Herramienta de definición de comportamientos de personajes no jugadores en videojuegos educativos

Grado en Ingeniería en Informática

Especialidad en Ingeniería de Computadores

Trabajo de Fin de Grado

Segundo cuatrimestre
Curso 2012-2013

Autor:

Javier Abellán Fernández

Tutor:

Telmo Zarraonandia Ayo

Índice de contenido

Índice de ilustraciones

Índice de tablas

1. INTRODUCCIÓN

En este capítulo se presenta el contexto del proyecto, la motivación que ha llevado a su desarrollo, sus objetivos, los medios empleados y la estructura del resto de la memoria.

1.1 Contexto

Los avances tecnológicos de las últimas décadas han supuesto un cambio drástico en la manera de entender la vida. Su uso se ha ido extendiendo sobre los más diversos campos de aplicación, abarcando desde el ocio y el entretenimiento hasta el área de la educación. Al respecto, una de las industrias de entretenimiento que más se ha beneficiado de los distintos avances en la tecnología ha sido la del videojuego, que ha sabido sacar provecho de los mismos para ampliar su público objetivo desde el tradicional jugador habitual a un tipo de jugador más esporádico que hasta ahora no se había acercado a este tipo de ocio. De hecho, en los últimos años los videojuegos han ido mejorando su reputación y las tradicionales connotaciones negativas con las que hasta ahora eran asociados han ido dando paso a su vinculación con nuevos usos y aplicaciones más allá del mero entretenimiento. Entre estos usos destaca, sin duda, el de la educación, donde los videojuegos ofrecen interesantes posibilidades como herramienta didáctica debido a su capacidad de motivar al alumno.

En cualquier caso, si bien cada vez es más común encontrarnos con videojuegos educativos, los costes de diseño y desarrollo de este tipo de herramienta siguen siendo muy elevados. Por una parte su programación requiere un conocimiento técnico elevado que únicamente puede ofrecer personal altamente cualificado. Por otro lado requieren además de la colaboración de diseñadores gráficos que se encarguen de proporcionar una estética atractiva. Por último suele ser necesario la asistencia o participación en la fase de diseño de educadores y expertos en la materia que garanticen que el producto final sirva a su propósito didáctico.

En este contexto el presente proyecto se enmarca dentro del desarrollo de una plataforma (**Game Rules Scenario Player – GREP**) que trata de aliviar esta situación permitiendo a educadores reducir el tiempo y coste de diseño e implementación de juegos que puedan ser utilizados dentro de sus procesos educativos, minimizando o prescindiendo incluso de asistencia técnica en el proceso.

La plataforma GREP es capaz de interpretar diseños de juegos educativos especificados en ficheros **XML**, de generar el correspondiente entorno virtual que permita al alumno jugar al juego y de almacenar información sobre la partida de forma que el tutor o profesor pueda conocer cuál ha sido el desarrollo de la misma. Haciendo uso de las diversas herramientas de la plataforma un educador puede, actualmente, diseñar juegos mono-jugadores para ser jugadores en navegadores.

1.2 Objetivo Principal

Con el fin de permitir a los educadores diseñar e implementar experiencias de juego más complejos, en este proyecto se plantea extender la actual implementación de la plataforma GREP para ofrecer una solución que permita el diseño y definición de comportamientos de aquellas entidades del juego que el jugador no controla pero con las que puede interactuar a lo largo de una partida.

1.3 Objetivos Específicos

Como ya se ha mencionado, el objetivo general del proyecto consiste en ofrecer a usuarios con un perfil técnico bajo una herramienta eficaz e intuitiva, que les permita definir y/o configurar una serie de comportamientos para los personajes de sus juegos con el fin de crear una seudo Inteligencia Artificial (IA). Esto es, que los personajes no jugadores posean una lógica por la cual puedan parecer inteligentes.

Los objetivos específicos de este proyecto, se pueden resumir en los siguientes puntos:

- **Diseño de los comportamientos directamente sobre el escenario del juego**: la herramienta proveerá al usuario de una interfaz que le permitirá definir los comportamientos de los jugadores directamente sobre el escenario del juego. Con este objetivo se busca reducir la complejidad que puede suponer a un usuario con bajo perfil técnico definir los comportamientos mediante herramientas o lenguajes auxiliares que requieran de un aprendizaje adicional.

- **Definición de eventos que disparan los comportamientos de los personajes**: para que el comportamiento de los personajes no jugadores tenga lógica, será preciso permitir al educador establecer una lista de eventos que los desencadenen o interrumpan y que, en último término, estarán relacionados con las acciones que el jugador va desarrollando en el juego.

- **Exportación de los comportamientos diseñados**: se deberá dar la opción de exportar los diseños realizados de forma que puedan ser incluidos en otros juegos.

- **Extensión de la plataforma GRE Player para la interpretación y ejecución de los comportamientos diseñados**: finalmente, será preciso ampliar la funcionalidad del módulo encargado de ejecutar la lógica del juego en la plataforma GREP de tal forma que sea capaz de ejecutar los comportamientos que le diseñador ha especificado cada vez que se detecte que los eventos por él definidos tienen lugar.

Como resultado del proyecto se espera por tanto obtener dos aplicaciones o módulos distintos:

- Una herramienta de autoría que permita al educador el diseño e implementación de los comportamientos de los personajes, a la que llamaremos "**NPCs GREP Behaviour Authoring Module**".
- Una extensión de la plataforma GRE Player que permita la integración y ejecución de los comportamientos previamente diseñados dentro de una partida concreta a un videojuego. Esta extensión se denominará "**NPCs GREP Behaviour Player Module**".

La plataforma GRE Player ha sido implementada sobre el motor de juegos Unity 3D, que actualmente es uno de los motores con mayor nivel de crecimiento del mercado gracias, en parte, a su dinamismo y portabilidad entre plataformas. El desarrollo del proyecto requerirá por tanto del estudio del lenguaje de programación usado por este motor y del conjunto de herramientas que proporciona para su programación, y de la familiarización con la arquitectura y el código de la actual implementación del GRE Player.

1.4 Motivación

La motivación personal a la hora de decidir abordar la realización de este proyecto ha venido especialmente fundamentada en las siguientes razones:

- **Investigación y desarrollo sobre motores gráficos.**

  Se puede considerar el motivo principal. La industria del videojuego ha alcanzado un importante nivel dentro del mercado del ocio. Con los avances en hardware de los últimos años, se están desarrollando nuevos motores gráficos cada vez más capaces de alcanzar un nivel de realismo espectacular. Es por ello que el hecho de realizar un estudio sobre sus capacidades, alcance y limitaciones sea de gran interés para mí.

- **Colaboración en el desarrollo de sistemas educativos.**

  Además, este proyecto se encuentra igualmente relacionado con la investigación y desarrollo de nuevas formas de aprendizaje, que rompan con la tradicional dinámica de estudio, y permitiendo explorar las posibilidades que los juegos pueden ofrecer en la educación. Se trata por tanto, en último término, de ofrecer un método alternativo de estudio, entretenido y divertido.

1.5 Medios empleados

En este apartado se describen los distintos medios, hardware y software, que se han empleado para el desarrollo del proyecto.

1.5.1 Elementos hardware

Tanto para la implementación como para el desarrollo de la documentación, se han empleado dos equipos informáticos como soporte hardware:

- **Ordenador de sobremesa**: dotado de un procesador AMD Phenom 9600, 4GB de memoria RAM, tarjeta gráfica AMD 4890 y sistema operativo Windows 7 Profesional.
- **Ordenador portátil**: provisto de un procesador Intel Core i7 2500K, 8GB de memoria RAM, disco duro SSD de 256GB, tarjeta gráfica NVIDIA GeForce GTX 650 y sistema operativo Windows 7 Profesional.

Los equipos se han dispuesto y están preparados para soportar la importante carga de procesamiento que precisa el desarrollo y renderizado de gráficos; aspecto especialmente importante para este proyecto.

1.5.2 Elementos software

En la implementación se han utilizado los siguientes elementos software:

- **Unity SDK**: Completo entorno de desarrollo para la programación de aplicaciones con el motor gráfico de Unity. Se han empleado los lenguajes de programación C# y UnityScript que ofrece Unity, mediante scripting.
- **MonoDevelop**: Conocido IDE diseñado para trabajar con C# y lenguajes de programación .NET. Se ha empleado tanto para edición como para depuración, gracias al motor de depuración de código que integra.
- **Microsoft Office Profesional 2010**: Suite de Microsoft para la elaboración de documentación. Se ha empleado para realizar el presente documento, así como de sus anexos.
- **Dropbox:** Repositorio multiplataforma de alojamiento de archivos en la nube. Ofrece un servicio de sincronización automática de los documentos compartidos Se ha empleado a modo de respaldo de los ficheros del proyecto.

1.6 Estructura de la memoria

La memoria del proyecto consta de las siguientes secciones y/o capítulos:

- **Introducción**: En este capítulo se tratan aspectos generales del proyecto, y se presenta el trabajo que se va a llevar a cabo.
- **Estado del Arte**: Se plantea un análisis del estado actual de la industria del videojuego, ofreciendo una comparativa de los principales motores gráficos que disponen de entornos de desarrollo accesibles, situando el proyecto en su contexto. También se incluye un estudio sobre las diferentes técnicas de obtención e implementación de comportamientos automáticos.
- **Análisis funcional**: En este capítulo se extraen las funcionalidades del sistema planteado, representadas mediante de un modelo de casos de uso y apoyado por un análisis de requisitos.
- **Diseño Técnico**: Recoge la especificación de la arquitectura empleada, las tecnologías y los detalles de la implementación de los módulos.
- **Gestión del Proyecto**: Presenta la planificación utilizada durante el proyecto, su descomposición en fases y el presupuesto aplicado.
- **Conclusiones**: En este capítulo se recogen las ideas finales, experiencias y problemas surgidos durante el proyecto, exponiendo un resumen global sobre lo aprendido, así como sobre las líneas futuras de dicho proyecto.
- **Referencias**: Se lista la bibliografía empleada para la obtención de información aplicada a la redacción de esta memoria.

Por otro lado, como anexo, se encuentra un **manual de usuario** que presenta paso a paso y con apoyo gráfico, las funcionalidades de la aplicación.

2. ESTADO DEL ARTE

En este capítulo se analizarán los principales motores gráficos que fueron considerados candidatos a la hora de soportar el desarrollo de la aplicación, así como de una serie de algoritmos y técnicas comúnmente empleadas a la hora de definir e implementar comportamientos de personajes no jugadores (NPCs). Además, y en primer lugar, se presentará también el modelo teórico que ha sido utilizado como guía para el desarrollo de la plataforma GRE Player y que establece la forma en que los juegos que éste interpreta deben ser descritos.

2.1 El modelo GRE (Game Rules and scEnario).

El proyecto GREP se guía por un modelo conceptual conceptual (**Game Rules and Scenarios Model – GREM**) [**1**] que facilita la reutilización de partes del diseño de juegos educativos y la producción de variantes de un juego que proveen un conjunto de elementos configurables a partir de un vocabulario básico. En la Ilustración 1 - Modelo de diseño GREP se muestra el planteamiento de dicho modelo, en el cual los elementos son organizados en dos submodelos independientes: el **modelo de reglas del juego** (a la izquierda) y el **modelo de escenario** (a la derecha). El primero describe las reglas que especifican cómo se juega al juego; el segundo define el entorno sobre el cual se va a jugar y las interfaces provistas para interactuar con él.

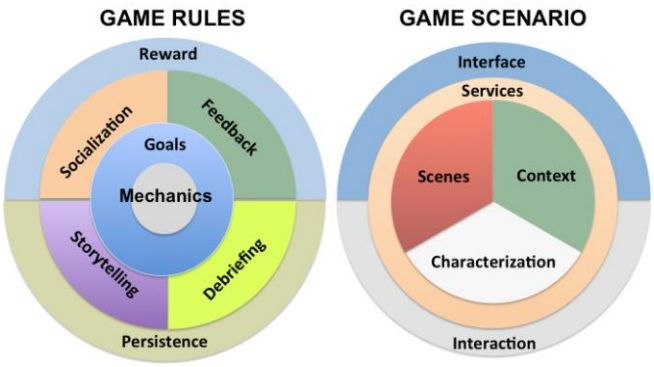


Ilustración 1 - Modelo de diseño GREP [1]

La idea es permitir jugar a un juego en múltiples escenarios y que sobre éstos mismos se pueda jugar a diferentes tipos de juegos. Además, para facilitar su reutilización, los elementos del modelo se organizan en distintas capas, de modo que cada capa está definida por los elementos de las capas inferiores. Siguiendo esta aproximación, los dos niveles inferiores de la perspectiva del modelo de reglas de juego proveen de los elementos que describen las mecánicas y objetivos de cada juego, que los jugadores deben seguir y lograr, respectivamente. Por su parte, los niveles 3 y 4 permiten a los diseñadores extender la definición del juego organizando los objetivos anteriormente definidos en una secuencia de desafíos que conforman la línea argumental, describir las reglas que controlan las interacciones sociales entre jugadores, especificar el tipo y la cantidad de reacciones que afectan al jugador o añadir recompensas o mecanismos de persistencia.

En relación al modelo de escenario, las capas inferiores permiten a los diseñadores definir las representaciones de las entidades del juego, las escenas, características y representaciones de los caracteres del juego, así como describir otros elementos que pueden ser útiles para establecer el contexto de ciertas situaciones que tengan lugar sobre el escenario. Por otro lado, las capas externas permiten a los diseñadores especificar los servicios que incrementarán las posibilidades de los diferentes juegos que podrán ser llevados a cabo en el escenario y que definen, de un modo abstracto, el aspecto sobre el cual los elementos de representación y los servicios serán organizados y presentados al jugador por cada dispositivo y el tipo de interacciones que será capaz de realizar a través de ellos.

Una vez definido un conjunto de reglas y un escenario los diseñadores obtendrán el diseño final del juego relacionando las entidades de una y otra perspectiva de la manera que mejor se adapte a sus necesidades.

2.2 Motores gráficos y entornos de desarrollo

2.2.1 Source Engine

El Source Engine [**2**] es un motor gráfico diseñado originalmente para juegos de acción en primera persona, aunque posteriormente ha sido empleado también en el desarrollo de otros tipos de juegos como rol, puzzles y acción en otras perspectivas. La version actual es la segunda del motor creado por Valve, en 2004. Desde entonces, no se han publicado nuevas versiones, sino que se ha seguido un protocolo basado en actualizaciones sin numeración.

Uno de los puntos que hace al Source Engine motivo de mención en este documento es, que es uno de los que mejor escalabilidad multi procesador tiene, ofreciendo un rendimiento muy bueno en máquinas de gama baja actualmente.

Ilustración 2 - Source Engine primeras versiones

Aunque no es uno de los motores más potentes gráficamente, incorpora un destacable sistema de renderizado e iluminación, flujo de líquidos y un motor propio de animación facial, entre otros aspectos.

Con el Source Engine, Valve provee de un entorno de desarrollo completo, desde el cual se pueden crear desde escenarios y animaciones hasta versiones completas de un videojuego.

Valve ha desarrollado en paralelo el software Source Filmmaker, capaz de realizar y editar explícitamente animaciones. Con él se han llevado a cabo trailers de algunos de los últimos juegos de Valve [**3**].

Ilustración 3 - Source Engine actual

2.2.2 Unity

Unity [**4**] es el resultado de un intento de crear un motor gráfico y una serie de herramientas de desarrollo que no requieren de un nivel muy alto de conocimientos para usarse, ofreciéndose, además, a un bajo precio.

Fue diseñado originalmente para proyectos en computadores Mac, pero con el continuo proceso de actualización e incorporación de funcionalidad, se ha ido extendiendo hacia otras plataformas.

Su primera versión se data en 2005, pero es a partir de Unity 3 cuando se da a conocer masivamente sobre todo gracias a la integración con herramientas de diseño como 3ds Max, Blender, Cinema 4D, Maya, Photoshop, entre otras.

Unity mantiene un sistema de coherencia de datos, por el que cualquier cambio realizado sobre un elemento del proyecto actualice todas sus referencias, de modo que mejora notablemente la usabilidad. Además, soporta las librerías gráficas Direct3D (sistemas Windows), OpenGL (sistemas Windows, Mac y Linux), OpenGL ES (dispositivos móviles iOS y Android) y librerías propietarias, como las de Wii e incorpora un potente motor de renderizado de modelos y texturas, efectos, iluminación y sombras, postprocesado y con soporte de físicas PhysX [**5**], resultando uno de los motores gráficos más completos

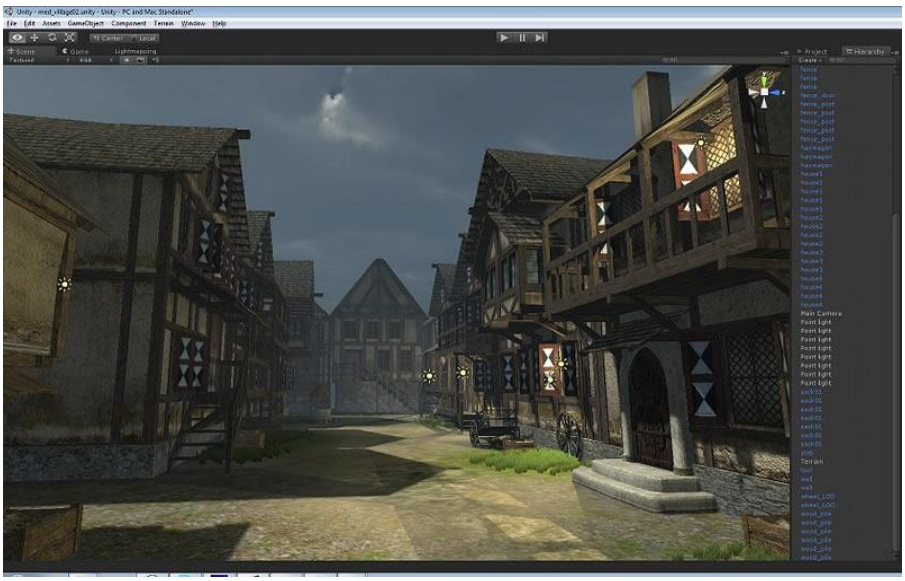


Ilustración 4 - Unity Pro, renderizado

En lo relativo a la programación, el código de un juego desarrollado mediante Unity se estructura en torno a un conjunto de scripts basados en Mono, una implementación de código libre de .NET Framework, aunque también existe la posibilidad de programar con UnityScript, C# o Boo. Uno de los puntos fuertes de Unity es su plataforma de desarrollo, caracterizada por disponer de una interfaz no muy compleja, en comparación con otros entornos de desarrollo.

Como es habitual también, incorpora su propio editor de código, incluyendo sistemas de depuración. A partir de la version 3.5 se incluyen nuevos modos de cálculo de rutas (pathfinding) y evasión de obstáculos y se mejora notablemente el procesado multi-hilo. También añade soporte a Flash Player. Unity 4, anunciado en Junio del 2012, incluye mejoras internas y algunas herramientas que favorecen la usabilidad de la aplicación.

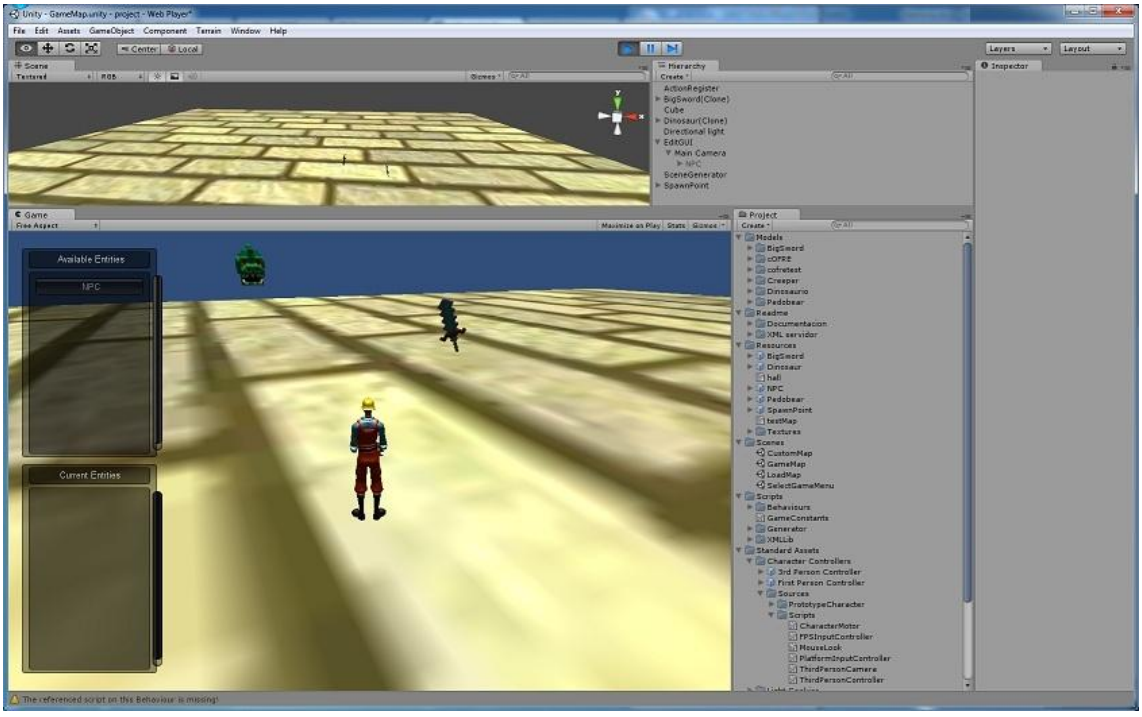


Ilustración 5 - Unity, entorno de desarrollo

Uno de los aspectos que ha hecho que Unity sea un gran candidato para el desarrollo de este proyecto ha sido su integración con navegadores web, ofreciendo la posibilidad de diseñar la aplicación independientemente del sistema operativo sobre el que se vaya a ejecutar.

Desde la herramienta de desarrollo se puede exportar un proyecto directamente a la plataforma que se desee, con ciertas limitaciones asociadas al tipo de licencia empleada:

La licencia gratuita permite hacer uso de la mayor parte de la funcionalidad, pudiendo exportar los proyectos como ejecutables para PC y Mac, en Flash y para navegadores web, mientras que para exportar a iOS, Android, y plataformas privadas, como Xbox 360, PlayStation 3 o Wii se requieren licencias independientes.

La licencia estándar de pago incorpora algunos cambios en la interfaz que facilitan el desarrollo, así como el acceso a herramientas que permiten gestionar mejor los proyectos.

2.2.3 Unreal Engine

Desarrollado desde 1998 por Epic Games, el Unreal Engine [**6**] es uno de los motores más estables creados hasta la fecha. Su facilidad de adaptación a múltiples plataformas y las herramientas de diseño que ofrece han hecho de él una de las principales opciones a tener en cuenta para la realización de proyecto.

Existen tres versiones oficiales: Unreal Engine 1, Unreal Engine 2 y Unreal Engine 3.

La primera de las versiones, Unreal Engine 1, aparece en 1998 incluyendo un poderoso motor de renderizado. Destaca por incorporar un complejo motor de Inteligencia Artificial (IA) y un sistema de comunicación de red, aunque lo más llamativo de la propuesta fue el uso de un lenguaje de programación propio que facilitaba que usuarios con un nivel técnico avanzado pudieran crear modificaciones del mismo. En 2002 aparece una nueva version, Unreal Engine 2m en la que se reescribe el motor prácticamente en su totalidad, resolviendo numerosos errores e incorporando mejoras importantes, como el motor de físicas Karma physics, el cual dotaba de un sistema de colisiones de partículas mucho más realista. Finalmente, en 2006 aparece una nueva version diseñada esta vez para DirectX. Es sin duda la versión más destacable de todas en cuanto a nivel de procesamiento. Ofrece técnicas muy avanzadas de renderizado, iluminación y sombras y se encuentra disponible para casi cualquier plataforma: consolas, Mac OS X, iOS, Android, Flash Player 11, entre otros.

Comparativa de renderizado entre versiones:

Ilustración 6 - Comparativa de renderizado Unreal Engine

Haciendo uso de esta última version se encuentra disponible la herramienta **UDK** (Unreal Development Kit)[**7**], ofrecida por Epic Games como entorno completo de desarrollo. Permite tanto diseño, como desarrollo y despliegue de escenarios, incluyendo su propio sistema de scripts. Su licencia gratuita permite realizar eventos no comerciales, teniendo que adquirir una versión de pago para tal fin.

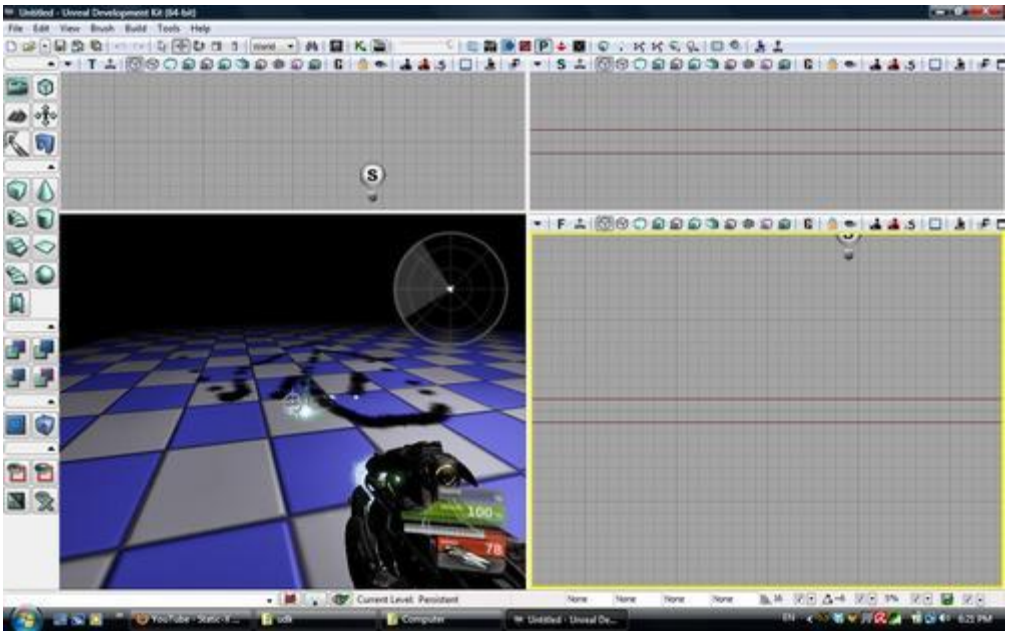


Ilustración 7 - Unreal Engine, entorno de desarrollo

2.2.4 CraftStudio

CraftStudio [**8**] no es un motor gráfico, aunque incorpora el suyo propio.

Es una plataforma de desarrollo 3D cooperativo en tiempo real. Esto es, que los proyectos se realizan tanto local, como remotamente, a través de la interfaz proporcionada, en la que varios participantes pueden interactuar con los mismos elementos “simultaneamente”. Por ejemplo, un diseñador puede estar creando un modelo 3D mientras otros realiza la textura o la animación.

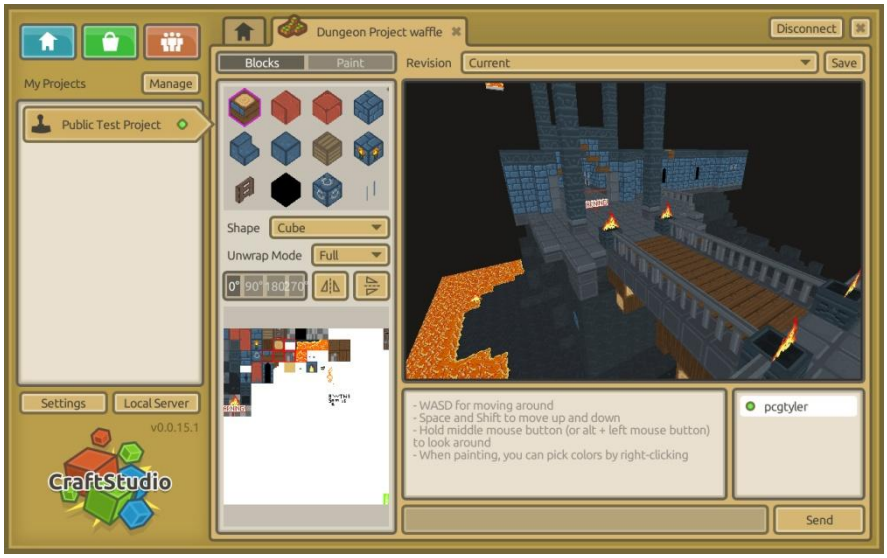


Ilustración 8 - CraftStudio, entorno de desarrollo

Se caracteriza por emplear formas cúbicas, tanto para los modelos como para los escenarios, lo que lo hace relevante para este proyecto, ya que se ajusta a la ideología principal con la que fue concebido: diseño amigable cargado de linealidad, sin curvas.

Es muy reciente y se encuentra aún en versiones iniciales. Sin embargo, ofrece la funcionalidad básica para realizar un juego completo, con la limitación de no incorporar funciones multijugador.

La licencia gratuita permite participar en la creación de modelos, texturas, animación y escenarios de un proyecto. La version de pago habilita la creación de scripts, basados en Lua, para la modificación de la lógica del juego.

2.3 ¿Por qué Unity?

A continuación se ofrece una comparativa de los entornos de desarrollo de los motores gráficos estudiados para la viabilidad de este proyecto, así como una serie de impresiones obtenidas, basada en los siguientes parámetros:

- **Múltiples plataformas**: especifica si la herramienta permite exportar la aplicación a varias plataformas.
- **Usabilidad**: entendida como el nivel de conocimientos que debe poseer un usuario para emplear la herramienta.
- **Integración externa**: indica el grado con el que la herramienta es capaz de interactuar con herramientas externas.
- **Renderizado**: generalización que engloba todos los aspectos gráficos: básico, medio, avanzado.
- **Motor de físicas**: especifica si emplea o no procesado de físicas.
- **Licencia gratuita**: indica con qué condiciones se puede emplear la herramienta con una licencia gratuita.

	Múltiples plataformas	Usabilidad	Integración externa	Renderizado	Motor de físicas	Licencia gratuita
Source Engine	Si, no web	-Nivel técnico alto	-Limitado, herramientas de diseño gráfico	-Medio	-Propietario	-Uso no comercial
Unity	Si	-Nivel técnico medio	-Amplio, herramientas de diseño gráfico	-Avanzado	-PhysX	-Uso no comercial -Licencia de pago con funcionalidad extra
Unreal Engine	Si	-Nivel técnico alto	-Limitado, herramientas de diseño gráfico	-Avanzado	-PhysX	-Uso no comercial
CraftStudio	No	-Nivel técnico bajo	-No	-Básico	-No	-Funcionalidad muy limitada

Tabla 1 - Comparativa motores gráficos

Tal y como se aprecia en la comparativa, Unity muestra cierta ventaja respecto a las otras opciones en cuanto a usabilidad, accesibilidad y escalabilidad, aunque el principal factor que ha inclinado finalmente la balanza hacia su elección es la disponibilidad de un amplio número de herramientas externas de diseño sobre las que ofrece facilidades de integración.

2.4 Técnicas de definición de comportamientos

Desde la aparición de los juegos de lógica, como el Ajedrez o las Damas, se ha buscado recrear el comportamiento de un jugador que les permitiese competir como si se tratase de una persona real. Con los años, las técnicas de generación de estos comportamientos han ido incrementando su complejidad, siendo uno de los campos de aplicación más habituales de las distintas técnicas y algoritmos de la *Inteligencia* Artificial (IA). Es esta sección se describen brevemente algunas de esas técnicas y algoritmos que más frecuentemente se utilizan en videojuegos.

Ilustración 9 - Genibo: el perro robot

2.4.1 Algoritmos de IA

Máquinas de estados

Una máquina de estados es un modelo matemático de computación, concebido como una máquina abstracta que nos permite determinar el comportamiento de una entidad, en base al estado en el que se encuentre (Máquinas de Moore) y una serie de "inputs", o señales de entrada (Máquinas de Mealy) [**9**].

Las máquinas de estado pueden ser utilizadas en muchos aspectos y niveles, según la complejidad del comportamiento. Un ejemplo básico de una máquina de estados puede ser un semáforo peatonal [**10**]:

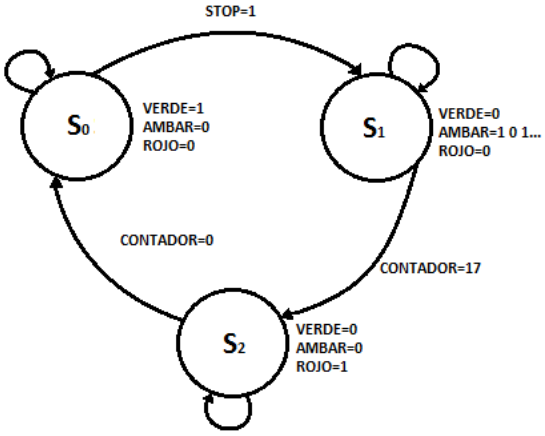


Ilustración 10 - Máquina de estados: semáforo

El funcionamiento de la máquina del diagrama se puede definir mediante el siguiente razonamiento:

- Inicialmente se encuentra en el estado **verde** (S0).
- Cuando se presiona el botón, cambia su estado a **ámbar** (S1).
- A los 3 segundos, cambia su estado a **rojo** (S2).
- Tras 17 segundos, vuelve al estado inicial **verde**.

La complejidad de las máquinas de estados se eleva exponencialmente según el comportamiento a representar. Sin embargo, suponen una metodología de definición de comportamientos sencilla y práctica a niveles básicos, que no precisan de conocimientos muy avanzados para llevarse a cabo. Por el contrario, presenta una serie de problemas concretos, como es la limitada flexibilidad y adaptabilidad o la baja reusabilidad. Al contrario que las redes de neuronas, ofrecen un comportamiento rígido y predecible, lo que conlleva obtener un resultado poco realista a la larga. Sin embargo, los costes de estos algoritmos se ven notablemente reducidos, tanto en implementación como en eficiencia. Debe destacarse que este último aspecto tiene especial importancia en el desarrollo del software, ya que esto permite que dicho software sea ejecutado en una rama más amplia de computadores, al no requerir tantos recursos.

En relación a la industria del videojuego, se han ganado el título de la forma más básica de definición de comportamientos automáticos [**11**], con conocidos ejemplos como el Pac-Man, donde el estado de cada entidad de tipo fantasma es definido en función de los eventos que pueden efectuarse, acorde a una máquina finita de estados, representado en la Ilustración 11.

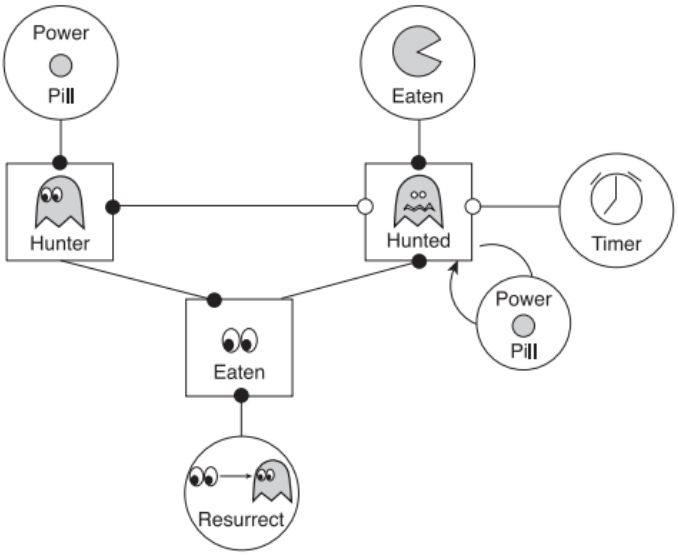


Ilustración 11 - Máquina finita de estado: Pac-Man

La limitación de esta metodología está clara: cuando se pretende manejar una cantidad elevada de estados y transiciones posibles, tanto la definición como el mantenimiento de cada máquina de estado resulta ineficientemente costosa, hasta el punto de considerarse inviable.

Por otro lado, los scripts con comportamientos predefinidos se pueden entender como una implementación básica de una serie de máquinas de estados individuales, cada una capaz de especificar un comportamiento concreto y/o detener la ejecución de otras máquinas.

Algoritmos genéticos

Los algoritmos genéticos son un conjunto de procedimientos de búsqueda con capacidad adaptativa. Simplificando mucho, se basan en la idea de resolver los problemas de la misma manera que lo hace la naturaleza, imitando la evolución biológica. Su origen se remonta a los años 50 y 60, cuando varios científicos comenzaron a estudiar los sistemas evolutivos como herramientas en problemas de optimización en ingeniería [**12**]. El problema principal que plantea un algoritmo genético es su excesiva complejidad, lo que implica un importante enfrentamiento entre eficiencia y la eficacia. Este motivo supone una inversión de recursos de procesamiento considerable, limitando las plataformas sobre las que desplegar un sistema informático. Otro inconveniente es su elevado coste de implementación, ya que son algoritmos que requieren un alto nivel de conocimientos tanto teóricos como prácticos, lo que los aleja de ser reutilizables y/o escalables. Sin embargo, son herramientas muy potentes, empleadas comúnmente en la industria de investigación y desarrollo como medios de optimización de variables y patrones en problemas de minería de datos [**13**].

En la industria del videojuego, los algoritmos genéticos se emplean sobre todo para crear comportamientos de inteligencia artificial, ofreciendo resultados muy avanzados y realistas. La principal característica por la que destacan es su capacidad de aprendizaje: el algoritmo evoluciona a partir de los datos que ha obtenido tras su última ejecución.

Un ejemplo práctico aplicado es el **bot** de muchos juegos de disparos en primera persona, o **FPS**. Los bots pretenden simular el comportamiento de un jugador humano, incorporando incluso motores de procesamiento del lenguaje, lo que les permite "hablar" con otros bots o jugadores.

Ilustración 12 - Quake III: bots

Pathfinding

La técnica del camino más corto, o pathfinding, es un procedimiento mediante el cual se calcula el camino a seguir para llegar desde un punto origen hasta uno destino, obteniendo el camino más corto [**14**]. Este es un problema habitual en el área de los videojuegos donde es necesario que los caracteres sean capaces de conducirse por un escenario por sí mismos, siguiendo distintas rutas que pueden variar dependiendo del objetivo que se persigue. Aunque, dada la naturaleza de este proyecto, no se aplique explícitamente como técnica de definición de comportamientos, merece la pena dedicarle unos párrafos, ya que es la base para la optimización de la implementación de una máquina de estados.

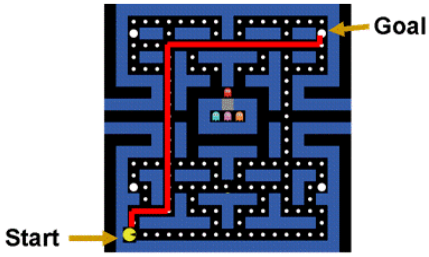


Ilustración 13 - Ejemplo de pathfinding: Pacman

De esta forma, dada una máquina de estados representada mediante un grafo, si se pondera cada salto (arista) con el coste que implica tomar dicho camino, se obtiene un mapa con las posibles opciones para llegar de un estado a otro:

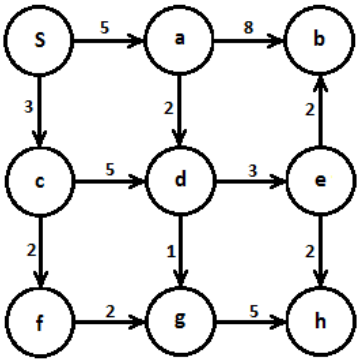


Ilustración 14 - Grafo ponderado

Esto plantea tres posibles problemas:

- El camino más corto **desde el origen** a todos los demás.
- El camino más corto para **alcanzar un estado final**.
- El camino más corto **entre todos los estados**.

La solución para el primer problema se puede alcanzar mediante el algoritmo de Dijskstra [**15**] resultando que el camino más corto desde **S** hacia todos los demás vértices se encuentra definido del modo:

De modo que podemos concluir que el camino más corto desde **S** a **h** es:

S, h *=>* ***S*** *- c - f - g –* ***h*** *con un peso de 15*

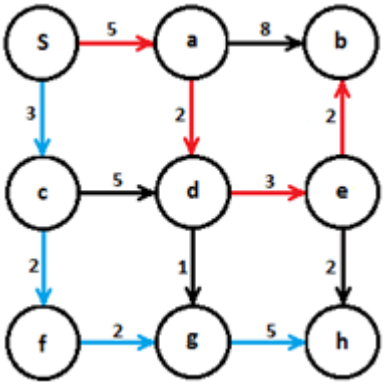


Ilustración 15 - Solución Shortest Path Problem

Existen algoritmos específicos para la resolución de cada problema, sobre los cuales se podría desarrollar todo un proyecto; algunos de los cuales podrían ser resueltos mediante heurística.

Ilustración 16 - RTS: Age of Empires

La aplicación de estos problemas es un aspecto fundamental en videojuegos del género **RTS** (*Real Time Strategy*). Tales como Age of Empires o Starcraft.

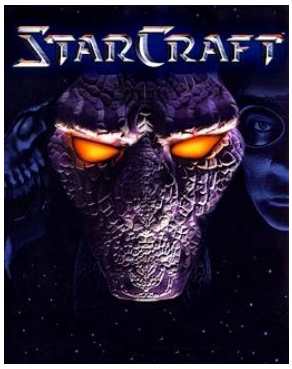


Ilustración 17 - RTS: StarCraft

3. UNITY SDK

El SDK, o entorno de desarrollo de software, de Unity es un conjunto de herramientas muy potente a través del cual se llevan a cabo proyectos con el motor gráfico Unity 3D. Por este motivo, y dado que este proyecto se ha realizado con mencionadas herramientas, es menester hacer un breve estudio sobre los requisitos, entorno y consideraciones varias al respecto.

3.1 Requisitos del Sistema

En esta apartado se hará una exposición sobre los requisitos, tanto de hardware como de software, que el entorno de desarrollo de Unity precisa para funcionar con fluidez.

3.1.1 Requisitos de Hardware

Los componentes físicos mínimos que debe poseer un equipo, según el nivel de carga de recursos, son los siguientes:

Gráficos 2D

Requisitos de Hardware para Gráficos 2D
Procesador de doble núcleo a 1.66GHz o similar
Tarjeta gráfica con soporte DirectX 9 (shader 2.0)
2GB de memoria RAM
Disco duro estándar a 5400rpm

Tabla 2 - Requisitos de Hardware para gráficos 2D

Gráficos 3D (básico)

Requisitos de Hardware para Gráficos 3D (básico)
Procesador de doble núcleo a 2.93GHz o similar
Tarjeta gráfica con soporte DirectX 9 (shader 3.0)
2GB de memoria RAM
Disco duro estándar a 7200rpm

Tabla 3 - Requisitos de Hardware para gráficos 3D (básico)

Gráficos 3D (high-end)

Requisitos de Hardware para Gráficos 3D (high-end)
Procesador de cuádruple núcleo a 2.66GHz o similar
Tarjeta gráfica con soporte DirectX 11 (shader 5.0)
4GB de memoria RAM
Disco duro de alto rendimiento a 7200rpm Recomendable disco duro de estado sólido (SSD)

Tabla 4 - Requisitos de Hardware para gráficos 3D (high-end)

3.1.2 Requisitos de Software

Los elementos de software mínimos dependen en gran medida de la plataforma final sobre la que será desplegado el proyecto:

Unity iOS

Requisitos de Software para Unity iOS
Sistema operativo Snow Leopard 10.6 o superior
Librerías de desarrollo Unity OS X
Entorno de desarrollo Xcode versión 4 en adelante

Tabla 5 - Requisitos de Software para Unity iOS

Unity Android

Requisitos de Software para Unity Android
Sistema operativo Snow Leopard 10.6 o superior Windows XP Service Pack 2 o superior
Kit de desarrollo de Android (SDK) y Java (JDK)
Dispositivo Android (emulado o físico) con sistema operativo versión 2.0 o superior,CPU con soporte ARMv7 y GPU con soporte OpenGLES 2.0

Tabla 6 - Requisitos de Software para Unity Android

Unity para ordenadores

Requisitos de Software para ordenadores
Sistema operativo Snow Leopard 10.6 o superior Windows XP Service Pack 2 o superior
Librerías gráficas OpenGL 2.0 y DirectX 9 (solo Windows) o superior

Tabla 7 - Requisitos de Software para ordenadores

3.1.3 Requisitos técnicos

Para aprovechar las características del SDK, los desarrolladores deben estar familiarizados con alguna de los siguientes tecnologías:

Mono

Mono [**16**] es un proyecto de código abierto desarrollado por Xamarin para crear un estándar Ecma sobre un conjunto de herramientas compatibles con el framework .NET, entre otras, un compilador de C# y una máquina virtual de .NET Common Language Runtime, o CLR.

Su propósito principal es lograr que las aplicaciones .NET sean multiplataforma, pero sobre todo tratan de facilitar herramientas de desarrollo para desarrolladores en Linux. Mono puede ejecutarse en muchas sistemas: distribuciones de Linux (incluido Android), OSX, Windows, Solaris y otras plataformas como Xbox 360 o PlayStation 3.

Con el lanzamiento del framework de .NET en el año 2000 por parte de Microsoft, Miguel de Icaza consideró que .NET tenía el potencial necesario para aumentar la productividad en el ámbito de la programación, por lo comenzó a investigar para el desarrollo de una versión para Linux. Por este motivo, lanzaron la primera versión del proyecto en julio de 2001, en la conferencia O'Reilly.

En junio del 2004 se lanzó la versión oficial de Mono 1.0, extendiendo la idea inicial para dar soporte a un amplio grupo de arquitecturas y sistemas operativos.

En el 2011, Miguel de Icaza anunció que el proyecto sería desarrollado desde entonces por la compañía Xamarin, que pretendían reescribir las librerías de .NET de iOS y Android. Este hecho cuestionó el futuro del proyecto Mono, ya que se estaba usando con unos fines que no eran los originales. Ese mismo año, se aprobaron las licencias para el desarrollo de MonoTouch y Mono for Android con ese propósito.

C#

C#, pronunciado C Sharp, es un lenguaje de programación desarrollado por Microsoft como parte de la iniciativa .NET y posteriormente aprobado como estándar por Ecma y ISO. Es uno de los lenguajes diseñados para la Infraestructura Común de Lenguaje, o CLI.

Es un lenguaje orientado a objetos, fuertemente tipado, imperativo, declarativo y funcional. Se diseñó para ser de propósito general, simple y moderno.

C# se construyó durante el desarrollo del framework .NET en el año 2000, haciendo su primera aparición pública en 2002, con la versión 1.0 de .NET Framework y Visual Studio 2002 [**17**]. Su diseñador principal fue Anders Hejlsberg, el cual formó parte en el diseño de Turbo Pascal, CodeGear Delphi y Visual J++.

Tuvo una gran controversia, cuando James Gosling, creador de Java en 1994 anunció que C# era una imitación de Java. Los creadores de C++, Klaus Kreft y Angelika Langer, junto con otros desarrolladores confirmaron su parecido, hasta el 2005 donde C# y Java divergen haciendo menos evidente su similitud.

C# incorpora importantes características que facilitan la programación funcional, soportando los framework de expresiones lambda, extensión de métodos, tipos anónimos y "closures".

Se encuentra actualmente en la versión 5.0, acorde a .NET Framework 4.5 y Visual Studio 2012.

.NET

.NET Framework [**18**] es un framework de software desarrollado por Microsoft inicialmente para el sistema operativo Microsoft Windows. Incluye una gran cantidad de librerías y se caracteriza por proveer de interoperabilidad entre múltiples lenguajes, en los que cada uno de ellos puede utilizar código escrito en otros diferentes.

.NET se ejecuta sobre la máquina virtual conocida como Common Language Runtime (CLR) [**19**] que implementa diversos mecanismos de seguridad, gestión de memoria y control de excepciones.

Las librerías de .NET proveen de clases para el manejo de datos, interfaces, conectividad con bases de datos, algoritmia, redes, criptografía y desarrollo web. Su intención es extenderse y ser empleado por la mayoría de aplicaciones creadas para la plataforma Windows. Para ello, Microsoft puso a disposición de los desarrolladores el IDE Visual Studio.

Comenzó su desarrollo en los últimos años de los 90, bajo el nombre de Next Generation Windows Services (NGWS), surgiendo la primera versión beta de .NET en el 2000.

Hasta la fecha han salido al mercado las versiones 1.1, 2.0, 3.0, 3.5, 4.0 y 4.5, acompañado por actualizaciones de Visual Studio y distribuido a través del sistema operativo Windows, desde la versión 1.1 con Windows Server 2003, hasta la 4.5 actualmente vigente en Windows 8 y Windows Server 2012.

Entre sus características principales, destacan su interoperabilidad entre las distintas versiones de .NET, permitiendo ejecutar aplicaciones antiguas en sistemas operativos Windows posteriores; su máquina virtual CLR, la cual garantiza ciertos aspectos de seguridad, gestión de memoria y manejo de excepciones, antes mencionados; el simplificado despliegue que requiere para ser ejecutado; la portabilidad entre sistemas provista por proyectos como Mono y la independencia del lenguaje que ofrece, permitiendo la implementación de librerías entre cualquiera de los lenguajes que integra .NET.

Javascript

JavaScript [**20**] es un lenguaje de programación interpretado originalmente diseñado como parte de los navegadores web.

Se ejecutaba por el lado del cliente, aunque existe una variante del lado del servidor (Server-side JavasScript, o SSJS) permitiendo al navegador interactuar con el usuario, hacer peticiones asíncronas y modificar el contenido de las páginas dinámicamente.

Es un lenguaje orientado a objetos, imperativo, funcional y basado en prototipos. Lo que lo califica como uno de los lenguajes más potentes. Es por ello que se ha ido extendiendo hacia otros usos entre numerosas plataformas.

Desarrollado originalmente por Brendan Eich, de Netscape, como un lenguaje ligero que complementase la maquina virtual de Java Sun Microsystems, que requería programadores profesionales. Su nombre oficial, con el lanzamiento de Netscape 2.0 en el año 1994, fue LiveScript, pero fue finalmente modificado en una revisión posterior. La nomenclatura produjo confusión por su similitud con Java, aspecto que favoreció su expansión resultando ser una importante medida de marketing.

En el año 1996, Microsoft incorporó JavaScript a su navegador Internet Explorer [**21**], cuya implementación renombró por JScript para evitar problemas con la marca.

Ese mismo año, Netscape propuso a Ecma International la estandarización de JavaScript, resultando en la aparición del ECMAScript. Actualmente existen cuatro ediciones, estando esta última aún en desarrollo. La tercera edición del ECMA-262, publicada en diciembre del 1999, es la versión actual de la mayoría de navegadores.

Uno de los usos más prácticos de JavaScript es su integración con numerosas herramientas, ofreciendo una sintaxis similar, lo que propicia su desarrollo. Unity script, un ejemplo de esta aplicación, es la que nos ha llevado a la realización de este proyecto.

Boo

Boo [**22**] es un lenguaje de programación orientado a objetos de propósito general para la Common Language Infraestructure. Está basado en tipos estáticos y tiene una sintaxis similar a Python. Entre sus características están los generadores, multimétodos, inferencia de tipos, macros y "closures".

Boo es de código libre, bajo la licencia MIT/BSD y es compatible con los frameworks .NET y Mono, habiendo módulos para integrar Boo con IDEs como Visual Studio. La compatibilidad con estos frameworks es por lo que se hace mención en este documento.

Cabe destacar el buen soporte que hay detrás de Boo, como suele suceder con estas tecnologías de código libre, existiendo numerosa documentación al respecto.

3.2 Entorno de desarrollo

El entorno de desarrollo de Unity está compuesto por un editor principal, desde el cual se gestionan los proyectos, y está apoyado en una serie de herramientas externas que abarcan todo tipo de funcionalidades, desde programación hasta contenido multimedia.

En este apartado se hará una descripción de los dos componentes principales, sobre los que se ha desarrollo el proyecto que atañe a este documento.

3.2.1 Unity editor

Mediante el editor de Unity se pueden gestionar todos los elementos y aspectos, tanto generales como específicos, de un juego. Dispone de numerosas herramientas visuales para el tratamiento de contenido, tales como un editor de animaciones o un generador de terreno. Además, la interfaz principal tiene la capacidad de poder ser configurada a gusto, incorporando o eliminando aquellos elementos que se deseen.

El editor de Unity ofrece en una única vista toda la información relevante para el desarrollo de un proyecto: la **jerarquía de ficheros**, donde se encuentran los archivos referentes a scripts, modelos, texturas, sonidos, etc.; el **inspector**, que muestra y permite editar propiedades de los elementos del proyecto o durante la propia ejecución del juego; la **vista de escena**, desde la cual se puede navegar a través del escenario en tiempo real. Resulta especialmente útil para comprobar que todo se está llevando a cabo acorde a las especificaciones. Finalmente, la **vista de juego**, una visualización integrada de la ejecución del juego, independiente de la plataforma final sobre la que se desplegará el proyecto. Esta última vista es fundamental para realizar simulaciones.

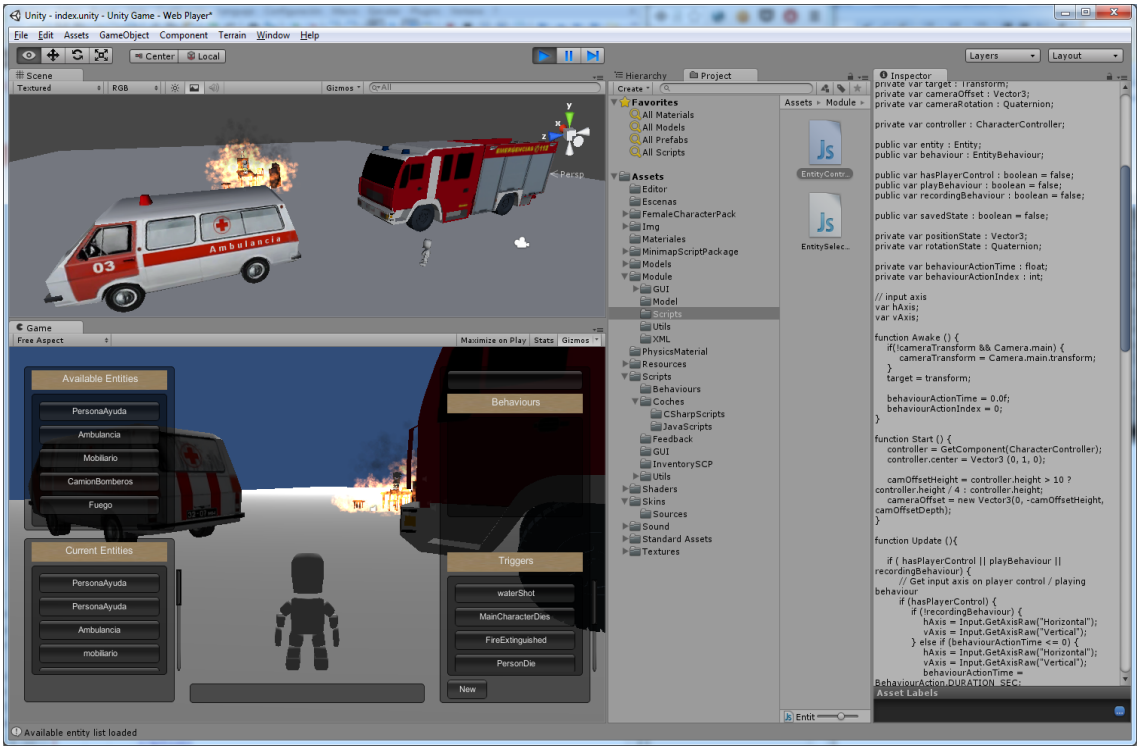


Ilustración 18 - Unity Editor: Interfaz

3.2.2 Mono Develop

El entorno de desarrollo integrado, o IDE, MonoDevelop [**23**], es una herramienta multiplataforma diseñada principalmente para el desarrollo sobre el lenguaje de programación C# y otros lenguajes basados en .NET. Permite portar aplicaciones .NET creadas con otras herramientas como Visual Studio y facilita la programación de aplicaciones sobre Linux, Windows y Mac OSX.

Una característica fundamental, que hace que este IDE haya sido clave para la realización de este proyecto, es su herramienta de depuración que incorpora, lo que facilita en gran medida la búsqueda de errores de implementación.

Otro aspecto importante es su integración el SDK de Unity, pudiendo realizar modificaciones sobre los scripts del proyecto sin precisar ninguna acción para mantener la sincronización de los ficheros. Su interfaz es totalmente configurable, pudiendo mostrar y ocultar las vistas que interesen o sobren, mediante un sistema de ventanas integradas. Merece la pena mencionar, que tiene un buen soporte para el autocompletado de código respecto de varios lenguajes, hecho que agiliza bastante la programación.

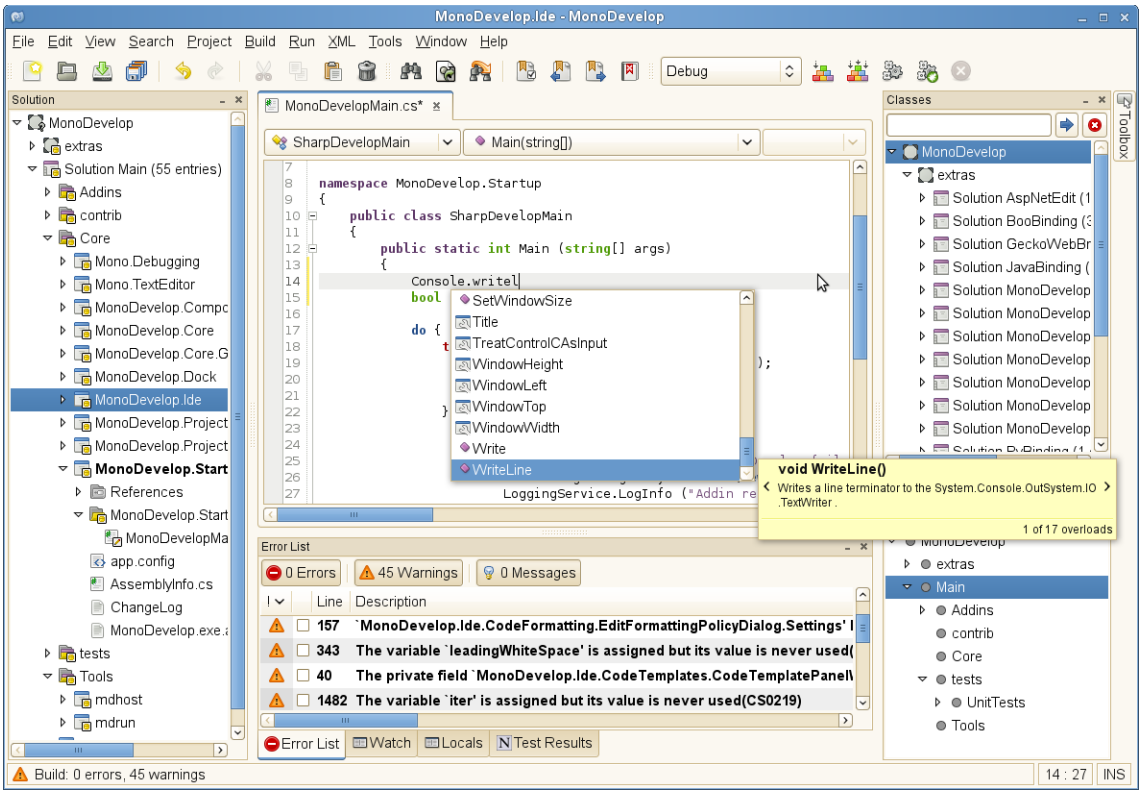


Ilustración 19 - MonoDevelop: Interfaz

3.3 Instalación del SDK de Unity

A continuación se ofrece un sencillo tutorial sobre la instalación de Unity para ordenador. No se incluyen los pasos para la instalación del SDK de Android o iOS ya que no están relacionados con este proyecto.

Este tutorial recoge los pasos que hacen falta para poner en funcionamiento el SDK en el sistema.

1. En primer lugar, vaya a la página http://unity3d.com/unity/download/ y obtenga la última versión del SDK de Unity (a fecha de hoy, la 4.1.2).

2. Es conveniente desinstalar previamente cualquier otra versión anterior de Unity instalada en el equipo, para evitar posibles incompatibilidades.

3. Asegúrese de que posee los últimos controladores de su tarjeta gráfica. Puede comprobarlo en http://www.nvidia.es/Download/index.aspx si posee una tarjeta Nvidia, o en http://support.amd.com/us/gpudownload/Pages/index.aspx si posee una Amd.

4. Si va a trabajar con DirectX11 (Windows únicamente), compruebe que dispone de la última versión de las librerías entrando en la página de soporte de Microsoft http://www.microsoft.com/en-us/download/default.aspx

5. Efectúe la instalación del SDK de Unity desde el fichero descargado en el punto 1. El proceso puede tardar unos minutos.

6. Compruebe que la aplicación se ha instalado correctamente ejecutándolo desde el acceso directo que ha aparecido en el escritorio.

7. Finalmente, introduzca la información referente a la licencia de uso, si lo precisa. Recuerde que la versión gratuita no ofrece limitaciones de funcionalidad, sin embargo no permite el acceso a numerosas librerías y algunas herramientas que simplifican el desarrollo.

4. ANÁLISIS FUNCIONAL

En esta sección se realizará un análisis de las funcionalidades de la aplicación, con la ayuda de un modelo de casos de uso. Este proceso se ha llevado a cabo a través de una serie de entrevistas con el tutor del proyecto, encargado del desarrollo. En dichas reuniones se han ido definiendo las distintas funcionalidades, en base a los procesos de investigación tratados en la sección anterior.

4.1 Visión general de la aplicación

El enfoque que se ha mantenido durante todo el ciclo de vida del proyecto ha sido permitir que el educador, diseñador en ámbitos generales, especifique los comportamientos que el NPC lleve a cabo, bien tomando el control de la entidad y registrándolo, o añadiendo acciones manualmente, definiendo posteriormente el momento en que quiere que ejecute.

Antes de comenzar, deben describirse los elementos principales sobre los que trabaja la aplicación. Estos son entidades, comportamientos y eventos.

- Una **entidad** es un elemento con el cual el usuario u otros elementos del escenario de un juego pueden interactuar.
- Un **comportamiento** es un conjunto de instrucciones, o acciones simples, que una entidad de cierto tipo puede llevar a cabo. La ejecución de un comportamiento está condicionada al cumplimiento de las condiciones de un evento.
- Los **eventos** agrupan las distintas condiciones que deben cumplirse para que se apliquen una serie de consecuencias sobre el estado de cierta entidad, desencadenen otros eventos o se lance la ejecución de un comportamiento.

El objetivo del proyecto será ofrecer un servicio a través del cual los educadores puedan generar y reproducir comportamientos para personajes (entidades) no jugadores. Para ello se deberán desarrollar dos módulos distintos: NPCs GREP Behaviour Authoring y NPCs GREP Behaviour Player.

El módulo *NPCs GREP Behaviour Authoring* permitirá editar y definir comportamientos para los NPCs definidos en un determinado juego y asociarlos con eventos que dispararán su ejecución. Este módulo cargará inicialmente todos los elementos (entidades, eventos y comportamientos ya predefinidos) del juego que el usuario selecciona. A continuación, y a través de una interfaz gráfica sencilla, el usuario creará o eliminará instancias de las entidades de NPCs del juego, posicionándolas en diversos puntos del escenario.

Una vez localizadas las entidades comenzará el proceso de definición de sus comportamientos que podrá ser ejecutado en dos modos: directo e indirecto. El modo directo permitirá registrar los movimientos básicos del personaje en tiempo real. Esto significa que el usuario tomará el control de una entidad, y en tiempo real va realizando la secuencia de movimientos y acciones que conformarán el comportamiento de la entidad. Por otra parte, mediante el modo indirecto, el usuario podrá ir añadiendo movimientos y acciones al comportamiento de la entidad de forma manual, seleccionándolos desde un listado, o bien reordenarlos o eliminarlos según desee. Una vez terminada la definición del comportamiento de la entidad, el usuario podrá asociarlo con uno o varios eventos que determinarán cuándo deberá ejecutarse dicho comportamiento y cuándo detenerlo. En cualquier momento el usuario podrá además pre-visualizar los comportamientos que vaya definiendo, reproduciendo las acciones registradas en cada uno de ellos.

Por otro lado, el módulo *NPCs GREP Behaviour Player* se encargará de ejecutar los comportamientos definidos para cada entidad durante una partida. Este módulo es parte del subsistema de ejecución de eventos de la aplicación GREP, por lo que no se requiere de ninguna acción especial por parte del usuario para su activación.

4.2 Usuarios del sistema

Como se puede inferir, un actor es un elemento externo que se comunica con el sistema y que interactúa con el mismo. En lo que refiere a los módulos de este proyecto, se encuentran dos actores:

- El **profesor** lleva el rol de diseñador del juego. Se encarga de la ejecución del módulo de edición y de llevar a cabo la generación o modificación de los comportamientos, entidades y/o eventos, cuya motivación se centra en la creación de un juego como complemento o soporte a un proceso educativo. El diseño de un juego según el modelo GREP requiere de la definición de un escenario y de un conjunto de reglas de juego, que determinan cuáles son las entidades que forman parte del juego, las acciones que cada una de ellas puede realizar y determina cuáles estarán bajo control del jugador (Player Controlled Entities) y cuáles no (NPCs). De este modo, el profesor se encargará de especificar cuándo y de qué forma se irán ejecutando las acciones de los NPCs.

- El **alumno**, cuya interacción radica únicamente en la ejecución directa del juego. Las acciones y movimientos que lleve a cabo sobre las entidades que controle irán disparando la ejecución de los NPCs de manera transparente para el usuario, ya que se encuentra integrada como parte del GRE Player. No existen restricciones respecto a la edad o formación del alumno, por lo que recae sobre el profesor la responsabilidad de diseñar una experiencia de juego que se corresponda con las necesidades del perfil del alumno.

4.3 Funcionalidades del sistema

Para poder delimitar exactamente las funcionalidades que van a tener los módulos de definición de comportamientos de la aplicación GREP y, por lo tanto, establecer las directrices a seguir durante el diseño y la implementación del sistema es preciso realizar un modelo de casos de uso.

4.3.1 Modelo de casos de uso

Los casos de uso ofrecen una visión de la funcionalidad basada en la perspectiva de cada actor que interactúa con la aplicación. Resultan muy efectivos para observar las interacciones típicas entre los usuarios y el propio sistema, describiendo qué se puede hacer, sin entrar en detalles sobre el cómo lo hace.

A continuación se procede a describir detalladamente cada caso de uso, de modo que queden identificadas las interacciones entre el actor y el sistema, las condiciones que se precisan para que pueda darse dicho caso de uso, así como el flujo de interacción requerido para ser efectuado. Estas descripciones han sido desarrolladas de acuerdo a la plantilla mostrada en la Tabla 8 que se descompone en:

- **Identificador**: Código que identifica de manera única cada caso de uso. La nomenclatura se basa en el formato CU-XX, donde XX es un número correlativo incremental, respecto de cada caso de uso.
- **Nombre**: Nombre descriptivo de cada caso de uso.
- **Actor**: Nombre del actor que interactúa con el sistema mediante el caso de uso especificado.
- **Descripción**: Conciso detalle sobre el funcionamiento del caso de uso.
- **Flujo**: Breve enumeración de los pasos intermedios que precisa el caso de uso para ser ejecutado.
- **Precondiciones**: Condiciones previas que son requeridas para que el caso de uso pueda llevarse a cabo.
- **Postcondiciones**: Condiciones que se darán una vez finalizada la ejecución del caso de uso.

Identificador		Nombre	
Actor			
Descripción			
Flujo			
Precondiciones			
Postcondiciones			

Tabla 8 - Plantilla descriptiva de los casos de uso

Casos de uso del actor Profesor

La Ilustración 20 muestra un diagrama con los distintos casos de uso del actor profesor. Estos casos han sido agrupados en cuatro grupos de funcionalidades: funcionalidad básica, gestión de entidades, gestión de comportamientos y gestión de eventos.

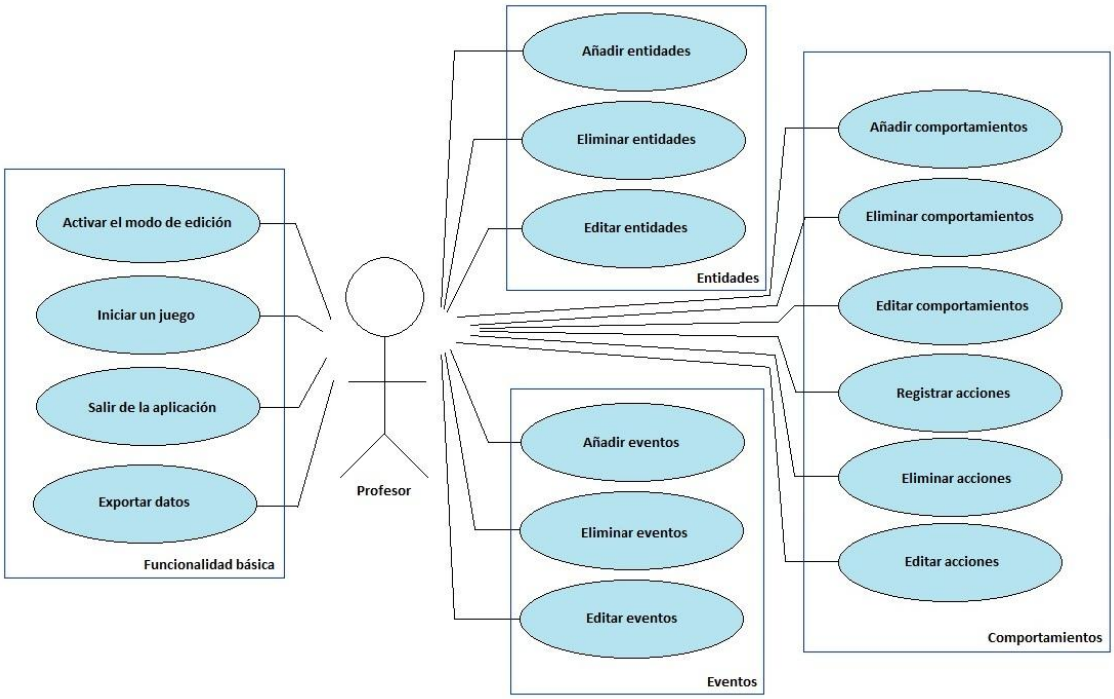


Ilustración 20 - Diagrama de casos de uso de los módulos NPCs GREP Behaviour: profesor

Funcionalidad básica

Se han definido cuatro casos de uso relacionados con la configuración y funcionalidad básica de la aplicación: activar el modo de edición, el cual arrancaría el modulo NPCs GREP Behaviour Authoring, iniciar el juego para el que se van a definir comportamientos, salir de la aplicación y exportar los comportamientos generados.

Identificador	CU-01	**Nombre**	Activar el modo de edición
Actor	Profesor		
Descripción	El usuario activa la funcionalidad del módulo de edición seleccionando el modo Edición.		
Flujo	1. El usuario lanza la aplicación. 2. Termina el proceso de carga.		
Precondiciones	El usuario debe encontrarse en el menú principal del juego, tras la carga inicial de datos.		
Postcondiciones	Se cargará el modo de edición al iniciar un juego.		

Tabla 9 - CU-01. Activar el modo de edición

Identificador	CU-02	**Nombre**	Iniciar un juego
Actor	Profesor		
Descripción	El usuario inicia la ejecución del juego.		
Flujo	1. Lanzar la aplicación 2. Esperar a que el proceso de carga termine		
Precondiciones	Tener acceso a la aplicación.		
Postcondiciones	Comienza la ejecución del juego según el modo elegido.		

Tabla 10 - CU-02. Iniciar un juego

Identificador	CU-03	**Nombre**	Salir de la aplicación
Actor	Profesor		
Descripción	El usuario finaliza la ejecución de la aplicación.		
Flujo	1. Se hace “click” sobre el botón de salida del juego.		
Precondiciones	La aplicación debe encontrarse en ejecución		
Postcondiciones	La ejecución de la aplicación es finalizada.		

Tabla 11 - CU-03. Salir de la aplicación

Identificador	CU-04	Nombre	Exportar datos
Actor	Profesor		
Descripción	Permite al usuario hacer persistentes los cambios efectuados en el modo de edición.		
Flujo	1. Se selecciona el botón correspondiente para la exportación de los datos.		
Precondiciones	Debe haberse iniciado el juego en el modo de edición.		
Postcondiciones	Los cambios en el juego se hacen persistentes para su posterior uso.		

Tabla 12 - CU-04. Exportar datos

Gestión de Entidades

En este apartado se detallan los distintos casos de uso relacionados con la creación, eliminación y modificación de instancias de las entidades básicas del juego.

Identificador	CU-05	Nombre	Añadir entidades
Actor	Profesor		
Descripción	El usuario añade una nueva instancia de una entidad al escenario.		
Flujo	1. Se selecciona una entidad de la lista de entidades recuperadas del fichero XML que contiene la definición del juego. 2. Se hace “click” en el lugar del escenario donde se desee añadir dicha entidad.		
Precondiciones	Debe haberse seleccionado previamente la entidad a añadir, de la lista de entidades disponibles.		
Postcondiciones	La nueva instancia de la entidad es generada sobre el escenario y se añade al listado global de entidades, con sus parámetros por defecto.		

Tabla 13 - CU-05. Anadir entidades

Identificador	CU-06	Nombre	Eliminar entidades
Actor	Profesor		
Descripción	El usuario elimina del escenario una instancia de una entidad concreta.		
Flujo	1. Se selecciona una entidad del listado global de entidades. 2. Se hace “click” sobre el botón “Eliminar”.		
Precondiciones	Debe haberse seleccionado previamente la entidad a eliminar, de la lista global de entidades.		
Postcondiciones	La entidad seleccionada es eliminada del escenario y de la lista global de entidades.		

Tabla 14 - CU-06. Eliminar entidades

Identificador	CU-07	Nombre	Editar entidades
Actor	Profesor		
Descripción	El usuario modifica los parámetros de estado de una entidad.		
Flujo	1. Se selecciona una entidad del listado global de entidades. 2. Se hace “click” sobre el botón “Editar”. 3. Se modifican los atributos de la entidad a través de la interfaz.		
Precondiciones	Debe haberse seleccionado previamente la entidad a editar, de la lista global de entidades.		
Postcondiciones	Los parámetros de la entidad quedan modificados.		

Tabla 15 - CU-07. Editar entidades

Gestión de Comportamientos

Categoría que abarca los casos de uso que afectan directamente al control y gestión de los comportamientos.

Identificador	CU-08	Nombre	Añadir comportamientos
Actor	Profesor		
Descripción	El usuario añade un nuevo comportamiento para un tipo concreto de entidad.		
Flujo	1. Se hace “click” sobre el botón “Añadir” del recuadro de control de comportamientos.		
Precondiciones	Debe haberse seleccionado previamente una entidad, del listado global de entidades.		
Postcondiciones	El nuevo comportamiento es añadido al listado global de comportamientos para el tipo de entidad especificado.		

Tabla 16 - CU-08. Añadir comportamientos

Identificador	CU-09	Nombre	Eliminar comportamientos
Actor	Profesor		
Descripción	El usuario elimina un comportamiento para un tipo concreto de entidad.		
Flujo	1. Se selecciona un evento del listado global de comportamientos. 2. Se hace “click” sobre el botón “Eliminar”.		
Precondiciones	Debe haberse seleccionado previamente un comportamiento, del listado global de comportamientos.		
Postcondiciones	El comportamiento es eliminado del listado para ese tipo de entidad.		

Tabla 17 - CU-09. Eliminar comportamientos

Identificador	CU-10	Nombre	Editar comportamientos
Actor	Profesor		
Descripción	El usuario modifica los parámetros de un comportamiento		
Flujo	1. Se selecciona un comportamiento del listado global de comportamientos. 2. Se hace “click” sobre el botón “Editar”. 3. Se modifican los parámetros del comportamiento a través de la interfaz.		
Precondiciones	Debe haberse seleccionado previamente un comportamiento del listado global de comportamientos.		
Postcondiciones	Los parámetros del comportamiento quedan modificados.		

Tabla 18 - CU-10. Editar comportamientos

Identificador	CU-11	Nombre	Registrar acciones
Actor	Profesor		
Descripción	El usuario registra acciones para un comportamiento concreto. Puede hacerlo de manera automática (1) o manual (2)		
Flujo	(1) 1. Se hace “click” sobre el botón “Tomar control” para manejar la entidad. 2. Se hace “click” sobre el botón “Comenzar grabación” para registrar automáticamente las acciones. 3. Se hace “click” sobre el botón “Finalizar grabación” para detener el registro de acciones. (2) 1. Se hace “click” sobre el botón “Añadir acción” 2. Se selecciona una acción de la lista		
Precondiciones	Debe haberse seleccionado un comportamiento del listado global de comportamientos para un tipo de entidad y haber hecho “click” en el botón “Editar”.		
Postcondiciones	Las acciones son añadidas a la lista de acciones del comportamiento.		

Tabla 19 - CU-11. Registrar acciones

Identificador	CU-12	Nombre	Eliminar acciones
Actor	Profesor		
Descripción	El usuario elimina una acción del conjunto de acciones que definen un comportamiento.		
Flujo	1. Se selecciona una acción del listado de acciones. 2. Se pulsa en el botón “Eliminar” para eliminar la acción.		
Precondiciones	Debe haberse seleccionado una acción del listado de acciones de un comportamiento.		
Postcondiciones	La acción es eliminada del conjunto de acciones del comportamiento.		

Tabla 20 - Editar acciones

Identificador	CU-13	Nombre	Editar acciones
Actor	Profesor		
Descripción	El usuario modifica el orden de una acción dentro del conjunto de acciones que definen un comportamiento.		
Flujo	1. Se selecciona una acción del listado de acciones. 2. Se pulsa sobre los botones “Mover izq.” o “Mover der.” para recolocar la acción.		
Precondiciones	Debe haberse seleccionado una acción del listado de acciones de un comportamiento.		
Postcondiciones	El orden de la acción dentro del conjunto de acciones del comportamiento es modificado.		

Tabla 21 - Editar acciones

Gestión de Eventos

En este apartado se detallan los distintos casos de uso relacionados con la gestión de los eventos del juego.

Identificador	CU-14	Nombre	Añadir eventos
Actor	Profesor		
Descripción	El usuario añade un nuevo evento al juego.		
Flujo	1. Se hace “click” sobre el botón “Añadir” del recuadro de control de eventos. 2. Se rellenan los atributos del evento, las condiciones de activación y se introducen los identificadores de los comportamientos relacionados con el evento.		
Precondiciones	Encontrarse el modo de edición activo en la ejecución del juego.		
Postcondiciones	El nuevo evento es añadido al listado global de eventos.		

Tabla 22 - Añadir eventos

Identificador	CU-15	Nombre	Eliminar eventos
Actor	Profesor		
Descripción	El usuario elimina un evento del juego.		
Flujo	1. Se selecciona un evento del listado global de eventos. 2. Se hace “click” sobre el botón “Eliminar”.		
Precondiciones	Debe haberse seleccionado previamente un evento, del listado global de eventos.		
Postcondiciones	El evento es eliminado del listado global de eventos, y por lo tanto de la posterior ejecución del juego.		

Tabla 23 - Eliminar eventos

Identificador	CU-16	Nombre	Editar eventos
Actor	Profesor		
Descripción	El usuario modifica los parámetros de un evento, las condiciones que lo disparan y/o los comportamientos que se ejecutan como respuesta.		
Flujo	1. Se selecciona un evento del listado global de eventos. 2. Se hace “click” sobre el botón “Editar”. 3. Se modifican los parámetros, condiciones y comportamientos asociados.		
Precondiciones	Debe haberse seleccionado previamente un evento del listado global de eventos.		
Postcondiciones	Los parámetros del evento quedan modificados.		

Tabla 24 - Editar eventos

Casos de uso del actor Alumno

Dado que el modulo de integración de comportamientos en la ejecución del juego se efectúa de manera transparente para el usuario, el actor Alumno únicamente posee el caso de uso “Iniciar un juego”. Los casos de uso referentes a la ejecución del juego quedan fuera de las competencias de este proyecto.

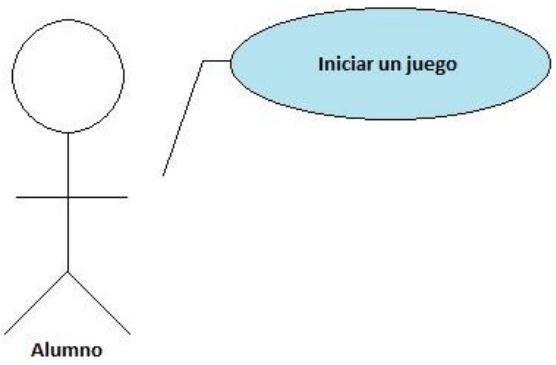


Ilustración 21 - Diagrama de casos de uso de los módulos *NPCs GREP Behaviour*: alumno

Identificador	CU-17	Nombre	Iniciar un juego
Actor	Alumno		
Descripción	El usuario inicia la ejecución del juego.		
Flujo	1. Lanzar la aplicación. 2. Esperar a que el proceso de carga termine.		
Precondiciones	Tener acceso a la aplicación.		
Postcondiciones	Comienza la ejecución del juego.		

Tabla 25 - Iniciar un juego

4.3.2 Análisis de Requisitos

Como parte del proceso de análisis del proyecto, en este apartado se tratará el análisis de los requisitos. A partir de los casos de uso, se describirá una serie de requisitos específicos, que recogen las necesidades o condiciones que deben cumplirse. Esto permite identificar las dependencias y/o conflictos de funcionalidad, garantizando que la fase de diseño tenga una base sólida.

Es importante destacar que, dado que el desarrollo de software es un proceso iterativo, los casos de uso no son la única fuente de obtención de requisitos, sino que ambos elementos se complementan recíprocamente.

Se distinguen dos grupos de requisitos: **funcionales** y **no funcionales**.

- **Requisitos funcionales (Capacidad)**: aquellos requisitos que describen el funcionamiento del sistema entran dentro de esta categoría. Definen el comportamiento interno del software que, simplificando, responden a la pregunta: ¿Qué hay que hacer?

- **Requisitos no funcionales (Restricción)**: son los requisitos que especifican criterios que aportan restricciones al sistema. Complementan a los funcionales respondiendo a ¿Cómo hay que hacerlo?

El análisis de requisitos tiene como base algunos estándares, en los cuales se detallan reglas para establecer una correcta especificación de los requisitos.

El estándar IEEE 830 Recommended Practice for Software Requirements Specifications declara una serie de características [**24**] que deben tenerse en consideración:

- **Ambigüedad**: Cada requisito debe tener una única interpretación posible, describiéndose con términos únicos que eviten confusión.
- **Verificabilidad**: Los requisitos deben poder ser verificados mediante procesos sencillos, permitiendo comprobar que la implementación satisface dicho requisito.
- **Consistencia**: No pueden existir conflictos entre requisitos, generando escenarios contradictorios.
- **Modificabilidad**: La especificación de los requisitos debe permitir ser modificada. Por lo que debe tiene que poseer una organización coherente, evitando la redundancia.
- **Completitud**: La especificación de requisitos debe ser completa, abarcando toda la funcionalidad del sistema.

Para definir los requisitos del sistema, se hará uso de una plantilla como la mostrada a continuación:

Identificador		**Título**	
Fecha		**Versión**	
Descripción			
Necesidad			
Estabilidad			
Prioridad			

Tabla 26 - Plantilla descriptiva de los requisitos

Detalle de la tabla:

- **Identificador**: Código único que identifica al requisito. La nomenclatura se basa en el formato RX-YY, donde X es la inicial del tipo de requisito (Restricción o Capacidad) y donde YY es un número correlativo incremental.
- **Título**: Título descriptivo de cada requisito.
- **Fecha**: Fecha de creación o última modificación del requisito.
- **Versión**: Identificador numérico de las variaciones del estado del requisito a lo largo de la vida del proyecto. Es un número incremental con inicio en 1.
- **Descripción**: Breve descripción del requisito. Debe ser coherente y evitar la ambigüedad.

- **Necesidad**: Puede tener los valores: esencial, deseable u opcional. Este campo muestra lo importante que es el requisito para la resolución del problema.
- **Estabilidad**: Puede tomar los valores: alta, media o baja. Indica lo variable que puede ser el requisito a lo largo del desarrollo del proyecto.
- **Prioridad**: Puede tomar los valores: alta, media o baja. Este campo indica la preferencia que tiene un requisito frente a otro en caso de existir conflicto.

Requisitos de capacidad

Identificador	RC-01	**Título**	Activación/desactivación del modo de edición.
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá activar y desactivar el modo de edición desde la interfaz del menú principal.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Media		

Tabla 27 - RC-01. Activación/desactivación del modo de edición

Identificador	RC-02	**Título**	Elección del juego
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario deberá poder elegir un juego del listado para su edición.		
Necesidad	Esencial		
Estabilidad	Alta		
Prioridad	Alta		

Tabla 28 - RC-02. Elección del juego

Identificador	RC-03	**Título**	Exportación de datos
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá exportar los datos del juego editado a través de la interfaz de edición.		
Necesidad	Esencial		
Estabilidad	Baja		
Prioridad	Alta		

Tabla 29 - RC-03. Exportación de datos

Identificador	RC-04	**Título**	Estado inicial del escenario
Fecha	18/01/2013	**Versión**	1
Descripción	El módulo de edición permitirá visualizar el escenario del juego cargado en su estado inicial.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Media		

Tabla 30 - RC-04. Estado inicial del escenario

Identificador	RC-05	**Título**	Navegación del jugador
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá navegar a través del escenario con los controles del teclado.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Media		

Tabla 31 - RC-05. Navegación del jugador

Identificador	RC-06	Título	Listado de entidades
Fecha	18/01/2013	Versión	1
Descripción	El módulo de edición mostrará un listado con todos los tipos de entidades definidos en el juego y las instancias de las mismas añadidas al escenario.		
Necesidad	Esencial		
Estabilidad	Media		
Prioridad	Alta		

Tabla 32 - RC-06. Listado de entidades

Identificador	RC-07	Título	Listado de comportamientos
Fecha	18/01/2013	Versión	1
Descripción	El módulo de edición mostrará un listado con todos los comportamientos ya definidos previamente para el juego.		
Necesidad	Esencial		
Estabilidad	Media		
Prioridad	Alta		

Tabla 33 - RC-07. Listado de comportamientos

Identificador	RC-08	Título	Listado de eventos
Fecha	18/01/2013	Versión	1
Descripción	El módulo de edición mostrará un listado con todos los eventos ya definidos previamente para el juego.		
Necesidad	Esencial		
Estabilidad	Media		
Prioridad	Alta		

Tabla 34 - RC-08. Listado de eventos

Entidades

Identificador	RC-09	**Título**	Adición de entidades
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá seleccionar entidades de la lista de entidades disponibles, y situarlas en el escenario mediante un “click” del ratón.		
Necesidad	Esencial		
Estabilidad	Media		
Prioridad	Alta		

Tabla 35 - RC-09. Adición de entidades

Identificador	RC-10	**Título**	Eliminación de entidades
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá eliminar entidades del escenario a través de un botón de la interfaz.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Alta		

Tabla 36 - RC-10. Eliminación de entidades

Identificador	RC-11	**Título**	Acceso a los atributos de una entidad
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá acceder al listado de atributos de una entidad mediante un botón de la interfaz.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Alta		

Tabla 37 - RC-11. Acceso a los atributos de una entidad

Eventos

Identificador	RC-12	**Título**	Adición de eventos
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá añadir nuevos eventos al juego, a través de un botón de la interfaz de edición.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Alta		

Tabla 38 - RC-12. Adición de eventos

Identificador	RC-13	**Título**	Eliminación de eventos
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá eliminar eventos del juego mediante un botón de la interfaz.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Alta		

Tabla 39 - RC-13. Eliminación de eventos

Identificador	RC-14	**Título**	Acceso a los parámetros de un evento
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá acceder al listado de parámetros de configuración de un evento a través de un botón de la interfaz.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Alta		

Tabla 40 - RC-14. Acceso a los parámetros de un evento

Comportamientos

Identificador	RC-15	**Título**	Adición de comportamientos
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá añadir nuevos comportamientos a través de un botón de la interfaz.		
Necesidad	Esencial		
Estabilidad	Alta		
Prioridad	Alta		

Tabla 41 - RC-15. Adición de comportamientos

Identificador	RC-16	**Título**	Eliminación de comportamientos
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá eliminar comportamientos mediante un botón de la interfaz.		
Necesidad	Esencial		
Estabilidad	Alta		
Prioridad	Alta		

Tabla 42 - RC-16. Eliminación de comportamientos

Identificador	RC-17	**Título**	Acceso a la interfaz de edición de comportamientos
Fecha	18/01/2013	**Versión**	1
Descripción	El usuario podrá acceder a la interfaz de edición de comportamientos desde un botón de la interfaz.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Media		

Tabla 43 - RC-17. Acceso a la interfaz de edición de comportamientos

Identificador	RC-18	Título	Registro manual de acciones
Fecha	18/01/2013	Versión	1
Descripción	El registro manual de acciones de comportamientos deberá poder realizarse mediante selección en un listado de acciones desplegable.		
Necesidad	Esencial		
Estabilidad	Alta		
Prioridad	Alta		

Tabla 44 - RC-18. Registro manual de acciones

Identificador	RC-19	Título	Registro automático de acciones en tiempo real
Fecha	18/01/2013	Versión	1
Descripción	El registro automático de acciones de comportamientos deberá poder realizarse en tiempo real, mientras dure activado el modo.		
Necesidad	Esencial		
Estabilidad	Alta		
Prioridad	Alta		

Tabla 45 - RC-19. Registro automático de acciones en tiempo real

Identificador	RC-20	Título	Activación del modo automático del registro de acciones
Fecha	18/01/2013	Versión	1
Descripción	El registro automático de acciones de comportamientos deberá poder activarse y desactivarse mediante un botón de la interfaz.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Media		

Tabla 46 - RC-19. Activación del modo automático del registro de acciones

Identificador	RC-21	Título	Control de una entidad
Fecha	18/01/2013	Versión	1
Descripción	El usuario podrá tomar el control de una entidad previamente seleccionada, mediante un botón de la interfaz.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Alta		

Tabla 47 - RC-21. Control de una entidad

Identificador	RC-22	Título	Eliminación de acciones de un comportamiento
Fecha	18/01/2013	Versión	1
Descripción	Las acciones registradas podrán ser eliminadas de la cola de acciones mediante un botón de la interfaz.		
Necesidad	Esencial		
Estabilidad	Media		
Prioridad	Alta		

Tabla 48 - RC-22. Eliminación de acciones de un comportamiento

Identificador	RC-23	Título	Reordenación de acciones de un comportamiento
Fecha	18/01/2013	Versión	1
Descripción	Las acciones registradas podrán ser reordenadas (izquierda o derecha) mediante dos botones respectivos de la interfaz.		
Necesidad	Deseable		
Estabilidad	Baja		
Prioridad	Media		

Tabla 49 - RC-23. Reordenación de acciones de un comportamiento

Requisitos de restricción

Identificador	RR-24	Título	Ejecución de comportamientos en el modo de juego.
Fecha	18/01/2013	Versión	1
Descripción	La ejecución de los comportamientos del juego deberá ser transparente para el usuario en el modo de juego.		
Necesidad	Esencial		
Estabilidad	Alta		
Prioridad	Alta		

Tabla 50 - RR-24. Ejecución de comportamientos en el modo de juego

Identificador	RR-25	Título	Estándar de formato en la exportación de los datos.
Fecha	18/01/2013	Versión	1
Descripción	La exportación de los datos deberá seguir el estándar de formato XML.		
Necesidad	Deseable		
Estabilidad	Alta		
Prioridad	Alta		

Tabla 51 - RR-25. Estándar de formato en la exportación de los datos

Identificador	RR-26	Título	Modificación de los atributos de una entidad
Fecha	18/01/2013	Versión	1
Descripción	Los atributos de una entidad deberán ser modificables desde la interfaz, mediante selectores y campos de texto.		
Necesidad	Esencial		
Estabilidad	Media		
Prioridad	Alta		

Tabla 52 - RR-26. Modificación de atributos de entidad

Identificador	RR-27	Título	Modificación de los parámetros de un evento
Fecha	18/01/2013	Versión	1
Descripción	Los parámetros de configuración de un evento deberán ser modificables desde la interfaz, mediante selectores y campos de texto.		
Necesidad	Esencial		
Estabilidad	Media		
Prioridad	Alta		

Tabla 53 - RR-27. Modificación de los parámetros de un evento

Identificador	RR-28	Título	Modos de registro de acciones
Fecha	18/01/2013	Versión	1
Descripción	Deberán existir dos modos de registro de acciones de comportamientos: modo manual y modo automático.		
Necesidad	Esencial		
Estabilidad	Alta		
Prioridad	Alta		

Tabla 54 - RR-28. Modos de registro de acciones

Identificador	RR-29	Título	Disponibilidad de comportamientos
Fecha	18/01/2013	Versión	1
Descripción	Los comportamientos generados deberán estar disponibles para todas las entidades del mismo tipo.		
Necesidad	Deseable		
Estabilidad	Media		
Prioridad	Media		

Tabla 55 - RR-29. Disponibilidad de comportamientos

Identificador	RR-30	Título	Idioma de la documentación
Fecha	21/01/2013	Versión	1
Descripción	El idioma de la documentación deberá ser el castellano, dado que su uso será meramente académico dentro del ámbito de una universidad española.		
Necesidad	Esencial		
Estabilidad	Media		
Prioridad	Alta		

Tabla 56 - RR-30. Idioma de la documentación

Identificador	RR-31	Título	Proceso de aprendizaje
Fecha	21/01/2013	Versión	1
Descripción	El proceso de aprendizaje que se requerirá deberá tener una duración corta.		
Necesidad	Esencial		
Estabilidad	Alta		
Prioridad	Alta		

Tabla 57 - RR-31. Proceso de aprendizaje

4.3.3 Diagrama de Actividad

La Ilustración 22 muestra un diagrama que tiene como finalidad ofrecer una visión general del flujo de acciones que el usuario del módulo de edición *NPCs GREP Behaviour* realizará durante la definición de una serie de comportamientos para los NPCs de un determinado juego.

Tal y como se muestra en la figura, durante la carga inicial se obtienen los datos del servidor, ofreciendo al usuario el listado de juegos disponibles. En este punto, el usuario puede activar pulsar el botón del modo de edición para activar o desactivar dicho modo. En caso de activarlo, se realizará la carga de la interfaz de edición, y se instanciarán todas las entidades que componen el escenario. En caso opuesto, se iniciará la ejecución normal del juego.

Una vez cargado el escenario, la sencilla interfaz del modo de edición ofrecerá acceso a las funcionalidades de gestión de entidades, eventos y comportamientos.

El usuario podrá entonces añadir nuevas entidades al escenario, o seleccionar una ya disponible y eliminarla o editar sus propiedades. Para facilitar el uso de la herramienta, la dinámica es similar para la gestión de eventos y comportamientos: el usuario puede añadir un elemento nuevo, o seleccionarlo y editarlo o eliminarlo. Cada vez que el usuario seleccione una entidad determinada, el sistema mostrará el listado de comportamientos asociado a la misma.

La edición de estos comportamientos se efectúa mediante el registro y/o eliminación de acciones, bien de forma manual (seleccionándolas de un listado) o automáticamente (tomando el control de la entidad y ejecutando los movimientos y acciones directamente).

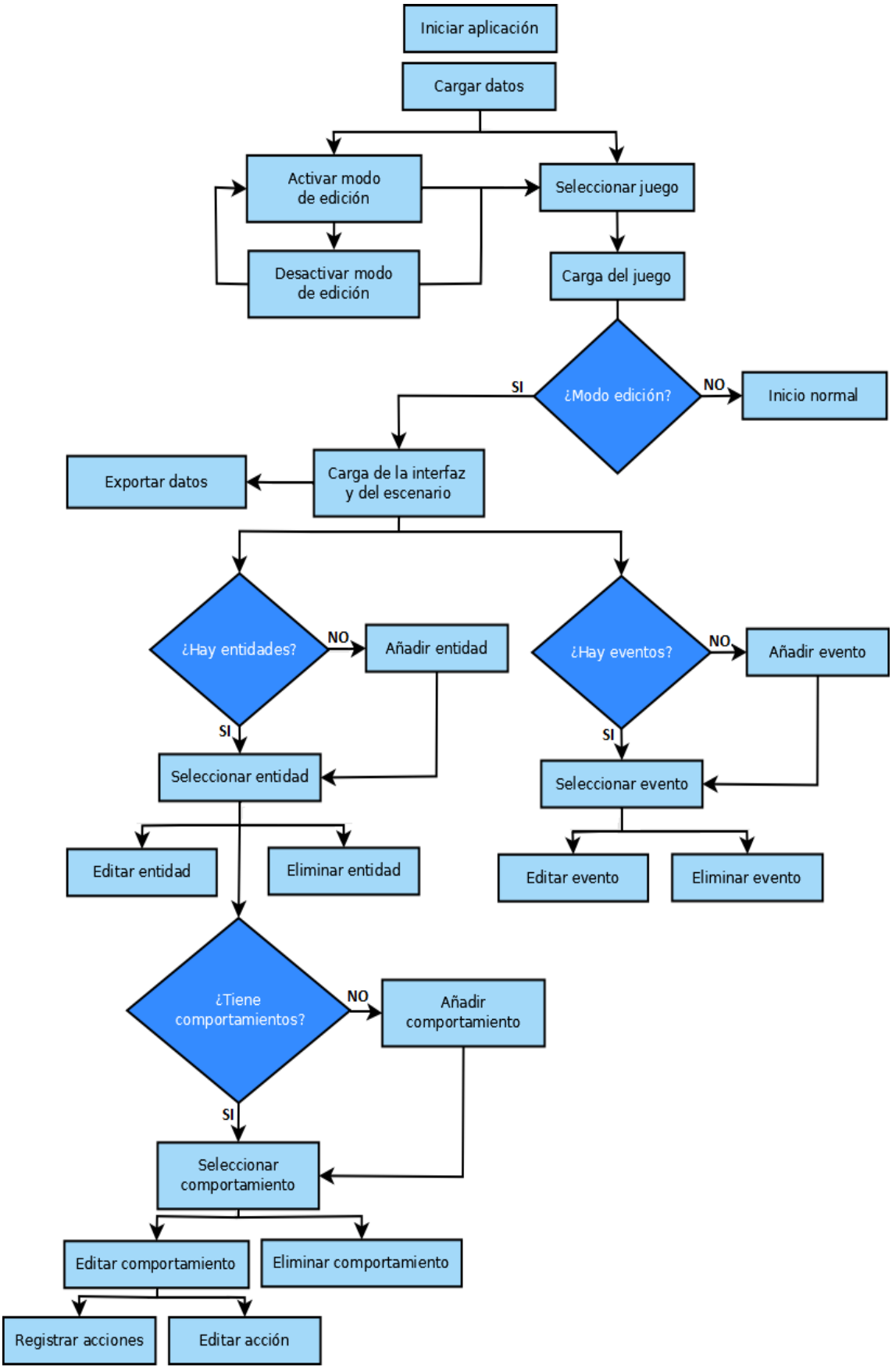


Ilustración 22 - Diagrama de Actividad

5. DISEÑO

En esta sección se especificará la arquitectura de cada módulo del sistema, el modelo de datos y el diseño inicial de la interfaz de usuario.

5.1 Arquitectura

La arquitectura de la aplicación GREP se basa en la interpretación de descripciones de juegos educativos a través de ficheros XML y la generación automática de los escenarios en tres dimensiones. Estas descripciones siguen el esquema del modelo GRE tratado anteriormente, donde se especifican los componentes de las reglas y del escenario de manera independiente. La Ilustración 23 e Ilustración 24 recogen este proceso relacionando, además, las interacciones de los módulos NPCs GREP Behaviour Player y NPCs GREP Behaviour Authoring y la aplicación GRE Player:

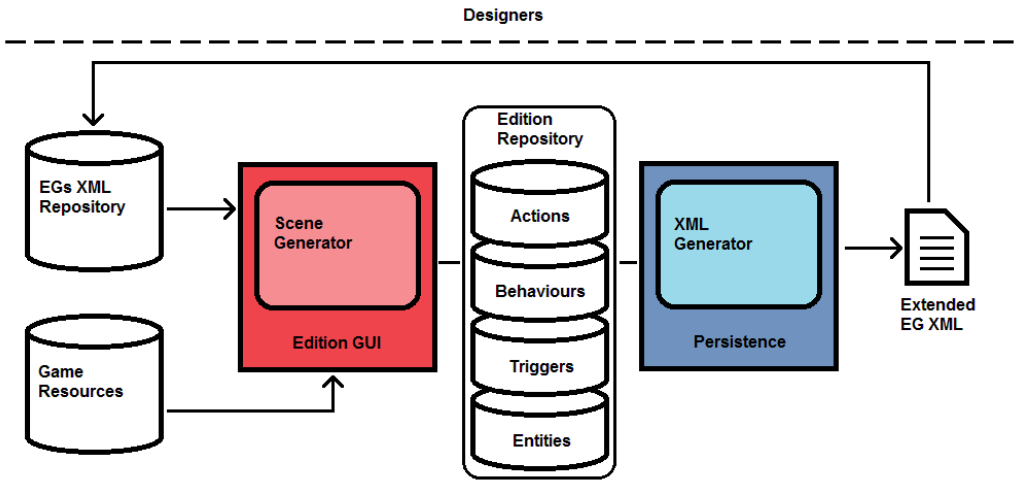


Ilustración 23 - Arquitectura del sistema - NPCs GREP Behaviour Authoring

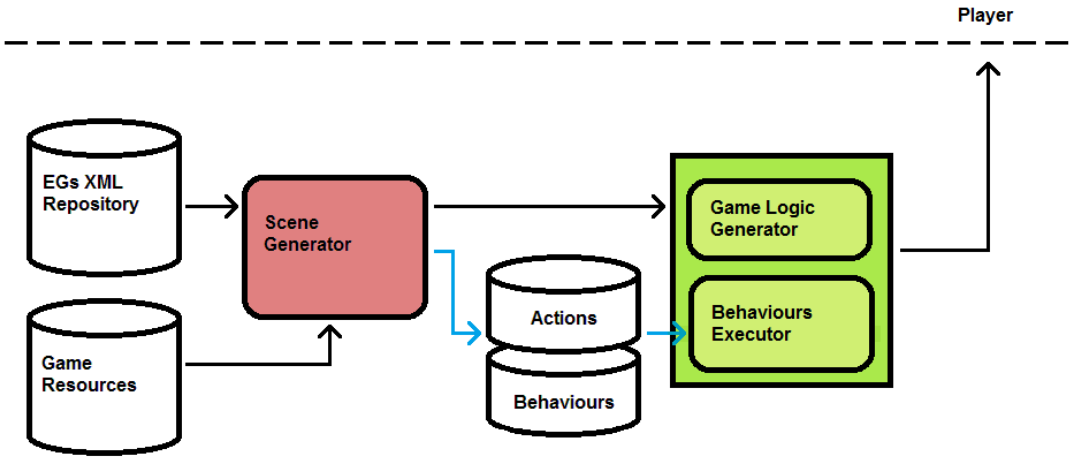


Ilustración 24 - Arquitectura del sistema - NPCs GREP Behaviour Player

5.1.1 Módulo GREP Behaviour Authoring

El módulo de edición se divide en los subsistemas gestores de **entidades**, **comportamientos**, **eventos** y **exportación de datos**.

Gestor de entidades	
Propósito	Su objetivo es ofrecer una serie de herramientas que permitan al usuario tratar las entidades que forman parte del juego.
Funciones	• Deberá proporcionar una interfaz básica con la información de las entidades de las que el juego predispone. • Deberá proporcionar una interfaz básica con la información de las entidades actualmente presentes en el escenario. • Deberá permitir al usuario añadir nuevas entidades al escenario. • Deberá permitir al usuario eliminar entidades del escenario. • Deberá permitir al usuario editar las propiedades de cada entidad.

Tabla 58 - Subsistema: Gestor de datos

Gestor de comportamientos	
Propósito	Su objetivo es ofrecer la funcionalidad capaz de generar y manejar comportamientos de las entidades del juego.
Funciones	• Deberá proporcionar una interfaz básica con la información de los comportamientos disponibles para cada tipo de entidad. • Deberá permitir al usuario añadir nuevos comportamientos para un tipo de entidad. • Deberá permitir al usuario eliminar comportamientos de un determinado tipo de entidad. • Deberá ofrecer al usuario una interfaz sencilla para la gestión de acciones. • Deberá permitir al usuario tomar el control de una entidad, como parte del proceso de registro automático de acciones. • Deberá proporcionar una interfaz que le permita al usuario añadir acciones manualmente al comportamiento. • Deberá permitir al usuario eliminar acciones del comportamiento. • Deberá permitir al usuario organizar el orden de las acciones de un comportamiento.

Tabla 59 - Subsistema: Gestor de comportamientos

Gestor de eventos	
Propósito	El objetivo de este subsistema es el de proporcionar la funcionalidad necesaria para que el usuario pueda manejar los eventos del juego.
Funciones	• Deberá proporcionar una interfaz básica con la información de los eventos configurados para el juego. • Deberá permitir al usuario añadir nuevos eventos al juego. • Deberá permitir al usuario eliminar eventos del juego. • Deberá permitir al usuario editar las propiedades de cada evento.

Tabla 60 - Subsistema: Gestor de eventos

Exportación de datos	
Propósito	Su finalidad consiste en ofrecer la funcionalidad necesaria para construir y exportar el modelo datos de la aplicación.
Funciones	• Deberá proporcionar una interfaz básica que permita al usuario hacer persistentes las modificaciones del juego. • Deberá exportar los datos según el estándar XML incorporando los cambios al fichero del juego editado.

Ilustración 25 - Subsistema: Exportación de datos

5.1.2 Módulo GREP Behaviour Player

El módulo de integración de comportamientos en la ejecución del juego está basado en la arquitectura provista por el proyecto GREP, extendiendo su funcionalidad para incorporar el motor de ejecución de comportamientos.

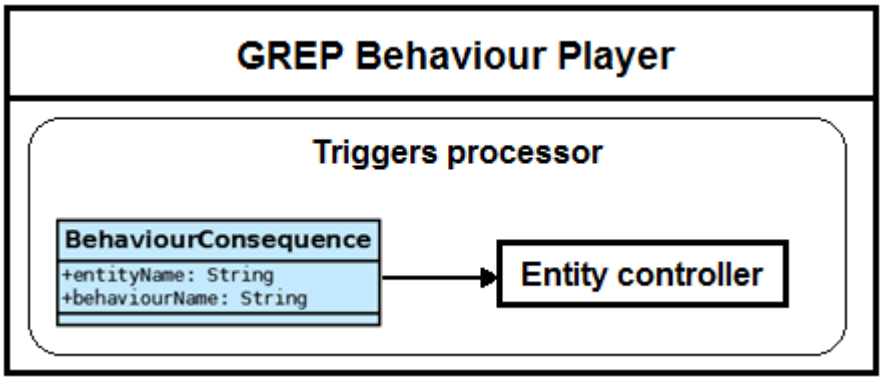


Ilustración 26 - Arquitectura: Componente GREP Behaviour Player

5.2 Modelo de datos

En este apartado se definirá el diseño del modelo de datos de los módulos del sistema GREP a través de modelos de clases. En su mayor parte, se presenta como una abstracción del XML del juego que facilita su lectura y modificación.

5.2.1 Atributo

La clase **Attribute** representa a cada atributo del juego. Éstos pueden requerir de un operador matemático empleado en el procesamiento de las condiciones de un evento.

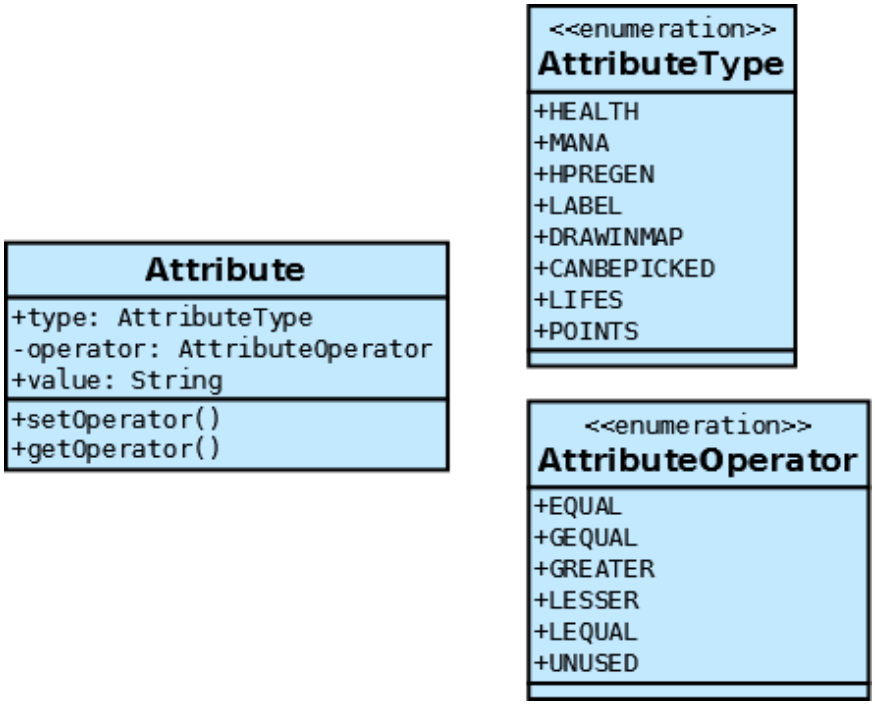


Ilustración 27 - Modelo de datos: Atributo

5.2.2 Entidad

Esta clase se puede considerar como un “envoltorio” que recoge los datos de estado de una entidad específica y lo asocia con la instancia del juego que lo representa, denominado “prefab”. Ofrece además la funcionalidad que gestiona la toma de control de dicha entidad.

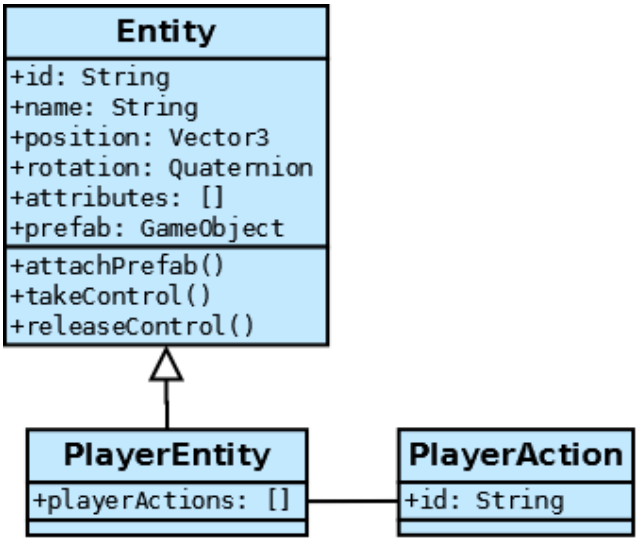


Ilustración 28 - Modelo de datos: Entidad

La clase **PlayerEntity** es una especialización de una entidad que representa al jugador. Posee un listado con las acciones que el jugador puede realizar durante la ejecución del juego.

5.2.3 Consecuencia

Las consecuencias de los eventos se tratan a través de este conjunto de clases: especializaciones de la clase **Consequence** que identifican cada tipo de consecuencia disponible en el juego.

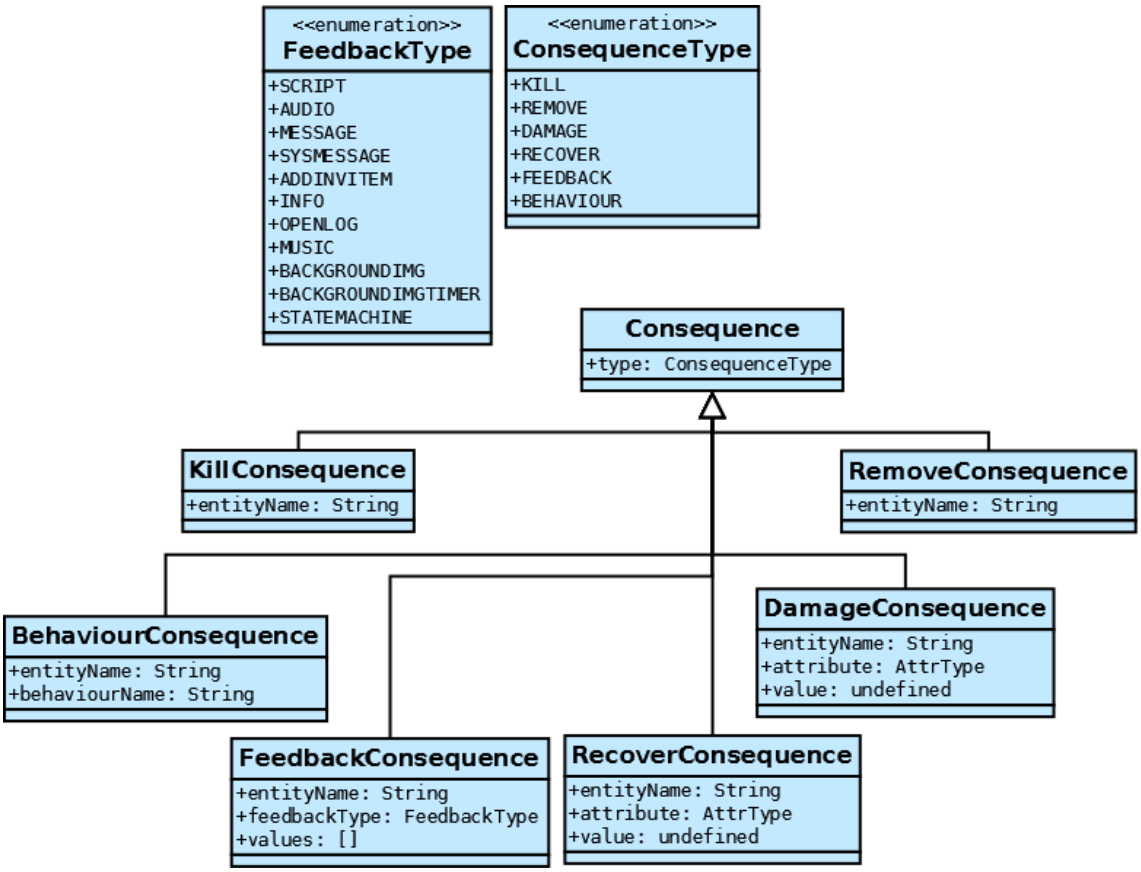


Ilustración 29 - Modelo de datos: Consecuencia

En el módulo de edición no ofrecen funcionalidad específica: su uso se limita a su representación, modificación y exportación de los datos al XML del juego. Sin embargo, en el módulo de ejecución se ha añadido el tratamiento de la clase **BehaviourConsequence** para incorporar la funcionalidad de reproducción de comportamientos a la aplicación GREP.

5.2.4 Comportamiento

Los comportamientos del juego se estructuran de manera sencilla, a través de las clases **EntityBehaviour** y **BehaviourAction**.

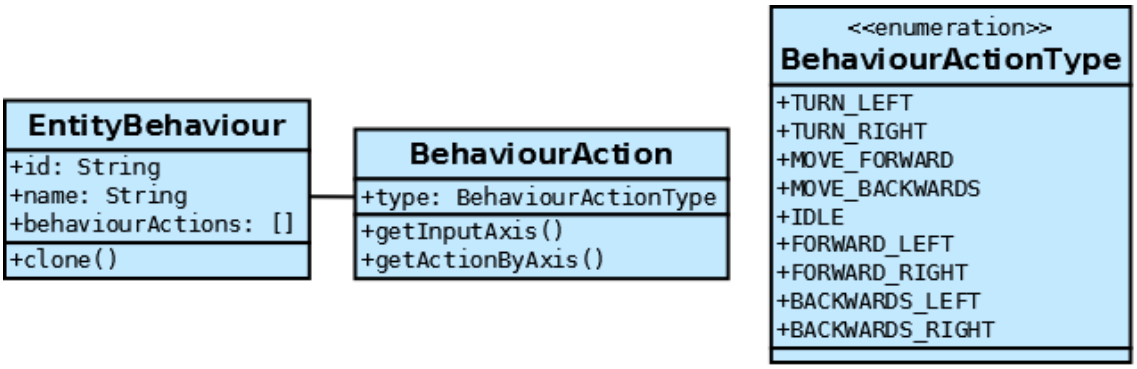


Ilustración 30 - Modelo de datos: Comportamiento

La clase **EntityBehaviour** asocia un tipo de entidad del juego con el listado de acciones que forma un comportamiento específico. Los tipos de acciones disponibles en el juego están definidos en la enumeración **BehaviourActionType**, sobre la cual se establecen los valores de los ejes de dirección (horizontal y vertical) a través de la clase **BehaviourAction** que realiza la transformación.

5.2.5 Evento

Estas clases representan los diferentes tipos de eventos disponibles en el juego. Como ya se ha mencionado anteriormente, cada evento dispone de una serie de condiciones según su tipo, cuya evaluación positiva desencadenará la ejecución de una serie de consecuencias.

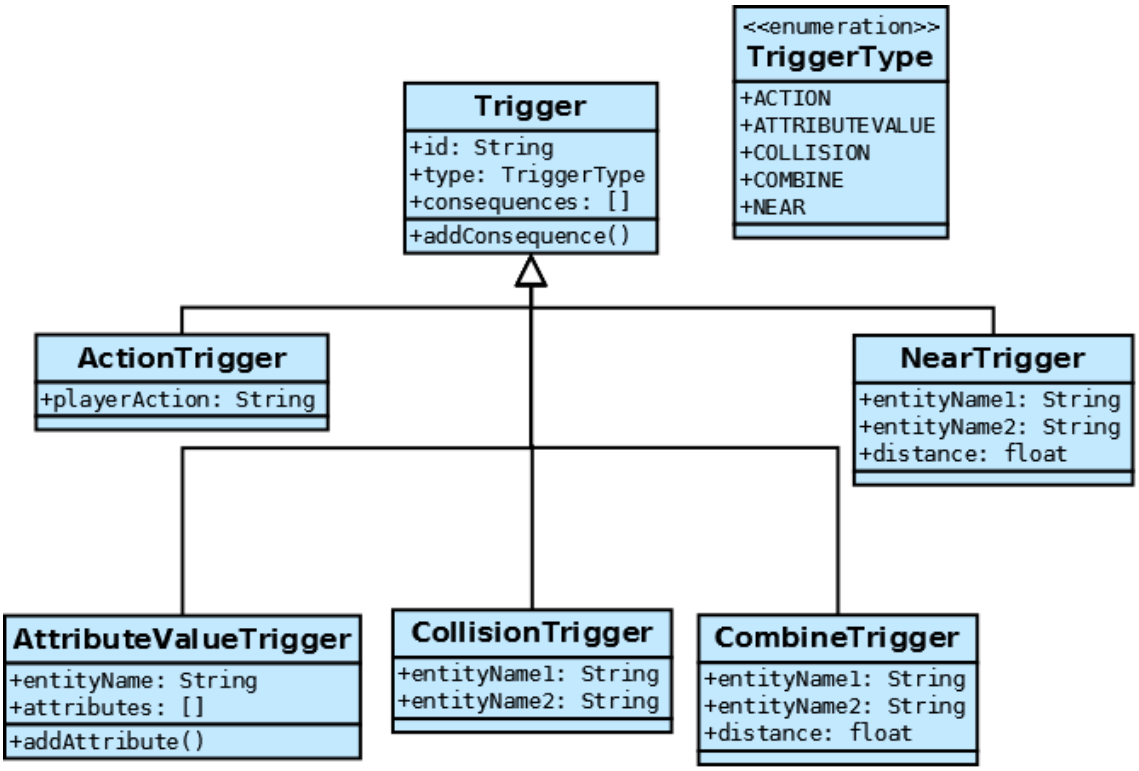


Ilustración 31 - Modelo de datos: Evento

En el módulo de edición, al igual que con el modelo de **consecuencia**, su uso se limita a su representación, modificación y exportación de los datos al XML del juego

5.2.6 Plataforma/Muro

Los “bloques” del juego que forman las plataformas y los muros del escenario están representados por la clase **Cube**, la cual recoge los datos sobre su identificación, situación y textura y encapsulan la funcionalidad que permite instanciarlos sobre el escenario.

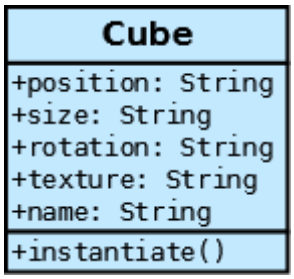


Ilustración 32 - Modelo de datos: Cubo

5.3 Interfaces

En este apartado se definirán las distintas interfaces gráficas del módulo de edición, representadas a través de unos bocetos, o “mockups”.

5.3.1 Pantalla principal GREP

La interfaz principal del juego deberá ser extendida para incluir un botón que permita la activación o desactivación del modo de edición:

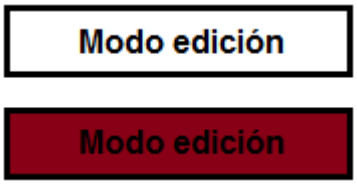


Ilustración 33 - Interfaz: Activador del modo de edición

El botón se encontrará en color gris cuando el modo está desactivado, y en color rojo cuando se active.

5.3.2 Interfaz del módulo de edición

La interfaz completa del modo de edición está basada en una serie de ventanas y diálogos emergentes que recogen toda la funcionalidad de edición del juego de manera sencilla e intuitiva.

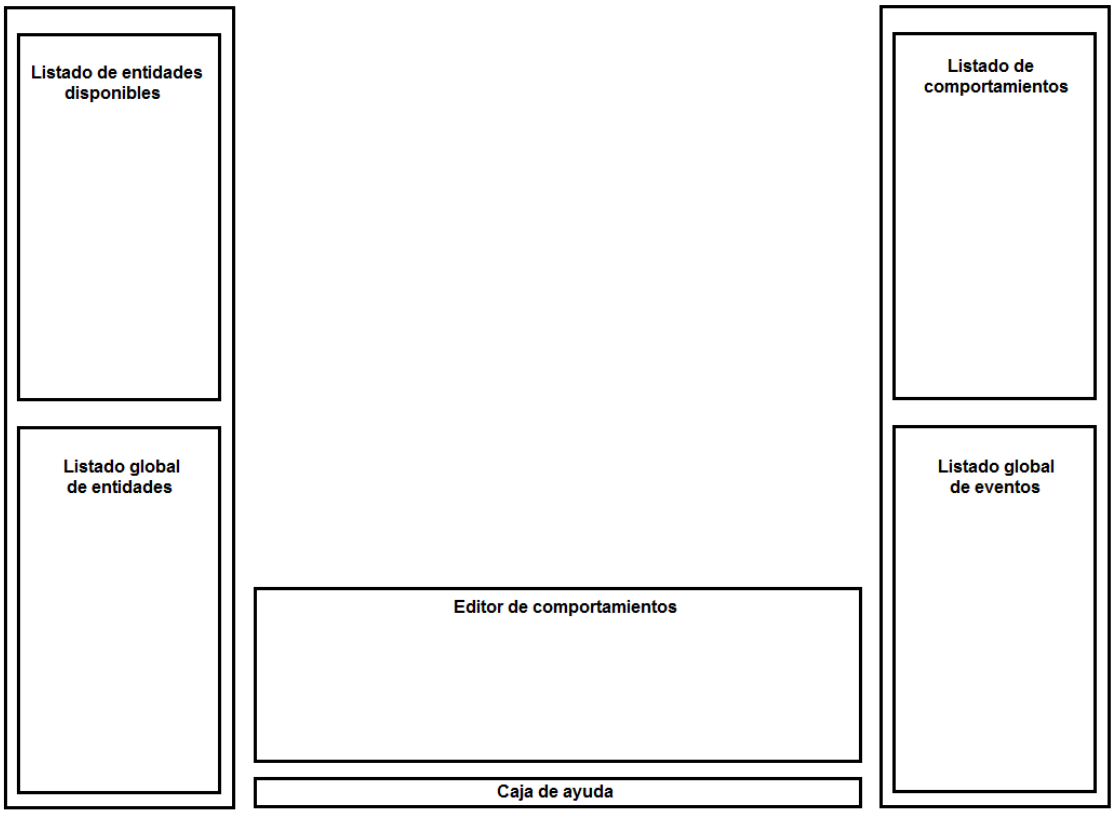


Ilustración 34 - Interfaces: Vista general del modo de edición

A continuación se procederá a describir cada elemento de manera individual.

- **Listado de entidades disponibles**

Este recuadro lista las entidades que el juego actual permite incorporar al escenario.

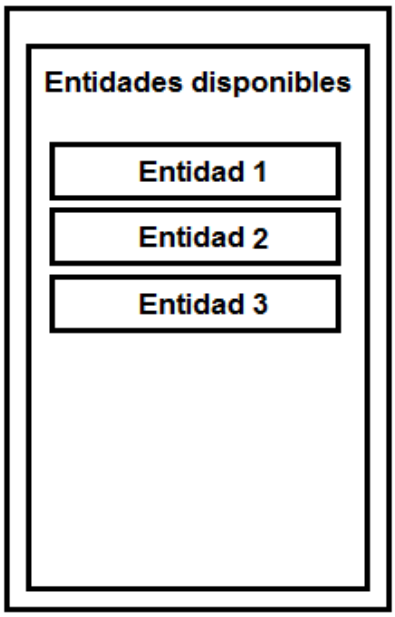


Ilustración 35 - Interfaz: Listado de entidades disponibles

- **Listado global de entidades**

A través de esta interfaz el usuario puede seleccionar y editar o eliminar una entidad concreta, previamente añadida al escenario mediante el listado de entidades disponibles, o como resultado de la carga inicial del juego. Además, incluye el botón con el cual se puede tomar el control de la entidad seleccionada, así como devolver el control al personaje principal.

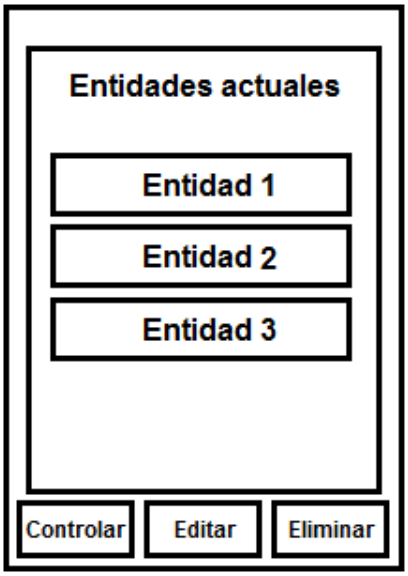


Ilustración 36 - Interfaces: Listado global de entidades

- **Listado de comportamientos**

Este recuadro lista los comportamientos disponibles para un determinado tipo de entidad (PersonaAyuda, en la ilustración). Incluye los botones para añadir y eliminar comportamientos, así como para renombrarlos.

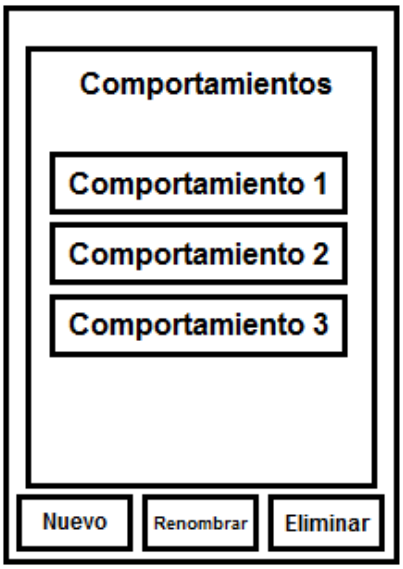


Ilustración 37 - Interfaces: Listado de comportamientos

- **Listado global de eventos**

Mediante esta interfaz el usuario puede añadir o seleccionar y eliminar o editar un evento del juego previamente añadido.

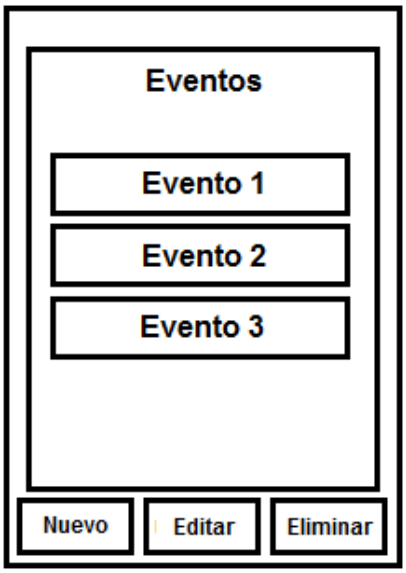


Ilustración 38 - Interfaces: Listado global de eventos

- **Editor de comportamientos**

Con la interfaz de edición de comportamientos, el usuario puede acceder al modo manual de registro de acciones mediante el botón “Añadir”, pudiendo seleccionar acciones de la cola de acciones y eliminarlas. Por otro lado, incluye el botón “Reproducir todo” que permite previsualizar el comportamiento, y el botón “Restaurar” que recoloca a la entidad asociada en su lugar original.

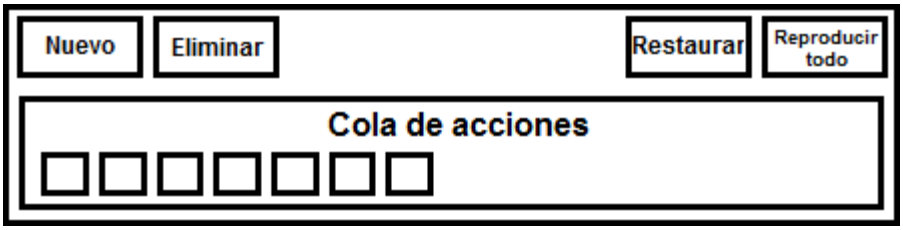


Ilustración 39 - Interfaces: Editor de comportamientos

- **Caja de ayuda**

Este recuadro, visible en todo momento, muestra ayuda dinámica al usuario, ofreciendo una pequeña descripción de la funcionalidad del elemento visual al que apunta con el puntero del ratón.

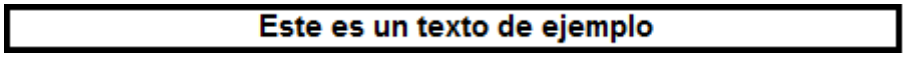


Ilustración 40 - Interfaces: Caja de ayuda

- **Editor de entidades**

El usuario es capaz de modificar los parámetros de una entidad a través de esta sencilla interfaz.

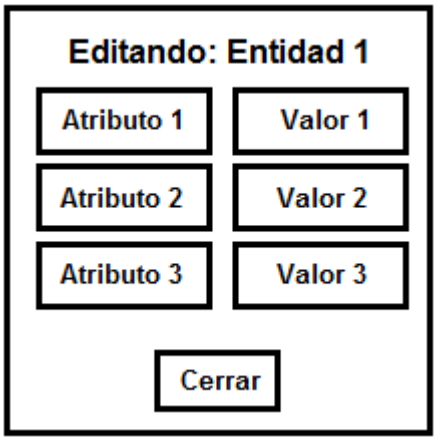


Ilustración 41 - Interfaces: Edición de entidades

- **Editor de eventos**

Se ha dispuesto una serie de elementos visuales para la edición de un evento: la vista principal, con los parámetros de identificación del evento; la vista de edición de los atributos que desencadenarán el evento; y la vista de edición de las consecuencias que se ejecutarán en tal caso.

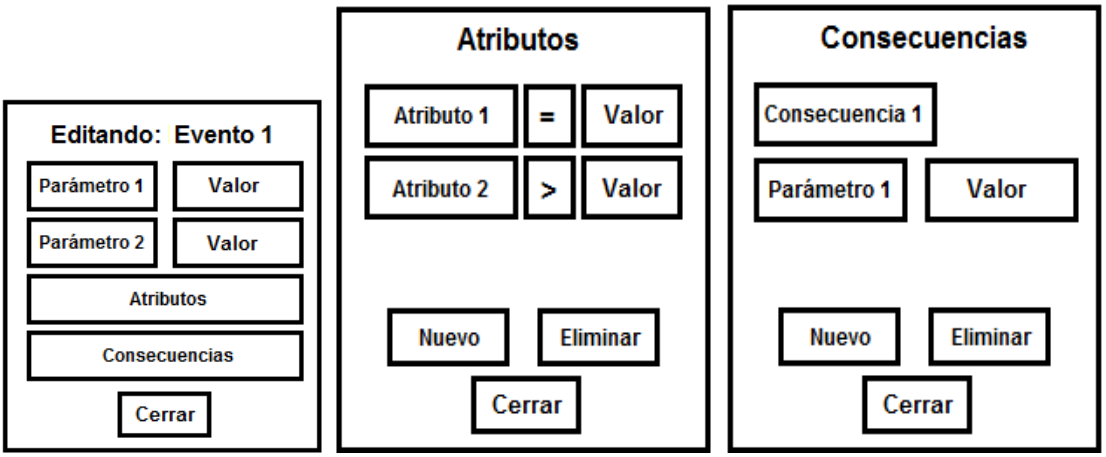


Ilustración 42 - Interfaces: Editor de eventos

El usuario podrá añadir, eliminar o modificar condiciones al evento, así como las consecuencias, mediante las tres sencillas interfaces descritas que incluyen campos de texto editables y listas desplegables.

- **Botones varios**

Botón de comienzo / fin del registro automático de acciones. Con este botón el usuario puede iniciar o detener la toma de acciones en tiempo real.

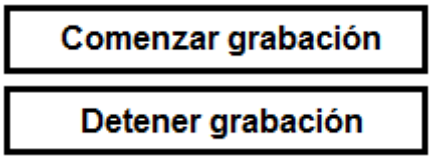


Ilustración 43 - Interfaces: Activador del registro automático de acciones

Botón de exportación de los datos. Al pulsar, se generará un fichero XML con los datos del juego modificados en el modo de edición.

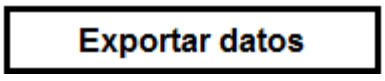


Ilustración 44 - Interfaces: Botón de exportación de datos

- **Elementos genéricos**

A continuación se muestran elementos genéricos de la interfaz.

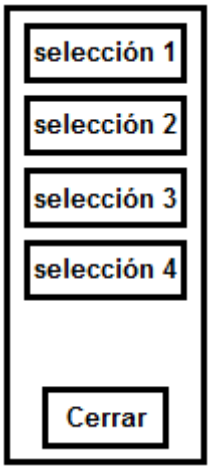


Ilustración 46 - Interfaces: Lista de selección

Las listas de selección permiten al usuario seleccionar un elemento de un conjunto global.

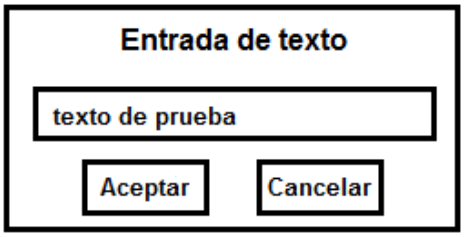


Ilustración 45 - Interfaces: Diálogo de entrada de texto

Los diálogos de entrada de texto permiten que un usuario introduzca un valor con el teclado, pudiendo cancelarlo en cualquier momento.

6. IMPLEMENTACIÓN

Tal y como se ha mencionado en apartados anteriores, la implementación se ha llevado a cabo mediante las herramientas ofrecidas por el Unity SDK: Unity Editor y MonoDevelop. Se ha empleado el lenguaje de programación UnityScript, basado en JavaScript, debido principalmente a que es uno de los lenguajes más potentes del mercado, cuya tendencia de uso en Unity ha promovido el desarrollo de numerosa documentación, un aspecto fundamental en este proyecto ya que se ha dedicado una importante carga de tiempo en el aprendizaje y uso de mencionadas herramientas.

En esta sección se describirá la estructura general de los XML, la estructura de ficheros del proyecto y la implementación de los módulos NPCs GREP Behaviour Authoring y NPCs GREP Behaviour Player.

6.1 Estructura de los XML del juego

En este apartado se identifican las tres secciones fundamentales de los XML con las que trabaja la aplicación GREP y se describen aquellos aspectos que han sido añadidos y/o modificados para incorporar la funcionalidad de los módulos NPCs GREP Behaviour.

En el ANEXO I: Ficheros XML, como ejemplos, se encuentran disponibles los dos XML correspondientes al juego “Bomberos”.

- **Muros/plataformas**: identificados por la etiqueta **floors**, indican dónde y cómo se sitúa cada “bloque” que forma el escenario del juego.
- **Entidades**: identificadas por la etiqueta **entities** contienen la información sobre el estado de cada entidad del escenario (tipo, situación…) y sus atributos correspondientes.
- **Eventos**: identificadas por la etiqueta **events**, representa los eventos disponibles del juego actual. Cada tipo de evento tiene una serie de condiciones específicas que deben cumplirse para desencadenar las acciones recogidas en las etiquetas **consequences**.

Para dar soporte a la funcionalidad de incorporación de entidades al escenario se ha añadido la entrada **availableEntities** con la dirección URL del XML donde se encuentra la información de las entidades disponibles del juego para su instanciación. Dicho XML tiene una estructura similar a la sección de Entidades provista por la aplicación GREP, con la salvedad de que los atributos se encuentran inicializados con valores por defecto.

Los comportamientos generados son añadidos en la entrada **behaviours**, donde se recoge cada acción de dicho comportamiento, asociado a un tipo de entidad específico. Las acciones similares consecutivas están representadas por una única entrada que incluye un parámetro groups para cuantificar el número de acciones y reducir la sobrecarga de información en el XML.

Por último, en los eventos ha sido preciso incorporar un nuevo tipo de consecuencia que desencadene la ejecución de un comportamiento para un determinado tipo de entidad, por lo que dispone de los parámetros entity-name y behaviour.

6.2 Jerarquía de ficheros

El IDE de desarrollo de Unity organiza los ficheros separando los recursos del proyecto del resto de componentes, permitiendo ser flexible en cuanto a la jerarquía o estructura a emplear. La Ilustración 47 muestra cómo se han distribuido los ficheros.

Todos los componentes del módulo NPCs GREP Behaviour se encuentran recogidos en la carpeta independiente “Module”, y aquellos recursos que han requerido adaptaciones se han incluido en una subcarpeta “Edition” dentro de los Resources.

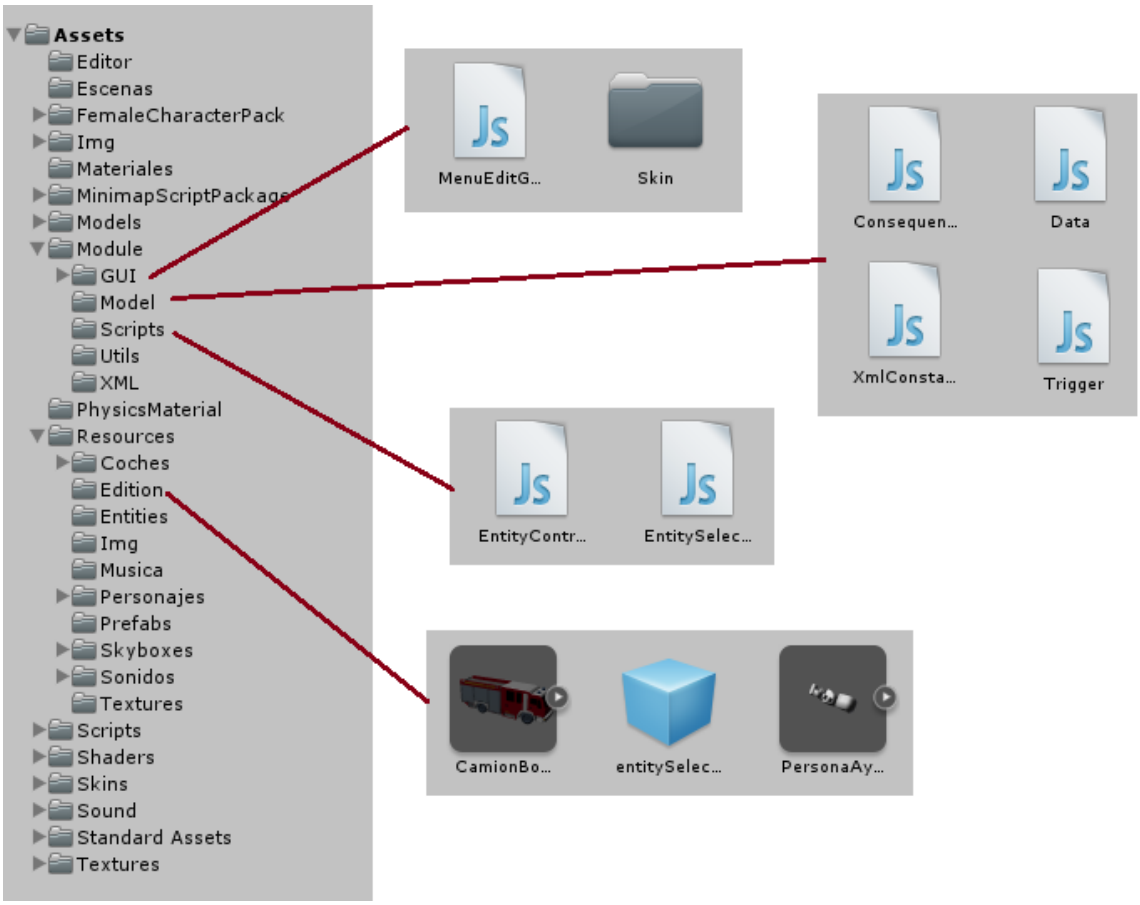


Ilustración 47 - Jerarquía de ficheros

6.3 NPCs GREP Behaviour Authoring: Edición

El módulo de edición contiene toda la interfaz de usuario dedicada a la modificación en tiempo de ejecución de entidades, comportamientos y eventos de un juego y la lógica del registro de acciones que conforman cada comportamiento.

6.3.1 Interfaz de edición

El tratamiento de interfaces en Unity se realiza a través del hilo de ejecución declarado mediante la función:

```
function OnGUI(){
```

Esta función es llamada numerosas veces por cada imagen generada (típicamente, 60 imágenes por segundo). En dicha función se deben incluir los elementos relacionados con la interfaz, evitando incorporar funcionalidad que pueda bloquear la ejecución, provocando “tirones” que afectan negativamente a la fluidez.

El modelo de datos se ha construido mediantes clases sencillas que representan e incorporan la funcionalidad básica para el tratamiento de los datos. Para ello, se realizan las modificaciones dentro del hilo de ejecución declarado como:

```
function Update() {
```

Esta función es llamada en torno a 1 vez por cada imagen generada.

```
function showAttributeListDialog() {
    var width = buttonWidth + hMargin * 2;

    var posX = screenWidth / 2 - width / 2 ;
    var posY = defaultDialogYPosition;

    var i = 0;
    var entityName = selectedConsequenceAttribute != null ?
                selectedConsequenceAttribute.entityName : currentTrigger.entityName;

    var showAttribute = function ( attributes : Array ) {

        for ( var attr : Attribute in attributes) {
            var button = Rect(posX + hMargin, posY + vMargin + (buttonHeight + vMargin) * i,
                              buttonWidth, buttonHeight);
            if ( GUI.Button(button, attr.type.ToString()) ) {
                var attribute : Attribute;
                if (attr.type == AttrType.CANBEPICKED) {
                    attribute = new AttributeCanbePicked("","","","");
                } else {
                    attribute = new Attribute(attr.type, "");
                }
                if (selectedConsequenceAttribute != null) {
                    selectedConsequenceAttribute.attribute = attr.type;
                    selectedConsequenceAttribute = null;
                } else {
                    (currentTrigger as AttrValueTrigger).addAttribute(attribute);
                }
            }
            i++;
        }
    };

    var attrList : Array;
    // show the current attribute list of the entity provided by entity-name trigger
    if (GameVars.playerEntity.id == entityName) {
        attrList = GameVars.playerEntity.attributes;
    } else {
        for ( var ent : Entity in GameVars.currentEntities ) {
            if (ent.id == entityName ){
                attrList = ent.attributes;
                break;
            }
        }
    }

    // container
    var height = vMargin * 2 + (buttonHeight + vMargin) * ( (attrList!= null ? attrList.Count : 0) + 1);
    var box = Rect(posX, posY, width, height);
    GUI.Box(box, "");

    showAttribute(attrList);

    var close = Rect(box.xMax - box.width / 2 - buttonWidth / 2, box.yMax - vMargin * 2 - buttonHeight,
                     buttonWidth, buttonHeight);

    if (GUI.Button(close, Strings.close)){
```

Ilustración 48 - Implementación: Interfaz (fragmento)

En este fragmento de código se puede observar la dinámica de generación de interfaces en Unity, mediante posicionamiento absoluto o relativo de cada componente visual dentro de la ejecución del hilo de interfaz. Cabe destacar la similitud con otras conocidas librerías como Swing, de Java. Si bien resulta ser bastante eficiente, es una metodología de desarrollo muy lenta que ha penalizado en cierta medida al desarrollo de este proyecto.

Se puede considerar que Unity se encuentra en una fase temprana de desarrollo con respecto a la generación de interfaces y deberían tratar este aspecto para futuras versiones.

6.3.2 Registro de acciones

El registro de acciones se ha realizado mediante una cola. Esto es, una estructura de datos caracterizada por que cada elemento es almacenado por orden de llegada, y accedido del mismo modo (Véase FIFO [**25**]).

El modo manual para registrar acciones se realiza a través de una ventana de diálogo que muestra un listado con las acciones disponibles. Por otro lado, el modo automático emplea marcas de tiempo que determinan el comienzo y fin de la acción que se está efectuando en ese momento.

```
function showNewBehaviourActionDialog() {

    var width = buttonWidth + hMargin * 2;

    var posX = screenWidth / 2 - width / 2 ;
    var posY = defaultDialogYPosition;

    var height = vMargin * 2 + (buttonHeight + vMargin) *
                    (System.Enum.GetValues(BehaviourActionType).length + 1);
    var box = Rect(posX, posY, width, height);
    GUI.Box(box, "");

    var i = 0;
    for ( var bAction : BehaviourActionType in System.Enum.GetValues(BehaviourActionType) ) {
        var button = Rect(posX + hMargin, posY + vMargin + (buttonHeight + vMargin) * i,
                            buttonWidth, buttonHeight);
        if ( GUI.Button(button, bAction.ToString()) ) {
            currentBehaviour.behaviourActions.Add(new BehaviourAction(bAction));
            newBehaviourActionDialog = false;
        }
        i++;
    }

    var close = Rect(box.xMax - box.width / 2 - buttonWidth / 2,
                    box.yMax - vMargin * 2 - buttonHeight, buttonWidth, buttonHeight);

    if (GUI.Button(close, Strings.close)){ newBehaviourActionDialog = false; }
}
```

Ilustración 49 - Implementación: Registro manual de acciones

La captura de acciones se lleva a cabo mediante la monitorización de las entradas de teclado, identificando los ejes horizontal y vertical, para el caso de los movimientos.

```
/**
*   Represents the actions within a behavior
*/
public class BehaviourAction {

    public static final var DURATION_SEC : float = .5f;

    public var type : BehaviourActionType;

    function BehaviourAction(type : BehaviourActionType) {
        this.type = type;
    }

    /**
    *   Parse a BehaviourActionType into an int array wich
    *   contains the values of the horizontal and vertical axes
    *
    *   return an int array with the axes values
    */
    function getInputAxis() : int[] {
        var res : int [];
        switch(type) {
            case BehaviourActionType.TURN_LEFT:
                res = [ -1, 0 ];
                break;
            case BehaviourActionType.TURN_RIGHT:
                res = [ 1, 0 ];
                break;
            case BehaviourActionType.MOVE_FORWARD:
                res = [ 0, 1 ];
                break;
            case BehaviourActionType.MOVE_BACKWARDS:
                res = [ 0, -1 ];
                break;
            case BehaviourActionType.IDLE:
                res = [ 0, 0 ];
                break;
            case BehaviourActionType.FORWARD_LEFT:
                res = [ -1, 1 ];
                break;
            case BehaviourActionType.FORWARD_RIGHT:
                res = [ 1, 1 ];
                break;
            case BehaviourActionType.BACKWARDS_LEFT:
                res = [ -1, -1 ];
                break;
            case BehaviourActionType.BACKWARDS_RIGHT:
                res = [ 1, -1 ];
                break;
        }
        return res;
    }

```

Ilustración 50 - Implementación: Captura de ejes de movimiento

6.4 NPCs GREP Behaviour Player: Reproducción

El módulo de reproducción es el encargado de procesar los eventos que desencadenarán las ejecuciones de los comportamientos generados en el módulo de edición. Al contrario que el módulo NPCs Behaviour Authoring, no se precisa de interfaz, ya que el proceso se realiza de manera transparente para el usuario.

6.4.1 Gestor de eventos

La plataforma GREP incorporaba de antemano la funcionalidad de procesamiento de eventos. Sin embargo, ha sido necesario modificar las funciones asociadas a dicho proceso para incorporar y tratar el modelo de datos sobre comportamientos.

```
855 /**
856 *   Makes the game object runs the specified behaviour
857 *   author: Javier Abellan Fernandez
858 */
859 public static function runBehaviour( go : GameObject, entityName : String, behaviourName : String ){
860     var component : EntityController = go.GetComponent(EntityController);
861 
862     if ( component != null && GameVars.behaviours.ContainsKey(entityName)){
863 
864         // Look for the specified behaviour and set to play
865         for (var behaviour : EntityBehaviour in GameVars.behaviours[entityName]) {
866 
867             if (behaviour.name == behaviourName) {
868                 component.behaviour = behaviour;
869                 component.playBehaviour = true;
870             }
871         }
872     }
873 }
874 
```

Ilustración 51 - Implementación: Gestor de eventos (Player)

6.4.2 Reproducción de comportamientos

La reproducción de movimientos viene dada por la incorporación de un script de control asociado a cada entidad del escenario del juego. Dicho script dispone de un "buffer" de acciones que especifican qué movimientos debe realizar durante la reproducción.

```
80      if (playBehaviour && behaviourActionTime <= 0) {
81          var axis = getNextBehaviourActionAxis();
82          hAxis = axis[0];
83          vAxis = axis[1];
84      }
85
86      // rotate entity on horizontal axis
87      target.Rotate(0, hAxis * rotateSpeed, 0);
88
89      // move entity forward / backward
90      controller.SimpleMove(target.TransformDirection(Vector3.forward) * moveSpeed * vAxis);
91
92      // manage entity animation
93      if ( animation != null ) {
94          var velocity = controller.velocity;
95          if(velocity.sqrMagnitude < 0.1) {
96              if ( animation["idle"] ) {
97                  animation.Play("idle");
98              }
99          } else {
100             if ( animation["run"] ) {
101                 animation["run"].speed = Mathf.Clamp(velocity.magnitude, 0.0, 1);
102                 animation.CrossFade("run");
103             }
104         }
105     }
```

Ilustración 52 - Implementación: Reproducción de comportamientos (fragmento1)

Cuando un evento dispara la ejecución de un comportamiento, la entidad asociada a dicho evento recibe la cola de acciones que debe reproducir, interpretando los ejes horizontal y vertical y aplicando las transformaciones en su posición y rotación correspondientes.

```
154 function getNextBehaviourActionAxis() : int[] {
155     if ( hasNextBehaviourAction() ) {
156         var action : BehaviourAction = behaviour.behaviourActions[behaviourActionIndex];
157         behaviourActionIndex++;
158
159         return action.getInputAxis();
160     } else {
161         playBehaviour = false;
162         behaviourActionIndex = 0;
163         return [ 0, 0 ];
164     }
165 }
```

Ilustración 53 - Implementación: Reproducción de comportamientos (fragmento2)

6.5 Exportación y despliegue

A continuación se va a detallar el proceso de exportación del proyecto, así como de los requisitos necesarios para su despliegue.

A través del Unity Editor se realiza la compilación del proyecto acorde a la plataforma de despliegue que se desee, recordando que se requieren licencias específicas para determinadas plataformas. En lo que se refiere a este proyecto, se realizará la exportación para reproducción en navegadores web.

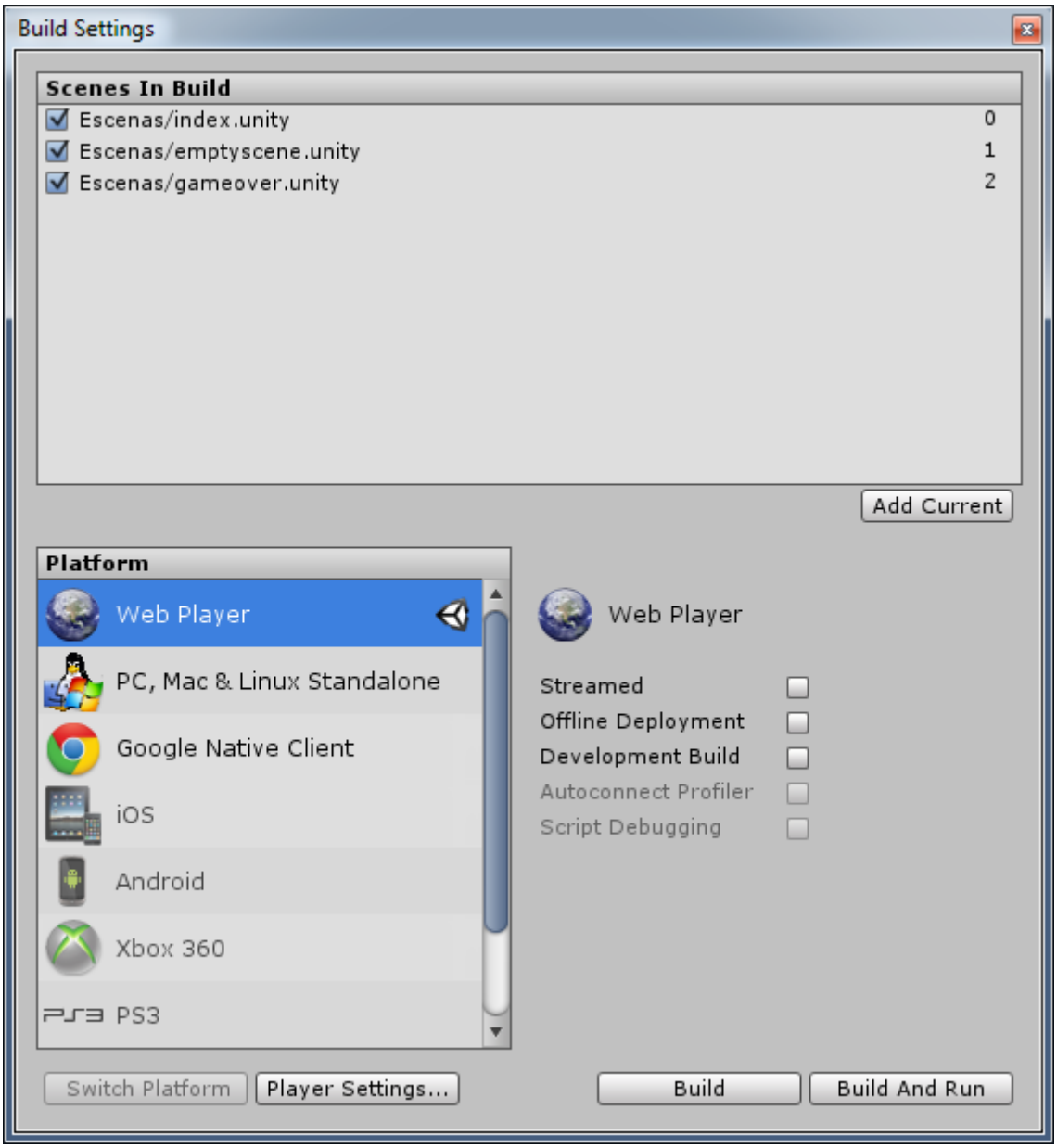


Ilustración 54 - Implementación: Opciones de exportación

Para la ejecución de la aplicación se requiere que el fichero compilado sea procesado mediante un componente web, por lo que debe estar alojado en un servidor. El proceso de compilación genera automáticamente un fichero HTML por defecto que puede ser modificado para incorporar información o mostrar una consola de depuración.

El único requisito para que un usuario pueda ejecutar la aplicación, ya que no requiere instalación previa, es que disponga del plugin ***Unity Web Player***, disponible para descarga en: http://unity3d.com/webplayer/

El plugin es compatible con los navegadores Internet Explorer, Chrome, Firefox, Safari y Opera, siendo recomendable su actualización frecuente debido a que se encuentra en continuo desarrollo.

7. PRUEBAS

En este capítulo se recoge la especificación de las tareas relacionadas con la verificación de cada funcionalidad del sistema.

La mayoría de las pruebas se han ido realizando durante la fase de implementación del proyecto, como comprobación de que el desarrollo se estaba llevando a cabo acorde a lo esperado. Además, se ha realizado una serie concreta de pruebas con la intención de verificar que el diseño de los módulos cumple con el objetivo fundamental del proyecto: permitir que usuarios sin un perfil técnico avanzado puedan definir comportamientos de NPCs.

7.1 Especificación del plan de pruebas

Para la especificación del plan de pruebas se ha hecho uso de una plantilla como la mostrada a continuación:

Identificador	
Objetivo	
Entrada	
Salida	
Resultado	

Tabla 61 - Plantillas descriptiva del plan de pruebas

Detalle de la tabla:

- **Identificador**: Código único que identifica a la prueba. La nomenclatura se basa en el formato PXX-ZZ, donde XX es un código con la inicial de la categoría de la prueba F si es Funcional, o UY si es de usuario, donde Y es el número de la prueba realizada; ZZ es un número correlativo incremental.
- **Objetivo**: Descripción de la finalidad de la prueba.
- **Entrada**: Condiciones, o flujo de acciones requeridas para cumplir el objetivo especificado.
- **Salida**: Resultado obtenido tras la ejecución del flujo de acciones de entrada.
- **Resultado**: Parámetro que califica el resultado de la prueba. Puede tomar los valores: CORRECTO o FALLIDO.

7.1.1 Pruebas funcionales

En este apartado se detallan las pruebas relacionadas con la funcionalidad del sistema.

Identificador	PF-01
Objetivo	Comprobar la incorporación de entidades al juego.
Entrada	Seleccionar una entidad del listado de entidades disponibles y hacer “click” sobre el escenario.
Salida	Aparece una nueva entidad sobre el escenario, en la posición indicada y se añade una entrada en la lista global de entidades.
Resultado	CORRECTO

Tabla 62 - PF-01. Comprobar la incorporación de entidades al juego.

Identificador	PF-02
Objetivo	Comprobar que una entidad es eliminada del juego
Entrada	Seleccionar una entidad del listado global de entidades y pulsar sobre el botón “Eliminar”
Salida	La entidad desaparece del escenario, así como su entrada del listado global de entidades.
Resultado	CORRECTO

Tabla 63 - PF-02. Comprobar que una entidad es eliminada del juego

Identificador	PF-03
Objetivo	Comprobar la modificación de los atributos de una entidad.
Entrada	Seleccionar atributos de distintos tipos de la entidad y realizar modificaciones en su contenido.
Salida	Los atributos son modificados satisfactoriamente.
Resultado	CORRECTO

Tabla 64 - PF-03. Comprobar la modificación de los atributos de una entidad.

Identificador	PF-04
Objetivo	Comprobar la incorporación de nuevos comportamientos de un determinado tipo.
Entrada	Seleccionar una entidad de un tipo concreto y pulsar sobre el botón “Añadir” del recuadro de control de comportamientos.
Salida	Aparece una nueva entrada en el listado de comportamientos únicamente para el tipo de entidad seleccionado previamente.
Resultado	CORRECTO

Tabla 65 - PF-04. Comprobar la incorporación de nuevos comportamientos de un determinado tipo.

Identificador	PF-05
Objetivo	Comprobar la eliminación de un comportamiento de un determinado tipo.
Entrada	Seleccionar una entidad de un tipo concreto, seleccionar un comportamiento del recuadro de control de comportamientos y pulsar sobre el botón “Eliminar”.
Salida	La entrada del listado de comportamientos desaparece el tipo de entidad seleccionado previamente.
Resultado	CORRECTO

Tabla 66 - PF-05. Comprobar la eliminación de un comportamiento de un determinado tipo.

Identificador	PF-06
Objetivo	Comprobar el registro de acciones para un comportamiento a través de la interfaz.
Entrada	Pulsar sobre el botón “Añadir” de la interfaz de edición de comportamientos y seleccionar varios acciones del listado.
Salida	Las acciones seleccionadas son añadidas a la línea temporal de acciones del comportamiento.
Resultado	CORRECTO

Tabla 67 - PF-06. Comprobar el registro de acciones para un comportamiento a través de la interfaz.

Identificador	PF-07
Objetivo	Comprobar el registro de acciones para un comportamiento mediante la toma de control de una entidad.
Entrada	Pulsar sobre el botón “Comenzar grabación”, realizar una serie de movimientos sobre el escenario durante unos segundos y pulsar sobre el botón “Finalizar grabación”.
Salida	Las acciones cometidas durante el proceso son añadidas a la línea temporal de acciones del comportamiento.
Resultado	CORRECTO

Tabla 68 - PF-07. Comprobar el registro de acciones para un comportamiento mediante la toma de control de una entidad.

Identificador	PF-08
Objetivo	Comprobar la incorporación de eventos al juego.
Entrada	Pulsar sobre el botón “Añadir” del recuadro de control de eventos.
Salida	Se añade una nueva entrada en el listado global de eventos.
Resultado	CORRECTO

Tabla 69 - PF-08. Comprobar la incorporación de eventos al juego.

Identificador	PF-09
Objetivo	Comprobar la eliminación de eventos del juego.
Entrada	Seleccionar un evento del listado global de eventos y pulsar sobre el botón “Añadir” del recuadro de control de eventos.
Salida	El evento seleccionado desaparece del listado global de eventos.
Resultado	CORRECTO

Tabla 70 - PF-09. Comprobar la eliminación de eventos del juego.

Identificador	PF-10
Objetivo	Comprobar la modificación de los parámetros de un evento.
Entrada	Seleccionar diferentes parámetros de un evento y realizar modificaciones en su contenido.
Salida	Los parámetros son modificados satisfactoriamente.
Resultado	CORRECTO

Tabla 71 - PF-10. Comprobar la modificación de los parámetros de un evento.

Identificador	PF-11
Objetivo	Comprobar que los datos son exportados acorde a las modificaciones realizadas y en el formato adecuado.
Entrada	Realizar modificaciones varias sobre el juego y pulsar sobre el botón “Exportar”.
Salida	Las modificaciones quedan correctamente reflejadas en el fichero XML correspondiente.
Resultado	CORRECTO

Tabla 72 - PF-11. Comprobar que los datos son exportados acorde a las modificaciones realizadas y en el formato adecuado.

Identificador	PF-12
Objetivo	Comprobar que los comportamientos generados en el modo de edición se reproducen adecuadamente en el módulo de ejecución.
Entrada	Iniciar el modo de juego y cumplir la o las condiciones del evento especificado en el modo de edición que desencadenará la ejecución de un comportamiento.
Salida	La entidad asociada al comportamiento realiza las acciones del comportamiento generado al cumplir las condiciones del evento.
Resultado	CORRECTO

Tabla 73 - PF-12. Comprobar que los comportamientos generados en el modo de edición se reproducen adecuadamente en el módulo de ejecución.

7.1.2 Pruebas de usuarios

En este apartado se detallan las pruebas llevadas a cabo por usuarios externos, con el fin de comprobar el nivel de usabilidad del sistema.

Usuario 1

Identificador	PU1-01
Objetivo	Comprobación del nivel de usabilidad del sistema.
Entrada	El usuario ejecutará la aplicación y realizará las acciones de añadir, eliminar y editar entidades.
Salida	El usuario se desenvuelve correctamente y encuentra con facilidad los menús que busca para realizar cada tarea.
Resultado	CORRECTO

Tabla 74 - PU-01. Primera comprobación del nivel de usabilidad del sistema.

Identificador	PU1-02
Objetivo	Comprobación del nivel de usabilidad del sistema.
Entrada	El usuario ejecutará la aplicación y realizará las acciones de crear, modificar y eliminar un evento.
Salida	El usuario se desenvuelve correctamente y encuentra con facilidad los menús que busca para realizar cada tarea.
Resultado	CORRECTO

Tabla 75 - PU-02. Segunda comprobación del nivel de usabilidad del sistema.

Identificador	PU1-03
Objetivo	Comprobación del nivel de usabilidad del sistema.
Entrada	El usuario ejecutará la aplicación y creará un comportamiento, realizando el registro de acciones.
Salida	El usuario hace una breve consulta del manual de usuario (Véase ANEXO II: Manual de usuario) tras lo cual realiza correctamente las acciones de creación y registro de acciones, tanto manual como dinámicamente.
Resultado	CORRECTO

Tabla 76 - PU2-02. Tercera comprobación del nivel de usabilidad del sistema.

Usuario 2

Identificador	PU2-01
Objetivo	Comprobación del nivel de usabilidad del sistema.
Entrada	El usuario ejecutará la aplicación y realizará las acciones de añadir, eliminar y editar entidades.
Salida	El usuario consulta el manual de usuario (Véase ANEXO II: Manual de usuario) tras lo cual realiza correctamente las tareas de añadir, eliminar y modificar los atributos de una entidad.
Resultado	CORRECTO

Tabla 77 - PU2-01. Primera comprobación del nivel de usabilidad del sistema.

Identificador	PU2-02
Objetivo	Comprobación del nivel de usabilidad del sistema.
Entrada	El usuario ejecutará la aplicación y realizará las acciones de crear, modificar y eliminar un evento.
Salida	El usuario se desenvuelve correctamente y encuentra con facilidad los menús que busca para realizar cada tarea.
Resultado	CORRECTO

Tabla 78 - PU2-02. Segunda comprobación del nivel de usabilidad del sistema.

Identificador	PU2-03
Objetivo	Comprobación del nivel de usabilidad del sistema.
Entrada	El usuario ejecutará la aplicación y creará un comportamiento, realizando el registro de acciones.
Salida	El usuario se desenvuelve correctamente y encuentra con facilidad los menús que busca para realizar cada tarea.
Resultado	CORRECTO

Tabla 79 - PU2-03. Tercera comprobación del nivel de usabilidad del sistema.

7.2 Observaciones y resultados

La importante carga de interfaces que precisa este proyecto ha requerido principalmente pruebas relacionadas con el uso y navegación entre ellas, siendo especialmente importantes las realizadas a través de usuarios sin conocimientos técnicos. Cabe destacar que apenas ha habido errores en la ejecución de la batería de pruebas, y que estos han servido para localizar y corregir tanto los fallos que atañen, como otros derivados de los mismos.

Las pruebas de usuario han sido especialmente esclarecedoras para confirmar el nivel de usabilidad del sistema. Es importante mencionar que los usuarios realizaron las pruebas en el orden indicado y sin tener conocimiento previo de la aplicación, con la excepción de poder consultar el manual de usuario (Véase ANEXO II: Manual de usuario).

El **usuario 1** comenzó muy atento la ejecución de cada prueba, encontrándose especialmente cómodo con la interfaz de menú. Realizó las pruebas PU1-01 y PU-02 sin mayor problema. Durante la ejecución de la prueba PU1-03 tuvo que hacer uso del manual de usuario apartado 5.5 Registrar acciones dinámicamente para localizar cómo iniciar el registro dinámico de acciones, tras lo cual finalizó la prueba satisfactoriamente.

"Se maneja todo fácilmente, aunque me perdí un poco para encontrar cómo iniciar el registro en tiempo real, pero luego es igual."

El **usuario 2** realizó en la primera prueba, antes de comenzar, una breve consulta al manual, concretamente al apartado 3.1 Interfaz de usuario tras lo cual no tuvo problemas para finalizar correctamente las pruebas PU2-01, PU2-02 y PU2-03.

"Es sencillo. No me habría hecho falta consultar el manual: cada sección se ve rápidamente. Es muy intuitivo"

8. GESTIÓN DEL PROYECTO

En este capítulo se analizan las tareas relacionadas con la planificación y presupuesto. El objetivo es especificar las diferentes fases que se han llevado a cabo y el coste que ha supuesto el desarrollo de este proyecto.

8.1 Fases del proyecto

Se identifican las siguientes fases:

- **Investigación**: Esta primera fase abarca todo el proceso de búsqueda de información para realizar un estudio sobre los diferentes motores gráficos actuales, así como de las diversas técnicas de generación y tratamiento de comportamientos de personajes no jugadores. De dicho estudio se obtendrán las bases para el posterior proceso de análisis y diseño.

- **Análisis**: En esta etapa se exploran en detalle las necesidades de la aplicación (casos de uso), identificando los problemas que debe solucionar (requisitos) y funcionalidades.

- **Diseño**: Esta fase trata la estructuración del proyecto, basada en los modelos obtenidos en el análisis, para especificar la estructura de la aplicación, identificando las partes y las decisiones técnicas tomadas.

- **Implementación**: Esta etapa está dedicada al proceso de aplicación y codificación de las directrices establecidas en las fases anteriores, que concluyen en la generación de los módulos *NPCs GREP Behaviour Authoring* y *NPCs GREP Behaviour Player*.

- **Pruebas**: En paralelo a la fase de implementación se lleva a cavo el conjunto de pruebas necesario para garantizar que el producto final satisface los requisitos acordados con el cliente en la fase de análisis.

- **Documentación**: Además de la documentación del propio código llevaba a cabo a lo largo de la fase de implementación, será necesario redactar dos documentos adicionales:

    - **Memoria**: Documento principal del proyecto.

    - **Manual de usuario**: Anexo con las instrucciones básicas de manejo de los módulos a nivel de usuario.

8.2 Planificación

Los diagramas de Gantt, mostrados a continuación, resumen la planificación llevaba a cabo en el proyecto, indicando las distintas fechas de inicio y de fin de cada fase, en sus estimaciones inicial y final. Los detalles sobre las tareas de cada fase se encuentran en los apartados 8.2.1 Planificación inicial del proyecto y 8.2.2 Planificación final del proyecto.

Para mejorar la legibilidad de los gráficos, se ha desglosado cada mes en cuatro semanas.

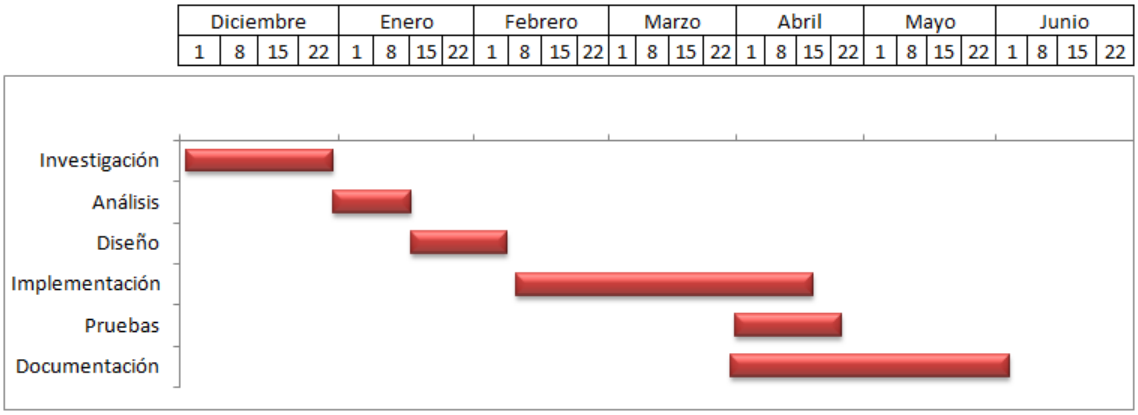


Ilustración 55 - Diagrama de Gantt: planificación inicial

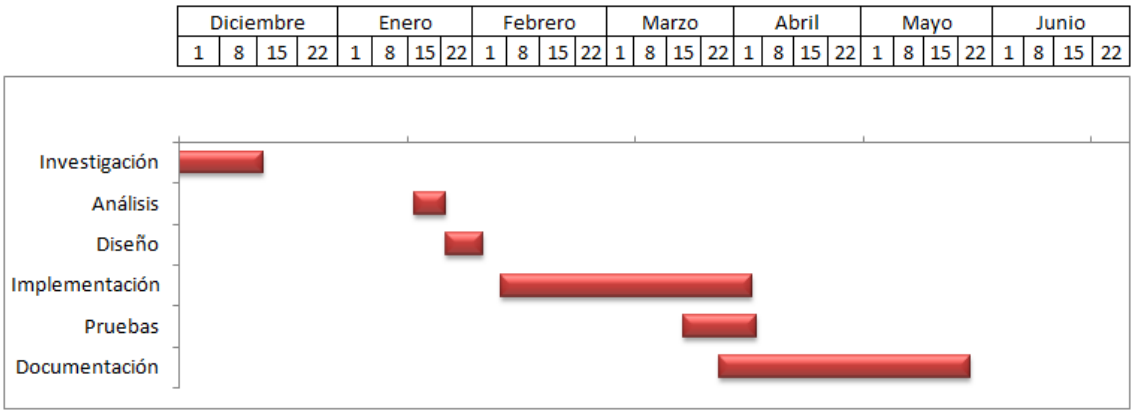


Ilustración 56 - Diagrama de Gantt: planificación final

Si bien se puede observar un lapso de tiempo vacío entre finales de diciembre y principios de enero debido a las festividades de la época, en general el desarrollo del proyecto se ha realizado acorde a las estimaciones previstas, resultando en un ahorro temporal de unos 39 días, tal y como se detalla en los apartados mostrados a continuación.

Tareas

La Tabla 80, enumera las distintas tareas en las que se subdivide cada fase del proyecto, las cuales llevan asociadas un código único identificativo.

Fase	Tarea	Código
Investigación	Estudio sobre motores gráficos	INV_01
	Estudio sobre técnicas de definición de comportamientos	INV_02
	Familiarización y aprendizaje sobre el entorno de desarrollo del motor gráfico elegido	INV_03
Análisis	Especificación de casos de uso	AN_01
	Especificación de requisitos	AN_02
Diseño	Arquitectura del sistema	DS_01
	Especificación de la interfaz de usuario	DS_02
Implementación	Preparación del entorno	DEV_01
	Preparación del modelo de datos	DEV_02
	Registro de acciones	DEV_03
	Reproducción de acciones	DEV_04
	Interfaces	DEV_05
Pruebas	Especificación de las pruebas	TEST_01
	Realización de simulaciones	TEST_02
Documentación	Memoria	DOC_01
	Manual de usuario	DOC_02

Tabla 80 - Listado de tareas del proyecto

8.2.1 Planificación inicial del proyecto

En la Tabla 81 se listan las distintas tareas mencionadas en el apartado anterior y se indican las fechas de inicio y fin, permitiendo visualizar de manera desglosada la estimación inicial de los días dedicados al proyecto.

Fase	Tarea	Fecha de inicio	Fecha de fin	Estimación inicial (días)
Investigación	INV_01	01/12/12	10/12/12	10
	INV_02	10/12/12	25/12/12	15
	INV_03	01/12/12	31/12/12	25
Análisis	AN_01	01/01/13	06/01/13	4
	AN_02	06/01/13	15/01/13	7
Diseño	DS_01	15/01/13	31/01/13	10
	DS_02	01/02/13	08/02/13	4
Implementación	DEV_01	08/02/13	09/02/13	1
	DEV_02	09/02/13	18/02/13	9
	DEV_03	22/02/13	15/03/13	19
	DEV_04	22/03/13	01/04/13	9
	DEV_05	01/04/13	17/04/13	12
Pruebas	TEST_01	01/04/13	10/04/13	10
	TEST_02	10/04/13	25/04/13	15
Documentación	DOC_01	28/03/13	15/05/13	50
	DOC_02	15/05/13	01/06/13	10
TOTAL				**210 días**

Tabla 81 - Planificación inicial del proyecto

En la planificación inicial se estimó una distribución aproximada de unas 2 horas diarias a lo largo de 210 días, lo que resulta en 420 horas de dedicación, acorde a lo esperado en un Trabajo de Fin de Grado.

8.2.2 Planificación final del proyecto

A continuación, la Tabla 82 recoge los datos sobre la planificación final del proyecto, realizando la estimación final sobre los días dedicados al proyecto.

Fase	Tarea	Fecha de inicio	Fecha de fin	Estimación final (días)
Investigación	INV_01	28/11/12	07/12/12	9
	INV_02	09/12/12	20/12/12	11
	INV_03	10/12/12	11/12/12	11
Análisis	AN_01	15/01/13	17/01/13	2
	AN_02	18/01/13	21/01/13	3
Diseño	DS_01	21/01/13	28/01/13	7
	DS_02	25/01/13	29/01/13	4
Implementación	DEV_01	08/02/13	09/02/13	1
	DEV_02	09/02/13	16/02/13	7
	DEV_03	16/02/13	15/03/13	27
	DEV_04	17/03/13	28/03/13	11
	DEV_05	25/03/13	05/04/13	11
Pruebas	TEST_01	16/03/13	20/03/13	4
	TEST_02	20/03/13	29/03/13	9
Documentación	DOC_01	25/03/13	22/05/13	45
	DOC_02	18/05/13	27/05/13	9
TOTAL				**171 días**

Tabla 82 - Planificación final del proyecto

Durante todo el desarrollo del proyecto, se ha pretendido cumplir con las fechas de finalización de cada tarea establecidas en la planificación inicial, habiendo obtenido de una manera global, resultados positivos.

Con los datos obtenidos en las planificaciones anteriores, se construye la Tabla 83, la cual recoge las estimaciones de tiempo inicial y final, y lista la diferencia entre ambas. Este cálculo ofrece una visión global sobre el estado del proyecto durante su desarrollo.

Fase	Tarea	Estimación inicial	Tiempo real	Diferencia (días)
Investigación	INV_01	10	9	1
	INV_02	15	11	4
	INV_03	25	11	14
Análisis	AN_01	4	2	2
	AN_02	7	3	4
Diseño	DS_01	10	7	3
	DS_02	4	4	0
Implementación	DEV_01	1	1	0
	DEV_02	9	7	2
	DEV_03	19	27	-8
	DEV_04	9	11	-2
	DEV_05	12	11	1
Pruebas	TEST_01	10	4	6
	TEST_02	15	9	6
Documentación	DOC_01	50	45	5
	DOC_02	10	9	1
TOTAL (días)		**210**	**171**	**39**

Tabla 83 - Comparativa planificación real y estimada

La comparativa muestra que se han empleado en el desarrollo del proyecto 39 días menos respecto de la estimación inicial. Esto es debido principalmente a que el proceso de familiarización y aprendizaje sobre el entorno de desarrollo de Unity ha resultado ser más sencillo de lo previsto, habiéndose aplicado en paralelo durante la fase de implementación.

8.3 Presupuesto

En este apartado se detallan los costes presupuestales del proyecto por fases, personal y material empleado.

8.3.1 Personal

El personal que forma parte del proyecto es el siguiente:

Nombre	Función	Coste mensual
Javier Abellán Fernández	Analista / Programador	2203,22 €
Telmo Zarraonandia Ayo	Ingeniero Senior	4234,81 €

Tabla 84 - Personal del proyecto

8.3.2 Distribución de los recursos

Los recursos (expresados en horas) se han distribuido tal y como se muestra a continuación:

	Javier Abellán Fernández	Telmo Zarraonandia Ayo	Total
Investigación	28	2	30
Análisis	17	1	18
Diseño	19,5	0	19,5
Implementación	243	0	243
Pruebas	27	0	27
Documentación	134	0	134
Otros*	17	6	23
Total	485,5	9	494,5

Tabla 85- Distribución de los recursos por fase

* La fase “Otros” hace referencia a aquellas tareas que quedan fuera de las tareas principales, tales como reuniones, comunicación por correo, etc.

8.3.3 Coste del personal

A raíz de los datos expuestos en los apartados anteriores, se procede a calcular el coste de cada elemento del personal. Para ello, se aplica un factor de dedicación (**Ded**), tal que:

$$\textbf{Ded} = \frac{H}{131{,}25 h/hombre\ \ mes}$$

Donde H es el número total de horas empleadas, y 131,25 la referencia estándar estimada de dedicación mensual.

El coste total se obtiene multiplicando el factor de dedicación mensual, por el coste mensual indicado en el apartado de Personal:

	Horas totales	Factor de dedicación	Coste
Javier Abellán Fernández	485,5	3,699	8149,71 €
Telmo Zarraonandia Ayo	9	0,068	287,96 €
Total			8437,67 €

Tabla 86 - Coste del personal

8.3.4 Coste del material

El material que ha supuesto un coste, indicado a continuación, está descrito en el apartado 1.5 Medios empleados de este documento.

	Coste por unidad	Unidades	Uso (%)	Coste
Equipo de sobremesa	894,23 €	1	55	491,83 €
Equipo portátil	1224,73 €	1	30	367,42 €
Microsoft Office Profesional 2010	119 €	1	100	119,00 €
Total				978,25 €

Tabla 87 - Coste del material

8.3.5 Presupuesto final

El coste total, que abarca el coste de personal y el coste del material, asciende a NUEVE MIL CUATROCIENTOS QUINCE EUROS Y NOVENTA Y DOS CÉNTIMOS (9415,92 €).

9. CONCLUSIONES

En esta sección se presentarán las conclusiones alcanzadas a la finalización del proyecto así como algunas posibles líneas de ampliación del trabajo realizado.

9.1 Conclusiones

Durante este proyecto se han desarrollado los módulos NPCs GREP Behaviour Authoring y NPCs GREP Behaviour Player, unas extensiones de la aplicación GREP que permiten la definición, tratamiento y reproducción de comportamientos automáticos en personajes no jugadores (NPCs).

Estos módulos suponen una importante incorporación a la funcionalidad de la aplicación, tratando un tema “tabú” en la industria del videojuego desde sus comienzos. De este modo, proporcionan una vía cómoda e intuitiva de definir inteligencias artificiales sencillas con una inversión de tiempo reducida.

En relación a esto, el estudio sobre técnicas de definición de comportamientos ha permitido concluir que la definición de comportamientos mediante scripts basados en básicas máquinas de estados, supone la metodología que más se adecua para afrontar el problema principal que aborda este proyecto: la implementación dinámica de comportamientos sin un nivel de conocimientos avanzado. Esto ha sido posible gracias al continuo progreso que están sufriendo los entornos de desarrollo de videojuegos, que actualmente ofrecen un conjunto completo de herramientas apoyado por una comunidad de usuarios que aportan numerosa documentación y que permiten, cada vez más, que usuarios sin conocimientos previos puedan investigar y desarrollar por su cuenta. Quiero creer que esto supone un paso hacia delante dentro de la industria del videojuego (como bien ha demostrado la tendencia con el software libre) en la que cualquier usuario podrá, de aquí a unos años, generar sus propios videojuegos de manera sencilla, reduciendo los costes que este proceso acarrea. No obstante, a la par en el tiempo, los avances tecnológicos que conllevan la creación de nuevos motores de procesamiento, tanto gráficos como de físicas o comportamientos, seguirán aportando los beneficios que se merecen las compañías diseñadoras de videojuegos.

Merece la pena, sinceramente, echar un vistazo atrás, apenas una década, y darse cuenta del impresionante avance que ha tenido esta industria, que no únicamente tiene fines recreativos, sino que abarcan casi cualquier campo de estudio y que aportan cada vez más, potentes e innovadoras formas de cumplir sus objetivos.

Durante todas las fases del proyecto se ha cuidado especialmente la limpieza técnica, de interfaz y usabilidad de modo que se consiga un sistema “amigable” para el usuario final de la aplicación. De esta forma, se le anima a que investigue y aprenda por su propia intuición a manejarla, sin disponer de ningún conocimiento técnico previo. Como apoyo, se ha desarrollado un manual de usuario, desglosado en las diferentes acciones que puede llevar a cabo, donde se muestra de manera ilustrada los pasos que el usuario debe seguir.

Es menester mencionar, que se han realizado pruebas de uso con personas ajenas al proyecto que no disponían de un perfil técnico, cuyos resultados fueron satisfactorios, corroborando o, dando firmeza, al cumplimiento de los objetivos establecidos.

A lo largo del desarrollo del proyecto han ido surgiendo dificultades que han traído numerosos quebraderos de cabeza, principalmente relacionados con el continuo proceso de investigación y aprendizaje tanto de la herramienta Unity como del código ya implementado del player GREP.

En cualquier caso, tanto el proceso de investigación como el de desarrollo de este proyecto me ha resultado personalmente enriquecedor, proporcionándome una visión algo distinta en cuanto al modo en el que un ingeniero de software diseña una aplicación: la relación entre el complejo tema de la generación de comportamientos automáticos con la usabilidad, simpleza y jugabilidad, donde estos conceptos se mezclan para ofrecer una aplicación didáctica que servirá de apoyo como herramienta de aprendizaje.

Por otro lado, he podido afianzar conocimientos sobre algunos de los principales motores gráficos del mercado con los que trabajan muchas empresas, aumentando mi experiencia, además, con otros lenguajes de programación como C# y JavaScript. Especialmente este último, ya que se encuentra en alza actualmente, habiéndose extendido su uso en numerosas aplicaciones y plataformas que han desarrollado APIs para darle soporte, como es PhoneGap [**26**] para el desarrollo de aplicaciones móviles.

9.2 Líneas futuras

La primera línea en cuanto a desarrollos futuros supone la ampliación de la funcionalidad de los módulos NPCs GREP Behaviour. Si bien se ha cumplido con los objetivos que se establecieron inicialmente, sería posible continuar la investigación con el fin de aprovechar los últimos avances en las herramientas de desarrollo, permitiendo optimizar el rendimiento y realizar alguna mejora, como la definición de comportamientos complejos, formados por otros menores, de modo que tengan la capacidad individual de interrumpir o desencadenar otros comportamientos (esto último aspecto está ya en parte contemplado, con la limitación de estar ligado a eventos que deben ser definidos previamente).

Otra opción se centraría en la posibilidad de adaptar los módulos para desarrollar una aplicación independiente de la aplicación GREP, permitiendo individualizar tanto su uso como su desarrollo, ampliando el campo funcional sobre el que está destinado. Esta tarea abriría una nueva rama de desarrollo que requeriría la participación de un equipo de personas predispuestas para emprender mencionada tarea.

ANEXO I: Ficheros XML

1. Fichero “bomberos.xml”

```
<configuration>
 <config name="walkSpeed" id="walkSpeed" value="4"/>
 <config name="inventory" id="inventoryspace" value="1"/>
 <config id="initiallifes" value="10"/>
 <config id="skybox" value="Sunny3"/>
 <config id="music" value="The Hero-short vs-2011"/>
</configuration>

<availableEntities url="http://tfgservertest.hostei.com/Game/Config/bomberos_entities.xml" />

<scene>

<floors>
    <rect pos="0;-0.1;0" size="100;0.1;100" rotation="0;0;0"/>
</floors>

<behaviours>
    <entity name="PersonaAyuda">
        <behaviour name="roll">
            <action type="forward_left" groups="2" />
            <action type="move_forward" groups="1" />
            <action type="forward_right" groups="2" />
            <action type="move_forward" groups="1" />
        </behaviour>
    </entity>
</behaviours>

<entities>
    <main-character name="Fireman" position="-38.8;0;-4.7" rotation="55.78">
        <attribute type="health" value="5000"/>
        <attribute type="mana"  value="8000"/>
        <attribute type="hpregen"  value="1"/>
        <actions>
            <action name="moveLeft" id="actMoveL">
                <control button="left" alt="a" />
            </action>
            <action name="moveRight" id="actMoveR">
                <control button="right" alt="d" />
            </action>
            <action name="moveFront" id="actMoveF">
                <control button="up" alt="w" />
            </action>
            <action name="moveBack" id="actMoveB">
                <control button="down" alt="s" />
            </action>
             <action name="jump"  id="actJump" >
               <control button="left alt" alt="mouse 2"/>
             </action>
             <action name="run" id="actRun" >
                <control button="left shift" alt="right shift" />
             </action>
            <action name="Hose" id="actHose" >
                <control button="left ctrl" alt="right ctrl" />
                <requires attribute="mana" value="1"/>
            </action>
        </actions>
```

```
</main-character>
<entity id="PersonaAyuda" name="PersonaAyuda" pos="-3.993738;2.624559;2.4684">
    <attribute type="health" value="5000"/>
    <attribute type="canbepicked" maxDistance="2.8" invImage="personasalvada" invName=
    "Persona Rescatada" invClickScript="dejarpersonaenambulancia" />
    <attribute type="drawinmap" value="1"/>
</entity>
<entity id="PersonaAyuda" name="PersonaAyuda" pos="-11.73868;1.528397;3.137119">
    <attribute type="health" value="5000"/>
    <attribute type="canbepicked" maxDistance="2.8" invImage="personasalvada" invName=
    "Persona Rescatada" invClickScript="dejarpersonaenambulancia" />
    <attribute type="drawinmap" value="1"/>
</entity>
<entity id="Ambulancia" name="Ambulancia" pos="-34.44388;2.254424;9.480547">
</entity>
<entity id="mobiliario" name="mobiliario" pos="-11.1224;-0.04999681;3.106812">
    <attribute type="drawinmap" value="real"/>
</entity>
<entity id="CamionBomberos" name="CamionBomberos" pos="-32.40186;0.1767808;-15.7423">
</entity>
<entity id="Fuego" name="Fuego" pos="-3.993738;2.624559;2.4684">
    <attribute type="health" value="3500"/>
    <attribute type="hpregen" value="2"/>
</entity>
<entity id="Fuego" name="Fuego" pos="-7.748193;0;4.961168">
    <attribute type="health" value="3500"/>
    <attribute type="hpregen" value="2"/>
</entity>
<entity id="Fuego" name="Fuego" pos="-10.60062;0;-0.2395077">
    <attribute type="health" value="3500"/>
    <attribute type="hpregen" value="2"/>
</entity>
<entity id="Fuego" name="Fuego" pos="0.3863806;0;0.7975388">
    <attribute type="health" value="3500"/>
    <attribute type="hpregen" value="2"/>
</entity>
<entity id="Fuego" name="Fuego" pos="-8.316444;0;2.442708">
    <attribute type="health" value="3500"/>
    <attribute type="hpregen" value="2"/>
</entity>
<entity id="Fuego" name="Fuego" pos="-6.545052;0;-2.68299">
    <attribute type="health" value="3500"/>
    <attribute type="hpregen" value="2"/>
</entity>
<entity id="Fuego" name="Fuego" pos="0.5659609;0;4.606334">
    <attribute type="health" value="3500"/>
    <attribute type="hpregen" value="2"/>
</entity>
<entity id="Fuego" name="Fuego" pos="-16.22453;0;0.6937933">
    <attribute type="health" value="3500"/>
    <attribute type="hpregen" value="2"/>
</entity>
<entity id="Fuego" name="Fuego" pos="-15.27546;0;4.010962">
    <attribute type="health" value="3500"/>
    <attribute type="hpregen" value="2"/>
</entity>
<entity id="Fuego" name="Fuego" pos="-4.462069;0;1.622988">
```

```
        <attribute type="health" value="3500"/>
        <attribute type="hpregen" value="2"/>
    </entity>
</entities>

<events>
    <!-- event id se utiliza para la monitorización -->
    <event id="waterShot" type="action">
        <actions>
            <action-id name="actHose"/>
        </actions>
        <consequences>
            <consequence type="damage" entity-name="Fireman" attribute="mana" value="10" />
        </consequences>
    </event>

    <event id="MainCharacterDies"  type="attributeValue">
        <entities>
            <entity-name name="Fireman"/>
        </entities>
        <attributes>
            <attr id="health" value="0" operator="equal"/>
        </attributes>
        <consequences>
            <consequence type="kill" entity-name="Fireman" />
            <consequence type="feedback" feedback-type="info" value="Ten mucho cuidado! Tu
            personaje ha sido herido de gravedad" />
            <consequence type="feedback" feedback-type="music" value="Warhead"/>
            <consequence type="feedback" feedback-type="openlog" />
        </consequences>
    </event>

    <event id="FireExtinguished"  type="attributeValue">
        <entities>
            <entity-name name="Fuego"/>
        </entities>
        <attributes>
            <attr id="health" value="0" operator="equal"/>
        </attributes>
        <consequences>
            <consequence type="kill" entity-name="Fuego" />
            <consequence type="feedback" feedback-type="script" feedback-id="gainpoints"
            value="10" /><!-- este no estaba funcionando porque estaba así: <consequence
            type="feedback" feedback-type="gainpoints" value="10" /> -->
            <consequence type="feedback" feedback-type="sysmessage" value="Muy bien!! Fuego
            apagado!" />
        </consequences>
    </event>


    <event id="PersonDie"  type="attributeValue">
        <entities>
            <entity-name name="PersonaAyuda"/>
        </entities>
        <attributes>
            <attr id="health" value="0" operator="equal"/>
        </attributes>
```

```
    <consequences>
        <consequence type="kill" entity-name="PersonaAyuda" />
        <consequence type="feedback" feedback-type="sysmessage" value="Una de las
        personas que estaba en apuros ha muerto" />
    </consequences>
</event>

<event id="FireWater" type="collision">
    <entities>
        <entity-name name="Agua"/>
        <entity-name name="Fuego"/>
    </entities>
    <consequences>
        <consequence type="damage" entity-name="Fuego" attribute="health" value="10000"
        />
    </consequences>
</event>

<event id="FireDamages" type="near" distance="14.0">
    <entities>
        <entity-name name="Fuego"/>
        <entity-name name="Fireman"/>
    </entities>
    <consequences>
        <consequence type="damage" entity-name="Fireman" attribute="health" value="10" />
        <consequence type="feedback" entity-name="Fireman" feedback-type="script"
        feedback-id="cough" />
    </consequences>
</event>

<event type="near" distance="16.0">
    <entities>
        <entity-name name="Fuego"/>
        <entity-name name="PersonaAyuda"/>
    </entities>
    <consequences>
        <consequence type="damage" entity-name="PersonaAyuda" attribute="health" value=
        "2" />
        <consequence type="damage" entity-name="PersonaAyuda" attribute="health" value=
        "value1" />
    </consequences>
</event>

<event type="near" distance="9.0">
    <entities>
        <entity-name name="Humo"/>
        <entity-name name="Fireman"/>
    </entities>
    <consequences>
        <consequence type="damage" entity-name="Fireman" attribute="health" value="5" />
        <consequence type="feedback" entity-name="Fireman" feedback-type="script"
        feedback-id="cough" />
        <consequence type="feedback" feedback-type="sysmessage" value="El fuego te puede
        ahogar" />
    </consequences>
</event>
<event type="near" distance="10.0">
```

```
        <entities>
            <entity-name name="PersonaAyuda"/>
            <entity-name name="Fireman"/>
        </entities>
        <consequences>
            <consequence type="behaviour" entity-name="PersonaAyuda" behaviour="roll" />
        </consequences>
    </event>
</events>

<monitor>
    <counter event-id="FireExtinguished"/>
    <counter event-id="MainCharacterDies"/>
    <counter event-id="PersonDie"/>
    <timestamp event-id="FireExtinguished"/>
    <timestamp event-id="PersonDie"/>
    <gametime event-id="MainCharacterDies"/>
</monitor>

</scene>
```

2. Fichero “bomberos_entites.xml”

```
<entities>
    <entity id="PersonaAyuda" name="PersonaAyuda" pos="0;0;0">
        <attribute type="health" value="0"/>
        <attribute type="canbepicked" maxDistance="0" invImage="" invName="" invClickScript=
        "" />
        <attribute type="drawinmap" value="0"/>
    </entity>
    <entity id="Ambulancia" name="Ambulancia" pos="0;0;0" />
    <entity id="mobiliario" name="Mobiliario" pos="0;0;0">
        <attribute type="drawinmap" value="0"/>
    </entity>
    <entity id="CamionBomberos" name="CamionBomberos" pos="0;0;0" />
    <entity id="Fuego" name="Fuego" pos="0;0;0">
        <attribute type="health" value="0"/>
        <attribute type="hpregen" value="0"/>
    </entity>
</entities>
```

ANEXO II: Manual de usuario

1. Instalación

Para comenzar a usar la aplicación es necesario instalar el plugin de Unity Web Player.

Asegúrese de que dispone de los requisitos mínimos para su instalación: Sistema operativo Windows XP/Vista/7 o Mac OS X 10.5 o superior y navegador web Internet Explorer, Firefox, Chrome, Safari u Opera.

Descargue y ejecute el plugin desde: http://unity3d.com/webplayer/

Dependiendo de su navegador, puede que se requiera la confirmación de instalación del software: siga las instrucciones que se le indiquen hasta finalizar la instalación.

Ilustración 57 - Instalación de Unity Web Player

2. Menú principal

Al iniciar la aplicación, se le mostrará el menú principal:

Ilustración 58 - Menú principal

Desde esta pantalla podrá iniciar un juego del listado de juegos disponibles (1) y activar y/o desactivar el modo de edición (2). Para mayor detalle consulte los apartados: 2.1 Iniciar un juego y 2.2 Activar o desactivar el modo de edición.

2.1 Iniciar un juego

Para lanzar la ejecución de un juego, pulse sobre la opción del listado de juegos que desee (Véase Ilustración 58 - Menú principal). Espere mientras se realiza la carga del juego. Este proceso tardará unos segundos.

2.2 Activar o desactivar el modo de edición

Para activar el modo de edición, haga click sobre el botón 

del menú principal. El modo de edición se iniciará tras seleccionar un juego del listado de juegos.

Cuando el modo de edición está activo, el color de fondo del botón se tornará rojo. 

Para desactivar el modo de edición, pulse de nuevo sobre el botón.

3. Menú de edición

Desde el menú de edición podrá obtener acceso a toda la funcionalidad relacionada con la gestión de entidades, comportamientos y eventos.

3.1 Interfaz de usuario

Ilustración 59 - Menú de edición

Listado de entidades disponibles (1): Muestra los diferentes tipos de entidades que el juego tiene disponibles. Desde aquí se puede seleccionar y añadir una entidad al escenario (Véase el apartado 4.1 Añadir una entidad).

Listado global de entidades (2): Muestra las entidades que se encuentran actualmente añadidas al escenario. Desde esta ventana se puede seleccionar y editar/eliminar una entidad (Véanse los apartados 4.2 Eliminar una entidad y 4.3 Modificar los atributos de una entidad).

Nombre de la entidad seleccionada (3): Muestra el nombre de la entidad que se encuentra en ese momento seleccionada.

Listado de comportamientos (4): Muestra los comportamientos disponibles para el tipo de entidad seleccionado. Desde esta ventana se pueden añadir nuevos comportamientos, así como seleccionar y editar/eliminar uno existente (Véase la sección 5. Gestión de comportamientos).

Listado global de eventos (5): Muestra los eventos que se encuentran actualmente disponibles en el juego. Desde esta ventana se pueden añadir nuevos eventos, así como seleccionar y editar/eliminar un evento (Véase la sección 6 Gestión de eventos).

Campo de ayuda dinámica (6): Muestra una serie de mensajes de ayuda, o sugerencias, sobre los distintos elementos y/o acciones del menú o del escenario.

Editor de comportamientos (7): Muestra la lista de acciones del comportamiento seleccionado. Desde esta ventana se pueden añadir y/o eliminar acciones, reproducir el comportamiento o restaurar la posición original de la entidad seleccionada (Véase la sección 5 Gestión de comportamientos).

3.2 Moverse por el escenario

El usuario puede moverse libremente por el escenario del juego, mediante las teclas de dirección.

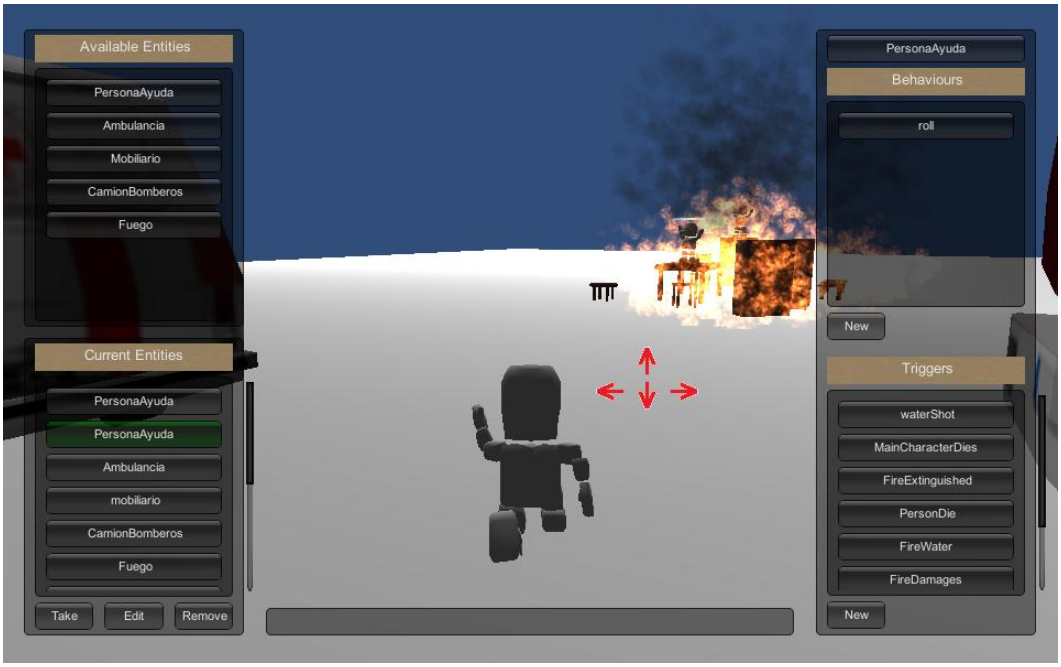


Ilustración 60 - Moviéndose por el escenario

4. Gestión de entidades

4.1 Añadir una entidad

Para añadir una nueva entidad, seleccione aquella que desee del listado de entidades disponibles. A continuación, haga “click” sobre el escenario, en el punto donde desee situar la nueva entidad.

Ilustración 61 - Añadiendo una entidad: 1

Las nuevas entidades se añadirán tanto al escenario como al listado global de entidades.

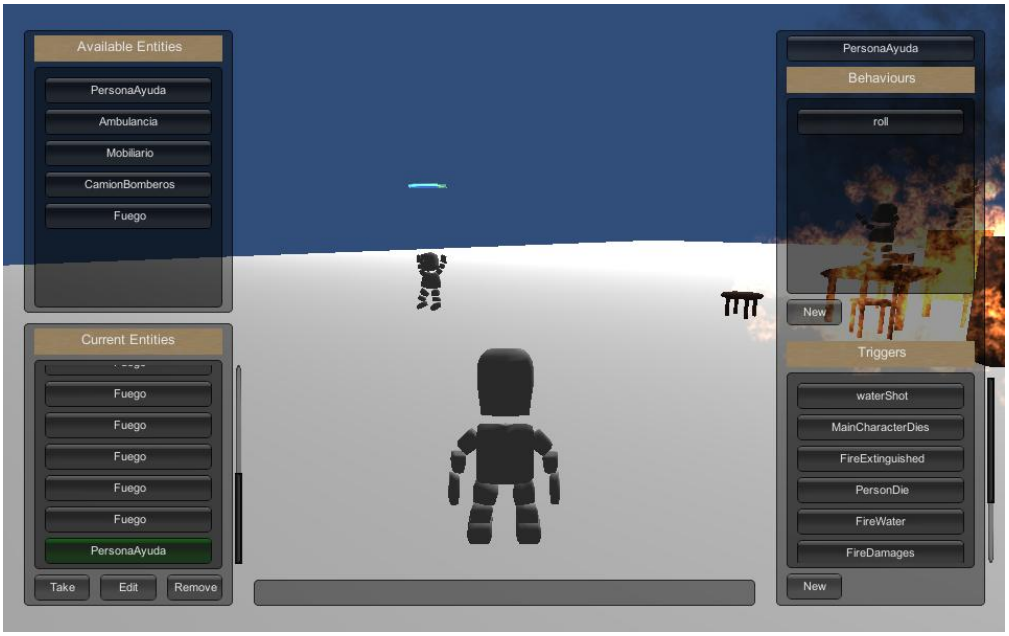
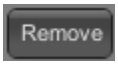


Ilustración 62 - Añadiendo una entidad: 2

4.2 Eliminar una entidad

Para eliminar una entidad del escenario, selecciónela en el listado global de entidades. A continuación, pulse sobre el botón 

de la ventana de control de entidades.

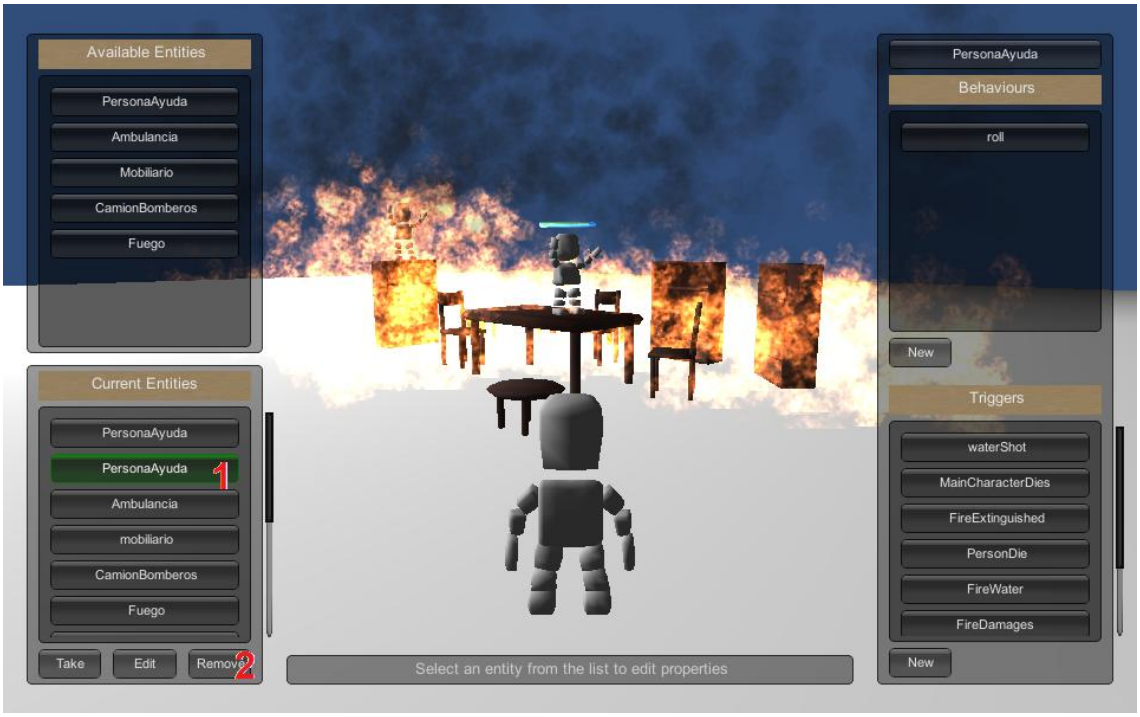


Ilustración 63 - Eliminando una entidad

La entidad será eliminada tanto del escenario como del listado global de entidades.

4.3 Modificar los atributos de una entidad

Para modificar los atributos de una entidad, selecciónela en el listado global de entidades. A continuación, pulse sobre el botón 

para desplegar la ventana de edición de atributos.

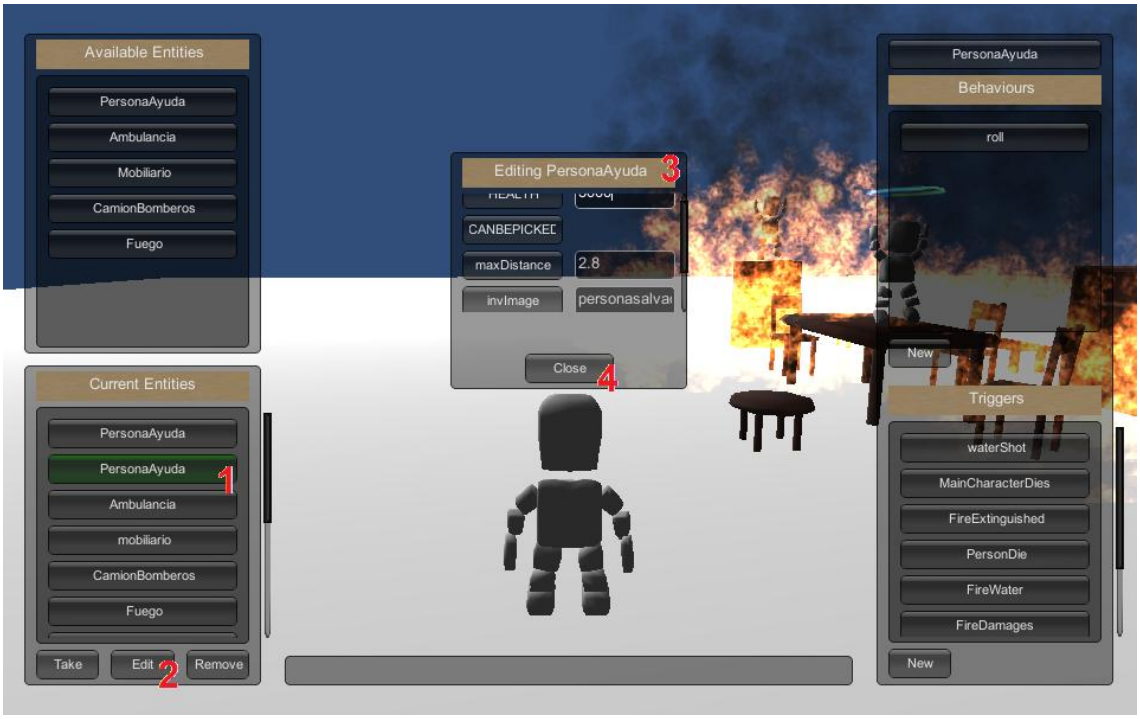


Ilustración 64 - Editando una entidad

Realice las modificaciones a través de los selectores y campos de texto de la ventana de edición de atributos. Para finalizar, pulse sobre el botón 

para cerrar la ventana.

Los atributos de una entidad están predefinidos, por lo que no se permite añadir o eliminar alguno de ellos, únicamente modificar sus valores.

4.4 Controlar una entidad

Puede tomar el control de una entidad para resituarla o iniciar el registro dinámico de acciones (Véase el apartado 5.5 Registrar acciones dinámicamente). Para ello, selecciónela en el listado global de entidades. A continuación, pulse sobre el botón 

para tomar el control directo de la entidad.

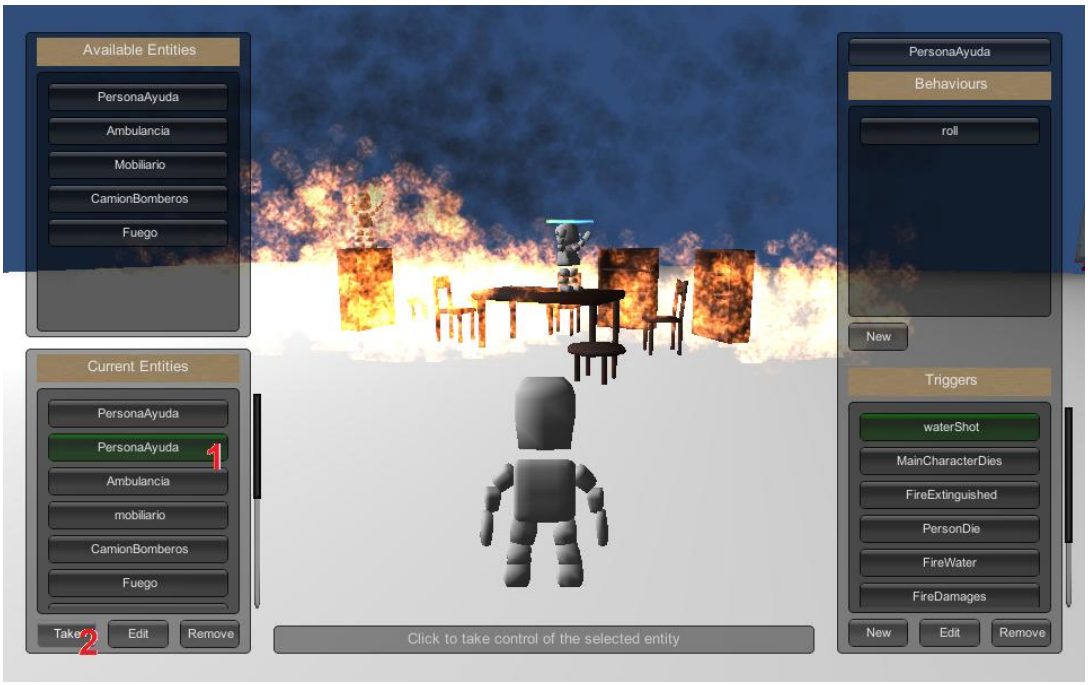


Ilustración 65 - Controlando una entidad: 1

Ilustración 66 - Controlando una entidad: 2

5. Gestión de comportamientos

5.1 Añadir un comportamiento

Para añadir un nuevo comportamiento, seleccione una entidad del listado de entidades disponibles. A continuación, pulse sobre el botón 

del recuadro de control de comportamientos.

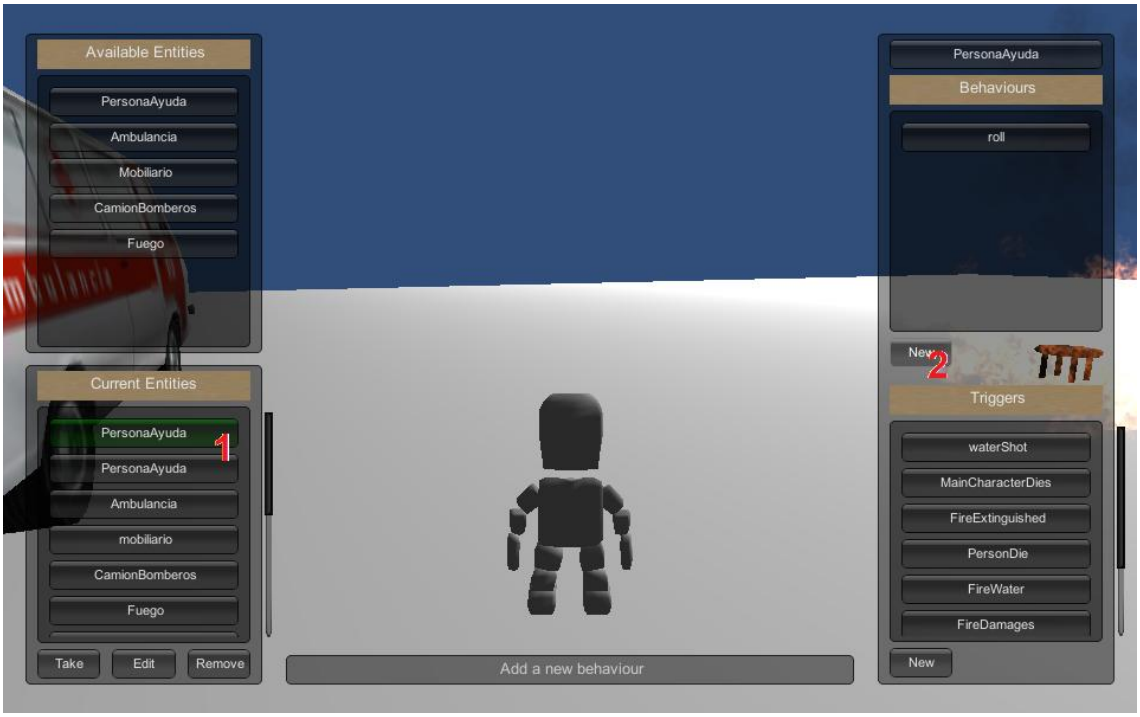
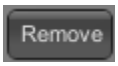


Ilustración 67 - Añadiendo un comportamiento

El comportamiento será añadido al listado de comportamientos para el tipo de entidad seleccionado.

5.2 Eliminar un comportamiento

Para eliminar un comportamiento de un determinado tipo de entidad, seleccione la entidad correspondiente del listado de entidades disponibles. A continuación, seleccione el comportamiento del listado de comportamientos y pulse sobre el botón 

del recuadro de control de comportamientos.

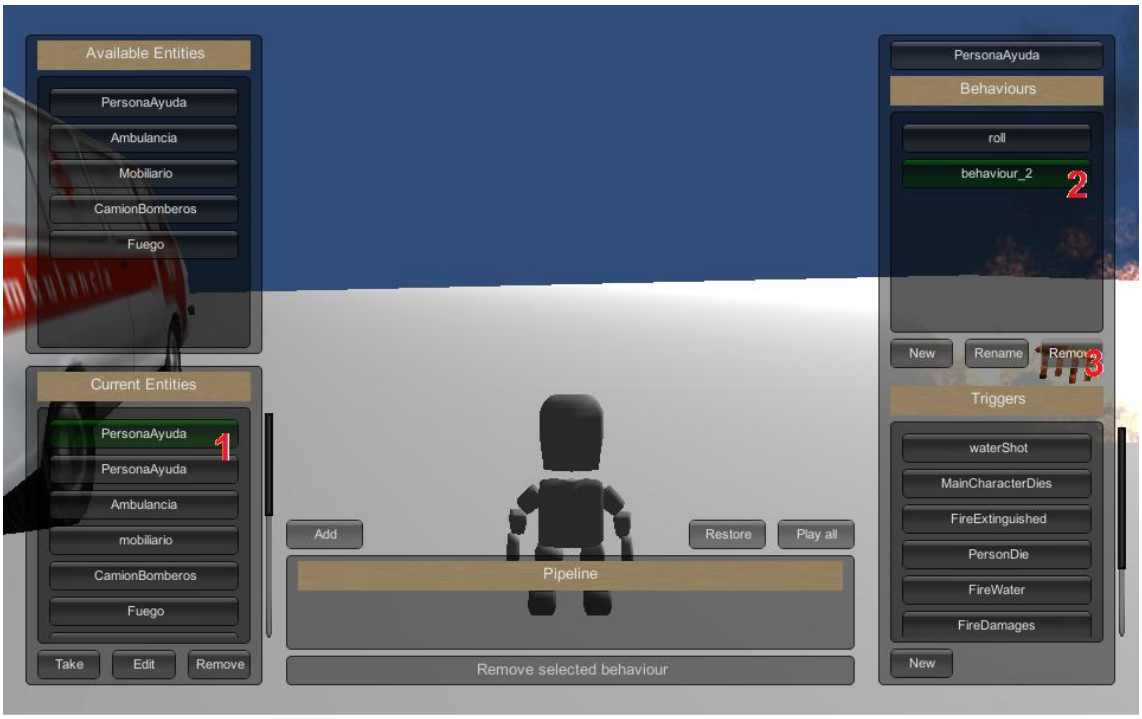


Ilustración 68 - Eliminando un comportamiento

El comportamiento será eliminado del listado de comportamientos para el tipo de entidad seleccionado.

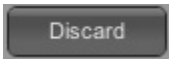
5.3 Renombrar un comportamiento

Para renombrar un comportamiento de un determinado tipo de entidad, seleccione la entidad correspondiente del listado de entidades disponibles. A continuación, seleccione el comportamiento del listado de comportamientos y pulse sobre el botón 

del recuadro de control de comportamientos.

Ilustración 69 - Renombrando un comportamiento

Introduzca un nuevo nombre para el comportamiento y pulse sobre el botón 

para realizar los cambios. Para cancelar, pulse 

5.4 Registrar acciones manualmente

Para registrar acciones de un comportamiento de manera manual, selecciónelo del listado de comportamiento para mostrar la ventana de edición de comportamientos.

A continuación, pulse sobre el botón 

del menú de edición para desplegar el diálogo de acciones disponibles y seleccione aquellas que desee. Para finalizar, pulse sobre el botón 

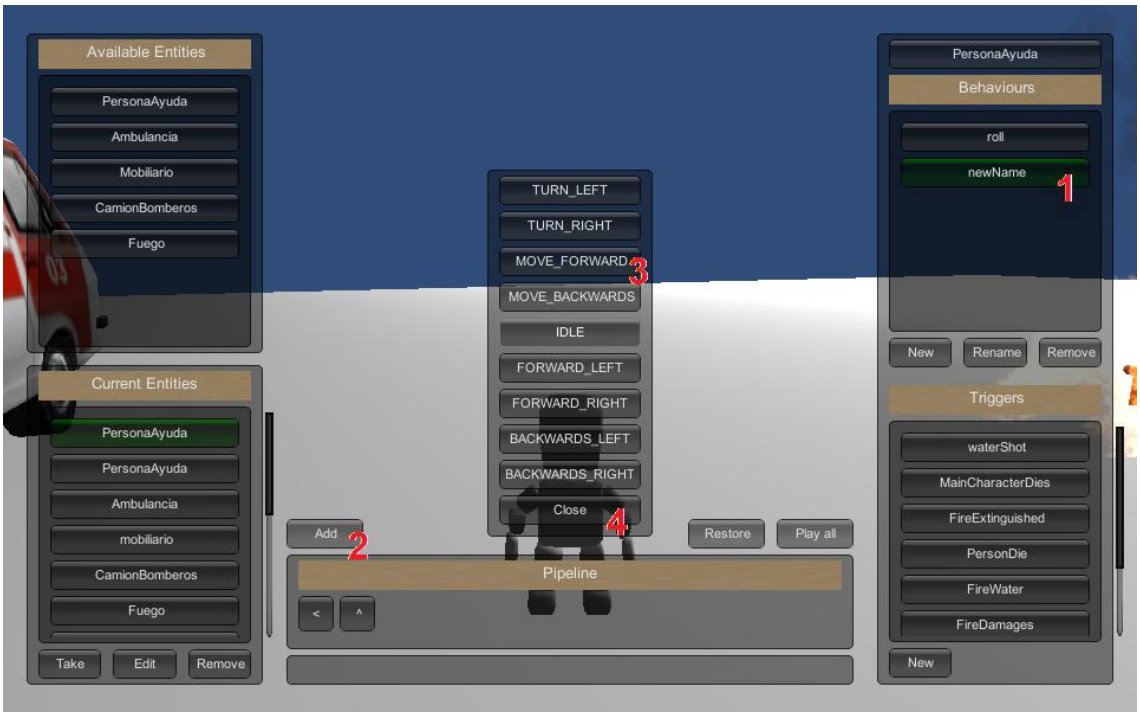


Ilustración 70 - Registrando acciones manualmente

Las acciones serán añadidas a la cola de acciones del comportamiento, agrupándose aquellas del mismo tipo.

5.5 Registrar acciones dinámicamente

Puede registrar acciones en tiempo real mediante el control directo de una entidad. Para ello, seleccione y tome el control de una entidad (Véase el apartado 4.4 Controlar una entidad).

Para comenzar el registro de acciones, seleccione o cree un comportamiento para desplegar el menú de edición de comportamientos. A continuación, pulse sobre el botón 

para iniciar la monitorización. Realice las acciones que desee registrar mediante las teclas de dirección.

Ilustración 71 - Registrando acciones en tiempo real

Las acciones serán añadidas automáticamente a la cola de acciones disponibles del comportamiento. Para finalizar el registro, pulse sobre el botón 

situado donde antes se encontraba el botón para iniciar el registro.

5.6 Eliminar acciones

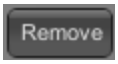
Para eliminar una acción de la cola de acciones de un comportamiento, selecciónela desde el menú de edición del comportamiento. A continuación, pulse sobre el botón 

Ilustración 72 - Eliminando acciones de un comportamiento

Las acciones serán eliminadas de la cola de acciones del comportamiento.

5.7 Previsualizar un comportamiento

Para previsualizar la ejecución de un comportamiento, selecciónelo en el listado de comportamientos de un determinado tipo de entidad. A continuación, pulse sobre el botón 

para comenzar la reproducción.

Ilustración 73 - Previsualizando un comportamiento

Asegúrese de que no está controlando la entidad correspondiente. En cuyo caso, la reproducción no tendrá efecto.

5.8 Restaurar la posición de una entidad

Puede restaurar la posición inicial de una entidad tras el registro dinámico de acciones. Para ello, pulse sobre el botón 

del menú de edición.

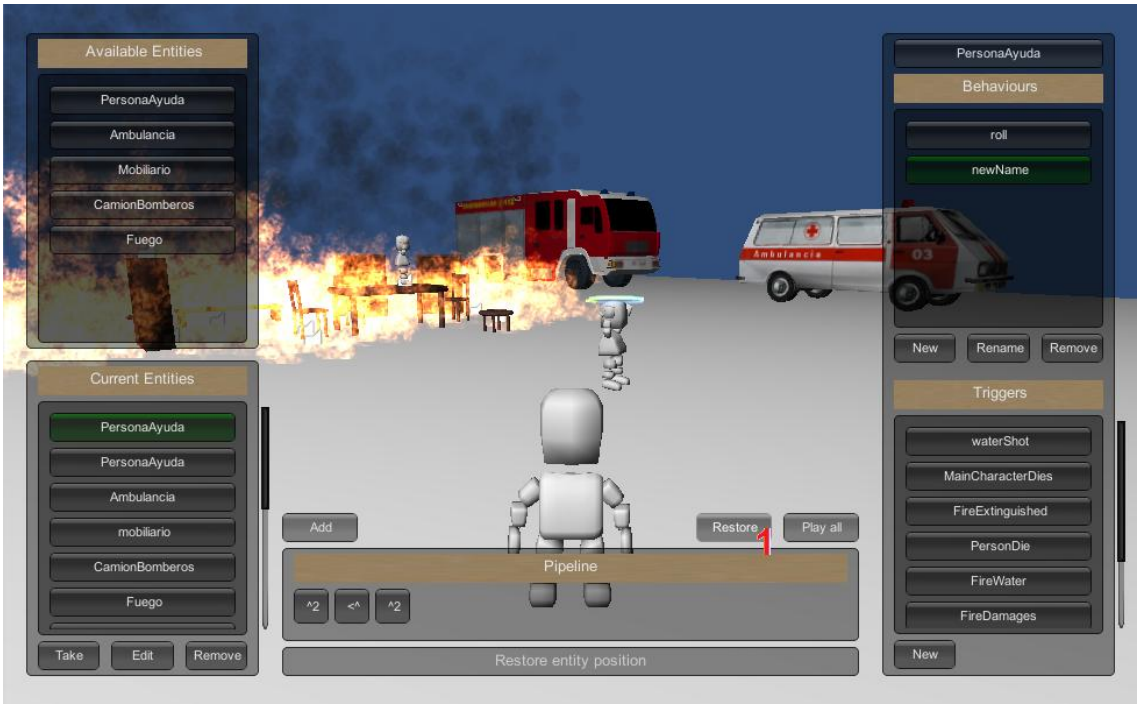


Ilustración 74 - Restaurando la posición de una entidad

La entidad recuperará la posición original, previa al registro de acciones.

6. Gestión de eventos

6.1 Añadir un evento

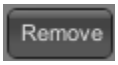
Para añadir un nuevo evento, pulse sobre el botón 

del recuadro de control de eventos. A continuación, sobre el diálogo emergente, seleccione el tipo que desee añadir.

Ilustración 75 - Añadiendo un evento

El evento será añadido al listado global de eventos.

6.2 Eliminar un evento

Para eliminar un evento del juego, selecciónelo del listado global de eventos. A continuación, pulse sobre el botón  del recuadro de control de eventos.

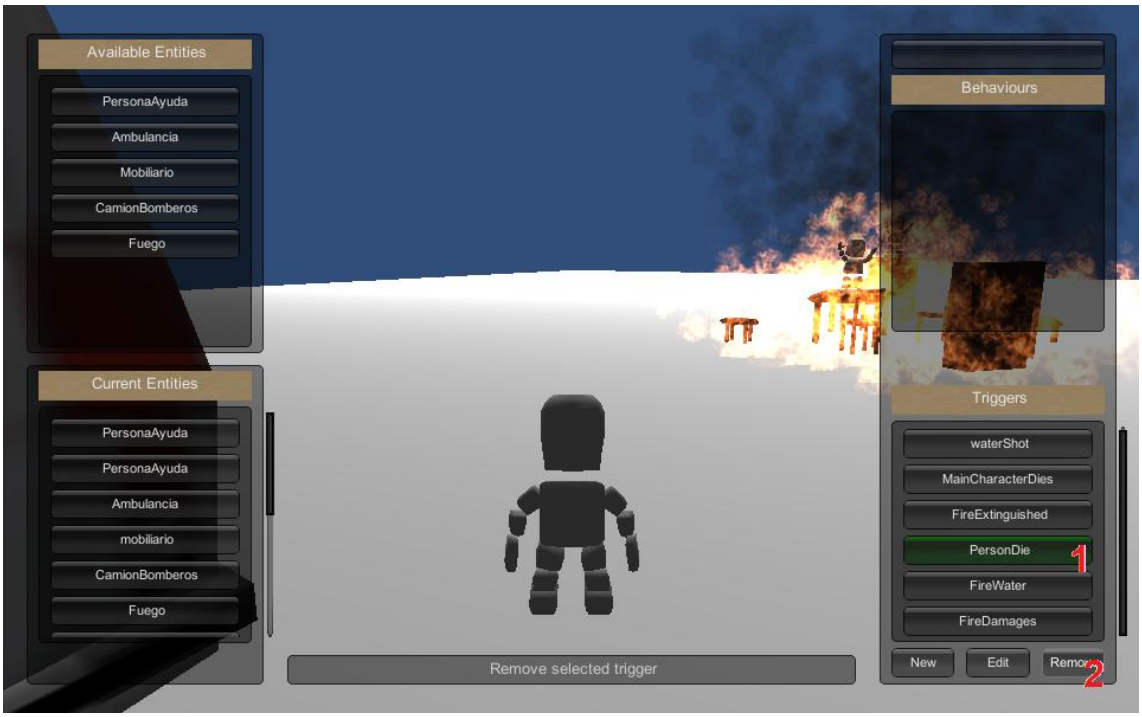


Ilustración 76 - Eliminando un evento

El evento será eliminado del listado global de eventos.

6.3 Modificar un evento

Para modificar los parámetros de un evento, selecciónelo de la lista global de eventos. A continuación, pulse sobre el botón 

para desplegar la ventana de edición de eventos.

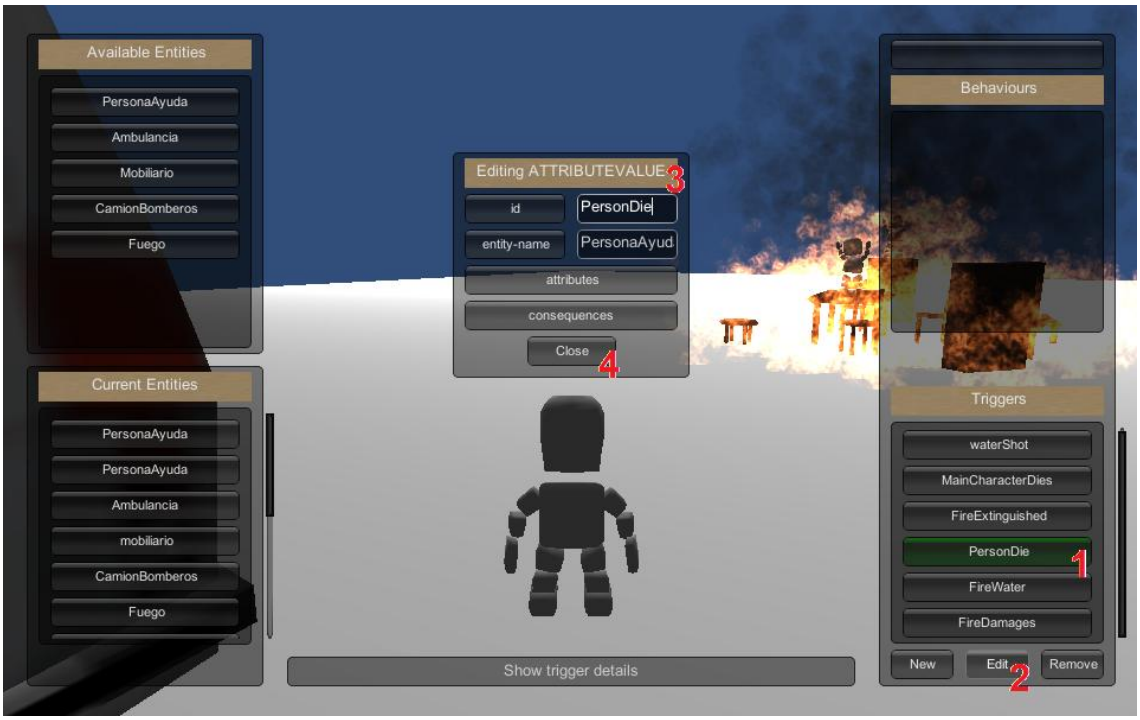


Ilustración 77 - Modificando un evento: parámetros

Realice las modificaciones a través de los selectores y campos de texto de la ventana de edición de eventos. Para finalizar, pulse sobre el botón 

para cerrar la ventana.

Para editar las condiciones que desencadenan un evento, pulse sobre el botón 

de la ventana de edición de eventos. A continuación, realice las modificaciones a través de los selectores y campos de texto.

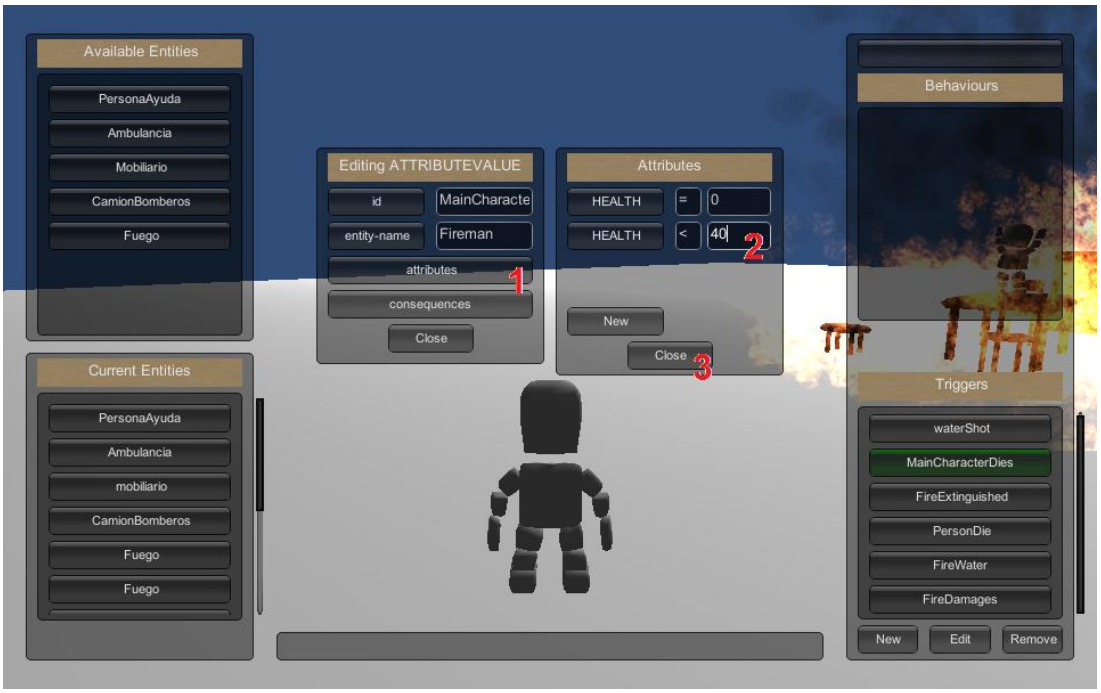
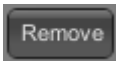


Ilustración 78 - Modificando un evento: atributos 1

Puede añadir y/o eliminar atributos pulsando respectivamente sobre el botón  del recuadro de control de atributos, o seleccionándolo del listado y pulsando sobre el botón 

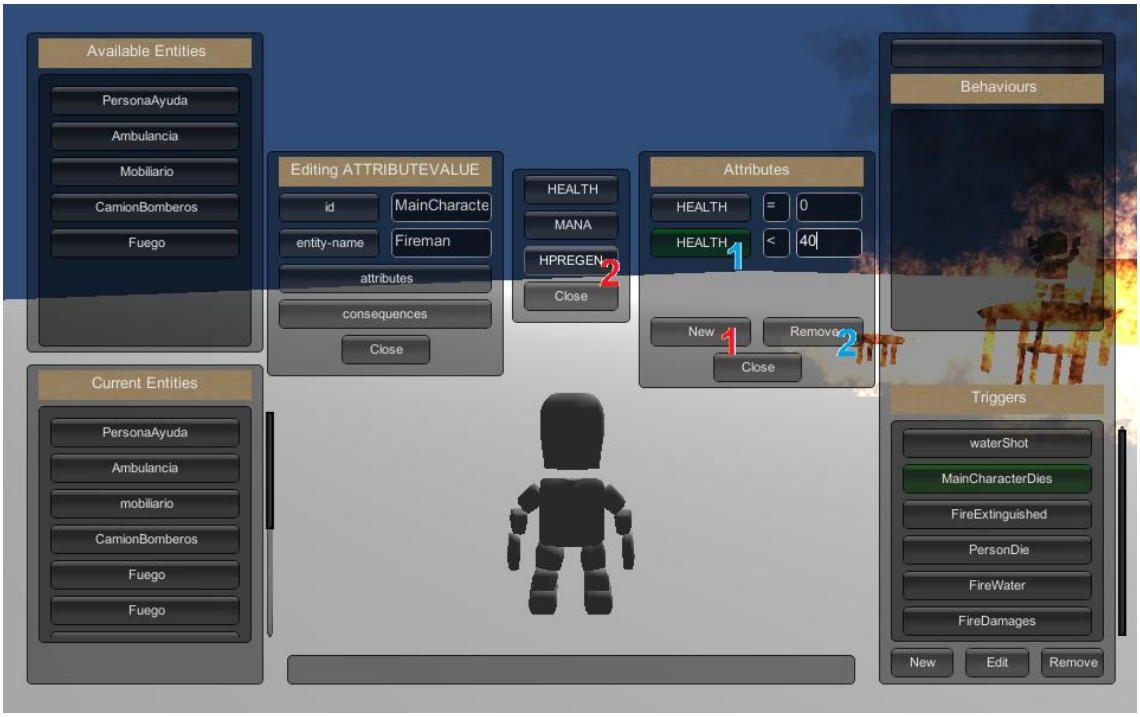
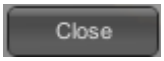


Ilustración 79 - Modificando un evento: atributos 2

Para editar las consecuencias que se ejecutarán cuando se cumplan las condiciones del evento, pulse sobre el botón 

de la ventana de edición de eventos. A continuación, realice las modificaciones a través de los selectores y campos de texto. Para finalizar, pulse sobre el botón 

para cerrar la ventana.

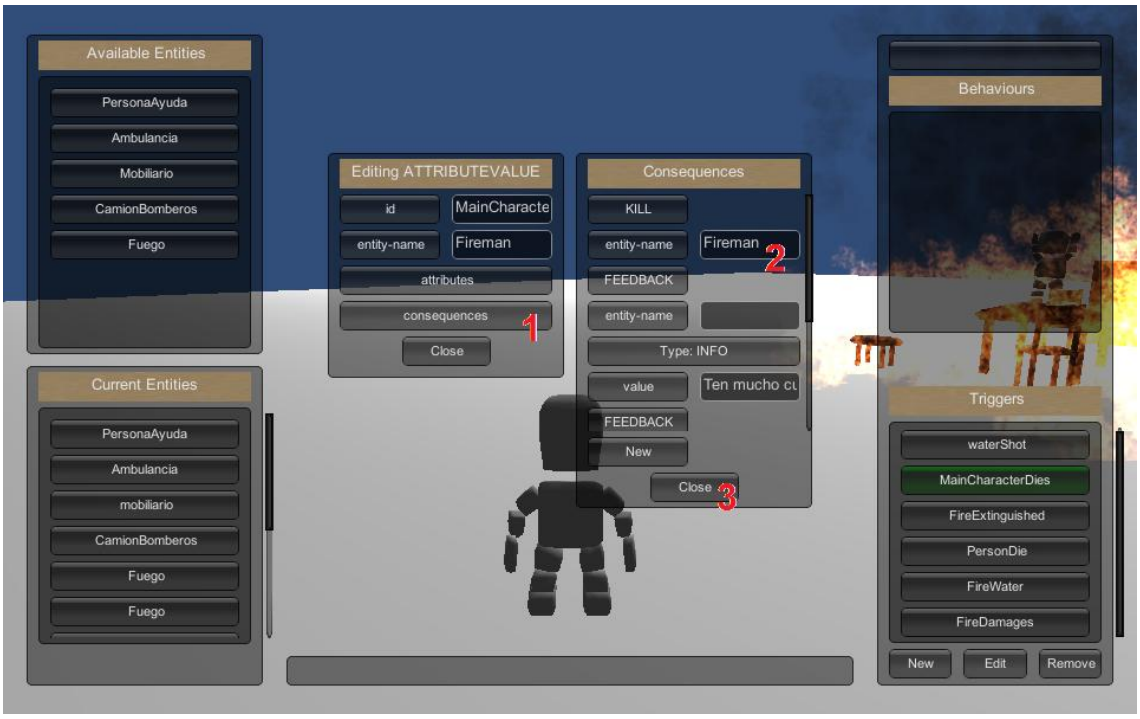
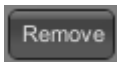


Ilustración 80 - Modificando un evento: consecuencias 1

Puede añadir y/o eliminar consecuencias pulsando respectivamente sobre el botón  del recuadro de control de consecuencias, o seleccionándolo del listado y pulsando sobre el botón 

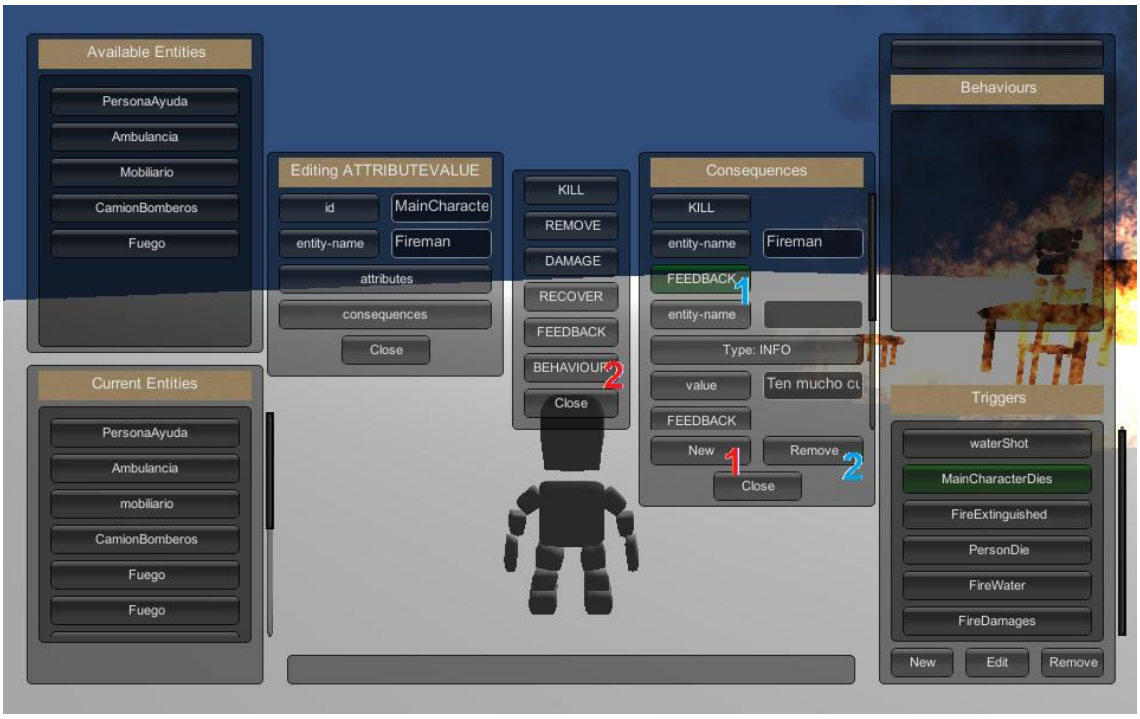


Ilustración 81 - Modificando un evento: consecuencias 2

7. Exportar datos

Para exportar los datos y hacer permanentes los cambios en el juego, pulse sobre el botón 

GLOSARIO

Acrónimos

IA: Inteligencia Artificial.

NPC: del Inglés, Non Player Character (Personaje no jugador).

SDK: del Inglés, Software Development Kit.

UDK: del Inglés, Unreal Development Kit.

GREM: del Inglés, Game Rules scEnario Model.

GREP: del Inglés, Game Rules scEnario Player.

IDE: del Inglés, Integrated Development Environment.

RTS: del Inglés, Real Time Strategy.

FPS: del inglés, First Person Shooter.

XML: del Inglés, Extensible Markup Language.

SSD: del Inglés, Solid State Disk.

JDK: del Inglés, Java Development Kit.

MVC: Modelo Vista Controlador.

Definiciones

En el contexto de este documento:

Click: acción de pulsar y soltar el botón de un periférico el cual tiene asociado un puntero que indica dónde se efectúa dicha acción.

Periférico: dispositivo auxiliar conectado a la unidad central de procesamiento de una computadora.

Hardware: componente físico de un computador.

Software: conjunto de programas de cómputo, procedimientos, reglas, documentación y datos asociados, que dormán parte de las operaciones de un sistema de computación.

Interfaz: herramienta gráfica que permite al usuario interactuar con la aplicación.

Lógica: conjunto de instrucciones o normas generales que permiten a una aplicación llevar a cabo tareas específicas de manera autónoma.

Renderizado: Proceso de generación de una imagen o vídeo a partir de un modelo en 3D.

Escenario: lugar donde se sitúan todos los elementos y personajes de cada juego y se desarrolla la ejecución.

Entidad: elemento con el cual el usuario u otros elementos del escenario de un juego, pueden interactuar.

Unity: Motor gráfico diseñado para el desarrollo de videojuegos.

Direct3D: Colección de librerías gráficas relacionadas con contenido multimedia, propiedad de Microsoft.

OpenGL: del Inglés, Open Graphics Library (Librerías gráficas de código libre) Colección de librerías gráficas para procesado de gráficos 2D y 3D. Caracterizado por ser software libre.

Script: Conjunto de instrucciones, o código, que llevan a cabo una determinada tarea.

Scripting: Metodología de programación basado en scripts.

.NET Framework: Conjunto de librerías específicas de Microsoft, con soporte para varios lenguajes de programación.

PhysX: Motor de físicas desarrollado por Ageia y propiedad de Nvidia desde 2008.

UnityScript: Lenguaje de programación propio de Unity, con sintaxis similar a Javascript.

Javascript: Lenguaje de programación interpretado orientado a objetos. Comúnmente conocido por su uso en programación Web.

C#: Lenguaje de programación orientado a objetos desarrollado por Microsoft.

Pathfinding: Técnica de obtención del camino más corto entre dos puntos. Empleado en conjunto con tecnologías de evasion de obstáculos para crear rutas de manera automatica por parte de una IA.

Grafo: Agrupación de objetos conectados entre sí que representa las relaciones entre los elementos de un conjunto.

Bot: Programa, o módulo, que permite automatizar tareas. En videojuegos: se trata de NPCs con habilidades para reproducir el comportamiento de un jugador humano.

High-end: Término empleado para indicar que un producto está destinado, o conlleva, un nivel de procesamiento muy alto.

Closure: Función característica de la programación funcional que extiende su entorno conteniendo una o más variables de otro entorno.

BIBLIOGRAFÍA

[1] Paloma Díaz, Ignacio Aedo, Mario Rafael Ruíz Vargas Telmo Zarraonandia, "Seeking Reusability of Computer Games Designs for Informal Learning," *Proceedings of ICALT*, pp. 455-459, 2011.

[2] Valve - Source Engine. [Online]. http://source.valvesoftware.com/

[3] Valve - Source SDK Base. [Online]. https://developer.valvesoftware.com/wiki/Source_SDK_Base

[4] Unity 3D. [Online]. http://unity3d.com/

[5] Geforce - PhysX. [Online]. http://www.geforce.com/hardware/technology/physx

[6] Unreal Engine. [Online]. http://www.unrealengine.com/

[7] Unreal Engine - UDK. [Online]. http://www.unrealengine.com/udk/

[8] CraftStudio. [Online]. http://craftstudio.wikia.com/wiki/CraftStudio_Wiki

[9] A.A.Puntambekar, *Formal Languages And Automata Theory*., 2008.

[10] CreacioDigital - Máquinas de estados. [Online]. http://creaciodigital.upf.edu/~smiguel/b13maquinaEstados.htm

[11] Andrew Rollings and Dave Morris, "Initial Design," in *Game Architecture and Design: A New Edition*. Estados Unidos: New Riders, 2003, ch. 17, pp. 457-509.

[12] Piedad Tolmos Rodríguez-Piñero. Universidad de Valencia. [Online]. http://www.uv.es/asepuma/X/J24C.pdf

[13] MSDN - Conceptos de minería de datos. [Online]. http://msdn.microsoft.com/es-es/library/ms174949.aspx

[14] Washington State University - The Shortest Path Problem. [Online].

http://www.eecs.wsu.edu/~ananth/CptS223/Lectures/shortestpath.pdf

[15] Peter Sanders Kurt Mehlhorn, *Algorithms and Data Structures: The Basic Toolbox*.: Springer, 2008.

[16] Mono-project. [Online]. http://www.mono-project.com/

[17] Microsoft - VisualStudio. [Online]. http://www.microsoft.com/visualstudio/

[18] Microsoft -.NET. [Online]. http://www.microsoft.com/net

[19] Microsoft msdn - Common Language Runtime. [Online]. http://msdn.microsoft.com/en-us/library/8bs2ecf4.aspx

[20] Mozilla Developer Network - Javascript. [Online]. https://developer.mozilla.org/en/docs/JavaScript

[21] Microsoft - Internet Explorer. [Online]. http://windows.microsoft.com/en-us/internet-explorer/download-ie

[22] Boo codehaus. [Online]. http://boo.codehaus.org/

[23] MonoDevelop. [Online]. http://monodevelop.com/

[24] "IEEE Std, IEEE Software Engineering Standard: Glossary of Software Engineering Terminology," *IEEE Computer Society Press*, 1993.

[25] Programación fácil - Colas. [Online]. http://www.programacionfacil.com/estructura_de_datos:colas

[26] PhoneGap. [Online]. http://phonegap.com/