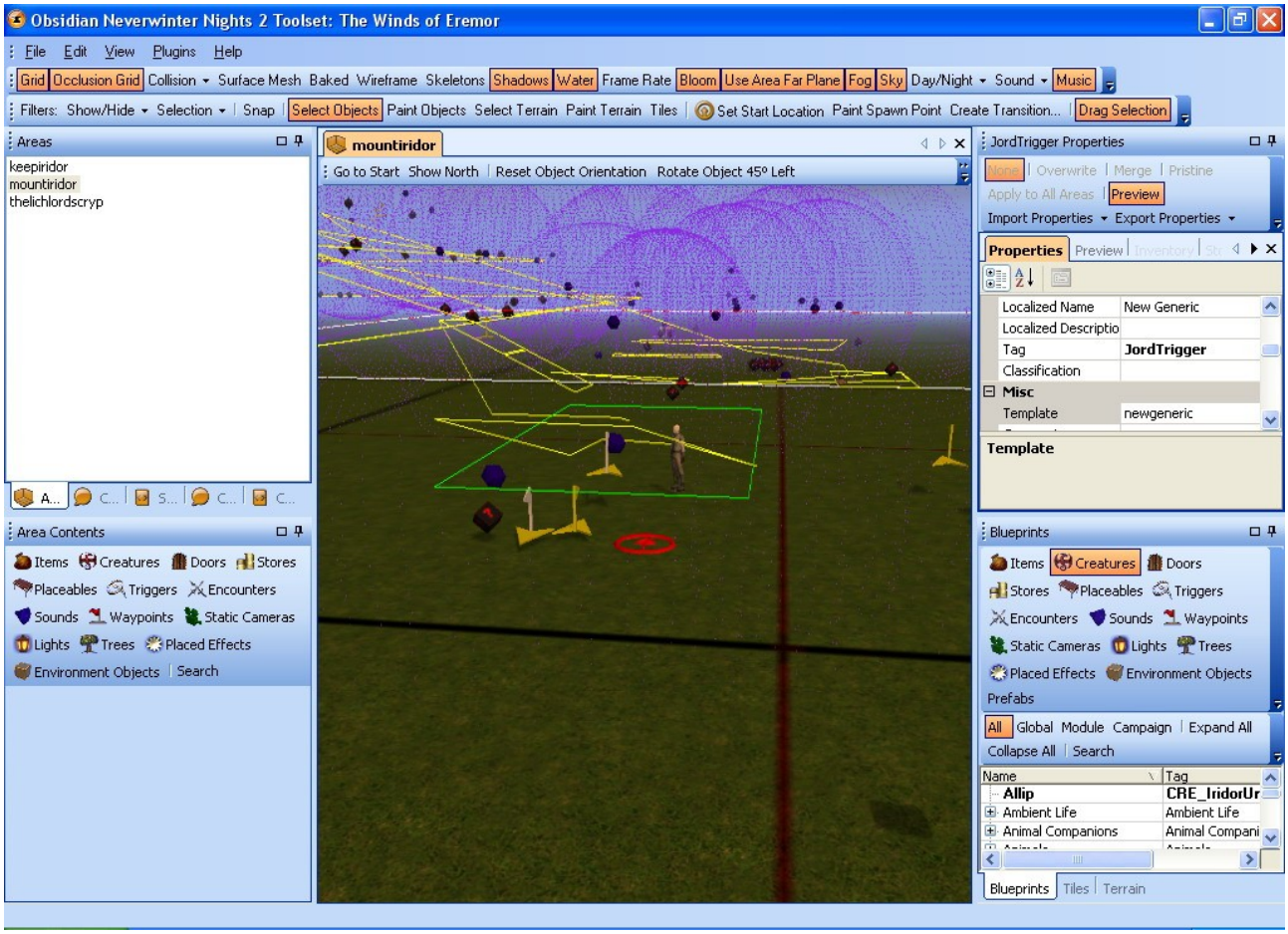


Fig. 20. Captura de Neverwinter Nights 2 Toolset.

Además de este entorno, que es una versión mejorada del que ya conocemos, y con el que ya hemos trabajado, sería muy interesante implementar adaptadores de otros entornos que no sean sólo de la temática de rol sino de otro tipo, ya sean shooters, juegos de aventura, juegos de estrategia, o cualquier otro. Nos podríamos plantear la implementación de adaptadores para los siguientes entornos: Knights of the Old Republic (que usa el mismo motor que NeverWinter Nights), Unreal Tournment 2007, Wargus (Warcraft 2 modificado), TIELT (Call to Power 2 u otros), Multi Theaft Auto (MOD del Grand Theft Auto 3).

Habiendo desarrollado la herramienta jRCEI y explorado las posibilidades del Neverwinter Nights como entorno de representación de diferentes problemas de Inteligencia Artificial, cabe destacar que se podría utilizar la documentación elaborada en esta memoria para otros proyectos que hagan uso de NeverWinter Nights o jRCEI. En la red se pueden encontrar algunos módulos elaborados por algunos usuarios de Neverwinter Nights para jugar al ajedrez [29], al BlackJack y algunos otros juegos. Este tipo de módulos están hechos para ser jugados de forma manual y no tienen implementada ningún tipo de Inteligencia Artificial. Esto podría programarse externamente con algoritmos o con un sistema inteligente y conectarse a través de jRCEI.

9. BIBLIOGRAFÍA

Software

- [1] Neverwinter Nights Diamond Edition (DVD que incluye SoU + HotU y es actualizable)
  http://nwn.bioware.com

- [2] Parche para la actualización de Neverwinter Nights 1.66 hot update
  Inglés: http://nwn.bioware.com/support/patch_english_hotu.html
  Español: http://nwn.bioware.com/support/patch_spanish_hotu.html

- [3] NWN Community Expansion Pack (CEP 2.0 más otros recursos disponibles en NWVault, en la obra de Mondego o en la comunidad del Reino de Aldor)
  http://nwn.bioware.com/players/cep.html
  http://nwvault.ign.com

- [4] Traducción de diálogos para NWN Diamond Edition
  http://nwvault.ign.com/View.php?view=Other.Detail&id=1153
  http://nwvault.ign.com/View.php?view=Other.Detail&id=1208

- [5] Neverwinter Nights Extender 2.6.1
  http://www.nwnx.org

- [6] ShadowDoor 1.0
  http://www.zubek.net/robert/software/shadow-door/

- [7] Peinado, F.: Knowledge-Intensive Interactive Digital Storytelling system (2007)
  http://federicopeinado.com/projects/kiids
  https://sourceforge.net/projects/kiids/

- [8] Peinado, F. and Navarro, A.: RCEI, A Remote-Controlled Enviroments Interface (2007)
  http://federicopeinado.com/projects/rcei
  http://code.google.com/p/rcei/

- [9] Protégé-OWL 3.2beta
  http://protege.stanford.edu

- [10] SWOOP 2.2.2
  http://www.mindswap.org/2004/SWOOP/
  http://code.google.com/p/swoop/

- [11] Pellet 1.3
  http://www.mindswap.org/2003/pellet/

- [12] WordGenerator
  http://www.kessels.com/WordGenerator/

- [13] NwnMultiLang
  http://nwvault.ign.com/View.php?view=Other.Detail&id=393#Files

- [14] RAD Video Tools (Herramienta de transformación de formatos de vídeo)
  www.radgametools.com

- [15] Camtasia Studio (Herramienta de edición de vídeo)
  http://www.techsmith.com

- [16] Neverwinter Nights MDL Compiler
  http://nwvault.ign.com/View.php?view=Other.Detail&id=1212

- [17] Java Compiler Compiler (JavaCC 4.0)
  https://javacc.dev.java.net/

- [18] Editor de textos (preferiblemente LaTeX, si no OpenOffice Writer o Microsoft Word)
  www.miktex.org
  www.latexeditor.org
  www.openoffice.org

Manuales, tutoriales e información del proyecto

- [19] The Seven Basic Plots: Why we tell stories. Booker, C. Contimuum (2004)

- [20] Peinado, F., Gervás, P.: "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 (Proceedings of the 2nd International Conference on Technologies for Interactive Digital Storytelling and Entertainment, TIDSE'04). 24-26 June, Darmstadt, Germany.
  http://www.fdi.ucm.es/profesor/fpeinado/publications/2004-peinado-transferring.pdf

- [21] Enciclopedia en CD-ROM Tolkien – David Day. Timun Mas (1991).

- [22] Manual de uso del conjunto de Herramientas de Neverwinter Nights incluido en el DVD original (ToolsetManualV103.pdf).

- [23] BioWare Aurora Neverwinter Nights Toolset Module Construction Tutorial
  http://nwn.bioware.com/builders/modtutorial.html

- [24] Manual Aurora en español
  http://www.terra.es/personal5/oupmfebh/ocio/nwn/nwn_00_indice.htm

- [25] Toolset Tutorials
  http://nwvault.ign.com/modules/tutorials/viewlets/

- [26] NWN ScriptEASE
  http://www.cs.ualberta.ca/~script/

- [27] Editor de tramas
  http://nwn.bioware.com/builders/plotwizard_tutorial6.html

- [28] Scrpting Tutorials
  http://nwn.bioware.com/builders/sctutorial.html

- [29] Adding Neverwinter Chess to a Module (Módulo para jugar al ajedrez desde NWN)
  http://nwn.bioware.com/builders/chess.html

- [30] Creating a starting Movie for a Module
  http://nwn.bioware.com/builders/movie_tutorial.html

Apéndice A. Manual de Instalación de NeverWinter Nights

Instalación:

1. Instalar Neverwinter Nights [1] + 2 expansiones: Shadows of the Undertide and Hordes of the Underdark.

2) Otra opción es instalar Neverwinter Nights Diamond Edition DVD sin todas las expansiones Kingmaker.

Los serials del juego vienen con el CD original. No es necesario instalar GameSpy ni registrarse en Atari.

Actualización:

Después de instalar el juego debemos instalar los últimos parches del juego [2] que corrijan pequeños bugs del juego o que tengan las últimas actualizaciones de algunas partes del juego. Para instalar el último parche del juego, si no lo tenemos debemos ir a la página web de Neverwinter Nights [2] y bajárnoslo (ocupa unos 70 MB). Además el menú del juego dispone de un botón para actualizar automáticamente la ultima versión del parche en caso de no estar instalado.

Lenguaje:

La selección del lenguaje depende de la versión del parche que instalemos [2], si instalamos la versión en español, todos los diálogos del juego estarán en español, sino estarán en inglés.

Además de esto existe un archivo en la raíz de la instalación del juego llamada dialog.tlk; en este archivo están todos los textos en inglés o en castellano. Para realizar la traducción a castellano es preciso descargarse este archivo de unos 12 MB disponible en la red [4] y reemplazarlo. También existe está misma actualización incluyendo además los subtítulos en castellano para todos los videos de la introducción de Neverwinter Nights de unos 200 MB. Después de esto, al ejecutar Neverwinter Nights, se configurará para una ejecución óptima.

Apéndice B. Manual de Instalación de NWN Extender y ShadowDoor

Instalación NWN Extender y Shadow Door:

1. Descargar la última versión de Neverwinter Nights eXtender [5].

2. Descargar la última versión de Shadow Door [6].

3. Descomprimir NWNX (nwnx_261.zip) y mover los archivos "NWNX2.exe", "nwnx-module.dll", y "madCHook.dll" a la raíz del directorio donde hayamos instalado Neverwinter Nights (usualmente *C:\Neverwinternights\NWN*).

4. Después de copiar los módulos de NWNX queremos usar Shadow Door, para lo cual tenemos que copiar el archivo nwnx_sd.dll a la raíz del directorio de Neverwinter Nights y el archivo shadowdoor.erf en la carpeta erf (si no está creada debemos crearla). Además debemos copiar el módulo de prueba Shadow Door Example Module con extensión .mod en la carpeta modules.

Ejecución NWN Extender:

1. Ejecutar NWNX2.exe, la ventana del servidor NWServer aparecerá automáticamente a los pocos segundos, después ya se puede cerrar NWNX2.

2. En la ventana de NWServer, debajo de la etiqueta llamada Module Name seleccionar el módulo Shadow Door Example Module.

3. Pulsar el botón Load, si el módulo se carga correctamente debemos ver en la barra de estado de la parte inferior el siguiente mensaje: "Running, login at will".

Instalar y ejecutar el sistema inteligente incluido en módulo Shadow Door:

El sistema inteligente que usa el ejemplo de Shadow Door es una pequeña aplicación en Lisp llamada Eliza para realizar un pequeño diálogo entre el jugador y la máquina.

1. Instalar Allegro Common Lisp 7.0 (ACL) y no actualizar ACL, es importante para que funcione.

2. Ejecutar la interfaz de Lisp (ACL). Pulsar en menu Select File, después elegir la opción Open Project.

3. Seleccionar el proyecto Lisp llamado eliza-demo-allegro.lpr, que esta dentro de la carpeta de Shadow Door que habíamos descomprimido inicialmente (en la carpeta Shadow Door\LISP Sources).

4. Debemos ver en la ventana de depuración de Lisp una consola, con el prompt CG-USER(1):. Debemos escribir los siguientes dos comandos en el prompt y pulsar enter como vemos a continuación:

   cg-user(1): (in-package common-lisp-user)
   cl-user(2): (shadow-door-eliza-loop)

   El primer comando es para cargar el paquete de Lisp y el segundo es para ejecutar el programa de Eliza incluido a modo de ejemplo. El programa Eliza se quedará ejecutando en un bucle y lo que responda Eliza sucederá en el juego.

Ejecutar el módulo Shadow Example Module:

Para ejecutar el módulo debemos hacerlo en Multiplayer como si jugáramos en red. Para ello es preciso registrar el juego Neverwinter Nights en la página web de Bioware [1] para tener un lógin y un password o tener el PC sin conexión a Internet para que no intente validar el lógin y password del juego.

1. Ejecutar el juego Neverwinter Nights normalmente. En el menu principal del juego debemos elegir la opción Multijugador. Realizar la autentificación con nuestro login y el password (si no estamos conectados podemos poner cualquiera).

2. Después debemos seleccionar la opción *Unirse a partida en LAN*. En la pantalla debe aparecer Shadow Door game, que habíamos cargado previamente desde NWServer.

3. Pulsamos el botón *Conectar* o al botón *Conectar Directamente* para empezar la partida.

4. Para probar la comunicación, podemos enviar mensajes, que se enviarán a través de sockets, desde la consola de LISP. En la documentación de Shadow Door vienen detallados varios ejemplos.

Apéndice C. Manual de Instalación de jRCEI

La distribución de jRCEI está compuesto por una serie de carpetas donde se encuentran cada una de las partes del proyecto. Estas carpetas son:

- ***adapters*** : en esta carpeta están los adaptadores a los entornos concretos de juego. En la versión 0.0.5 de jRCEI existe sólamente el adaptador para Neverwinter nights, que consiste en una DLL similar a la de Shadow Door. Su código en Visual Studio 2005 está también está disponible aquí.

- ***bin*** : los archivos .class del proyecto compilados del código JAVA.

- ***doc***: la documentación del código JAVA del proyecto en html. En esta documentación aparecen todas las clases con sus funciones comentadas en inglés.

- ***files*** : en esta carpeta están archivos XML con ejemplos del módulo de prueba para ser cargados desde el Wizard y comprobar el funcionamiento de la aplicación.

- ***images*** : en esta carpeta están las imágenes que necesitará el wizard para la visualización.

- ***rce*** : en esta carpeta están los archivos de configuración que cargará la aplicación para configurar el tipo de entorno y las propiedades del conector a dicho entorno.

- ***src*** : en esta carpeta el código fuente de todo el proyecto distribuido en distintos paquetes: uno para el parser, otro para el serializer, otro para el wizard y además la implementación de todas las clases necesarias para conectar con los entornos virtuales, y un API para poder ser usado como parte de otros proyectos.

- ***test*** : en esta carpeta hay un módulo de Neverwinter Nights de ejemplo para probarlo con el wizard.

Instalación de jRCEI:

1. Descargar la última versión de jRCEI [8].

2. Descomprimir el archivo zip en la ubicación que queramos(por ejemplo en el directorio raíz de la instalación de Neverwinter Nights).

3. Debemos copiar el adaptador nwnx_rcei.dll desde la carpeta *adapters\nwn1rceiadapter* hasta el directorio raíz donde tengamos instalado Neverwinter Nights (usualmente *C:\Neverwinternights\NWN*). También debemos copiar el archivo rcei.erf desde la carpeta *adapters\nwn1rceiadapter\erf* hasta la carpeta *erf* del directorio de Neverwinter Nights para poder construir nuestros

propios módulos usando jRCEI.

4. Copiar el módulo RCEI_Test.mod desde la carpeta *test* hasta la carpeta *modules* del directorio de Neverwinter Nights.

Ejecución NWN Extender:

1. Ejecutar NWNX2.exe, la ventana del servidor NWServer aparecerá automáticamente a los pocos segundos, después ya se puede cerrar NWNX2.

2. En la ventana de NWServer, debajo de la etiqueta llamada Module Name seleccionar el módulo *RCEI_Test.mod*.

3. Pulsar el botón Load, si el módulo se carga correctamente debemos ver en la barra de estado de la parte inferior el siguiente mensaje: "Running, login at will".

Ejecución de jRCEI usando el módulo de prueba RCEI_Test.mod:

Para ejecutar el módulo debemos hacerlo en Multiplayer como si jugáramos en red. Para ello es preciso registrar el juego Neverwinter Nights en la página web de Bioware [1] para tener un lógin y un password o tener el PC sin conexión a Internet para que no intente validar el lógin y password del juego.

1. Ejecutar el archivo jRCEIWizard.bat que está en la raíz de la carpeta jRCEI. Para ejecutarlo es preciso que el sistema operativo tenga instalada la máquina virtual de JAVA (JVM). Además en la variable de entorno PATH del sistema debe estar la ruta del archivo javac.exe. En este momento nos aparecerá una ventana con el wizard de jRCEI.

2. En primer lugar debemos realizar la conexión. Para ello debemos asegurarnos que tengamos abierto en NWNServer de los pasos previos y ya hayamos cargado el módulo. Pulsamos el boton *CONNECT* y la luz que indica la conexión debería pasar de rojo a verde.

3. Ejecutar el juego Neverwinter Nights normalmente. En el menu principal del juego debemos elegir la opción Multijugador. Realizar la autentificación con nuestro login y el password (si no estamos conectados podemos poner cualquiera).***

4. Después debemos seleccionar la opción *Unirse a partida en LAN*. En la pantalla debe aparecer *RCEI_Test*, que habíamos cargado previamente desde NWServer.

5. Pulsamos el botón *Conectar* o al botón *Conectar Directamente* para empezar la partida.

6. Para probar la comunicación, podemos enviar mensajes desde jRCEI, que se

enviarán a través de sockets, desde el wizard. Los mensajes están formados por varios comandos. Los comandos que enviamos se componen de un proceso que podremos elegir entre los distintos existentes en la gramática, un sujeto y diferentes complementos dependiendo del proceso. Tanto el sujeto como los complementos deben corresponderse con los nombres de los tags o etiquetas de los elementos del módulo que se esté tratando.

7. Cada mensaje está formado por varios comandos, y los mensajes pueden ser salvados en ficheros XML y cargados posteriormente. Para probar comandos sobre el módulo de prueba *RCEI_Test* podemos crear manualmente un mensaje y enviarlo o cargar algunos mensajes guardados en la carpeta *files\nwn1* de la carpeta de jRCEI. Para cargar cada uno de ellos debemos de pulsar en botón *Load RCEI file* y seleccionar uno cada vez para ir probándolos todos. Cada vez que carguemos un fichero RCEI, se mostrará su contenido en la tabla de la parte inferior.

8. Para enviar cada mensaje, ya lo hayamos creado manualmente o bien lo hayamos cargado, pulsamos en botón *Send*. Podremos ver que en la tabla superior el nuevo mensaje aparece como enviado. Además las respuestas del entorno aparecerán el la tabla superior como mensajes recibidos.

*** Para hacer pruebas en Neverwinter Nights del envío de mensajes desde jRCEI para ver como son recibidos y ejecutados por el entorno en una misma máquina (localhost) es bastante útil cambiar el modo de la pantalla. Es más fácil manejarse en modo ventana en vez de pantalla completa. Para cambiarlo debemos modificar el fichero de configuración nwn.ini, que está en la raíz de la carpeta del Neverwinter Nights. Dentro del apartado [Display Options] hay que poner FullScreen=0 AllowWindowedMode=1.

Apéndice D. Gramática reducida de RCEI

```
Message::= <RCEI>
           <messageNumber> numInt </messageNumber>
           LInstruction
           </RCEI>

LInstruction::= LCommand
LInstruction::= LFact

LCommand::= Command LCommand | Command

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

LFact::= Fact LFact | Fact

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

Subject::= environment | LAgent

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

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

Predicate::= <process> PMove </process>
             <dirComp> Movement </dirComp>

Predicate::= <process> PGive </process>
             <dirComp> Object </dirComp>
             <indComp> Agent </indComp>

Predicate::= <process> PGo </process>

Predicate::= <process> PTrans </process>
             <dirComp> Object </dirComp>

Predicate::= <process> PBuild </process>
             <dirComp> Agent </dirComp>
             <modeComp> Waypoint </modeComp>
             <placeComp> Location </placeComp>

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>

Predicate::= <process> PBegin </process>
             <concepts> Concepts </concepts>
             <relations> Relations </relations>

Concepts::= Concept Concepts | λ

Relations::= Relation Relations | λ

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>

Relation::= <is> Domain Range </is>
Relation::= <has> Domain Range </has>
Relation::= <ally> Domain Range </ally>
Relation::= <enemy> Domain Range </enemy>

Domain::=  LLocation
Domain::=  LLocation2
Domain::=  LAgent
Domain::=  LAgent2
Domain::=  LObject
Domain::=  LObject2
Domain::=  LLink
Domain::=  LLink2

Range::=  LLocation
Range::=  LLocation2
Range::=  LAgent
Range::=  LAgent2
Range::=  LObject
Range::=  LObject2
Range::=  LLink
Range::=  LLink2

LLocation::= <location> Location </location>
             LLocation | λ
LLocation2::= Location LLocation | λ

LAgent::= <agent> Agent </agent> LAgent | λ
LAgent2::= Agent LAgent | λ

LObject::= <object> Object </object> LObject | λ
LObject2::= Object LObject | λ

LLink::= <link> Link </link> LLink  | λ
LLink2::= Link LLink | λ

Movement::= none | drink | reed | greeting |
listen | talk | implore | furious | laugh |
victory| adore | harass | reverence | steal

PSpeak::= speak
PMove::= move
PGo::= go
PGive::= give
PBuild::= create | locatedAt
Waypoint::= num num num
PChange::= change
Atrans::= weather | sky | fog | light | camera
Mode::= id
PBegin::= begin
PTrans::= PGo | equip | unequip | take | drop |
attack | destroy | use

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