Universidad
Carlos III de Madrid

Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

SimEmergencias: Simulador para formación en atención de emergencias extra-hospitalarias

Autor: Álvaro García Uzquiano

Tutor: Mª Carmen Fernández Panadero

Leganés, Junio de 2011

Título: **SimEmergencias: Simulador para formación en atención de emergencias extra-hospitalarias**
Autor: Álvaro García Uzquiano
Director: M. Carmen Fernández Panadero

EL TRIBUNAL

Presidente: Pedro Muñoz Merino

Vocal: Ángel Arias Hernández

Secretario: Jaime José García Reinoso

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 21 de Junio de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL: Ángel Arias Hernández

SECRETARIO: Jaime José García Reinoso PRESIDENTE: Pedro Muñoz Merino

"La buena suerte... no es algo que depende del azar, sino el resultado de un conjunto de actitudes y valores ante la vida, en definitiva, una postura existencial".

(Álex Rovira)

Agradecimientos

A Mª Carmen Fernández Panadero, por su ejemplar labor como tutora. Gracias por haberme descubierto un sector tan bonito como es el de los juegos serios y juegos educativos.

A mis padres y hermana, sólo con su ayuda y apoyo he podido llegar a escribir estas líneas. Tantas cosas que agradecer que un par de líneas en blanco no son suficientes.

A la Residencia Fernando Abril Martorell, por ser más que cuatro paredes. Y a toda la gente que he conocido ahí, porque durante muchos años han sido mi familia. Algunos de ellos amigos incondicionales para toda la vida.

A todos mis amigos de BEST, porque siempre ponen la guinda en cada una de las universidades donde están.

A “Erasmo de Rotterdam”, porque de alguna forma, sin el poderlo haber imaginado, me ha hecho vivir uno de los mejores años de mi vida.

Resumen

Los simuladores para el aprendizaje han sido usados desde principios del siglo pasado y desde entonces han evolucionado mucho enriqueciendo su contenido con materiales multimedia y modelos 3D. Pero el proceso de elaboración de un simulador a medida es muy costoso y requiere equipos de trabajo multidisciplinares en el que haya expertos en programación, diseño gráfico, pedagogía y en el dominio de la aplicación en el que se vaya a desarrollar el simulador. Por lo que todo esto limita su uso a sectores con altos presupuestos.

Este proyecto consiste en la elaboración de una arquitectura software utilizando el motor de juegos Unity 3D que facilita la creación de simuladores de formación. Además se elaborará un caso de ejemplo que ilustre la funcionalidad que se puede conseguir con este tipo de simuladores.

El prototipo desarrollado pretende ser una prueba de concepto para demostrar que a través de una arquitectura modular se pueden construir nuevos simuladores de aprendizaje, de forma rápida y por tanto, de bajo coste. Estos simuladores se pueden modificar tanto en fase de desarrollo (por personal técnico), como en fase de producción (por personal no técnico) modificando únicamente ciertos ficheros de texto. Con esto se consigue que un experto en una materia (por ejemplo un profesor) pueda realizar nuevos casos para simular sin la necesidad de depender de un experto técnico.

Palabras clave: simulador 3D, Unity 3D, simulador aprendizaje, juego serio, juego educativo, SCXML, IMS-QTI Lite, emergencias.

Abstract

Learning simulators have been used since the beginning of last Century and since then they have evolved improving their content with multimedia and 3D models. Unfortunately, the tailored simulator development process is expensive and requires multidisciplinary working teams made of experts in several professional fields such as programming, graphic design, pedagogy and master the chosen simulator development tool. Because of all these facts, its use is limited to high budget sectors.

This project involves the implementation of a software architecture using the game development tool Unity 3D, which makes easier the development of learning simulators. A part from that, it is implemented a case of use to show the functionality these kind of simulators can provide.

The developed prototype aims to be a proof of concept to show that through a modular architecture it can be built new learning simulators, in a fast way, which significantly reduces costs. These simulators can be modified during the development phase (by technical staff), as during the production phase (by non-technical staff) just modifying specific text files. The aim of this feature is to allow a field expert, e.g., a teacher, to create new simulation cases without the need of a technical expert.

Keywords: 3D simulator, Unity 3D, learning simulator, serious game, educational game, SCXML, IMS-QTI Lite, emergencies.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

Este Proyecto Fin de Carrera tiene como objetivo final implementar un prototipo de simulador 3D de formación donde la configuración de los escenarios de simulación sea factible para el personal docente, incluso sin conocimientos técnicos sobre ninguna tecnología presente en la implementación de este simulador.

De esta forma, el docente puede configurar fácilmente el flujo que debe llevar cada escenario de simulación, y el contenido educativo a practicar, así como su sistema de evaluación modificando ficheros de texto.

1.1.1 Marco del proyecto

El proyecto se ha desarrollado con la finalidad de preparar una aplicación-demostración para una propuesta de contrato del hospital “La Paz” en Bata (Guinea Ecuatorial) para la formación de personal sanitario y no sanitario (bomberos, policía) en situaciones de emergencia fuera de las instalaciones hospitalarias. La ciudad de Bata

acogerá el próximo año 2012 la copa de África de fútbol, y aunque dispone de hospital no cuenta con un servicio específico para la atención de emergencias, de ahí la necesidad de formar al personal tanto sanitario (médicos, enfermeros y auxiliares) como no sanitario (bomberos y policía) en la atención de emergencias extrahospitalarias cada uno en función de sus competencias específicas (traslado, inmovilización, evacuación, diagnóstico, curas, etc…).

Así, este proyecto se definirá con respecto a los requisitos y especificaciones de esta propuesta de contrato y se centrará en los casos de formación de técnicos sanitarios en situaciones de emergencia fuera de las instalaciones hospitalarias. Este simulador se usará en el proceso educativo entre la formación teórica y la formación práctica.

Por otra parte, este proyecto también se encuentra en el marco de desarrollos de simuladores flexibles y escalables que el Departamento de Telemática cuyo objetivo es realizar arquitecturas de simuladores de formación que permitan la generación de nuevos simuladores de forma rápida y por tanto minimizando su coste de producción.

1.1.2 La importancia de los simuladores en el proceso educativo

La utilización de simuladores como herramienta educativa presenta sin dudas muchas ventajas: facilitan el aprendizaje del alumno, permiten una mayor comprensión de situaciones reales, a la vez que eliminan los efectos negativos del ensayo-error en la realidad [Rins 2010].

En el campo de las Ciencias Médicas, numerosos factores han afectado la enseñanza limitando las oportunidades de aprendizaje directo en escenarios clínicos y con pacientes reales. Afortunadamente el desarrollo de tecnologías como la simulación clínica y el aprendizaje virtual pueden complementar la enseñanza, facilitar el aprendizaje y mejorar en los estudiantes las habilidades clínicas, comunicativas, de trabajo en equipo y de respuesta ante situaciones de urgencia, a la vez que disminuyen los riesgos para el paciente; sin embargo, no substituyen a los escenarios clínicos reales ni el aprendizaje directo con los pacientes [Ruiz-Parra *et al.* 2009].

Mediante los simuladores se pueden practicar situaciones y casos reales sin el riesgo de cometer un error debido a la poca experiencia en casos similares o por situaciones de stress típicamente asociadas a emergencias. Así, las principales ventajas del empleo de simuladores en el aprendizaje son:

- Aprender no sólo el qué sino el cómo (por ejemplo el trato con el paciente), del mismo modo que lo haría en una situación real.
- Practicar con situaciones y datos realistas.
- Tener un "campo de entrenamiento" a disposición del alumno para entrenarse tantas veces como necesite con la posibilidad de analizar sus resultados y evolución.
- Poder practicar de forma específica técnicas o casos.

Además con los resultados obtenidos en la simulación se pueden realizar estudios sobre el comportamiento y la actuación de los alumnos en estos casos, para así obtener una realimentación en el proceso educativo.

1.2 Objetivos

Este proyecto tiene dos grandes objetivos funcionales:

Para el cliente final (personal docente): Implementar una prueba de concepto donde se muestre la funcionalidad de una arquitectura modular y flexible que permite mantener separados el flujo de la aplicación (diseño de escenas, personajes, etc…) del contenido educativo. De esta forma, el personal docente, ajeno a la tecnología base del simulador, puede modificar y crear nuevos escenarios de simulación fácilmente.

Para los desarrolladores de software: Crear una arquitectura "software" que haga posible la creación de nuevos simuladores de formación con diferente lógica de contenidos de una forma rápida y sencilla. Es decir, hacer escalable la producción de simuladores de formación.

A parte de estos objetivos principales, para la aplicación se han definido otros subobjetivos:

- La aplicación debe ser multilingüe, siendo suficiente para esta primera versión los idiomas: castellano e inglés.
- Desarrollar una interfaz de la aplicación agradable para el usuario en la que estén representados todos los posibles elementos que tendría la aplicación final: paneles de diálogo, pistas, feedback, puntuación, opciones de selección, etc…
- Permitir el seguimiento de las sesiones de los alumnos por el tutor mediante la creación de ficheros XML con la información de sesión de los alumnos.

Para conseguir llevar a cabo estos objetivos es necesario plantear (internamente) unos objetivos técnicos que hagan abordable la consecución de los objetivos funcionales. Así se proponen los siguientes retos técnicos:

- Modelar el flujo de estados del simulador mediante el uso de un estándar. Para ello se los estándares a estudiar son: SCXML [W3C 2011] y IMS-LD [IMS 2003]. Una vez elegido el estándar se implementará un motor UnityScript [Unity 2011a].
- Modelar las opciones de comportamiento del estudiante mediante el uso de un estándar. Los estándares estudiados para este objetivo son: IMS-QTI [IMS 2006] y su versión reducida, IMS-QTI Lite [IMS 2002]. Y se implementará un motor para el estándar elegido.
- Por parte del cliente se pidió la adecuación de una interfaz para leer (en un futuro) datos de instrumentos de diagnóstico. Estos datos no están asociados a ningún formato y se leerán de texto plano, al contrario que sucede en los dos objetivos anteriores.

1.3 Estructura de la memoria

Este documento pretende recopilar toda la información relativa al desarrollo del proyecto, por tanto aquí se encuentran todos los detalles técnicos y funcionales del proyecto que se han estructurado para atender a dos tipos de lectores: (1) personal técnico

que se enfrente a una futura versión del proyecto, para facilitar la transmisión de todo el conocimiento del diseño y del desarrollo adquirido durante la realización de este proyecto y facilitar desarrollos posteriores; (2) profesionales no necesariamente expertos en la materia que quieran tener una visión global del proyecto para comprender el trabajo realizado y valorar las posibilidades de esta tecnología en el ámbito de la educación.

De esta forma, los capítulos que conforman esta memoria son los siguientes:

- Introducción y objetivos. En este primer capítulo se ha ofrecido una introducción al proyecto, presentándose las motivaciones y los objetivos que se pretenden conseguir.
- Estado del arte. Se realiza un estudio de la situación actual respecto a los simuladores para formación bajo la perspectiva de los objetivos de este proyecto. A partir de este análisis se sacan las conclusiones que sirven como base para la implementación de este proyecto. Además, se tratan conceptos relacionados con los simuladores y emergencias médicas, así como se introducen las herramientas utilizadas en el desarrollo de este proyecto.
- Gestión del proyecto. Detalla todos los aspectos relacionados con la gestión del proyecto, tales como los recursos necesarios, plan de trabajo, y una estimación de los costes de implementación.
- Diseño y desarrollo del sistema. Recoge el conjunto de requisitos que ha de cumplir el sistema a implementar, la arquitectura diseñada para cubrir estos requisitos y la implementación realizada.
- Conclusiones y trabajo futuro. Recoge los resultados logrados y las conclusiones obtenidas tras la realización del proyecto, analizando el nivel de consecución de los objetivos iniciales y presentando las líneas futuras de ampliación y mejora.

Finalmente, para concluir la memoria, se adjunta (1) un glosario con definiciones y aclaraciones sobre algunos términos y acrónimos citados en la memoria; (2) las referencias bibliográficas utilizadas; y (3) un manual de usuario.

Capítulo 2

Estado del Arte

2.1 Introducción

El objetivo de este capítulo es analizar los simuladores para el aprendizaje y entrenamiento de profesionales sanitarios en situaciones de emergencia en el mercado actual a nivel nacional y global. Además, se analizará en un punto a parte cómo estos simuladores se apoyan en ficheros o sistemas externos para crear los escenarios de simulación y los contenidos de estos.

Posteriormente se establecen las conclusiones del análisis previo, considerando los recursos utilizados y posibles problemas o limitaciones. Para finalizar se comparan estos resultados con la solución propuesta para este proyecto.

2.2 Valoración del Estado del Arte

Hasta la fecha se han desarrollado diferentes simuladores en el campo de la atención médica y la atención médica en situaciones de emergencia. La mayoría de estos simuladores se han basado en tecnologías de 2 dimensiones como los simuladores del *Proyecto Agrega* [Agrega 2009]. Pero también se han lanzado al mercado simuladores en 3 dimensiones como *Zero Hour: America's Medic* [Virtual Heroes 2007] que han supuesto un gran referente para este proyecto ya que también lo son a nivel global [Unreal Technology 2009].

En este capítulo se analizan estos dos estilos de simulación y sus ventajas, inconvenientes y casos de uso. En concreto se van a tomar los dos simuladores presentados en el párrafo anterior como referentes para el análisis de cada estilo.

2.2.1 Simuladores médicos de 2 dimensiones: Proyecto Agrega

En el estudio previo de simuladores existentes con tecnologías 2D se han analizado los simuladores desarrollados dentro del Proyecto Agrega, en especial los relacionados con emergencias sanitarias como el que se muestra en Figura 1.

En cuanto a la selección de ejercicios, en este simulador se pueden practicar distintos escenarios típicos propuestos en el menú principal.

Figura 1 Pantalla principal del simulador para técnico en emergencias sanitarias del Proyecto Agrega

Estos simuladores se han desarrollado bajo la tecnología Flash [Adobe Flash] y tienen un aspecto visual de 2 dimensiones, o algunas veces también llamado falso 3D. En la Figura 2 se puede observar cómo es la apariencia que presenta el simulador. En este ejemplo el alumno debe ordenar las acciones a seguir cuando se presenta una emergencia.

Figura 2 Pantalla de simulación

Como se puede ver consiste en seleccionar una de las acciones, representada como un item circular, y colocarlo ordenadamente en uno de los huecos libres que se muestran bajo la lista (ver Figura 3).

Figura 3 Selección de una respuesta en la pantalla de simulación

Tras la respuesta del alumno ocurren inmediatamente 2 eventos en la pantalla del simulador (ver Figura 4): (1) se actualiza el panel de puntuación el número de intentos y el número de respuestas correctas indicando al alumno si su respuesta fue acertada; y (2) se muestra un panel de "feedback" indicando si la elección fue correcta o no con alguna información más detallada.

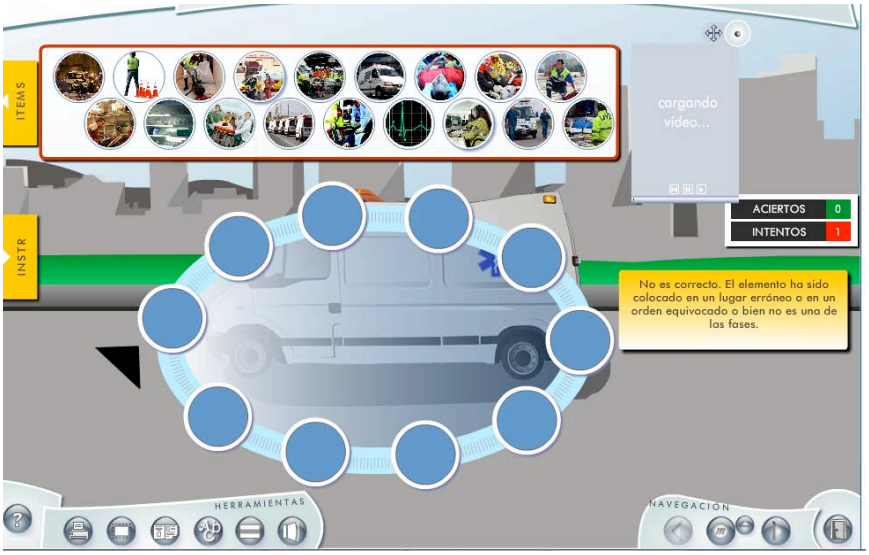

Figura 4 Muestra mensaje de 'feedback' en la pantalla de simulación

Los escenarios que el alumno puede practicar en el simulador analizado, "Técnico en emergencias sanitarias", son:

- Decálogo hospitalario de los servicios de emergencia
- Valoración inicial y secundaria del paciente traumatizado
- Protocolo de monitorización de valoración continua
- Simulación de registros electrocardiográficos básicos
- Protocolo de actuación para reconocimiento de parada cardiorrespiratoria
- Protocolo de actuación ante una parada cardiorrespiratoria básica
- Simulación del procedimiento de utilización de un desfibrilador externo automático
- Procedimiento de actuación tras una reanimación cardiopulmonar exitosa
- Procedimiento de triage en la atención de múltiples víctimas

A través de estos escenarios el alumno pone en práctica y refuerza la teoría aprendida previamente. Así, al alumno se le presenta un caso práctico (de los escenarios enumerados previamente) descrito a través de un texto y/o una grabación de audio. A partir de esta descripción el alumno debe realizar el ejercicio requerido, por ejemplo, ordenar las acciones a tomar en un protocolo.

El valor añadido del uso de un simulador similar a este con respecto a realizar un test o ejercicios por escrito (sin soporte informático) es que gracias al simulador el alumno recibe un feedback inmediato de su acción, conociendo si su elección fue correcta y recibiendo un mensaje de "feedback" (aunque muchas veces ocurre que este "feedback" es genérico y no aporta información, quedando toda la riqueza que podría aportar la tecnología anulada).

2.2.2 Simuladores médicos en 3 dimensiones: Zero Hour

El simulador de emergencias médicas "Zero Hour: America's Medic" fue diseñado entre 2006 y 2008 para el entrenamiento de técnicos médicos de emergencias en entornos virtuales extremos, y realistas, como son: terremotos, atentados terroristas, accidentes nucleares, etc… [Gaudiosi 2008]. Este simulador está siendo usado por la Iniciativa Nacional para la preparación de Servicios Médicos de emergencia [NEMSPI 2011] del Departamento de Defensa Interior de los Estados Unidos.

En cuanto a la parte técnica, este simulador ha sido desarrollado mediante la tecnología "Unreal Engine 3" de "Epic Games".

El objetivo de este simulador es ponerse en el papel de un técnico de emergencias médicas en cada escenario, evaluando la situación de emergencia y salvando el máximo número de vidas posible. Para ello, una de las claves y punto fuerte de este simulador es que el alumno debe diagnosticar a través de los síntomas de las víctimas, que se presentan con mucho detalle como se puede ver en la Figura 5 donde se puede apreciar cómo la víctima tiene los ojos irritados y de color rojizo.

Figura 5 Técnico sanitario interactuando con una víctima

Los principales objetivos que el alumno puede practicar, mejorar y reforzar con el simulador se dividen en cuatro grupos fundamentales:

- Detección de emergencias químicas, biológicas, radiológicas y nucleares.
- Triage y tratamiento pre-hospitalario.
- Recolección de información y reconocimiento de amenazas.
- Colaboración e información sobre los detalles del escenario.

Un ejemplo de escenario de este simulador se muestra en la Figura 6, donde se ha producido una emergencia química que ha afectado a muchas víctimas. De esta forma, el estudiante debe seguir las normas marcadas en el protocolo de emergencias químicas, informando a su llegada sobre los detalles del escenario que tiene ante sí, valorando la situación de las víctimas (como se ve en la Figura 6 algunas víctimas permanecen tumbadas en el suelo inmoviles, otras piden ayuda, etc…) y por último tomando las decisiones necesarias para salvar el mayor número de vidas.

Figura 6 Ejemplo de escenario de simulación en *Zero Hour*

Tras cada escenario se evalúa la actuación que el alumno ha tenido y se muestra en una pantalla como la del ejemplo de la Figura 7.

Figura 7 Pantalla de resultados

En esta pantalla se evalúa el tratamiento recibido por cada víctima o cómo se ha resuelto un objetivo. Si se ha completado satisfactoriamente se marcará como aprobado.

En cambio, si se ha cometido un error crítico se calificará como suspenso. Y por último, si el objetivo se ha completado, pero se podría mejorar la actuación, se calificará como suficiente.

De la misma manera cada objetivo tiene una lista más detallada de subobjetivos que también es evaluada como se muestra en la Figura 8.

Figura 8 Pantalla de resultados de subobjetivos

2.2.3 Uso de ficheros y sistemas externos en la creación de escenarios

Como se ha explicado en el capítulo de objetivos, establecer los contenidos de los escenarios de simulación, incluso los propios escenarios, a través de ficheros externos provee de una gran flexibilidad al sistema simulador y permite crear nuevos escenarios de una forma rápida y sencilla.

De esta forma, de los simuladores antes analizados, los simuladores del Proyecto Agrega tienen definidos tanto los textos de los escenarios como los contenidos educativos (tareas, objetivos, puntuaciones, feedback, etc…) en ficheros XML externos, y por tanto modificables por los usuarios. Estos ficheros XML siguen un formato propio de la

empresa que ha desarrollado estos simuladores, aunque son fácilmente comprensibles por cualquier persona ajena al equipo de desarrollo de los simuladores.

Por el contrario, el simulador *Zero Hour* no dispone de ningún fichero externo de donde obtenga la información para sus escenarios o misiones.

2.3 Conclusiones del Estado del Arte

Con el fin de elaborar el diagnóstico de la situación actual, en este punto se exponen las conclusiones obtenidas a partir del punto anterior y se identifican problemas, deficiencias, limitaciones y mejoras que se podrían hacer y que en la propuesta de este proyecto se han tenido en cuenta.

Como principal diferencia encontrada tanto en aspecto como en experiencia de usuario está la apariencia en 2 o 3 dimensiones. Se ha observado que para materias de aprendizaje estáticas, como por ejemplo ordenar las acciones a tomar en un protocolo, un simulador en 2 dimensiones proporciona un escenario de simulación correcto, incluso puede ocurrir que un escenario en 3 dimensiones pueda resultar demasiado complejo y se pierda el foco de atención del alumno en el continente en lugar de en el contenido.

Ahora bien, si la materia educativa que se está tratando tiene un alto componente situacional, es decir, la situación, el entorno, las condiciones donde se desarrolla la simulación dependen del entorno, entonces, solamente un simulador en 3 dimensiones puede proporcionar todos los detalles necesarios para reproducir de una forma realista un escenario.

De igual forma, cuando en el aprendizaje se requieren componentes multimedia, interactuar o cooperar con más personajes o estudiantes en el “mundo” virtual para resolver un objetivo, o aproximarse a algún objetivo o víctima, definitivamente un entorno en 3 dimensiones ofrece mayor realismo.

En el caso de simuladores de emergencias, se ajusta perfectamente a las necesidades descritas para los simuladores en 3 dimensiones. Por esta razón, en la propuesta para este proyecto se ha optado por implementar un simulador en 3 dimensiones.

	Proyecto Agrega	**Zero Hour**	**Propuesta de este proyecto**
Referencia	www.proyectoagrega.es	zerohour.nemspi.org	https://svn.gast.it.uc3m.es/PFCs/SimuladorDeEmergenciasConUnity_2/SimEmergencias (se requiere autorización)
Tecnología	Adobe Flash	Unreal 3	Unity 3
Apariencia	2D	3D	3D
Tipo de interacción	Ratón	Teclado, ratón y posición en el “mundo” virtual	Teclado, ratón y posición en el “mundo” virtual
Flexibilidad en fase de producción	Textos modificables a través de ficheros XML (formato propietario)	No modificable	Textos, datos historial médico, imágenes y multimedia modificables (si permitido) a través de ficheros XML (estándares SCXML y IMS-QTI Lite)
Clasificación de escenas	Por técnica	Por escenario	Por escenario y/o por técnicas (en siguiente versión)
Efectos sonoros clínicos	No	Si	Parcialmente

	Proyecto Agrega	**Zero Hour**	**Propuesta de este proyecto**
Límite de tiempo	No	Si	Si
Puntuación	Visible durante simulación	Visible tras la simulación	Visible durante y después de la simulación
Feedback	Durante simulación	No	Durante simulación

Tabla 1 Comparativa de simuladores sanitarios comerciales actuales

2.4 Definiciones y conceptos generales

En este apartado se abordan brevemente conceptos específicos de los simuladores 3D o del entorno clínico que aparecen a lo largo de este documento.

- Avatar: representación virtual de una persona en un "mundo" o entorno virtual.

- Mundo (virtual): representación de un entorno donde un o varios usuarios a través de avatares pueden interaccionar con los diversos elementos de su entorno y comunicarse entre ellos. En este documento se referirá como "mundo".

- Triage: método de selección y clasificación de pacientes basado en sus necesidades terapéuticas y los recursos disponibles para su atención [Vergara Olivares *et al.*]. En este proyecto se ha utilizado el Sistema Español de Triaje cuyo sistema de clasificación se describe en la siguiente tabla:

Blanco	cuando la víctima ha fallecido
Negro	cuando las posibilidades de recuperación son nulas
Rojo	cuando el paciente tiene posibilidad de sobrevivir y la actuación médica debe ser inmediata
Amarillo	es un paciente diferible, para ser vigilado mientras se le puede atender
Verde	paciente levemente lesionado, que puede caminar y su traslado no precisa medio especial

Tabla 2 Sistema Español de Triage

2.5 Uso de especificaciones y estándares

Para el desarrollo de este proyecto se han estudiado y comparado diferentes estándares, en concreto se han utilizado para definir los formatos estándar de los ficheros de datos externos que se analizan en el punto 4.4.1 Ficheros de contenido del escenario de simulación.

A continuación se analizan estos estándares.

2.5.1 XML

XML, siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML (Standard Generalized Markup Language) y permite definir la gramática de lenguajes específicos [W3C 2008].

2.5.2 SXCML

SCXML, siglas en ingles de State Chart XML: State Machine Notation for Control Abstraction (‘Diagrama de estados XML: notación de máquina de estados para control abstracto’). Es un lenguaje basado en el formato XML que proporciona una máquina de

estados genérica basada en diagramas de estado Harel. Los estados de Harel permiten crear superestados, regiones ortogonales así como actividades como parte de un estado.

Mediante SCXML se pueden describir maquinas de estado complejas, como por ejemplo: subestados, estados paralelos, sincronización o concurrencia [Wikipedia 2011d].

Este estándar al igual que XML está desarrollado por el World Wide Web Consortium [W3C 2011].

2.5.3 IMS-LD

IMS LD [IMS 2003] (siglas en ingles de Instructional Management Systems learning Design) es una especificación de un metalenguaje el cual define el modelo para los procesos de aprendizaje pedagógicamente neutro. Esta especificación está mantenida por el IMS Global Learning Consortium.

La especificación puede ser asemejada a un guión donde las personas actúan en diferentes roles. Estos roles están orientados hacia objetivos específicos llevando a cabo tareas de aprendizaje y/o actividades de apoyo.

2.5.4 SCXML vs IMS-LD

Para el desarrollo de este proyecto se ha decidido utilizar el estándar SCXML debido a que su funcionalidad, basada en estados, encaja perfectamente con los requisitos del sistema, configurar qué debe ocurrir en cada situación, modelada mediante un estado.

Además, otra de las razones por las que se descartó utilizar IMS-LD es su complejidad comparado con SCXML, que es mucho más ligero a la hora de diseñar y construir un motor.

2.5.5 IMS-QTI e IMS-QTI Lite

La especificación IMS-QTI [IMS 2006] (siglas en ingles de Instructional Management Systems Question and Test Interoperability specification) y su versión

reducida IMS-QTI Lite [IMS 2002] especifican un formato estándar para definir un contenido evaluable entre autor, usuario, repositorios y diferentes sistemas. La especificación consiste en un modelo de datos que define la estructura de preguntas, evaluaciones y resultados basados en un formato XML.

La razón por la que se ha decidido utilizar en este proyecto la especificación reducida, IMS-QTI Lite, se debe a que dados los requerimientos del sistema a implementar, utilizar la versión reducida cumplía con los requisitos. De esta forma se ahorró tiempo de implementación en esta parte para poder distribuirlo en otras tareas del proyecto.

2.6 Elección de Unity 3D como plataforma

La elección de Unity 3D como plataforma de desarrollo, frente a otras tecnologías como Flash, se ha basado en los siguientes factores [Unity Answers 2011]:

- Calidad gráfica 3D. Unity ofrece aceleración 3D mediante hardware, además hace uso de modernos sistemas de sombras, efectos gráficos y velocidad gráfica.
- Rendimiento. La implementación de JavaScript para Unity, UnityScript, es 20 veces más rápida que la usada para Flash [Unity 2011a].
- Amplio soporte entre plataformas. Actualmente Unity es soportando en Windows, Mac OS, navegadores web, iOS, Android y Wii, siendo el coste de desarrollo de adaptación de una plataforma a otra, nulo o mínimo.
- Fácil instalación del plugin (en el caso de navegadores web). Su instalación es muy rápida y simple, además se puede instalar sin necesidad de tener cuenta de administrador del sistema operativo.

2.7 Herramientas empleadas

En este apartado se describen las herramientas utilizadas para la implementación de este proyecto.

2.7.1 Herramientas de desarrollo

Unity 3D

Unity 3 es una herramienta integrada de autor para el desarrollo de videojuegos en 3D (tres dimensiones) u otro contenido interactivo como animaciones 3D en tiempo real o visualizaciones arquitectónicas.

La principal característica, de Unity es su entorno de desarrollo, que es gráfico, aunque también dispone de un editor de código en diferentes lenguajes de programación; JavaScript, C# y Boo (Phyton).

El entorno de desarrollo está disponible para los sistemas operativos Windows y Mac OS X y se pueden desarrollar aplicaciones para ser ejecutadas en: Windows, Mac OS, navegadores web, iPhone, iPad o las videoconsolas Nintedo Wii, Xbox 360 o PlayStation 3.

Para Windows, Mac y navegadores web se pueden desarrollar aplicaciones de forma gratuita aunque también existe la opción de adquirir una licencia profesional (Pro) que habilitan opciones de desarrollo más avanzadas como filtros de audio, reproducción de videos, sombras en tiempo real, etc… Esta licencia profesional es obligatoria si la empresa en cuestión superase los 100.000$ de facturación anuales. Para desarrollar aplicaciones para las demás plataformas es necesario adquirir una licencia específica para cada una.

Para el desarrollo de este proyecto se ha utilizado la versión de Unity 3.2.0f4 tanto para el sistema operativo Windows Vista como para Mac OS X Versión 10.6.6.

En cuanto al lenguaje de programación utilizado principalmente se ha decido utilizar JavaScript debido a que la mayor parte, casi la totalidad, de la documentación oficial está basada en JavaScript.

La aplicación está diseñada por razones de usabilidad y seguridad para ser una aplicación de escritorio para los sistemas operativos antes mencionados, Windows y Mac OS. Aunque no se descarta en un futuro incluir la versión Web.

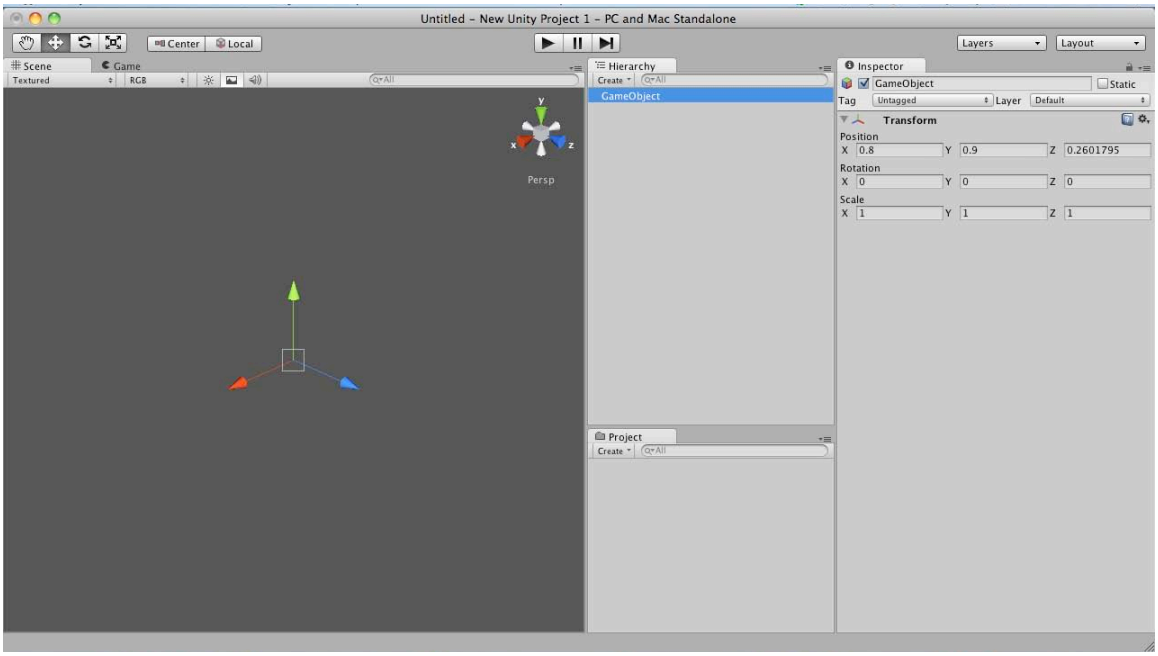

Figura 9 Vista del entorno gráfico de desarrollo de Unity.

Documentación oficial y foros técnicos

En este proyecto se han utilizado para consultar y resolver soluciones técnicas tanto información oficial como foros técnicos.

A continuación se muestran las fuentes más relevantes:

- Unity [Unity 2011b]: página oficial de Unity. Aquí se encuentran todos los recursos oficiales, incluido la página de descarga de la aplicación y la tienda para comprar las licencias.
- Unity Scripting Documentation [Unity 2011c]: documentación oficial de las librerías de Unity (también se incluyen ejempos).

- Mono [Mono 2011]: página oficial de Mono, donde se encuentra la documentación de sus librerías.
- UnityAnswers [UnityAnswer 2011b]: foro oficial de la comunidad de desarrolladores de Unity, basado en el formato de preguntas y respuestas (Q&A, questions and answers).
- Unity & RF Creación de juegos [Unity RF 2011]: foro de desarrolladores de videojuegos y simuladores. Principalmente la participación en este foro se ha centrado en el apartado de "scripting".

En especial, el foro oficial de Unity 3D, UnityAnswers, ha sido de gran ayuda debido a que la tecnología Unity 3D aún es muy joven y no dispone de muchas fuentes o ejemplos prácticos, por lo que se hace imprescindible la ayuda y cooperación entre desarrolladores y artistas 3D. Por esta razón durante la fase de implementación el desarrollador de este proyecto se llegó a posicionar dentro de los 150 miembros más activos de esta comunidad global.

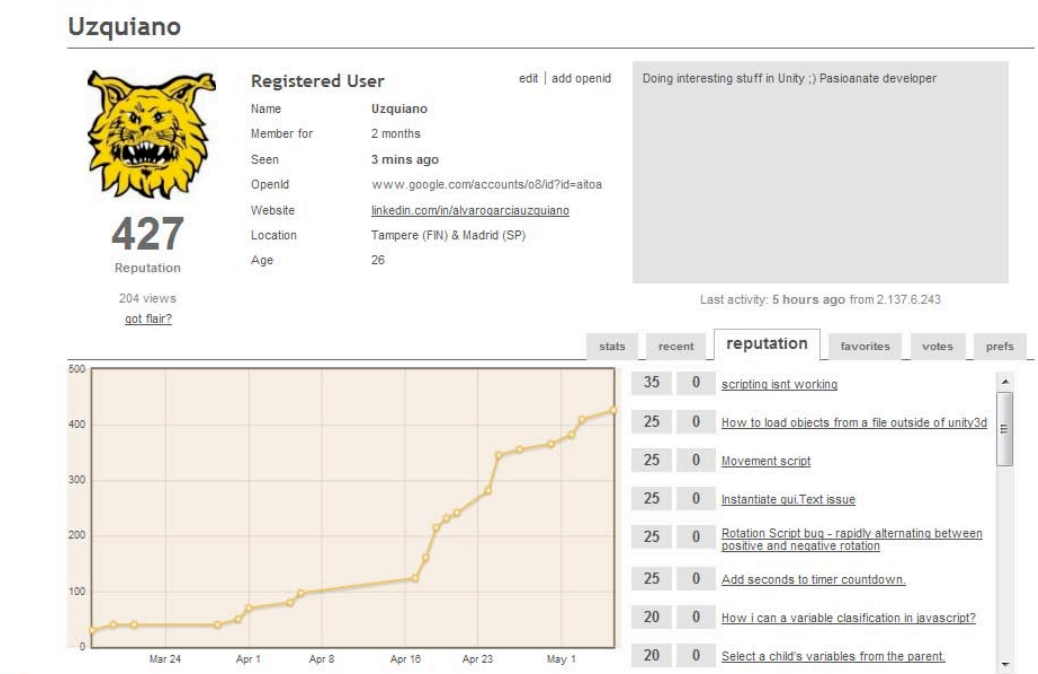

Figura 10 Pantalla con información de usuario en *UnityAnswer* del desarrollador Unity del proyecto

2.7.2 Extensiones de Unity 3D

El desarrollo de alguno de los elementos visuales del proyecto tales como el terreno, carreteras y caminos, y ríos, han hecho necesario el uso de algunas extensiones de Unity: "Terrain Toolkit" [*Six Times Nothing* 2009a], "River Tool" [*Six Times Nothing* 2009b] y "Road/Path Tool" [*Six Times Nothing* 2009c], que se describen a continuación:

Terrain Toolkit

Esta extensión fue creada para el "*2009 Unity Summer of Code"* por el grupo independiente de desarrollo de juegos *Six Times Nothing*. Actualmente, la herramienta está disponible para su descarga de forma gratuita [*Six Times Nothing* 2009a].

Terrain Toolkit es una solución integrada para la generación de terrenos en el motor de juegos Unity. El objetivo de esta herramienta es la creación de terrenos de forma sencilla a través de la vista de *Editor* de Unity.

Figura 11 Captura de pantalla de la extensión *Terrain Toolkit*

En el proyecto se ha utilizado esta herramienta para crear los escenarios donde se lleva a cabo la simulación. En la Figura 12 se muestra una imagen del escenario Carretera, aquí se puede apreciar una pequeña subida y en la parte exterior de la pista de tierra algo de maleza y dos tipos diferentes de árboles.

Figura 12 Imagen del escenario Carretera. Terreno creado con Terrain Toolkit

River Tool

Esta extensión fue creada en el año 2009 por Chris Morris y aunque se creo expresamente para el desarrollo del juego "Dawn Of The Tyrant" del estudio *Six Times Nothing*, con el tiempo ha pasado a ser gratuita y pública [*Six Times Nothing* 2009b].

La herramienta *River Tool* permite crear ríos en el motor de juegos Unity 3D a través de la vista de *Editor*. Para ello esta herramienta deforma el terreno necesario para crear por ejemplo el cauce.

Esta herramienta ha sido utilizada en la aplicación para crear un arroyo que desemboca en un gran lago, una imagen de la desembocadura se puede ver en la Figura 13. Esta imagen pertenece al escenario Bosque.

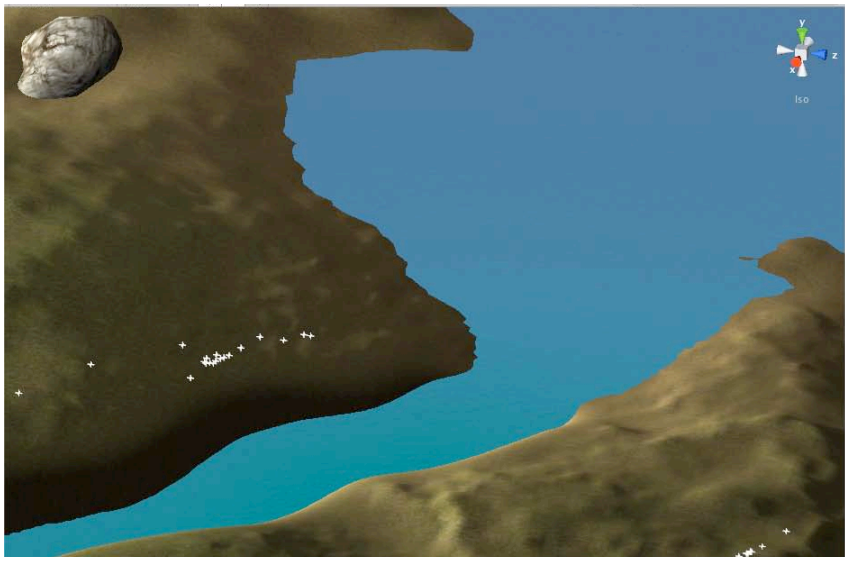

Figura 13 Imagen de un río desembocando en un lago en uno de los escenarios de simulación de la aplicación

A continuación en la Figura 14 se muestra el proceso de creación de un río. En esta imagen se muestra el proceso de diseño, se puede ver cómo se crean los meandros del río.

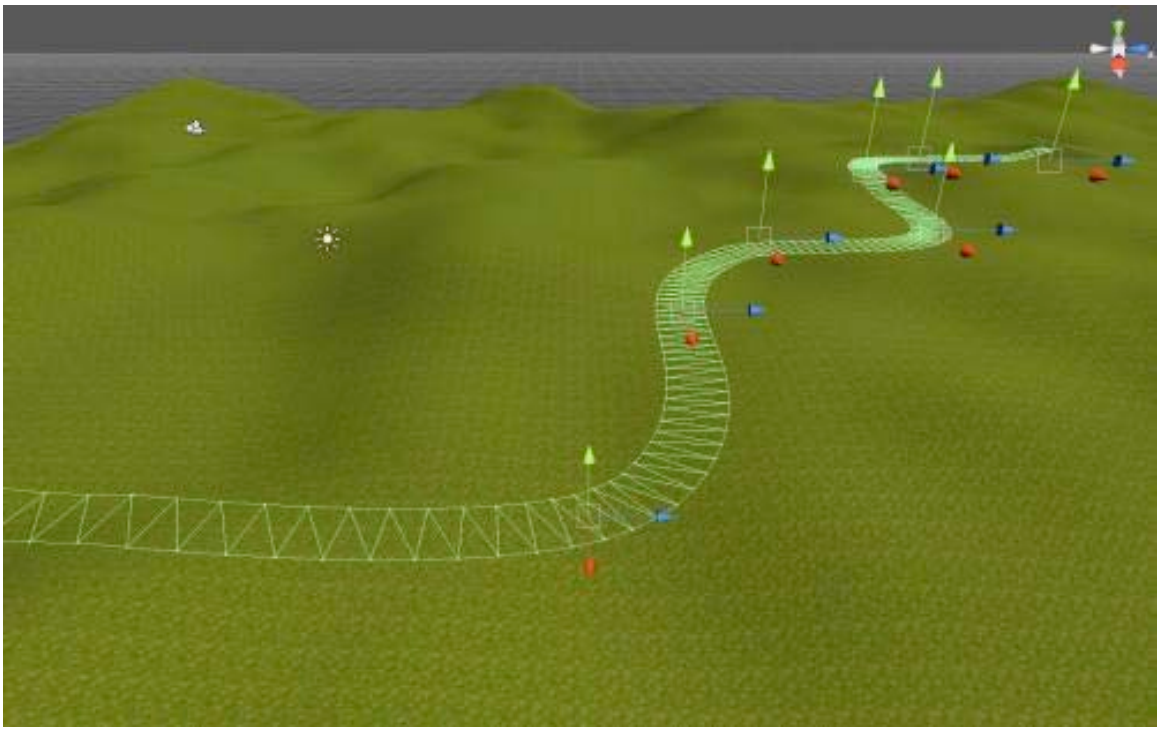

Figura 14 Vista del proceso de creación de un río con la herramienta *River Tool*

Después de crear los elementos básicos de un río, cauce y torrente, se puede añadir más realismo añadiendo más modelos 3D al entorno, como pueden ser piedras, árboles o hierba. Un ejemplo de estas mejoras se puede ver en la Figura 15

Figura 15 Ejemplo de escenario fluvial

Road/Path Tool

La herramienta *Road/Path Tool* permite crear carreteras o montañas en el motor de juegos Unity 3D a través de la vista de *Editor*. Esta herramienta es gratuita y se puede descargar desde la página del estudio *Six Times Nothing* [*Six Times Nothing* 2009c].

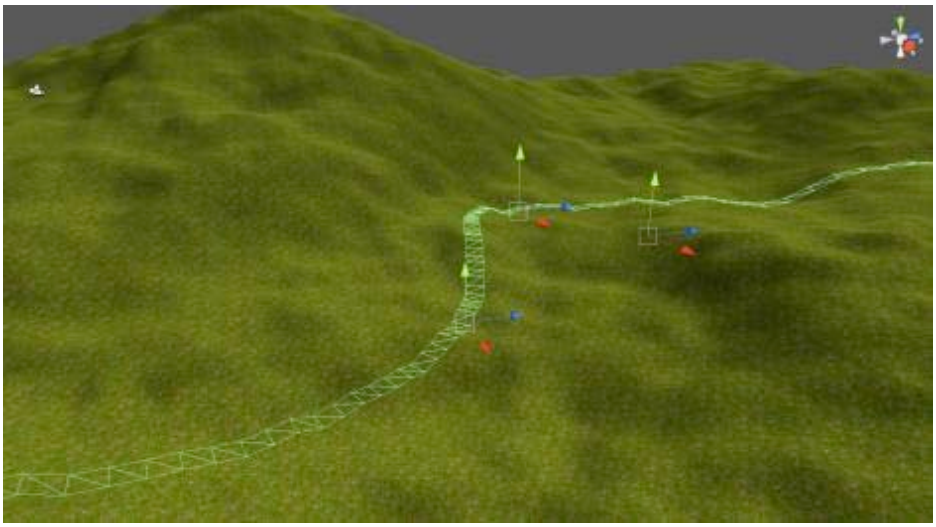

Figura 16 Vista del proceso de creación de una carretera/camino con la herramienta *Road/Path Tool*

Como se puede apreciar en la Figura 17 esta herramienta allana ligeramente el terreno para obtener un resultado más realista.

Como nota para posibles futuros desarrolladores se ha de señalar que este "allanamiento" automático supuso un problema en este proyecto dado que fue la primera vez que se utilizó este plugin. El problema en sí ocurre cuando el algoritmo que se encarga de allanar el terreno se encuentra con otros elementos 3D con el que "colisiona", cancelando la construcción de la carretera y generando un mensaje de error poco aclarativo. La recomendación del equipo de desarrollo de este proyecto es generar las carreteras o caminos inmediatamente después de generar el terreno para evitar problemas de este tipo.

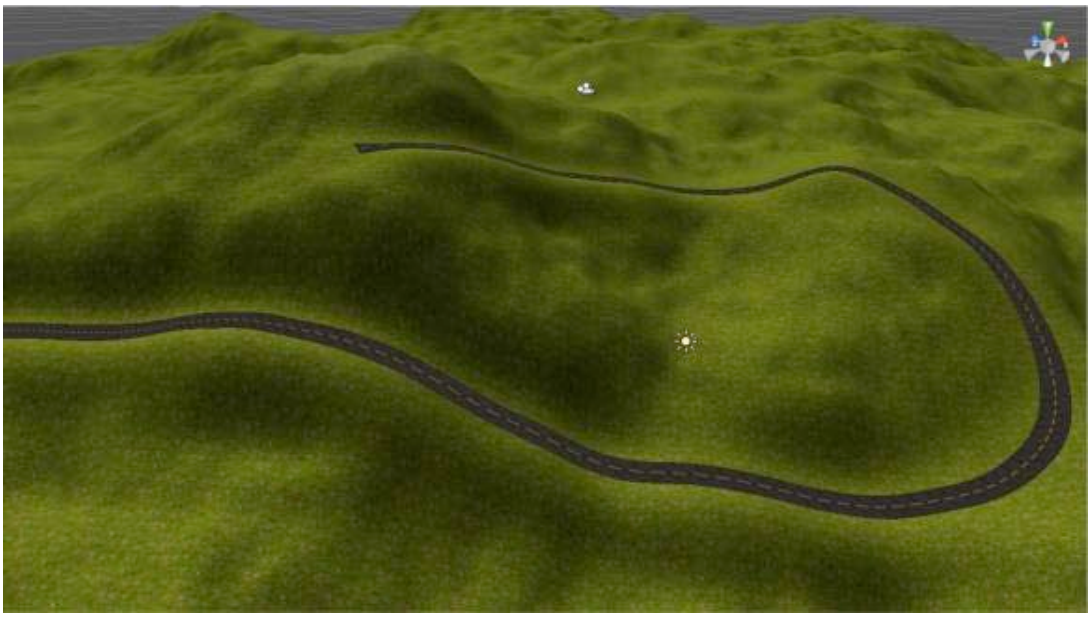

Figura 17 Ejemplo de carretera

En la figura 15 se muestra un ejemplo de camino.

Figura 18 Ejemplo de camino

2.7.3 Otras herramientas

A parte de las herramientas anteriormente citadas, que fueron fundamentales para el desarrollo del proyecto, en este apartado se enumerarán brevemente otras herramientas, que aunque su papel no fue relevante, fueron de gran utilidad en la gestión del proyecto.

Dropbox

Dropbox es un servicio de alojamiento de archivos multiplataforma en la "nube". El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre diferentes equipos, y además compartir archivos y carpetas con otros usuarios.

Dropbox ha sido de gran utilidad para el control de versiones y nos ha permitido trabajar de manera cómoda y segura, en el proyecto, desde diferentes equipos y sistemas operativos ya que el proyecto se ha desarrollado utilizando fundamentalmente 2 equipos con distinto sistema operativo, Windows Vista y Mac OS, como se indica en la sección 3.4.1 Estimación de costes hardware.

TortoiseSVN

TortoiseSVN es un cliente de *Subversion*, sistema de control de versiones, de código abierto que gestiona archivos y directorios en el tiempo.

En este proyecto se ha utilizado de forma asíncrona para guardar versiones del proyecto e intercambiar información con el tutor. Es decir, cumplido un hito del proyecto se procedía a guardar una versión manualmente. Se ha utilizado la versión 1.6.7 para el sistema operativo Windows Vista.

Capítulo 3

Gestión del proyecto

3.1 Introducción

En este capítulo se explica cómo se ha gestionado el proyecto, describiendo la metodología de desarrollo empleada, la planificación del proyecto y una estimación de los costes incurridos.

Como introducción se quiere recordar la definición que el “Project Management Institute” proporciona sobre el término “proyecto”: “Un proyecto es un esfuerzo temporal con la finalidad de lograr un único producto o servicio, con un comienzo y final definido y objetivos específicos que, cuando son logrados significa que se ha completado” [PMBOK 2004].

3.2 Metodología de desarrollo

Para el desarrollo de este proyecto se ha decido usar una metodología ágil, más concretamente Scrum. Esta metodología tiene como pilares básicos los cuatro puntos de su manifiesto, que se muestra a continuación:

Manifiesto Ágil

"Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a valorar:

- *A los individuos y su interacción, por encima de los procesos y las herramientas.*
- *El software que funciona, por encima de la documentación exhaustiva.*
- *La colaboración con el cliente, por encima de la negociación contractual.*
- *La respuesta al cambio, por encima del seguimiento de un plan.*

Aunque hay valor en los elementos de la derecha, valoramos más los de la izquierda."

Esta metodología propone el desarrollo incremental de la funcionalidad del sistema, entregando en cada *Sprint* (periodo de tiempo de entre 2 y 8 semanas) una versión de la aplicación potencialmente utilizable. Como se muestra en la Figura 19, a diferencia del clásico modelo en espiral, la salida de cada *Sprint* es teóricamente entregable.

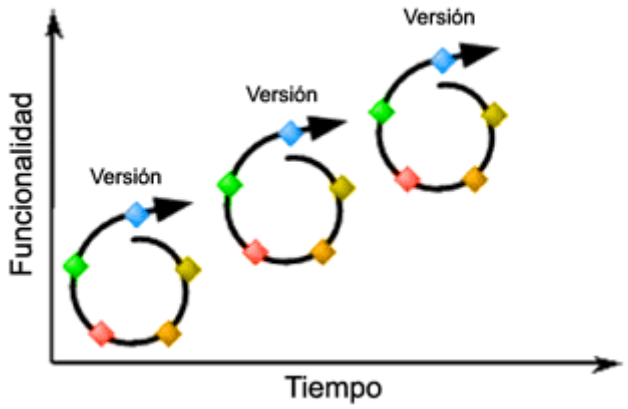

Figura 19 Esquema Funcionalidad vs Tiempo de Scrum

3.3 Organización del proyecto

El desarrollo de este proyecto se ha dividido en tres fases principales o sprints, que han quedado distribuidos en el tiempo de la siguiente forma:

- Sprint 1: del 28 de febrero al 21 de marzo
- Sprint 2: del 22 de marzo al 15 de abril
- Sprint 3: del 16 de abril al 23 de mayo

Hay que añadir que también ha habido dos fases adicionales no estrictamente de desarrollo como es la fase inicial y la fase de documentación.

Por tanto, las etapas de las que ha constado el proyecto son:

- Fase inicial (fase 1 en los cuadros de seguimiento del proyecto), se ha definido el alcance del proyecto y se han estudiado las soluciones existentes en el mercado.
- Desarrollo de funcionalidad en Unity 3D (fase 2 en los cuadros de seguimiento del proyecto), se ha desarrollado la funcionalidad del simulador así como todas las pantallas de la aplicación. Su mayor actividad se ha dado en el primer sprint
- Desarrollo de arquitectura (fase 3 en los cuadros de seguimiento del proyecto), se ha desarrollado el motor SCXML-UnityScript, el motor IMS-QTI Lite – UnityScript, y la arquitectura de soporte multi-idioma.
- Desarrollo de entorno y elementos 3D (fase 4 en los cuadros de seguimiento del proyecto).
- Fase de documentación, donde se ha creado toda la documentación.

El detalle del desarrollo se puede ver en este capítulo desde la Figura 20 hasta la Figura 27.

El seguimiento y comunicación de los integrantes del equipo ha seguido lo establecido según la metodología Scrum. Así, la comunicación ha sido diaria (en persona o email) comentando el trabajo llevado a cabo el día anterior y el planeado para el mismo día.

Además, cada diez días aproximadamente se han mantenido reuniones de seguimiento. También, pero más distanciadas en el tiempo, 4-8 semanas se ha mantenido reuniones de planificación.

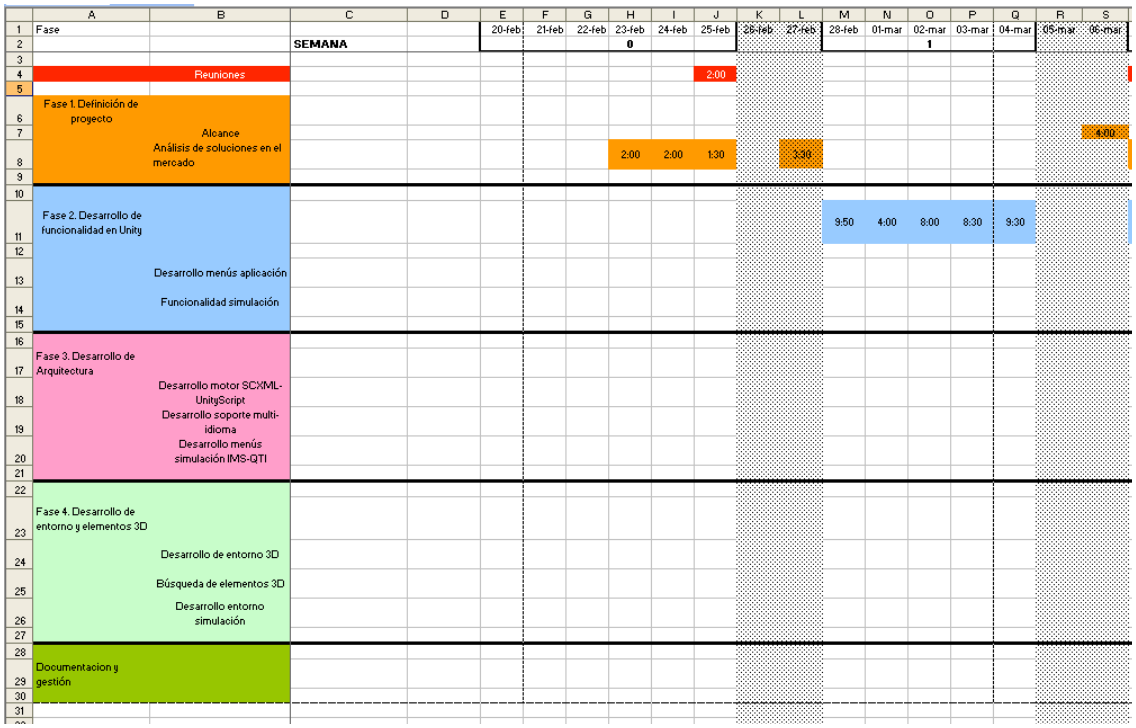

Figura 20 Cuadro de seguimiento del proyecto

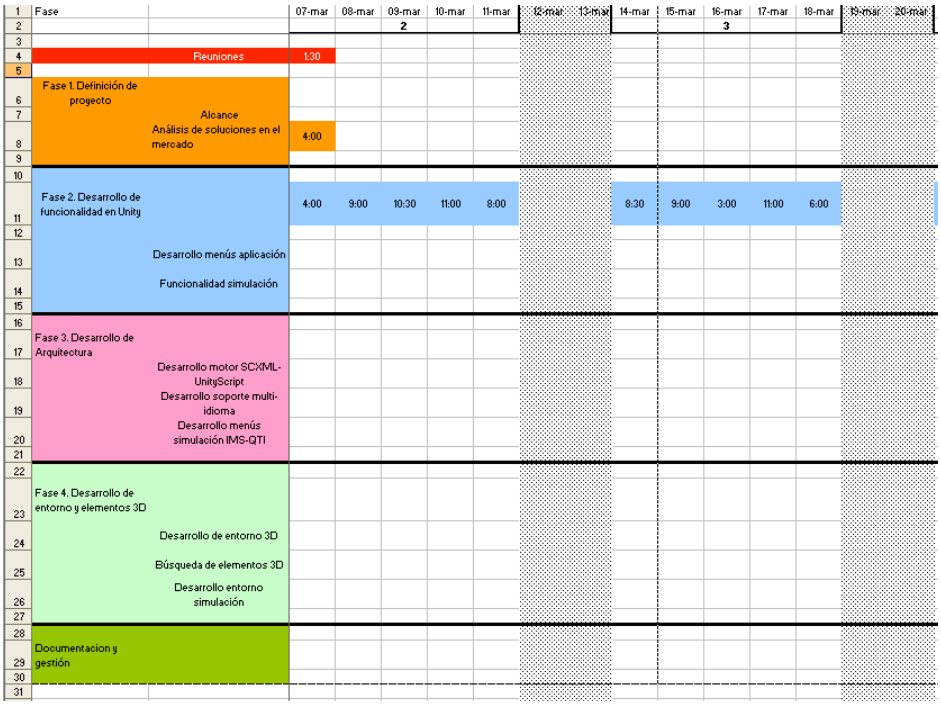

Figura 21 Cuadro de seguimiento del proyecto

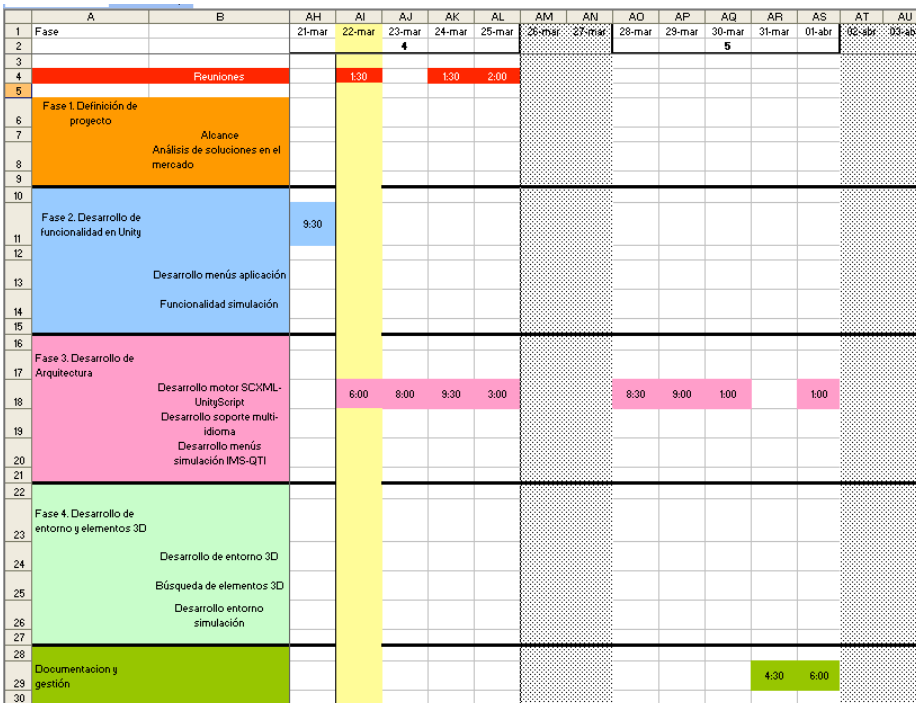

Figura 22 Cuadro de seguimiento del proyecto

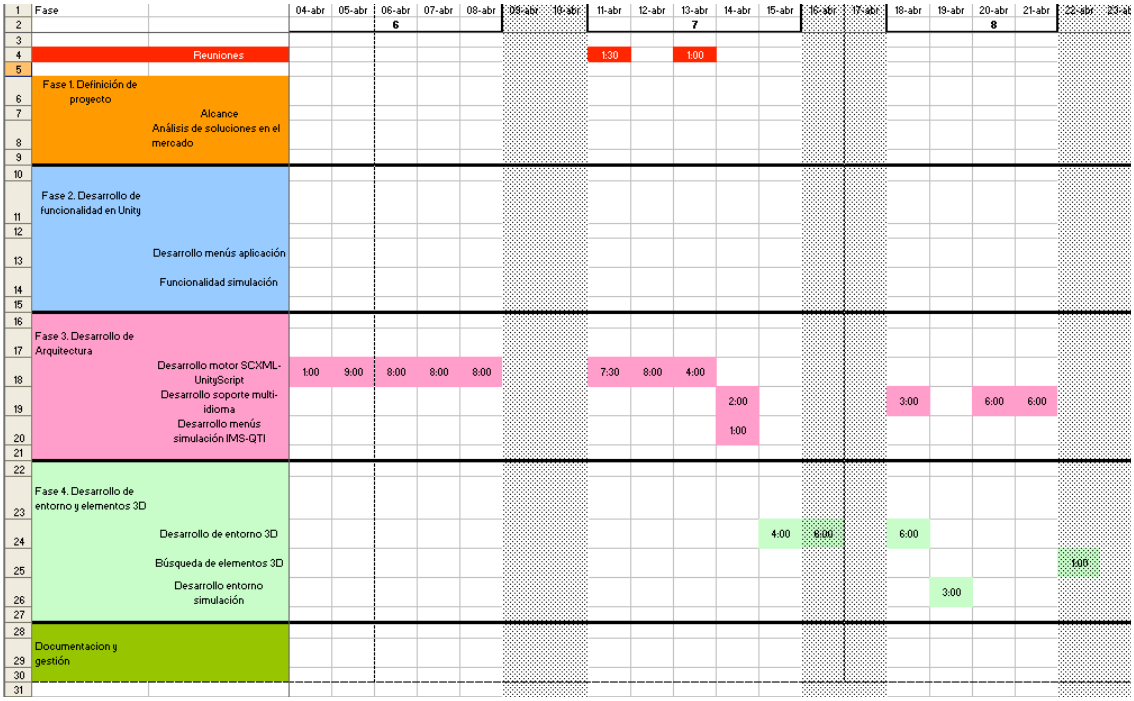

Figura 23 Cuadro de seguimiento del proyecto

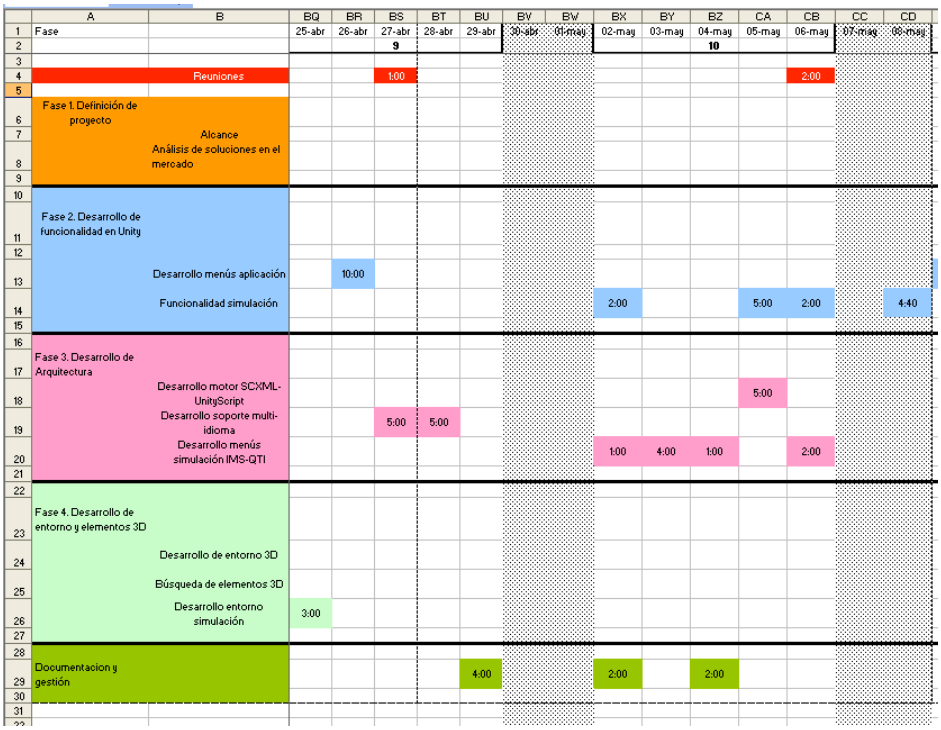

Figura 24 Cuadro de seguimiento del proyecto

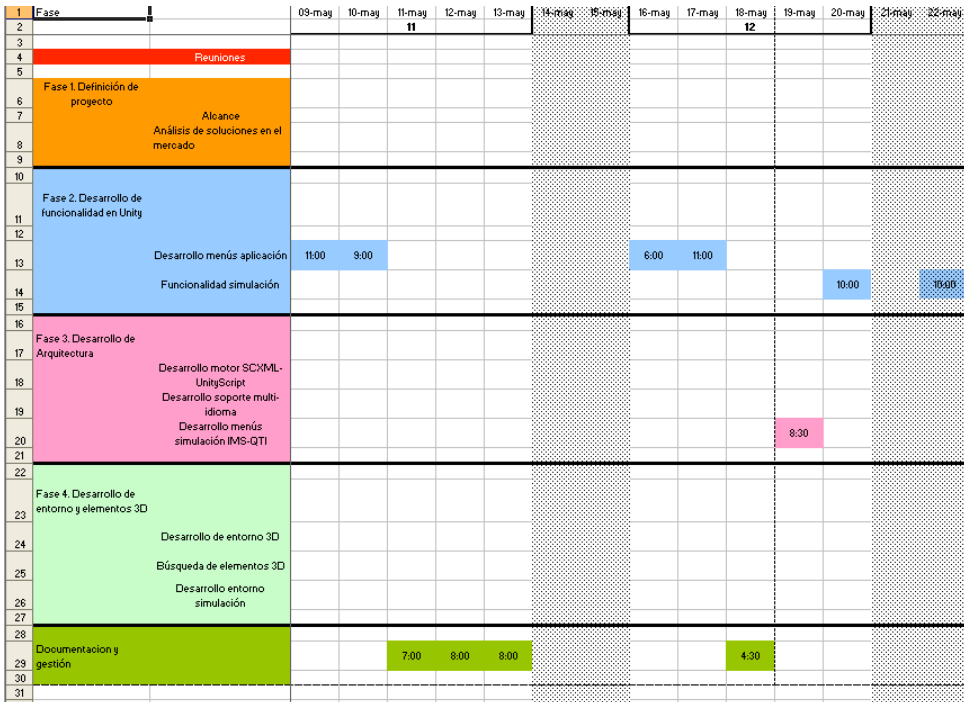

Figura 25 Cuadro de seguimiento del proyecto

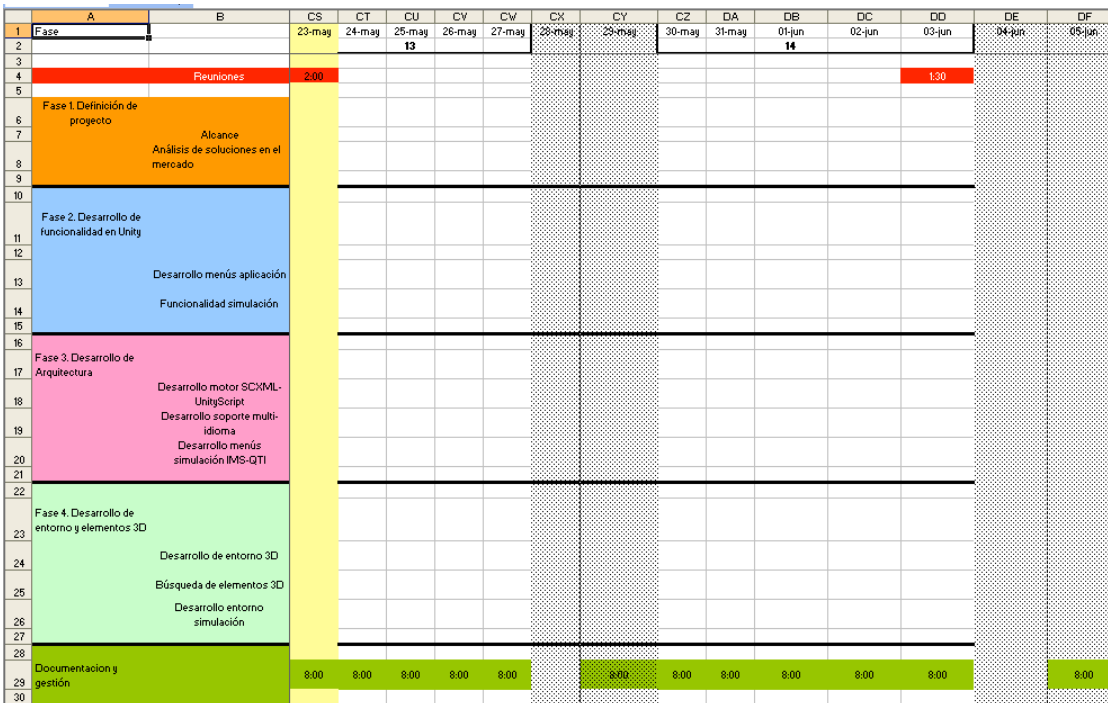

Figura 26 Cuadro de seguimiento del proyecto

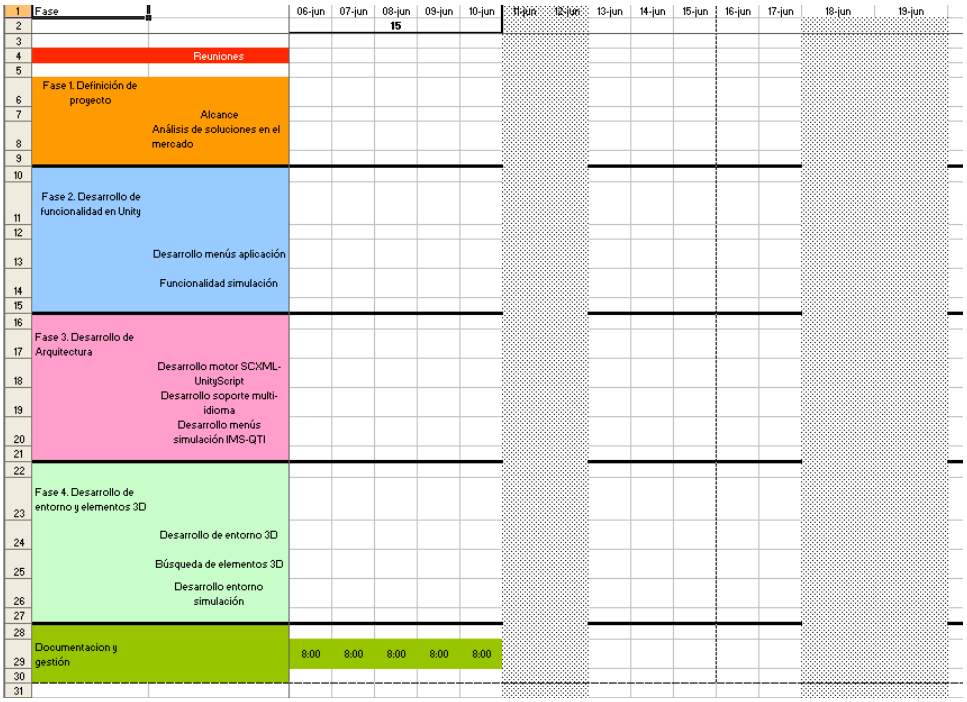

Figura 27 Cuadro de seguimiento del proyecto

Así, tras analizar los recursos utilizados en el desarrollo de la aplicación se presenta la Tabla 3 con el detalle de horas dedicadas a cada fase y la Figura 28 muestra la misma información de forma porcentual.

Dirección de proyecto	65:00:00
Reuniones (desarrollador)	17:00:00
Fase 1. Definición de proyecto	17:00:00
Fase 2. Desarrollo de funcionalidad en Unity	210:00:00
Fase 3. Desarrollo de Arquitectura	149:00:00
Fase 4. Desarrollo de entorno y elementos 3D	32:00:00
Gestión y documentacion del proyecto	182:00:00
TOTAL	672:00:00

Tabla 3 Recursos utilizados (en horas) para el desarrollo del proyecto

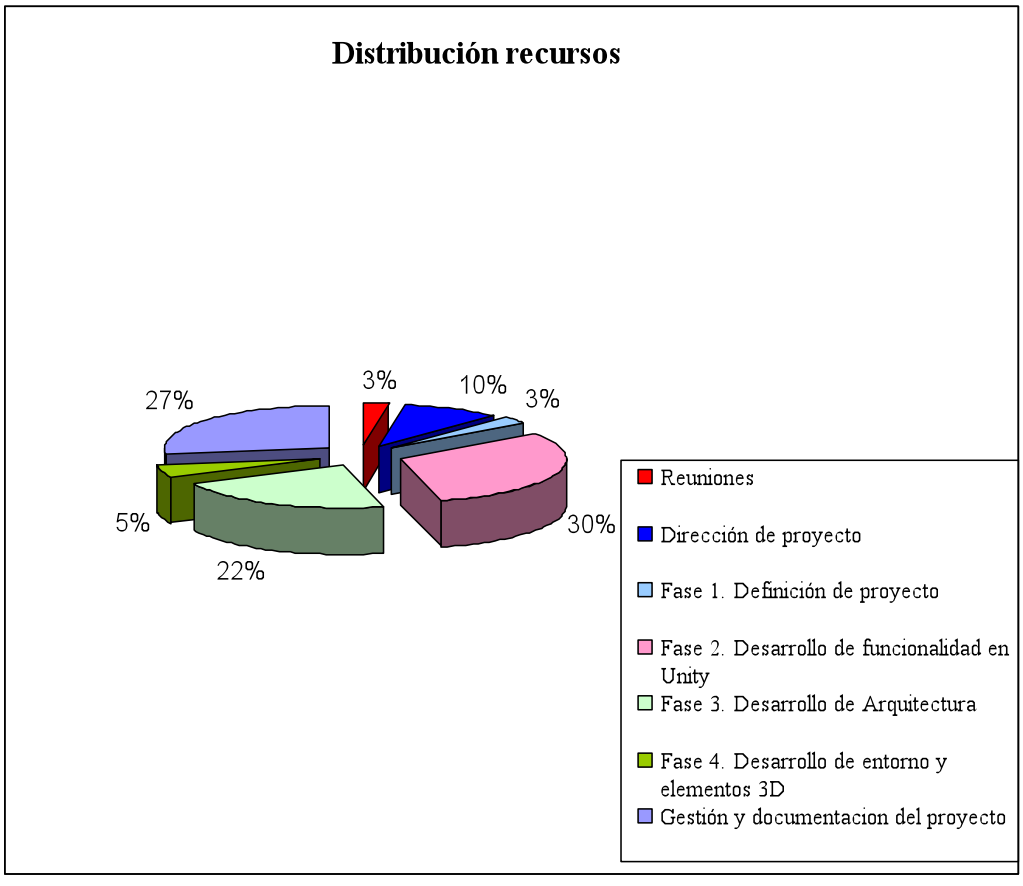

Figura 28 Distribución (%) de recursos humanos en el proyecto

3.4 Estimación de costes

Para calcular el coste estimado del proyecto se deben tener en cuenta tanto los gastos materiales como los costes humanos del desarrollo.

Además de estos costes, se estima que los costes indirectos del proyecto representan el 20% de los costes de recursos humanos.

3.4.1 Estimación de costes hardware

Para el desarrollo de este proyecto se ha utilizado el siguiente equipamiento hardware:

- Ordenador personal:
  - Sistema operativo: Windows Vista Home Basic
  - Procesador: Pentium Dual-Core E5300 @ 2.60GHz
  - Memoria (RAM): 4,00 GB
  - Tarjeta gráfica: NVIDIA GeForce 7300 LE

- Mac:
  - Sistema operativo: Mac OS X 10.6.6
  - Procesador: Intel Core 2 Duo @ 2.10GHz
  - Memoria (RAM): 2,00 GB
  - Tarjeta gráfica: Interl GMA X3100

Recurso	**Coste**
Ordenador personal	550€
MacBook	950€
TOTAL	**1500€**

Tabla 4 Tabla de costes hardware

3.4.2 Estimación de costes software

A continuación se muestra el presupuesto en forma de tabla de las herramientas software utilizadas para el desarrollo del proyecto:

Programas	**Licencia**
Unity 3D	Gratuita
MS Office 2010 Professional Academic	98,99€
Windows Vista Home (student version)	78,99€
Mac OS X Snow Leopard	30€
TOTAL	**207,98 €**

Tabla 5 Tabla de costes software

3.4.3 Estimación de costes de recursos humanos

Para calcular el coste de recursos humanos empleados durante todo el proceso se ha llevado un seguimiento diario, recogido en las figuras entre Figura 20 y Figura 27. El proyecto ha sido desarrollado por un jefe de proyecto (rol desempeñado por la tutora del proyecto) y un sólo ingeniero proyectista (desarrollando él mismo las funciones de: analista, diseñador y desarrollador). A continuación se muestra una relación entre las funciones que ha desempeñado cada perfil, el tiempo invertido en cada uno de ellos y su coste:

Jefe de Proyecto: Se encarga de la gestión del proyecto, su organización, planificación y supervisión a lo largo de todo el desarrollo del mismo.

Analista: Se encarga de obtener y redactar los requisitos, además de modelar los procesos y tareas a codificar.

Diseñador: Su tarea es el diseño de la arquitectura del sistema.

Diseñador gráfico: Responsable de todo lo relacionado con la apariencia del sistema y los modelos 3D.

Desarrollador: Se encarga de la codificación del sistema, así como de la ejecución de las pruebas necesarias del mismo.

Recursos Humanos	Horas	Coste/Hora	Total
Jefe de proyecto	65	40	2600 €
Analista	165	25	4125 €
Diseñador	100	25	2500 €
Diseñador gráfico	60	25	1500 €
Desarrollador	330	20	6600 €
SUBTOTAL			17325 €
I.V.A.		18 %	3118,5€
TOTAL			20443,5€

Tabla 6 Tabla de costes del proyecto

3.4.4 Estimación de coste total del proyecto

El presupuesto total de este proyecto asciende a la cantidad de VEINTISEIS MIL DOSCIENTOS CUARENTA EUROS.

Recursos hardware	1.500€
Recursos software	207,98€
Recursos humanos	20443,5€
Costes indirectos	4088,7€
TOTAL	26.240**€**

Leganés a 10 de Junio de 2011

El ingeniero proyectista

Fdo. Álvaro García Uzquiano

Capítulo 4

Diseño y desarrollo del sistema

4.1 Introducción

En este capítulo se analiza el diseño del sistema simulador implementado en este proyecto.

En un primer lugar, se hace un análisis de los requisitos necesarios que debe cumplir el sistema para después analizar la arquitectura diseñada para cubrir estos requisitos y finalmente se explica como ha sido implementado.

4.2 Análisis de requisitos del sistema

Como se ha explicado en el capítulo de introducción, este proyecto es una prueba de concepto correspondiente a la petición de un simulador de formación para el hospital “La Paz” en Bata (Guinea Ecuatorial). Este simulador tiene como objetivo facilitar el aprendizaje de paramédicos que han de desarrollar su trabajo en situaciones de

emergencia en el entorno geográfico de Guinea Ecuatorial. Además, el simulador debe ser configurable desde el punto de vista del personal docente para poder desarrollar nuevas escenas de simulación sin la necesidad de conocimientos sobre la tecnología subyacente.

Por otro lado, la universidad quiere una arquitectura flexible que se pueda reutilizar para otros modelos de negocio o migrar otros simuladores implementados con tecnologías 2D [Adobe Flash]. Esta flexibilidad se pretende conseguir a través de externalizar, o hacer independiente, el contenido de la lógica del simulador de la aplicación en sí. De esta forma se pueden conseguir desarrollos rápidos cambiando sólo los ficheros externos donde se define la lógica del simulador.

Estos requisitos presentarán una visión general de la aplicación, sin ahondar en aspectos técnicos, estableciendo las principales funcionalidades y restricciones, sirviendo de base a posteriores versiones de simuladores de formación.

La definición de requisitos está ligada a la definición de funcionalidad a mostrar en la prueba de concepto por lo que, en este caso todos los requisitos definidos serán de tipo funcional.

Para la toma de requisitos se han realizado una serie de reuniones presenciales donde se exponían las necesidades que la aplicación debía cubrir. En una primera reunión se definieron los objetivos globales del proyecto. Este proyecto, al haberse desarrollado dentro de una metodología ágil (Scrum) ha permitido que al inicio de cada “sprint” o ciclo se hayan podido revisar, actualizar y depurar alguno de los objetivos definidos en un primer momento.

A continuación se muestran mediante tablas los requisitos identificados para este proyecto.

Identificador	Req-001
Nombre	Fichero de definición de flujo
Descripción	El flujo del escenario de simulación se debe definir en un fichero externo a la aplicación y podrá ser modificable en la fase de producción. Este fichero debe estar basado en un estándar.
Precondiciones	Debe tratarse de un estándar.
Procedimiento	Se debe definir el fichero de flujo de simulación antes de iniciar la aplicación.
Observaciones	Se estudian los estándares IMS-LD y SCXML.

Tabla 7 Req-001 Fichero de definición de flujo

Identificador	Req-002
Nombre	Fichero de definición del contenido educativo
Descripción	El contenido educativo del escenario de simulación se debe definir en un fichero externo a la aplicación y podrá ser modificable en la fase de producción. Este fichero debe estar basado en un estándar.
Precondiciones	Debe tratarse de un estándar.
Procedimiento	Se debe definir el fichero de contenido educativo antes de iniciar la aplicación. El nombre del fichero en uso debe definirse en el fichero de contenido de flujo.
Observaciones	Se estudia utilizar el estándar IMS-QTI y IMS-QTI Lite.

Tabla 8 Req-002 Fichero de definición del contenido educativo

Identificador	Req-003
Nombre	Fichero de datos de instrumentos de diagnóstico
Descripción	Los datos obtenidos (virtualmente o de forma real) a través de instrumentos de diagnóstico en un fichero externo.
Precondiciones	Fichero con formato de texto plano para facilitar que en futuras versiones este fichero puede ser escrito por una aplicación externa.
Procedimiento	El nombre del fichero en uso debe definirse en el fichero de contenido de flujo.
Observaciones	

Tabla 9 Req-003 Fichero de definición de datos de instrumentos de diagnóstico

Identificador	Req-004
Nombre	Arranque de la aplicación
Descripción	Al iniciar la aplicación se mostrará la pantalla de menú de inicio. En este menú de inicio se presentarán las siguientes opciones: (1) Entrar; (2) Nuevo usuario; (3) Idioma; (4) Salir.
Precondiciones	
Procedimiento	Se iniciará la aplicación haciendo doble “click” en el icono de la aplicación. Una vez en la pantalla de menú de inicio el usuario deberá seleccionar mediante el uso del ratón la opción deseada.
Observaciones	

Tabla 10. Req-004 Arranque de la aplicación

Identificador	Req-005
Nombre	Validación de sesión
Descripción	El usuario deberá validar su sesión mediante los campos ‘nombre de usuario’ y ‘contraseña’ para poder acceder a los escenarios de simulación.
Precondiciones	Acceder a la opción (1) “Entrar” desde el menú de inicio.
Procedimiento	El usuario debe introducir en los campos de texto que aparecen en la pantalla su ‘usuario’ y ‘contraseña’. A continuación deberá pulsar el botón ‘Continuar’.
Observaciones	En el caso de una validación de sesión exitosa se pasará directamente a la pantalla de selección de escenario de simulación.

Tabla 11. Req-005 Validación de sesión

Identificador	Req-006
Nombre	Crear nueva cuenta de usuario
Descripción	Un nuevo usuario podrá crear su cuenta de usuario.
Precondiciones	Acceder a la opción (2) “Nuevo usuario” desde el menú de inicio.
Procedimiento	El usuario debe introducir en los campos de texto que aparecen en la pantalla su: “nombre y apellidos”, ‘nombre de usuario’ y ‘contraseña’. A continuación deberá pulsar el botón ‘Continuar’.
Observaciones	En el caso de una creación de cuenta exitosa se pasará directamente a la pantalla de selección de escenario de simulación.

Tabla 12. Req-006 Creación de nueva cuenta de usuario

Identificador	Req-007
Nombre	Cambiar idioma de la aplicación
Descripción	El usuario cambia el idioma de la aplicación.
Precondiciones	Acceder a la opción (3) “Idioma” desde el menú de inicio.
Procedimiento	El usuario pulsa en la opción del menú, si el idioma actual es castellano cambia a inglés y viceversa.
Observaciones	Los idiomas de la aplicación son castellano e inglés. Los textos en los diferentes idiomas se encuentran en un mismo fichero. El texto de cada idioma se especifica con su respectiva etiqueta.

Tabla 13. Req-007 Cambio de idioma de la aplicación

Identificador	Req-008
Nombre	Cierre de la aplicación
Descripción	El usuario sale de la aplicación.
Precondiciones	Acceder a la opción (4) “Salir” desde el menú de inicio.
Procedimiento	El usuario pulsa la opción del menú.
Observaciones	

Tabla 14. Req-008 Cierra de la aplicación

Identificador	Req-006
Nombre	Selección de caracteres
Descripción	El usuario podrá elegir el sexo del personaje en la pantalla de simulación.
Precondiciones	Acceder con una cuenta de usuario a la pantalla de selección de simulación.
Procedimiento	El usuario puede elegir si el personaje es hombre o mujer.
Observaciones	

Tabla 15. Req-009 Selección de sexo de personaje

Identificador	Req-010
Nombre	Selección de simulación
Descripción	El usuario podrá elegir la simulación a través de un listado de escenarios o técnicas a entrenar.
Precondiciones	Acceder con una cuenta de usuario a la pantalla de selección de simulación.
Procedimiento	El usuario puede elegir seleccionar la simulación por escenarios o por técnicas, una vez escogida la categoría se debe seleccionar la simulación entre las opciones mostradas en el menú desplegado.
Observaciones	La selección por técnicas o escenarios es uno de los requisitos principales del cliente, cuyo objetivo es hacer el entrenamiento más intensivo.

Tabla 16. Req-010 Selección de simulación por “Escenario” o “Técnica”

Identificador	Req-011
Nombre	Análisis de resultados
Descripción	Tras la simulación aparecerá una pantalla de evaluación de resultados. La información que aparece en la pantalla se guardará en un fichero externo.
Precondiciones	Haber terminado un escenario de simulación.
Procedimiento	Tras finalizar el escenario de simulación aparecerá automáticamente.
Observaciones	

Tabla 17. Req-011 Análisis de resultados

Identificador	Req-012
Nombre	Eventos por proximidad
Descripción	En el escenario de simulación se pueden incluir eventos según la posición del personaje con respecto a los objetos del “mundo”.
Precondiciones	El usuario se debe encontrar realizando una simulación. Para que este tipo de evento se dispare, el personaje debe acercarse a los diferentes objetos.
Procedimiento	Se debe activar el componente de proximidad en los objetos deseados.
Observaciones	

Tabla 18. Req-012 Eventos de proximidad durante simulación

Identificador	Req-013
Nombre	Panel de feedback
Descripción	En el escenario de simulación se puede mostrar un panel de “feedback” compuesto por una imagen y un texto.
Precondiciones	El usuario se debe encontrar realizando una simulación. Para que este panel se muestre, el usuario debe haber realizado una acción que tiene asignada un elemento de feedback.
Procedimiento	La información del panel se debe configurar bien en el archivo de definición de flujo o de definición de contenido educativo.
Observaciones	

Tabla 19. Req-013 Panel de feedback durante simulación

Identificador	Req-014
Nombre	Menú de acción
Descripción	En el escenario de simulación se pueden mostrar menús de acción con los que el alumno tenga que interactuar.
Precondiciones	El usuario se debe encontrar realizando una simulación. Para que este panel se muestre, el usuario debe haber realizado una acción que tiene asignada un menú de acción.
Procedimiento	La configuración del menú de acción se debe definir en el archivo de definición de flujo y su contendido se debe definir en el fichero de contenido educativo.
Observaciones	Se pueden añadir eventos a cada opción del menú.

Tabla 20. Req-014 Menú de acción

Identificador	Req-015
Nombre	Panel de diálogo
Descripción	En el escenario de simulación se puede mostrar un panel de diálogo para representar los diálogos entre personajes.
Precondiciones	El usuario se debe encontrar realizando una simulación. Para que este panel se muestre, el usuario debe haber realizado una acción que tiene asignada un panel de diálogo.
Procedimiento	La configuración del panel de diálogo se debe definir en el archivo de definición de flujo o en el de definición de contenido educativo.
Observaciones	

Tabla 21. Req-015 Panel de diálogo

Identificador	Req-016
Nombre	Cronómetro de cuenta atrás
Descripción	En el escenario de simulación se muestra un cronómetro de cuenta atrás indicando el tiempo límite para terminar los objetivos.
Precondiciones	El usuario se debe encontrar realizando una simulación.
Procedimiento	Cuando se inicia el escenario de simulación se activa el cronómetro.
Observaciones	

Tabla 22. Req-016 Cronómetro de cuenta atrás

Identificador	Req-017
Nombre	Panel de puntuación
Descripción	En el escenario de simulación se muestra un panel indicando la puntuación (acciones correctas e incorrectas) conseguidas hasta el momento por el usuario.
Precondiciones	Estar realizando la simulación.
Procedimiento	Cuando el usuario realiza una acción puntuable se actualiza el panel de puntuación.
Observaciones	

Tabla 23. Req-017 Panel de puntuación

Identificador	Req-018
Nombre	Panel de progreso
Descripción	En el escenario de simulación se muestra un panel indicando la puntuación total conseguidas hasta el momento por el usuario y el máximo que puede conseguir en el escenario.
Precondiciones	Estar realizando la simulación.
Procedimiento	Cuando el usuario realiza una acción correcta se actualiza el panel de puntuación.
Observaciones	

Tabla 24. Req-018 Panel de progreso

4.3 Arquitectura

Uno de los principales requisitos que debe cumplir el sistema es que el contenido de los escenarios de simulación debe estar separado, externo, del sistema simulador, es decir, del motor que indica qué elementos gráficos mostrar, su contenido, y en qué orden.

Cuando se habla de contenido de los escenarios de simulación, se hace referencia a todo lo que tiene que ver con la lógica de la temática del simulador, es decir, el contenido educativo, las tareas y ejercicios a llevar a cabo, material multimedia, etc... En el caso de este proyecto, la temática del simulador son las emergencias sanitarias.

Con esta separación, lo que se pretende es que se puedan generar nuevos escenarios de simulación (con diferentes objetivos y contenido educativo) sin necesidad de volver a la fase de diseño del simulador y por tanto, sin necesidad de tener conocimientos técnicos sobre la tecnología subyacente. A este tipo de reutilización, únicamente de modificación de contenidos, se la llamará reutilización parcial.

Por tanto, se va a optar por una arquitectura de capas como se muestra en la Figura 29. Cada capa tiene una misión específica y juntas componen la aplicación final.

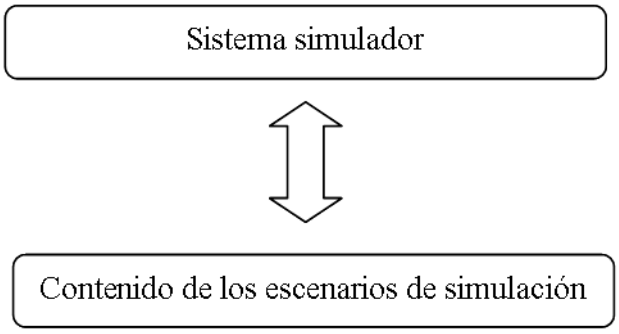

Figura 29 Arquitectura de capas

Anteriormente se ha descrito el tipo de reutilización parcial, que consiste en modificar el contenido de los escenarios, y más adelante en este mismo punto (en el apartado Subsistema de contenidos internos) se entrará en detalle en la reutilización total, que consiste en modificar también su estructura interna (sistema simulador) creando más pantallas y modificando los flujos de la aplicación. Para desarrollar este tipo de reutilización sí es necesario conocer, al menos, de forma básica la tecnología base, en este caso Unity 3.

A continuación se describirán estas dos capas de la aplicación en detalle.

4.3.1 Definición de niveles de arquitectura

Como antes se ha introducido, la arquitectura del sistema se puede dividir en dos capas. Así, la primera capa está compuesta por una serie de ficheros externos que definen los distintos escenarios de simulación. Mientras que la segunda capa, el sistema simulador, se encarga de reproducir los escenarios que se le asignen, definidos en los ficheros de la primera capa. Describiendo este sistema mediante un símil, el sistema simulador sería el reproductor de vídeo y los archivos que componen la información de escenarios de simulación las pistas de vídeo. Así, como se muestra en la Figura 30, se pueden distinguir fácilmente los dos bloques principales del sistema.

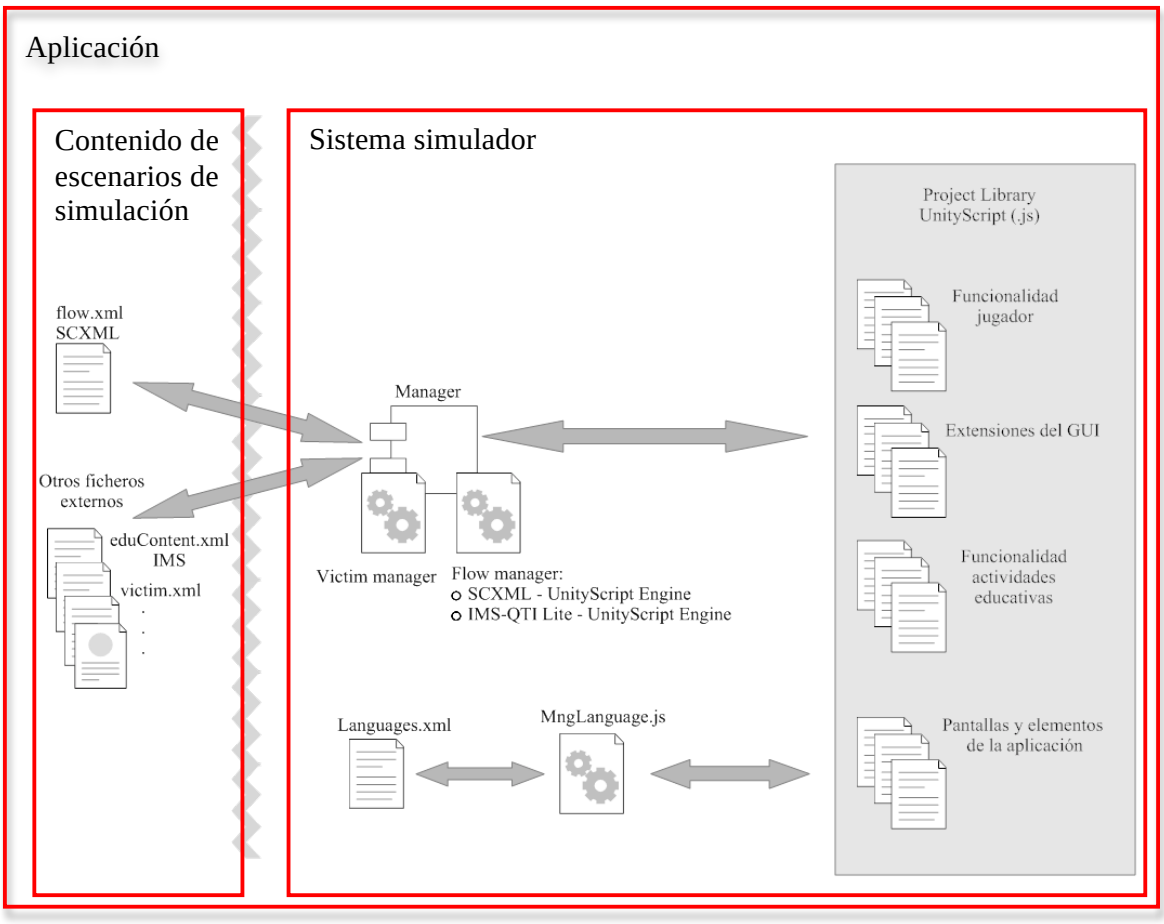

Figura 30 Arquitectura de la aplicación

Además, la capa del Sistema simulador está compuesta por tres subsistemas:

- Subsistema **intérprete de contenidos** de escenarios de simulación: es el encargado de interpretar la información suministrada por la capa anterior: el contenido de escenarios de simulación.
- Subsistema de **entorno de trabajo**: su objetivo es conseguir a partir de la salida del subsistema anterior reproducir las instrucciones obtenidas y ejecutar la funcionalidad asignada durante el escenario de simulación.
- Subsistema de **contenidos internos**: permite representar los contenidos de las pantallas de la aplicación (botones, menús, etc…) definidos en un archivo interno con formato XML. Este archivo se explicará más en detalle en el punto 4.4.3 Ficheros de interfaz de la aplicación.

En el siguiente apartado se analizan individualmente los sistemas y subsistemas introducidos en el punto anterior.

Capa de Contenido de escenarios de simulación

El bloque de Contenido de escenarios de simulación está compuesto por los ficheros que describen los estados existentes en el escenario de simulación, el contenido educativo, la descripción del historial médico de la víctima, así como otros ficheros con información adicional sobre el escenario de simulación. Todos estos ficheros se describen y detallan en el punto 4.4.1 Ficheros de contenido del escenario de simulación.

Capa del Sistema simulador

Esta capa, mediante sus tres subsistemas, aporta toda la funcionalidad y capacidad a la aplicación. A continuación se describen cada uno de ellos.

Subsistema intérprete de contenidos de escenarios de simulación

Como se ha indicado previamente, los ficheros que forman la capa de contenido de los escenarios de simulación tiene la información que debe aparecer durante la simulación, pero estos ficheros por sí mismos no tienen funcionalidad en el sistema. Es necesario, por tanto, interpretarlos y adecuarlos para poder utilizar la información contenida en el sistema simulador.

Por esta razón, este subsistema analiza, procesa y almacena esta información en estructuras para que los demás subsistemas y partes de la aplicación puedan trabajar con esta información. Así, como se ve en la Figura 31 a través del elemento “Manager” se obtiene la información de los archivos externos y después de procesar la información se obtienen estructuras de datos, con información sobre los estados de simulación, los contenidos educativos y la información relativa al historial médico de las víctimas

presentes en el simulador. Las dos primeras estructuras son generadas por el script *Engine.js*, mientras que la última es creada por el script *MngVictim.js*.

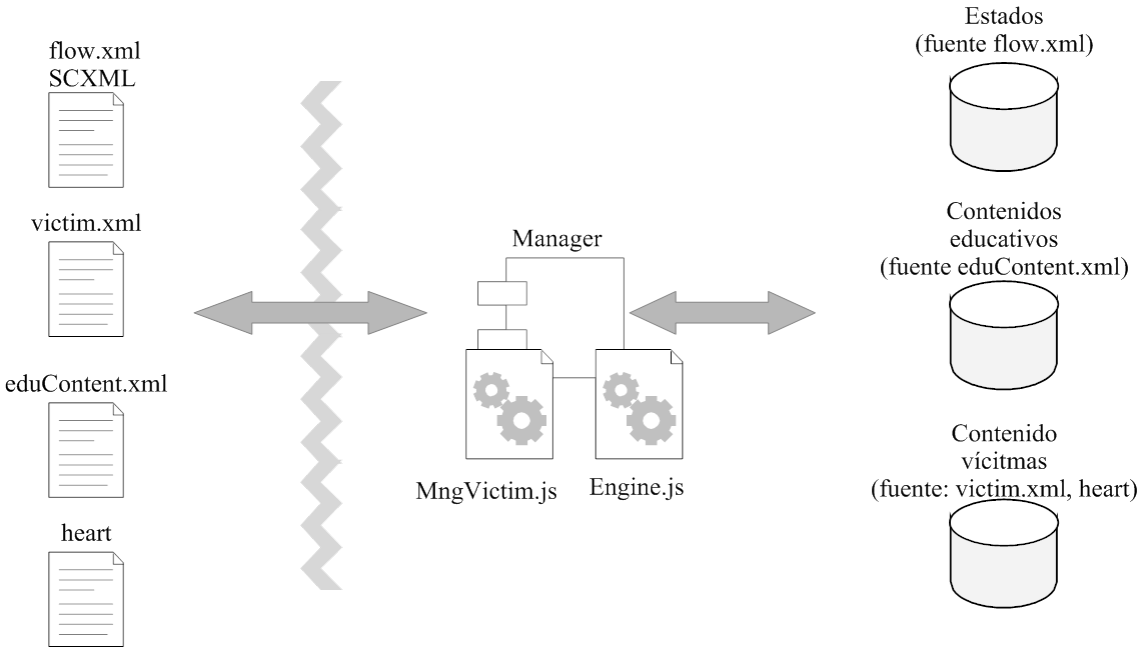

Figura 31 Arquitectura del subsistema intérprete de los ficheros de contenidos de escenarios de simulación

A continuación se va a analizar cómo están compuestas cada una de las tres estructuras de datos comentadas anteriormente.

En primer lugar, la estructura de estados está compuesta por objetos de la clase *State*. Esta clase (cuya definición se muestra en la Figura 32) sirve como estructura de datos en la que se agrupan todos los atributos (elementos que conforman un estado) que un estado puede tener. De esta forma, un estado se define: por un nombre de estado, una lista de transiciones a otros estados (objetos de la clase *TransitionClass*), una lista de objetos *Content* y otra lista de comandos, ambas asociadas con el evento de entrada al estado y por último una lista de objetos *Content y* otra lista de comandos, también asociadas con el evento de salida del estado.

```
/*
.*.Class.modeling.the.states
.*/
class.State.{
    var.stateName.:.String;
    var.trans.:.ArrayList;.//ArrayList.of.Transition.elements
    var.onenterContent.:.ArrayList;.//ArrayList.of.Content.elements
    var.onenterScript.:.String[];
    var.onexitScript.:.String[];
    var.onexitContent.:.ArrayList;.//ArrayList.of.Content.elements

}
```

Figura 32 Fragmento de código donde se define la clase *State*

Por otro lado, los objetos que modelan las transiciones entre estados pertenecen a la clase *TransitionClass*. Como se muestra en la Figura 33, una transición entre estados se modela mediante un evento que dispara la transición, un estado objetivo, y opcionalmente una condición, que si se cumple, la transición se ejecuta. Estos atributos se almacenan a modo de cadena de caracteres que serán tratados de forma oportuna en el manejador de eventos, función *Send()* del *script Engine* (ver la Figura 38 para comprobar el algoritmo).

```
/*
.*.Class.modeling.the.transitions
.*
.*.<transition.event="e".cond="x==1".target="s1"/>
.*.or
.*.//<transition.event="e".target="s1"/>
.*
.*/
class.TransitionClass.{
    var.event.:.String;
    var.trgt.:.String;
    var.cond.:.String;.//condition.MIGHT.be.NULL
}
```

Figura 33 Fragmento de código donde se define la clase *TransitionClass*

En cuanto a los objetos de tipo *Content* se puede ver cómo están definidos en la Figura 34. Estos objetos modelan los contenidos educativos o propios de la situación que se quiere simular como puede ser un panel de “feedback” o de diálogo. Más adelante, en este mismo capítulo, en el punto “4.4.1 Ficheros de contenido del escenario de simulación - Fichero de definición del flujo de estados” se detalla el uso de estos componentes.

```
/*
.*.Class.modeling.content
.*
.*/
class.Content.{
    var.id.:.String;
    var.txt.:.String;
    var.imagen.:.String;
    var.xpos:.int;
    var.ypos.:.int;
    var.contentWidth.:.int;
    var.contentHeight.:.int;
}
```

Figura 34 Fragmento de código donde se define la clase *Content*

Continuando con el análisis de las estructuras creadas, la estructura para el almacenaje y gestión de los contenidos educativos guarda objetos de tipo *ActionMenu* que son creados a partir de la información del fichero de contenidos educativos. En la Figura 35 se muestra la estructura de la clase.

```
/*
.*.Class.modeling.the.ActionMenus
.*/
class.ActionMenu.{

    var.ident.:.String;
    var.title.:.String;
    var.buttons:.String[];
    var.buttonsCmd.:.String[];
    var.buttonsPoints.:.float[];
    var.buttonsFeedbackImage.:.String[];
    var.buttonsFeedback:.String[];
    var.typePoints:.String;
    var.maxPoints:.float;
}
```

Figura 35 Fragmento de código donde se define la clase *ActionMenu*

Finalmente, la última estructura que se crea es la relacionada con la información de las víctimas que aparecen en el escenario de simulación. Esta estructura se crea en el *script* MngVictim.js. Cada paciente viene caracterizado por un identificador, nombre, edad, dolencia y pulsaciones del corazón. La clase que modela las víctimas se muestra en la Figura 36.

```
/*
.*.Class.modeling.Victims
.*/
class.VictimClass.{
    var.id.:.int;
    var.nameV.:.String;
    var.age.:.String;
    var.hb.:.String[];
    var.hbNum.:.int[];
    var.symptom.:.String;
}
```

Figura 36 Fragmento de código donde se define la clase *VictimClass*

Subsistema de entorno de trabajo

Cuando las estructuras de información anteriores están disponibles, el sistema simulador ya tiene toda lo necesario para ejecutar la simulación. De esta forma el elemento "Manager" empieza a intermediar entre los eventos generados durante la simulación y la información sobre los estados. La información sobre los estados hace referencia a: transiciones entre estados, contenidos asociados a cada estado y/o evento almacenados en las estructuras de información creadas anteriormente (ver Figura 37). Todas estas tareas las lleva a cabo el *script Engine.js*, el motor del sistema simulador, que consigue que los engranajes de cada estructura de información encajen con el desarrollo en tiempo real de la simulación.

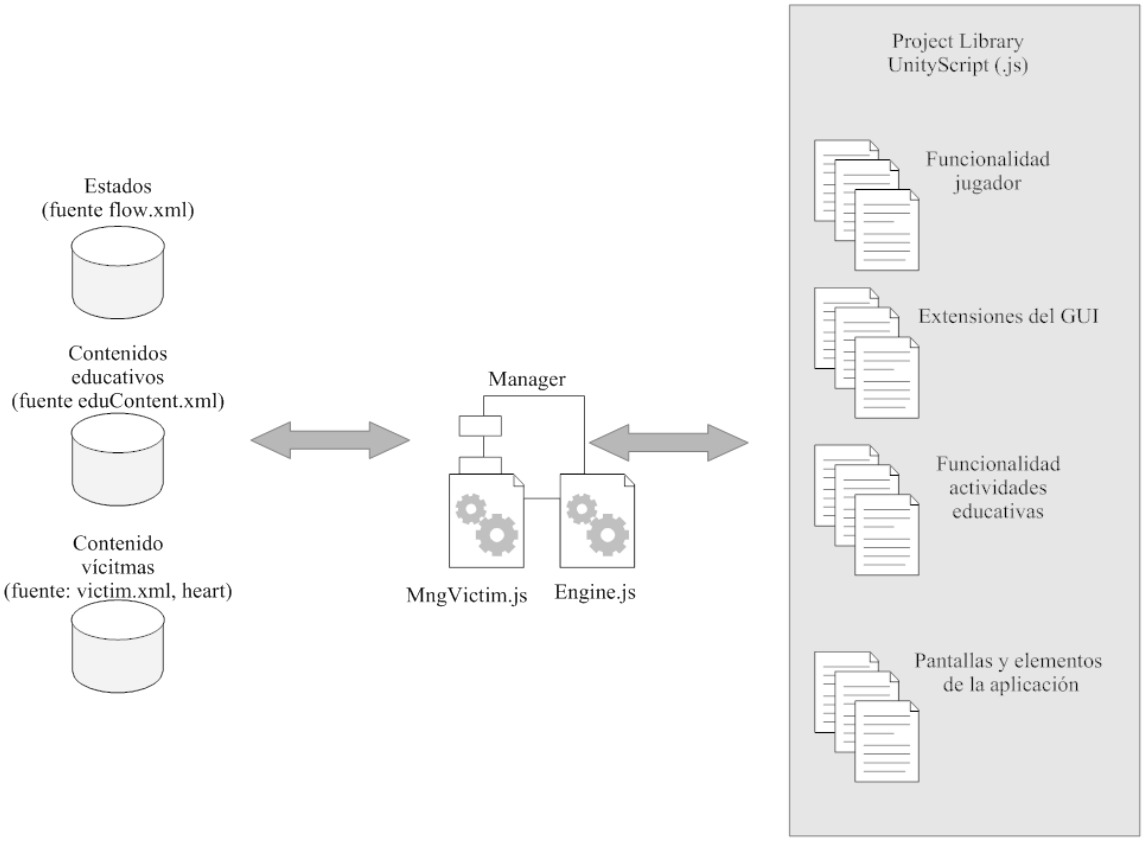

Figura 37 Arquitectura del subsistema motor del simulador

Para manejar los eventos, el *script Engine.js* utiliza la función estática *Send()*, accesible por todos los elementos activos en la escena, cuyo código se muestra en la Figura 38. De esta forma, cada vez que un elemento necesita lanzar un evento hace una llamada a esta función indicando por parámetro el evento en cuestión.

Cuando el evento disparado supone una transición entre estados, la función *Send()* hace una llamada a la función *Transition()*. En esta función se trata tanto la transición entre estados como las acciones indicadas para la entrada y/o salida del estado.

```
/*
 * Description: SCXML external communication
 *              <send>
 * Paremeters: ev, the event sent (happened)
 *
 *    How does it work?
 *    1) What is the current state? if parallel check the parent state too
 *    2) Check what <transition> is triggered by the event
 *    3) Check the conditions/prerequisites, if ok cond=true
 *    4) Call <transition>, the parameters event and target state are required
 */
static function Send (ev : String) {

    //search the current state in the stateArray
    var intCurrentState : int;
    var intEvent : int;

    for(intCurrentState = 0; intCurrentState < stateArray.Count; intCurrentState++) {
        if (stateArray[intCurrentState].stateName == currentState) {
            //search the transition children in the current state
            for(intEvent = 0; intEvent < stateArray[intCurrentState].trans.Count; intEvent++) {

                if (stateArray[intCurrentState].trans[intEvent].event == ev) {
                    //check conditions
                    if (stateArray[intCurrentState].trans[intEvent].cond != null){
                        for(var intCond = 0; intCond < condList.Count; intCond++){
                            if (stateArray[intCurrentState].trans[intEvent].cond == condList[intCond])
                                Transition(stateArray[intCurrentState].trans[intEvent], stateArray[intC

                return;
                        }
                    }
                    continue;
                }//end cond
```

Figura 38 Fragmento de código donde se define el manejador de eventos

En este subsistema, el script *MngVictim.js* sirve de apoyo en la gestión de las víctimas y de su información clínica. De esta forma cuando se está creando el contenido educativo se puede acceder a esta estructura como se muestra en la Figura 39.

```
            <conditionvar>
                <varequal respident="DialogVictim_ActionMenu">A</varequal>
            </conditionvar>
            <setvar>0.25</setvar>
            <displayfeedback linkrefid="A"/>
            <altmaterial>HudDialog.showHud(MngVictim.victimArray[0].nameV,"toy herido 50");.VictimVitalsTable.txt[0]
MngVictim.victimArray[0].nameV;</altmaterial><!--Button.command-->
        </respcondition>
        <respcondition title="B">
            <conditionvar>
                <varequal respident="DialogVictim_ActionMenu">B</varequal>
            </conditionvar>
            <setvar>0.25</setvar>
            <displayfeedback linkrefid="B"/>
            <altmaterial>HudDialog.showHud(MngVictim.victimArray[0].age,"toy herido 50",.0.5,.0.85,.0.3,.0.2);
VictimVitalsTable.txt[1]=MngVictim.victimArray[0].age;</altmaterial><!--Button.command-->
        </respcondition>
        <respcondition title="C">
            <conditionvar>
                <varequal respident="DialogVictim_ActionMenu">C</varequal>
            </conditionvar>
            <setvar>0.25</setvar>
            <displayfeedback linkrefid="C"/>
            <altmaterial>HudDialog.showHud(MngVictim.victimArray[0].symptom,"toy herido 50");.VictimVitalsTable.txt[2]
MngVictim.victimArray[0].symptom;</altmaterial><!--Button.command-->
```

Figura 39 Fragmento de código donde se define el contenido educativo y se accede a datos del historial médico de la víctima

Subsistema de contenidos internos

Para finalizar el análisis de la arquitectura se va a analizar el subsistema encargado de los elementos gráficos de las pantallas de la aplicación. Es decir, este sistema se encargará de crear la estructura de información necesaria para crear los títulos, botones y otros textos presentados en las pantallas de la aplicación como el menú principal, pantalla de registro, o la pantalla de selección de escena de simulación.

Este es un subsistema donde todos sus elementos son internos a la aplicación, por lo que el contenido de la aplicación no se puede editar en fase de producción (cuando ya ha sido entregado al cliente), pero sí puede hacerse en fase de desarrollo del sistema, que sólo es accesible por el equipo desarrollador. El objetivo de esta decisión de desarrollo se debe a dos motivos:

1. Evitar que el cliente final modifique partes de la aplicación que se quiere sean no editables.
2. Permitir y potenciar el desarrollo interno de nuevos simuladores reutilizando componentes para crear nuevos simuladores y adaptarlos a nuevas temáticas.

En el fichero *Languages.xml* están definidos todos los elementos gráficos que se mostrarán en la pantalla, como se puede ver en la Figura 63 en el punto 4.4.3 Ficheros de interfaz de la aplicación. En la Figura 40 se puede ver cómo mediante el script "MngLanguage.js" se crea una estructura donde se almacena toda la información que cada pantalla tiene que mostrar, por lo que a la hora de cargar una pantalla se debe ir a esta estructura y obtener la información a mostrar.

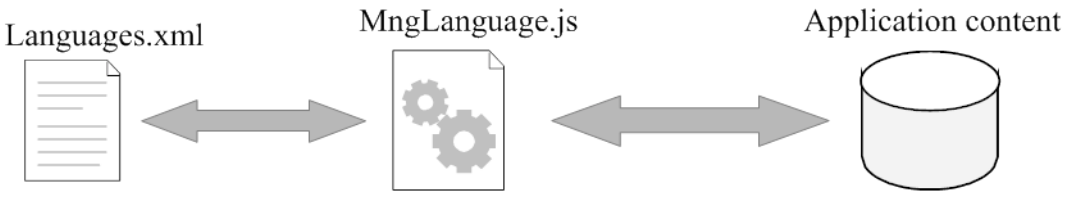

Figura 40 Arquitectura subsistema contenidos aplicación

Adicionalmente, si se desea incluir nuevas pantallas en la aplicación, reutilización total, se deberá recurrir a la fase de edición e incluirlas en los ajustes del proyecto tal y como se muestra en la Figura 41.

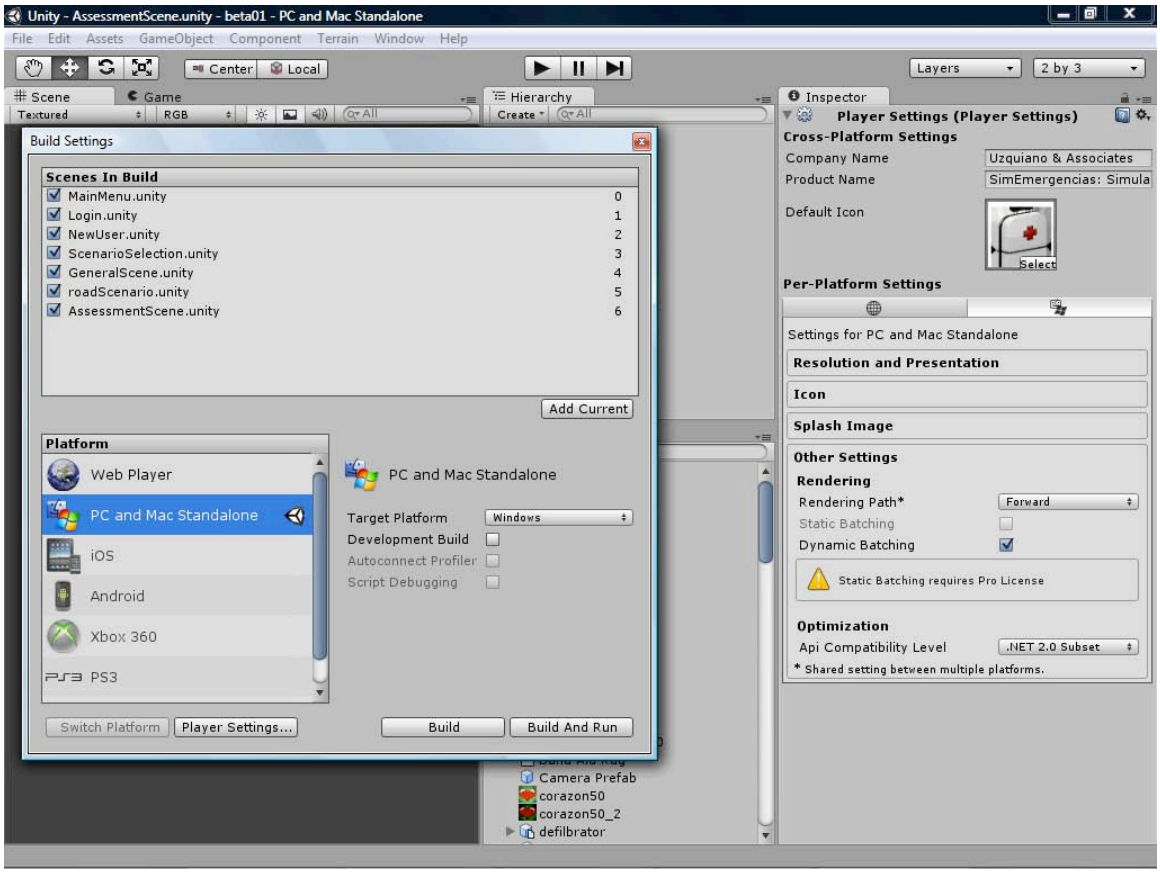

Figura 41 Entorno de trabajo de Unity 3 – Vista de ajustes del proyecto

4.4 Modelo de datos

El sistema está sustentado en ficheros tanto internos como externos a la aplicación que permiten separar el código propiamente dicho, del contenido, y del flujo de estados en los diferentes escenarios de la aplicación.

Los ficheros internos son aquellos que son compilados en la fase de construcción del proyecto. De esta forma, una vez construido el proyecto quedan encapsulados en la aplicación y por tanto no son accesibles para su modificación. Un ejemplo de este tipo de

ficheros son, por ejemplo, los scripts que dan funcionalidad al simulador, o el archivo XML donde están definidos los textos y elementos GUI (botones, menús, etc…) de la aplicación. Este último se analizará más en detalle en el apartado 4.4.3 Ficheros de interfaz de la aplicación.

Los ficheros externos, en cambio, sí pueden ser modificados después de construir el proyecto, pues como indica su nombre, son externos al proyecto. Mediante la utilización de este tipo de ficheros se conseguirá la flexibilidad y escalabilidad marcadas como objetivo en este proyecto, que permiten al usuario final modificar los contenidos sin necesidad de conocer el funcionamiento interno del simulador. En los apartados 4.4.1 Ficheros de contenido del escenario de simulación y 4.4.2 Ficheros de registro de actividad de la aplicación se verá el uso de los ficheros externos en este proyecto.

En el sistema existen tres tipos de ficheros de datos:

1. De **definición del contenido del escenario de simulación**: flujo de estados y contenido educativo. Se verá en detalle en el punto 4.4.1 Ficheros de contenido del escenario de simulación.
2. De **registro de actividad de la aplicación**, tanto registro de sesión y cuenta de usuario como de resultados de su interacción con el escenario de simulación. Se analizarán en el apartado 4.4.2 Ficheros de registro de actividad de la aplicación.
3. De **contenidos de la aplicación**: textos, imágenes, y menús propios de la aplicación. Como se ha dicho anteriormente se verán detenidamente en el apartado 4.4.3 Ficheros de interfaz de la aplicación.

4.4.1 Ficheros de contenido del escenario de simulación

En este apartado se verán los siguientes ficheros:

- Fichero de definición del flujo de estados
- Fichero de definición del contenido educativo
- Fichero de víctimas
- Fichero de datos de instrumentos de diagnóstico
- Ficheros multimedia

Fichero de definición del flujo de estados

En este fichero se definen los estados, y transiciones entre estados, además de los ficheros donde están definidos los contenidos educativos, las dimensiones por defecto de los elementos visuales de "feedback" o de cuadros de diálogo. El formato de este fichero sigue el estándar SCXML (State Chart XML) [W3C 2011] y debe llamarse "*flow.xml*", y estar situado en "*files/*". En la Figura 42 se muestra un ejemplo de definición de estado.

```
19  <state id="Initial">
20      <transition event="victim_near_distance" target="Victim"/>
21      <transition event="near_distance" target="Ambulance"/>
22
23      <onenter>
24          <script>HudDialog.hideHud();</script>
25          <script>ActionMenuWindow.hideFeedbackHud();</script>
26      </onenter>
27  </state>
```

Figura 42 Fragmento de código donde se define un estado inicial

En este fragmento se define un estado llamado "Initial" que permite pasar a otros dos estados llamados "Victim" (actividad Víctima, que consiste en atender a la víctima, definido como se muestra en la Figura 44) y "Ambulance" (actividad Ambulancia, donde hay que coger todo el material necesario de la ambulancia, definido como se muestra en la Figura 45). La transición se produce cuando el jugador se encuentra lo suficientemente cerca de la ambulancia o de la víctima.

Para definir un estado es necesario indicar un identificador en la etiqueta <*state*> y para definir una transición entre estados, se utiliza la etiqueta <*transition*> indicando el evento que dispara la transición y el estado de destino. Como se muestra, el estándar SCXML permite añadir una condición a esa transición como se indica en la Figura 43, donde se muestra que no podemos pasar al estado "AmbulanceTaskDone" hasta que no hayamos cogido todo el material necesario (*cond="inventoryDone"*).

```
<state id="Ambulance_ActionMenu">
<!--put always first the transitions with conditions-->
    <transition event="clickCloseActionMenu" cond="inventoryDone" target="AmbulanceTaskDone" />
    <transition event="clickCloseActionMenu" target="Ambulance" />
    <transition event="far_distance" target="Initial" />

    <onenter>
        <content id="actionMenu" ims_ident="IMS-QTI_AmbulanceInventory" />
    </onenter>
</state>
```

Figura 43 Fragmento de código donde se define una transición entre estados con condición

Es importante indicar que el nivel de prioridad de las transiciones viene indicado por el orden en el que éstas estén declaradas en el fichero. Es decir, en el caso de que dentro de un mismo estado se quiera definir dos transiciones disparadas por un mismo evento y como destino el mismo estado, se deberá escribir en primer lugar la transición que contenga una condición, y posteriormente la transición sin condición.

```
<state id="Victim">
    <transition event="onMouseOver" target="Victim_ActionMenu" />
    <transition event="victim_far_distance" target="Initial" />

    <onenter>
        <script>EnableScript("VictimActionMenu");</script>
        <script>HudDialog.showHud(MngVictim.victimArray[0].symptom,"toy.herido.50");</script>
    </onenter>

</state>
```

Figura 44 Fragmento de código donde se define el estado “Victim”

```
<state id="Ambulance">
    <transition event="onMouseOver" target="Ambulance_ActionMenu" />
    <transition event="far_distance" target="Initial" />

    <onenter>
        <script>EnableScript("AmbulanceActionMenu");</script>
        <content id="feedbackHud" txt="Recuerda coger el material necesario antes de tratar a las
víctimas" image="feedbackmedico50" xpos="0.87" ypos="0.85" width="0.24" height="0.14" />
    </onenter>
</state>

<state id="AmbulanceTaskDone">
    <transition event="onMouseOver" target="Ambulance_ActionMenu" />
    <transition event="far_distance" target="Initial" />

    <onenter>
        <content id="feedbackHud" txt="Ya has cogido todo lo necesario, ve ahora a ver cómo se
encuentran las victimas." image="feedbackmedico50" xpos="0.87" ypos="0.85" width="0.24" height="0.14" />
    </onenter>
</state>
```

Figura 45 Fragmento de código donde se define el estado “Ambulance”

Por otro lado, en los estados existe la posibilidad, mediante la etiqueta <*script*> y <*content*>, de definir acciones que se ejecutarán cuando se entra o sale de dicho estado, como se muestra en la Figura 46 en la que se indica que al entrar se muestra un menú de acción con el contenido indicado en la etiqueta <*item*> con identificador ”IMS-QTI_Victim” del fichero de contenidos educativos y al salir del estado “Victim_ActionMenu” se cierra la ventana de diálogo.

```
74  <state id="Victim_ActionMenu">
75      <transition event="clickTreatment" target="VictimTreatment" />
76      <transition event="clickCloseActionMenu" target="Victim" />
77      <transition event="victim_far_distance" target="Initial" />
78
79      <onenter>
80          <content id="actionMenuFixed" ims_ident="IMS-QTI_Victim" />
81      </onenter>
82      <onexit>
83          <script>HudDialog.hideHud();</script>
84      </onexit>
85  </state>
```

Figura 46 Fragmento de código donde se define acciones a la entrada y salida del estado

Para indicar las acciones que deben ocurrir al entrar al estado, éstas han de estar dentro de la etiqueta <*onenter*>. De igual forma, para indicar las acciones que se deben ejecutar antes de salir del estado, éstas se deben definir dentro de la etiqueta <*onexit*>.

Estas acciones pueden ser bien contenido ejecutable, utilizando la etiqueta <*script*>, o bien mostrar elementos del GUI, panel de “feedback” o paneles de diálogo, utilizando la etiqueta <*content*>. A continuación se describirá en detalle las posibilidades que ofrece el uso de estas dos etiquetas:

- <*script*>, el código definido dentro de esta etiqueta va ser ejecutado en el script (JavaScript). Las instrucciones disponibles son las siguientes:
    - *EnableScript("NombreScript");* : habilita la funcionalidad proporcionada por el script cuyo nombre se indica entre comillas. El uso de esta función queda restringida a los scripts presentes en la escena de simulación. Un ejemplo de su uso se puede ver en la Figura 47, donde se habilita la cámara que enfoca en primer plano al paciente y deshabilita la cámara principal.

```
87  <state id="VictimTreatment">
88  
89      <transition event="clickDialog" target="VictimDialog" />
90      <transition event="clickTriage" target="VictimTriage" />
91      <transition event="clickCloseActionMenu" target="Victim" />
92  
93      <onenter>
94          <content id="actionMenuFixed" ims_ident="IMS_QTI_TreatVictim" />
95          <script>EnableScript("victim01/CameraVictim");</script>
96          <script>DisableScript("Main Camera");</script>
97          <script>EnableScript("VictimVitalsTable");</script>
98      </onenter>
```

Figura 47 Ejemplo de uso de las funciones *EnableScript()* y *DisableScript()*

La descripción de la funcionalidad de cada script se detalla en el punto 4.6.5 Escenario de simulación.

- *DisableScript("Main Camera");* : deshabilita la funcionalidad proporcionada por el script cuyo nombre se indica entre comillas. Al igual que la función anterior, el uso de esta función queda restringida a los scripts presentes en la escena de simulación, un ejemplo de su uso se puede ver en la Figura 47. La descripción de la funcionalidad de cada script se detalla en el punto 4.6.5 Escenario de simulación.
- *Engine.Send("nombreEvento");* : dispara el evento indicado entre comillas.
- *HudDialog.showHud("texto a mostrar","imagen a mostrar");* : habilita y muestra el panel de diálogo con el texto e imagen indicados. En esta versión de la aplicación la única imagen a mostrar es: *"toy herido 50"*. La posición y dimensiones donde aparecerá el panel vienen definidos en la sección <*datamodel*> de este fichero, *flow.xml*. Los identificadores utilizados para este fin son: "DialogBeginX", "DialogBeginY", "Dialog_width", y "Dialog_height".

Figura 48 Captura de pantalla - Detalle del Panel de diálogo

- *HudDialog.showHud("texto a mostrar","imagen a mostrar", inicio horizontal, inicio vertical, ancho, alto);* : función similar a la anterior con la diferencia que en esta se indican implícitamente la posición y dimensiones del panel. Los parámetros relacionados con las dimensiones y posicionamiento son números decimales entre 0.0 y 1.0. El punto (0.0 , 0.0) representa el ángulo superior izquierdo y el punto (1.0 , 1.0) representa el ángulo inferior derecho. Los parámetros *ancho* y *largo* también son números decimales entre 0.0 y 1.0, donde 1.0 representa el ancho y largo de la pantalla respectivamente.

  Para una mejor compresión, en la Figura 48 se pueden ver señalados estos parámetros donde: "inicio horizontal" se corresponde con "Dialog_BeginX", "inicio vertical" con "Dialog_BeginY", "ancho" con "Dialog_width" y "alto" con "Dialog_height".
- *HudDialog.hideHud();* : esconde el panel de diálogo mostrado mediante las dos funciones anteriormente descritas.
- *ActionMenuWindow.addCondition("nombrecondición");* : añade una condición que se podrá utilizar a la hora de realizar transiciones como se muestra en la Figura 43.
- *ActionMenuWindow.hideFeedbackHud();* : esconde el panel de "feedback" que se ha lanzado en la etiqueta <*content*> con

id="feedbackHud". La etiqueta <*content*> se mostrará en el siguiente apartado.

- *ActionMenuWindow.CloseWindow();* : cierra la ventana que se ha lanzado en la etiqueta <*content*> con *id="actionMenu"* y *id="actionMenuFixed"*.
- *VictimHeartRate.StartHeartRate();* : comienza la animación multimedia que simula la toma de pulsaciones del paciente.

Figura 49 Captura de pantalla - Detalle de la animación de la toma de pulsaciones

- *VictimDiamond.setDiamond("color");* : hace visible el identificador visual de triage (ver Figura 50) con el color indicado entre comillas. Los colores disponibles son: rojo, amarillo, verde, blanco y negro.

Figura 50 Captura de pantalla - Detalle del identificador visual de triage

- *VictimVitalsTable.txt[num];* : modifica u obtiene la información mostrada en la tabla de información de la víctima, que se muestra un ejemplo de ésta en la Figura 51. El parámetro 'num' debe ser un número entero entre 0 y 2. Si 'num' es '0' se accederá al texto que se muestra junto a *"Nombre"* en la tabla de información del paciente. En el caso que 'num' es '1' se accederá al texto que se muestra con la edad del paciente. Si 'num' es '2' se accederá al texto que muestra los síntomas del paciente.

Figura 51 Captura de pantalla - Detalle de la tabla de información de la víctima

- *<content>*, lo definido dentro de esta etiqueta será interpretado y procesado por el motor SCXML-UnityScript. Así, las opciones disponibles son:
  - *<content id="feedbackHud" txt="Texto a mostrar" image="imagen a mostrar" xpos="inicio horizontal" ypos="inicio vertical" width="ancho" height="alto" />* : muestra el contenido indicado en el panel de "feedback". Los parámetros de posición y tamaño de esta opción están definidos de igual forma que los explicados para *HudDialog.showHud(…)* en el apartado anterior. La imagen por defecto para este panel es "*feedbackmedico50*".

- <content id="actionMenu" ims_ident="IMS-QTI_AmbulanceInventory" /> : muestra un menú donde se requiere la acción del usuario interaccionando con el menú. El parámetro *ims_ident* indica de donde se obtendrá el contenido a mostrar en el menú, este contenido se definirá en el archivo de contenidos educativos indicado en la etiqueta <*data*> con el identificador *"eduContent"* de la sección <*datamodel*> de este fichero, *flow.xml*. La posición donde aparece el menú es relativa a la posición del puntero del ratón.
- <content id="actionMenuFixed" ims_ident="IMS-QTI_Victim" /> : similar a la opción anterior, solamente se diferencia en la posición donde la ventana se muestra por pantalla que será por defecto.

Para finalizar el análisis del fichero que define el flujo de la aplicación se explicará el modelo de datos que se define mediante las etiquetas <*data*> de la sección <*datamodel*>. En esta sección se define la posición y dimensiones de algunos elementos de la interfaz y se definen los nombres (mediante el atributo *name*) y las rutas relativas (mediante el atributo *src*) de todos los ficheros externo.

- "*eduContent"*, archivo basado en el estándar IMS-QTI Lite donde se define el contenido educativo, contenido de las acciones a realizar por el alumno, contenido del "feedback", puntuaciones, etc… Se analizará más adelante en esta misma sección, en el apartado Fichero de definición del contenido educativo.
- "*victimInfo"*, archivo donde se define la información de las víctimas: nombre, edad y síntomas. Se verá la estructura de este archivo en este mismo capítulo, en el punto Fichero de víctimas.
- "*heartBeatSound"* : archivo multimedia con el sonido de latidos de corazón, ideado para reproducirlo cuando se realiza la acción de tomar pulsaciones.
- "*HeartBeats"* : archivo sin extensión donde se encuentran los datos relativos a las pulsaciones de una víctima. Los datos están separados por un carácter tabulador. Al igual que los archivos anteriores se detallará más adelante en este mismo capítulo.

```
<datamodel>
    <data name="eduContent" src="files/eduContent.xml"/>
    <data name="victimInfo" src="files/victim.xml"/>
    <data name="heartBeatSound" src="files/audio/heartbeat.wav"/>
    <data name="heartBeats" src="files/heart/heart"/>
    <data name="HudFeedbackBeginX" expr="0.87" />
    <data name="HudFeedbackBeginY" expr="0.85" />
    <data name="HUDFeedback_width" expr="0.24" />
    <data name="HUDFeedback_height" expr="0.14" />
    <data name="DialogBeginX" expr="0.5" />
    <data name="DialogBeginY" expr="0.85" />
    <data name="Dialog_width" expr="0.4" />
    <data name="Dialog_height" expr="0.2" />
</datamodel>
```

Figura 52 Fragmento de código donde se indican los ficheros de información externos

Fichero de definición del contenido educativo

En este archivo están definidas las actividades educativas del simulador. El formato que sigue este fichero es el estándar IMS-QTI Lite [IMS 2002]. Para determinar cual es el archivo utilizado en el escenario de simulación se ha de puntualizar la ruta relativa de este archivo en *flow.xml,* en la sección <*datamodel*> utilizando el identificador predefinido para este propósito, *"eduContent"*, como se muestra en la Figura 52. En el ejemplo de la figura anterior este fichero se encuentra en *"files/eduContent.xml"*.

El estándar IMS-QTI Lite, como se explicó en el capítulo del Estado del Arte, es una especificación creada para la definición de cuestionarios en el contexto de exámenes tipo test. En el marco de este proyecto se ha utilizado en un contexto más amplio ya que en este caso las respuestas serán acciones del usuario. Por ello, tanto las respuestas, como los resultados y feedback pueden tener una representación visual en el mundo virtual.

Cada actividad educativa vendrá definida por la etiqueta <*item*> que tiene como atributos el título de la actividad y un identificador único.

Una actividad educativa representará en el simulador una ventana emergente donde se requiere la acción del alumno. Una vez el alumno haya elegido una de las posibles respuestas, esta se procesará puntuando su elección, mostrando el "feedback" pertinente y demás acciones que el educador haya definido para la respuesta. Así pues, se debe entender una acción educativa como un ejercicio que se pide al alumno que resuelva. En

el caso de este simulador, el alumno cuando interacciona con los elementos del escenario surgirán ejercicios a resolver, acciones educativas. De esta forma, en el ejemplo de la Figura 53 se puede ver cómo se define la actividad educativa de triage, en la cual el alumno debe elegir la clasificación de la víctima entre las opciones mostradas en la ventana emergente, en este caso el triage rojo, amarillo, verde, blanco o negro.

```
<!--Triage_ActionMenu-->
<item title="Triage" ident="IMS-QTI_Triage">
    <presentation>
        <reponse_lid ident="Triage_ActionMenu">
            <render_choice>
                <response_label ident="A">
                    <material><mattext>Rojo</mattext></material>
                </response_label>
                <response_label ident="B">
                    <material><mattext>Amarillo</mattext></material>
                </response_label>
                <response_label ident="C">
                    <material><mattext>Verde</mattext></material>
                </response_label>
                <response_label ident="D">
                    <material><mattext>Blanco</mattext></material>
                </response_label>
                <response_label ident="E">
                    <material><mattext>Negro</mattext></material>
                </response_label>
            </render_choice>
        </reponse_lid>
    </presentation>

    <resprocessing>
        <outcomes>
            <!--defaultval is the max value the assessment can be, in this case-->
            <decvar varname="Valoración de Triage" defaultval="1.0"/>
        </outcomes>
        <respcondition title="A">
            <conditionvar>
                <varequal respident="Triage_ActionMenu">A</varequal>
            </conditionvar>
            <setvar>1</setvar>
```

Figura 53 Fragmento de código donde se define una actividad educativa

Así, si se traduce actividad educativa por ventana emergente (ver Figura 54), se puede decir que:

- el título de la ventana viene definido por el atributo *title* de la etiqueta <*item*>.
- los botones para seleccionar que aparecen en la ventana se definen en la sección <*presentation*>, más concretamente en la etiqueta <*responose_lid*>,

<*render_choice*>. Ahí, cada respuesta mostrada se definirá con la etiqueta <*response_label*>.

Figura 54 Ejemplo de acción educativa en el simulador

El tutor puede indicar dentro de cada <*item*> cómo debe el simulador reaccionar ante la respuesta del alumno. Así, si el alumno selecciona la respuesta correcta, se le puede sumar un punto a su nota y mostrar un mensaje de "feedback" diciendo que la respuesta ha sido correcta o incluso mostrar una animación. Este procesado de las respuestas se define dentro de la etiqueta <*resprocesing*> como parcialmente se muestra en la Figura 55.

```
321    <resprocessing>
322      <outcomes>
323        <!--defaultval.is.the.max.value.the.assessment.can.be,.in.this.case-->
324        <decvar.varname="Valoración.de.Triage".defaultval="1.0"/>
325      </outcomes>
326      <respcondition.title="A">
327        <conditionvar>
328          <varequal.respident="Triage_ActionMenu">A</varequal>
329        </conditionvar>
330        <setvar>1</setvar>
331        <displayfeedback.linkrefid="A"/>
332        <altmaterial>ActionMenuWindow.CloseWindow();VictimDiamond.setDiamond("red");</
   altmaterial>.<!--Button.command-->
333      </respcondition>
```

Figura 55 Fragmento de código donde se define la respuesta a una acción

En la figura anterior se puede ver cómo va a actuar el sistema cuando el alumno seleccione la opción con el identificador "A", "rojo". En este caso, al tratarse de una respuesta correcta, el tutor ha decidido puntuar con un punto la elección de esta respuesta definiéndolo en la etiqueta <*setvar*>, como se muestra más claramente en la Figura 56.

```
330          <setvar>1</setvar>
```

Figura 56 Ejemplo de asignación de puntuación a una respuesta

Esta puntuación se sumará a la categoría de puntuación definida en la sección <*outcomes*> como se muestra en la siguiente figura. En la definición de la categoría se ha de incluir el nombre y el valor máximo de puntos que puede valer dicha categoría, definido en el atributo *defaultval*.

```
321      <resprocessing>
322          <outcomes>
323              <!--defaultval is the max value the assessment can be, in this case-->
324              <decvar varname="Valoración de Triage" defaultval="1.0"/>
325          </outcomes>
```

Figura 57 Ejemplo de definición de categoría de puntuación

Además, en la sección de procesado de respuestas se define que se mostrará un panel de feedback enlazándolo con el elemento de "feedback", que se verá a continuación, con identificador "A".

```
331          <displayfeedback linkrefid="A"/>
```

Figura 58 Ejemplo de asignación de feedback a una respuesta

Para terminar la sección de procesado de respuestas, también se puede añadir como material alternativo, acciones en el sistema. En este ejemplo el tutor ha decidido que la ventana debe cerrarse tras elegir una respuesta y que la víctima debe mostrar visualmente la clasificación del triage.

```
332          <altmaterial>ActionMenuWindow.CloseWindow();VictimDiamond.setDiamond("red");</
  altmaterial><!--Button command-->
```

Figura 59 Ejemplo de material alternativo en el procesado de respuestas

Como se muestra en la Figura 58 se puede enlazar el contenido del mensaje de "feedback" cuando se selecciona una respuesta. Cada contenido de "feedback" se define independientemente en una sección <*itemfeedback*>, fuera de la sección <*resprocessing*>, como se muestra en la Figura 60.

El contenido del “feedback” está compuesto por una imagen, y un texto. Estos se definen dentro de la sección <*material*>, en la etiqueta <*matimage*> y <*mattext*> *respectivamente.* La imagen debe ser una imagen predefinida en el sistema, mientras que el texto lo puede decidir libremente el tutor.

```
364 ¬   ¬   ¬   ¬   <altmaterial>ActionMenuWindow.CloseWindow();VictimDiamond.setDiamond("black");</
  . altmaterial>.<!--Button.command-->¶
365 ¬   ¬   ¬   </respcondition>¶
366 ¬   ¬   </resprocessing>¶
367 ¶
368 ¬   ¬   <itemfeedback.ident="A">¶
369 ¬   ¬   ¬   <material>¶
370 ¬   ¬   ¬   ¬   <matimage>feedbackgreen50</matimage>¶
371 ¬   ¬   ¬   ¬   <mattext>Correcto!</mattext>¶
372 ¬   ¬   ¬   </material>¶
373 ¬   ¬   </itemfeedback>¶
374 ¬   ¬   <itemfeedback.ident="B">¶
375 ¬   ¬   ¬   <material>¶
376 ¬   ¬   ¬   ¬   <matimage>feedbackyellow50</matimage>¶
377 ¬   ¬   ¬   ¬   <mattext>Eso.a.estado.bien,.pero.has.de.tener.en.cuenta.su.grado.de.urgencia.</mattext>¶
378 ¬   ¬   ¬   </material>¶
379 ¬   ¬   </itemfeedback>¶
```

Figura 60 Ejemplo de definición de feedback

Fichero de víctimas

En este fichero se define la información propia de la víctima relacionada con su historia clínica que aparecerá en el escenario de simulación. Estos ficheros dotan de gran flexibilidad al simulador porque permiten cambiar la patología asociada a una víctima concreta o añadir nuevas víctimas. Al igual que ocurre con el fichero de contenidos educativos, para determinar cual es el archivo utilizado en el escenario de simulación se ha de especificar la ruta relativa de este archivo en *flow.xml*, en la sección <*datamodel*> utilizando el identificador predefinido para cargar la información de la víctima, *“victimInfo”*, y el atributo *“src”* para indicar su localización, *“files/victim.xml”*, como se muestra en la Figura 52.

Como se muestra en la Figura 61. La información perteneciente a la víctima es:

- <*id*>: número identificador de la víctima.
- <*name*>: nombre de la víctima.
- <*age*>: edad de la víctima.
- <*symp*>: síntoma que padece la víctima.

```
<victim-list>
    <victim><!--.victim.1.!.-->
        <id>00</id>
        <name>Laura.Pérez</name>
        <age>40.años</age>
        <symp>Me.duele.el.pecho..No.puedo.moverme</symp>
    </victim>
</victim-list>
```

Figura 61 Fragmento de código donde se define una víctima

Fichero de datos de instrumentos de diagnóstico

En algunos casos, el diagnóstico puede requerir tomar medidas adicionales mediante algún tipo de instrumentación (pulso, electrocardiograma, etc…).

En este proyecto, se ha implementado un fichero externo (Figura 62) que contiene la información de las pulsaciones en formato numérico (números enteros) separados por el carácter tabulador.

Como ya se ha comentado anteriormente, para determinar cual es el archivo utilizado en el escenario de simulación se ha de especificar la ruta relativa de este archivo en *flow.xml,* en la sección <*datamodel*> utilizando el identificador predefinido para cargar la información de las pulsaciones de la víctima, *"heartBeats"*, como se muestra en la Figura 52. En el ejemplo de la Figura 52 este fichero se encuentra en *"files/heart/heart"*.

heart

```
66	65	67	66	68	66	67
```

Figura 62 Ejemplo de fichero de pulsaciones del corazón

El objetivo mantener un fichero de texto separado para cada instrumento de diagnóstico es doble: por un lado, permitir en un futuro conectar el simulador con una base de datos que contenga información de diagnóstico real, y por otro conectar el simulador con dispositivos físicos que escriban esos ficheros en tiempo real.

Ficheros para diagnóstico multimedia

Para hacer la simulación más realista se ha habilitado la posibilidad de reproducir archivos multimedia con ejemplos de sonidos característicos que pueden presentar las víctimas, así los alumnos pueden entrenarse en la identificación visual o auditiva de pruebas de diagnóstico como por ejemplo cómo suena el estetoscopio un tipo concreto de patología cardiaca. En la versión actual del sistema, y para demostrar su funcionalidad esta posibilidad se ha restringido a la actividad de toma de pulsaciones de la víctima.

Los formatos aceptados para los archivos multimedia son: “wav” y “ogg”.

Al igual que ocurre con el fichero de contenidos educativos, para determinar cual es el archivo utilizado en el escenario de simulación se ha de especificar la ruta relativa de este archivo en *flow.xml*, en la sección <*datamodel*> utilizando el identificador predefinido para cargar la información de la víctima, “*heartBeatSound*”, y la ruta relativa se encuentra “*files/audio/heartbeat.wav*” (Figura 52).

4.4.2 Ficheros de registro de actividad de la aplicación

En esta primera versión del sistema, tanto los datos de los alumnos, como las calificaciones obtenidas se escribirán en archivos XML. En siguientes versiones está planificado que toda esta información se almacene y consulte a través de una base de datos.

Así, en este proyecto, los archivos donde se almacenará la información de los alumnos son:

- results.xml : registro de resultados del escenario de simulación, guarda los contenidos mostrados en la pantalla de resultados. Se genera al mostrar la pantalla de resultados de escenario de simulación.
- sessionInfo.xml : registro de la sesión de usuario. Se genera al entrar en el registro de usuario, opción accesible desde la pantalla del menú principal.
- UserInfo.xml : registro de la información del estudiante: nombre y apellidos del estudiante, nombre de usuario y contraseña. Se genera al crear un nuevo usuario, opción accesible desde la pantalla del menú principal.

4.4.3 Ficheros de interfaz de la aplicación

Para facilitar el mantenimiento de la interfaz y su internacionalización, todos los textos que aparecen en los componentes de la interfaz de usuario (menús, títulos, etc…) están definidos en un archivo XML interno, situado en la ruta "Assets/Languages/Resources/Languages.xml" (únicamente accesible para editar durante la fase de diseño). Los textos de la aplicación están definidos actualmente en dos idiomas: castellano e inglés.

```
<application>
    <screen id="MainMenu"><!-- Main Menu, first screen in the application. -->
        <button>
            <es>Entrar</es>
            <en>Login</en>
        </button>
        <button>
            <es>Nuevo usuario</es>
            <en>New user</en>
        </button>
        <button>
            <es>Idioma</es>
            <en>Language</en>
        </button>
        <button>
            <es>Salir</es>
            <en>Quit</en>
        </button>
    </screen>
```

Figura 63 Fragmento de código donde se define el menú principal

Por otro lado, las imágenes que forman la pantalla de selección de escenario de simulación se encuentran en la siguiente ruta: "files/images/".

4.5 Detalles de implementación Unity 3D

En este subcapítulo se analiza detenidamente la implementación del sistema incluyendo detalles específicos de Unity 3.

4.5.1 Tipos de recursos

Durante la fase de implementación se han generado los recursos necesarios para obtener la funcionalidad deseada y necesaria para cumplir los requisitos del sistema. Estos recursos son de varios tipos:

- *GameObjects*, u objetos del sistema: actúan como contenedores de componentes funcionales que modelan entidades en las pantallas del simulador, como por ejemplo la ambulancia que aparece en el simulador o los paramédicos. Estos componentes permiten encapsular funcionalidad y agrupar componentes funcionales en un mismo objeto.
- *Assets*, o recursos del sistema: estos elementos son fragmentos de código que implementan una funcionalidad concreta como por ejemplo la definida en un script. Los *assets* pueden ser scripts, aunque también pueden usarse para definir materiales, brillos, animaciones. Los *assets* se pueden organizar en componentes funcionales más complejos, *GameObjects*. Un *GameObject* puede tener uno o varios *assets* de tipo *script*, permitiendo de este modo ejecutar distintos comportamientos.
- Ficheros de configuración. Estos ficheros se describen en detalle en los apartados 4.4.1 Ficheros de contenido del escenario de simulación y 4.4.3 Ficheros de interfaz de la aplicación.
  - Los ficheros de definición del contenido educativo están en una ruta externa al proyecto Unity para que de esta forma puedan ser modificados en producción por el usuario final sin necesidad de volver a la fase de diseño del simulador.
  - En cambio, los ficheros donde se define la interfaz de la aplicación están situados dentro del proyecto Unity. Estos pueden ser modificados en fase de desarrollo pero al pasar de la fase de diseño a

producción, estos ficheros quedan encapsulados y no pueden ser modificados. El motivo de esta distribución es para que estos contenidos no sean modificados por el usuario final.

4.5.2 Sistema de compilación

Con respecto a los *scripts* implementados en este proyecto, es necesario conocer el sistema de compilación de Unity 3 para así entender totalmente en el siguiente punto la disposición que ha recibido cada *script* en el proyecto.

De esta forma, Unity compila todos los *scripts* a archivos dll de .NET. Estos ficheros dll a su vez son compilados en tiempo de ejecución (en inglés “JIT compilation”). De esta manera se consigue una ejecución de *scripts* increíblemente rápida, se estima que unas veinte veces más rápida que con JavaScript tradicional y un cincuenta por ciento más lento que con código nativo C++ [Unity].

Así, la compilación de *scripts* en Unity se realiza en cuatro pasos:

1. Primero, se compilan todos los *scripts* situados en las carpetas (en la fase de desarrollo): "*Standard Assets*", "*Pro Standard Assets*" y "*Plugins*". Los *scripts* situados en estas carpetas no pueden acceder directamente a otros *scripts* fuera de estas, pero pueden acceder mediante paso de mensajes.
2. Se compilan todos los *scripts* situados (en la fase de desarrollo) en "*Standard Assets/Editor*", "*Pro Standard Assets/Editor*" y "*Plugins/Editor*". Estos scripts pueden acceder a los scripts situados en las carpetas del grupo anterior.
3. Después son compilados todos los *scripts* situados en la carpeta “*Editor*” (en la fase de desarrollo). Los *scripts* en esta carpeta pueden acceder a los *scripts* de los dos grupos anteriores pero no a los del siguiente grupo.
4. Finalmente se compilan todos los *scripts* que no estén en ninguno de los grupos anteriores. Estos tienen acceso entre sí y a los *scripts* del primer grupo.

4.6 Desarrollo de pantallas de la aplicación

En este punto se van a analizar una a una todas las pantallas o escenas que componen la aplicación. Dentro del apartado de cada escena se detallarán los elementos o "GameObjects" que la forman, así como los componentes que a su vez conforman estos elementos. Además se explicará la funcionalidad que aporta cada uno de estos componentes.

4.6.1 Menú principal

En esta pantalla se presenta al usuario un menú con cuatro opciones (ver Figura 64):

- Entrar: el usuario puede entrar a la aplicación con su cuenta de usuario y practicar en el simulador.
- Nuevo usuario: permite crear una nueva cuenta de usuario.
- Idioma: con esta opción se puede cambiar el idioma de la aplicación, inglés o castellano. Por defecto el idioma principal es castellano
- Salir: sale de la aplicación.

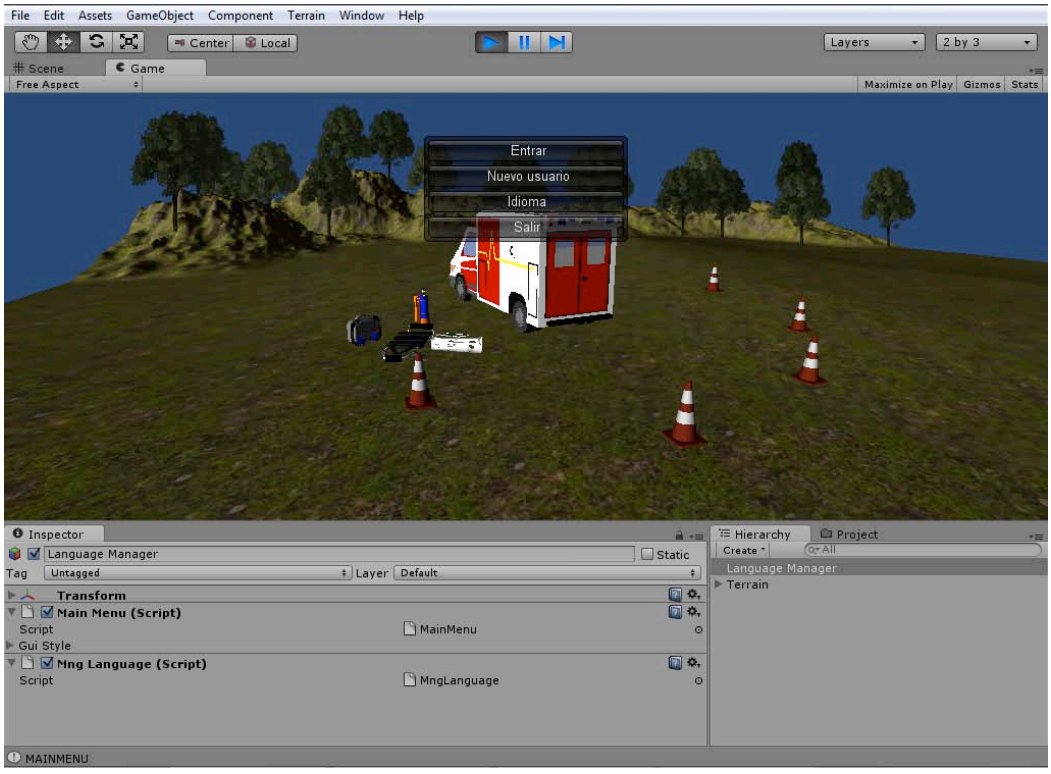

Figura 64 Vista en fase de edición de la pantalla Menú Principal

En las siguientes tablas se describen los componentes de los 'GameObject' presentes en la escena:

Language Manager	
Componentes:	
	Main Menu (script): • Muestra los componentes gráficos de interfaz de usuario definidos en el archivo *Languages.xml*. • El menú mostrado proporciona las opciones de acceso a: la pantalla de registro de usuario, la pantalla de registro de nuevo usuario, cambiar el idioma de la aplicación y salir de la aplicación.
	Mng Language (script) • Proporciona la interfaz de lectura entre el archivo *Languages.xml* donde se definen los elementos de GUI y la aplicación. Nota: *Languages.xml* está situado en la ruta *"/Assets/Languages/Resources/Languages.xml"*, de esta forma sólo puede ser modificado durante la fase de desarrollo, no de producción o comercialización. • Crea una estructura accesible para todos los elementos que componen la aplicación. **Nota:** por esta razón este componente permanecerá activo (no será destruido en el paso de escenas) durante toda la ejecución de la aplicación. • Realiza la función de cambio de idioma de la aplicación. • Este script está situado en la carpeta *"Standard Assets/Scripts"* para que su compilación sea anterior a la de los demás scripts [Unity. Overview: Script compilation] y éstos puedan utilizar la estructura creada con los elementos del GUI y de la aplicación.

Tabla 25 Language Manager – Elemento de la escena Main Menu

Terrain	
	Paisaje mostrado. Está formado por diferentes elementos gráficos: el propio 'Terrain', 'ambulancia', 'cones', 'defibrilator', 'Directional light', 'Main Camera', 'oxygentube' y 'stabilisers'. **Nota:** para crear este terreno se ha utilizado la herramienta Terrain Toolkit [Six Times Nothing 2009a] (ver Figura 65).

Tabla 26 Terrain – Elemento de la escena Main Menu

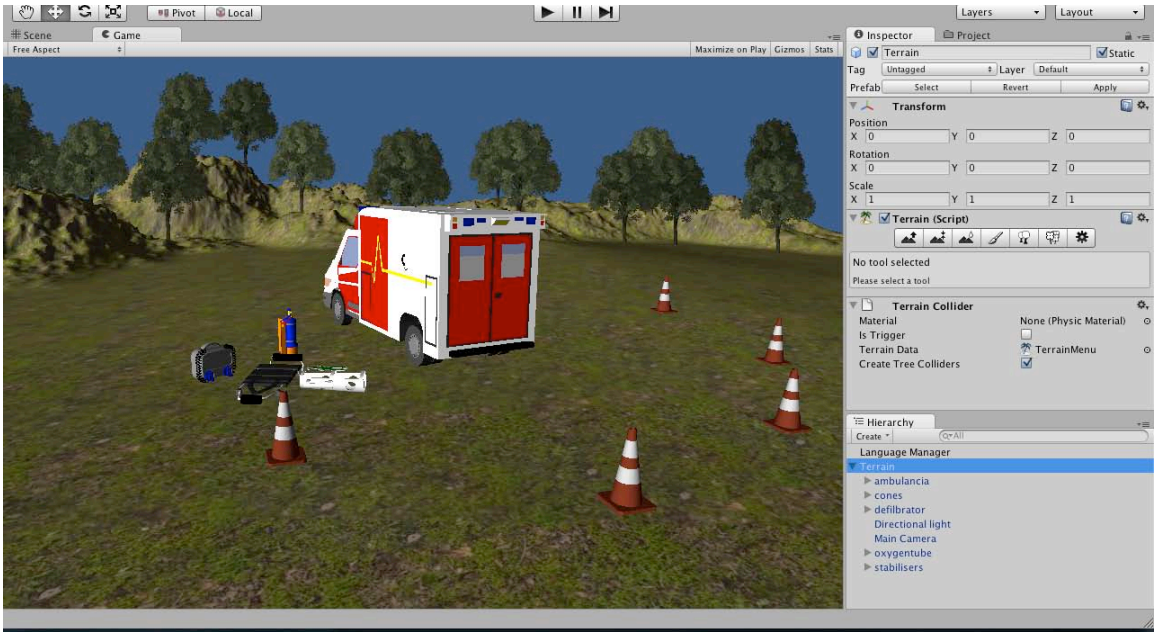

Figura 65 Vista en fase de edición del Terrain Toolkit

4.6.2 Pantalla de entrada de usuario

En esta pantalla un usuario ya registrado en la aplicación accede al simulador con su cuenta de usuario (ver Figura 66). Para ello, como se muestra en la pantalla, el alumno debe introducir su nombre de usuario y contraseña. También tiene la opción de volver al menú principal.

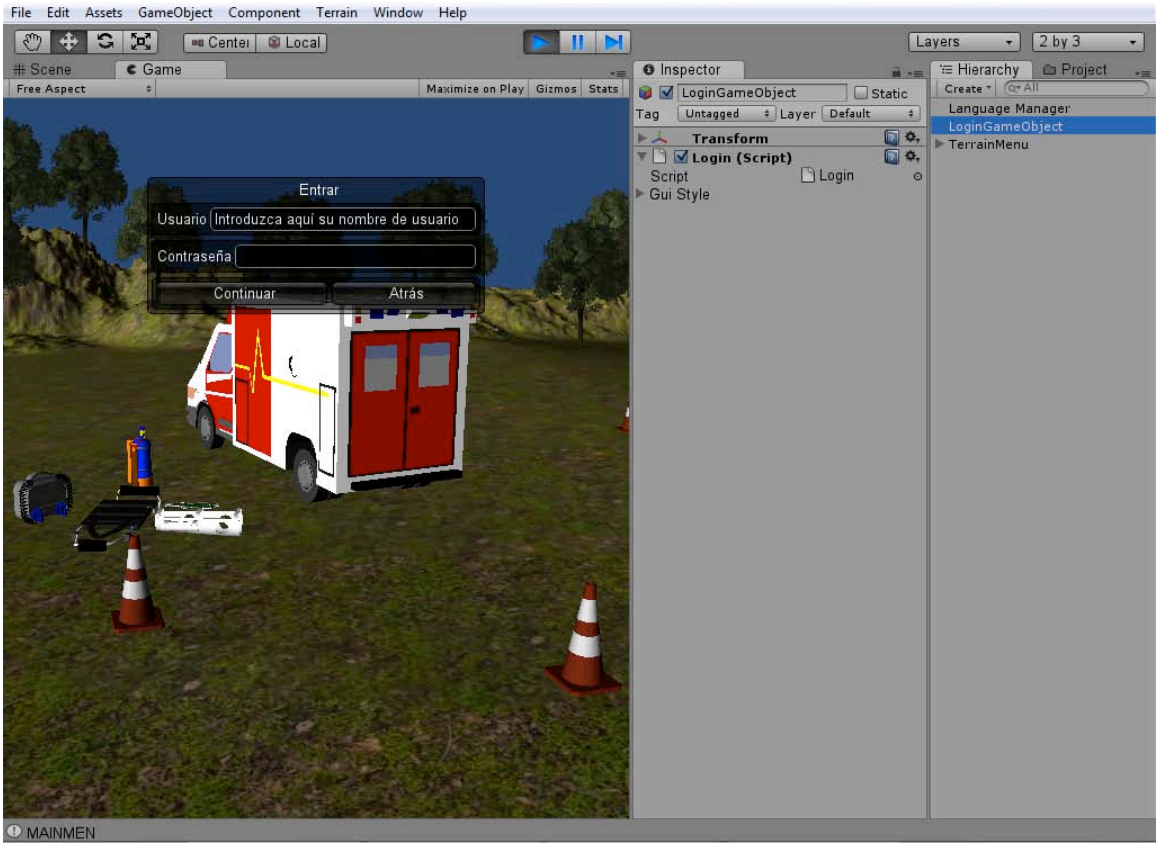

Figura 66 Vista en fase de edición de la pantalla de Login

En las siguientes tablas se describen los componentes de los 'GameObject' presentes en la escena:

LoginGameOject	
Componentes:	
	Login (script): • Muestra los componentes gráficos de interfaz de usuario definidos en el archivo *Languages.xml.* • El usuario debe introducir su nombre de usuario y contraseña, acto seguido puede pasar a la pantalla de selección de escenario. También tiene la opción de volver al escenario principal. • Escribe los datos de inicio de sesión en el archivo *sessionInfo.xml* situado en la ruta "*files/sessionInfo.xml*".

Tabla 27 LoginGameObject – Elemento de la escena Login

TerrainMenu (mismo GameObject que en el definido en la Tabla 26)

4.6.3 Pantalla de alta de nuevo usuario

En esta pantalla un nuevo usuario puede crear su cuenta de usuario. Como se muestra en la Figura 67 los campos necesarios para crear una nueva cuenta de usuario son: nombre y apellidos del alumno, usuario, y contraseña. Después de rellenar el formulario puede acceder a la pantalla de selección de escenario de simulación. También tiene la opción de volver al menú principal.

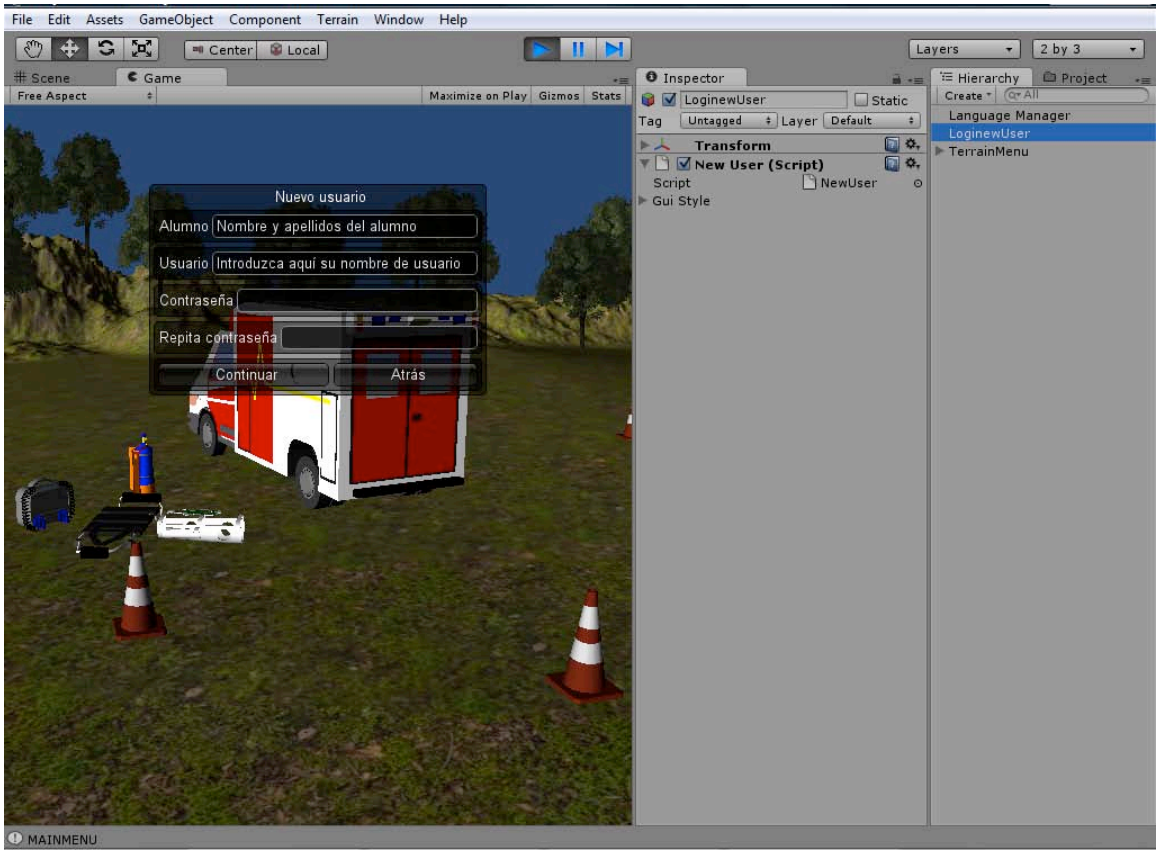

Figura 67 Vista en fase de edición de la pantalla de creación de nueva cuenta de usuario

En las siguientes tablas se describen los componentes de los 'GameObject' presentes en la escena:

LoginNewUser	
Componentes:	
	NewUser (script): • Muestra los componentes gráficos de la interfaz de usuario definidos en el archivo *Languages.xml.* • El usuario debe introducir su nombre y apellidos, usuario, y contraseña, acto seguido puede pasar a la pantalla de selección de escenario. También tiene la opción de volver al escenario principal. • Escribe los datos de inicio de sesión en el archivo *sessionInfo.xml* situado en la ruta "*files/sessionInfo.xml*". • Escribe los datos de la cuenta de usuario en el archivo *UserInfo.xml* situado en la ruta *"files/UserInfo.xml"*.

Tabla 28 LoginNewUser – Elemento de la escena NewUser

TerrainMenu (mismo GameObject que en el definido en la Tabla 26)

4.6.4 Pantalla de selección de escenario de simulación

En esta pantalla el usuario puede seleccionar la simulación a realizar. Puede elegir el género del carácter ("avatar"), y seleccionar la simulación mediante el menú de escenarios o de técnicas. En esta primera versión de esta aplicación sólo están disponibles dos escenarios, no pudiéndose seleccionar ninguna simulación a través de la opción del menú de técnicas. Una vez seleccionado el género del "avatar" y el escenario de simulación el usuario ya puede pasar a la simulación. También tiene la opción de volver al menú principal. Un ejemplo de esta pantalla se muestra en la Figura 68.

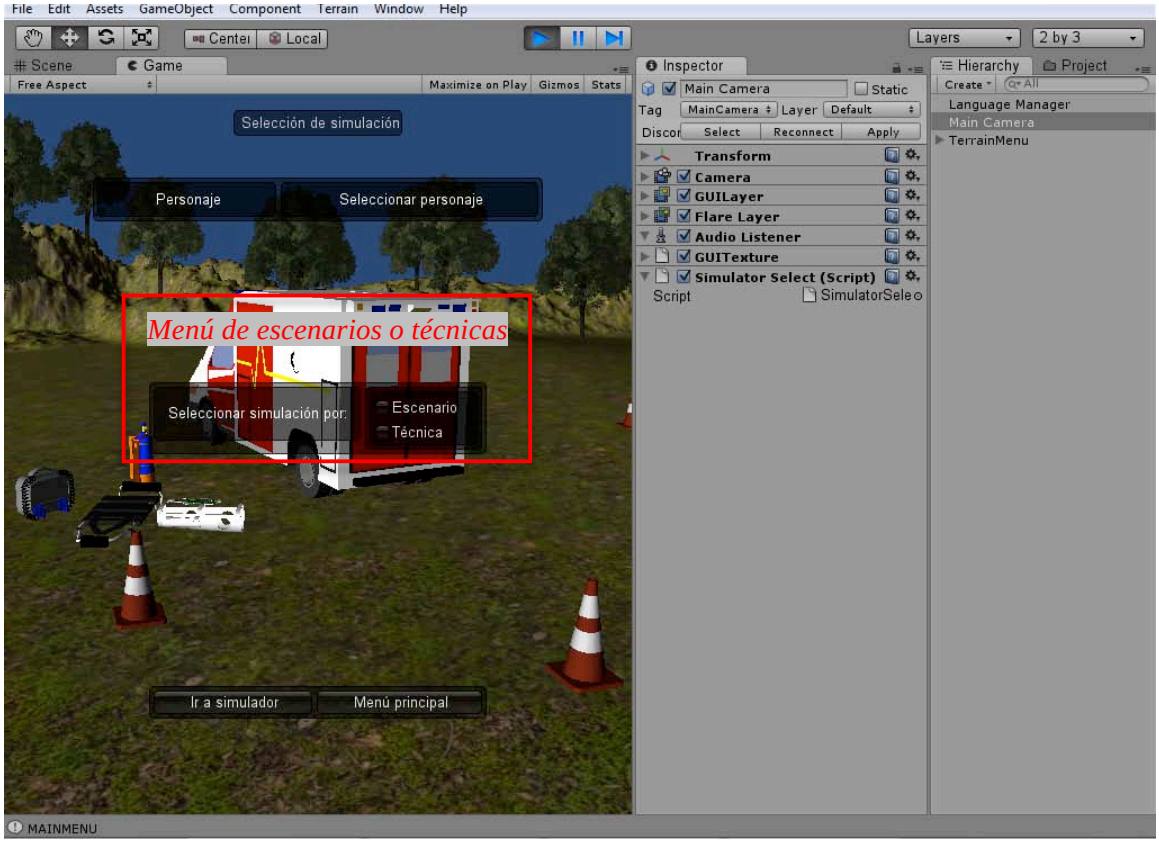

Figura 68 Vista en fase de edición de la pantalla de selección de escenario de simulación

Como punto a destacar, esta pantalla se ha diseñado con carácter genérico para dar cabida a nuevos escenarios y casos de uso que se están desarrollando en el marco de otros proyectos final de carrera pero el requisito de este proyecto es la implementación de un único escenario. Así, se podrían añadir fácilmente nuevas entradas al menú de escenarios símplemente modificando el archivo *Languages.xml* como se explica más en detalle en el punto 4.4.3 Ficheros de interfaz de la aplicación. En la Figura 68 se muestra el fragmento de código del fichero *Languages.xml* donde se definen las entradas del menú donde se selecciona el escenario.

```
114 ¬   ¬   <text>¶
115 ¬   ¬   ¬   <es>Seleccionar·simulación·por:</es>¶
116 ¬   ¬   ¬   <en>Select·simulation·type:</en>¶
117 ¬   ¬   </text>¶
118 ¬   ¬   <text>¶
119 ¬   ¬   ¬   <es>Escenario</es>¶
120 ¬   ¬   ¬   <en>Scenario</en>¶
121 ¬   ¬   </text>¶
122 ¬   ¬   <text>¶
123 ¬   ¬   ¬   <es>Técnica</es>¶
124 ¬   ¬   ¬   <en>Technique</en>¶
125 ¬   ¬   </text>¶
126 ¬   ¬   <text>¶
127 ¬   ¬   ¬   <es>Bosque</es>¶
128 ¬   ¬   ¬   <en>Forest</en>¶
129 ¬   ¬   </text>¶
130 ¬   ¬   <text>¶
131 ¬   ¬   ¬   <es>Carretera</es>¶
132 ¬   ¬   ¬   <en>Road</en>¶
133 ¬   ¬   </text>¶
134 ¬   ¬   <text>¶
135 ¬   ¬   ¬   <es>En·este·escenario·podrás·practicar·las·emergencias·típicas·que·ocurren·en·un·bosque</
  . es>¶
136 ¬   ¬   ¬   <en>In·this·scenario·you·could·train·the·most·usual·emergencies·that·happen·in·a·forest</
  . en>¶
137 ¬   ¬   </text>¶
138 ¬   ¬   <text>¶
139 ¬   ¬   ¬   <es>En·este·escenario·podrás·practicar·las·emergencias·típicas·que·ocurren·en·una·
  . carretera</es>¶
140 ¬   ¬   ¬   <en>In·this·scenario·you·could·train·the·most·usual·emergencies·that·happen·in·a·road</en>
141 ¬   ¬   </text>¶
```

Figura 69 Fragmento de código donde se definen los elementos del menú para seleccionar escenario

Este menú tiene como característica funcional que la imagen de fondo cambia como respuesta a la selección o posible selección del escenario de simulación. Así, cuando se pasa el ratón por alguna opción de simulación, la imagen de fondo muestra una imagen y un pequeño texto descriptivo sobre el escenario que se va a seleccionar. Un ejemplo de esta funcionalidad se puede ver en la Figura 70. Una vez seleccionado qué escenario se quiere entrenar, la imagen quedará permanentemente como imagen de fondo.

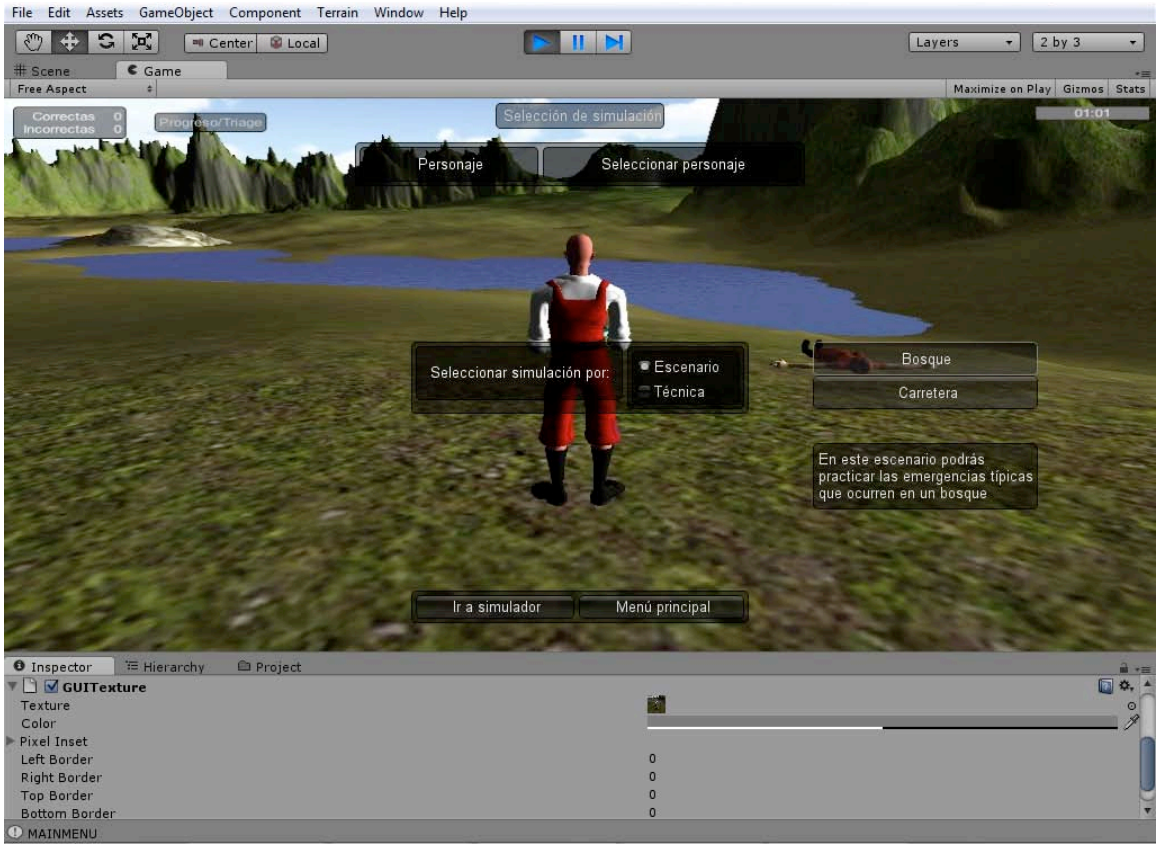

Figura 70 Vista en fase de edición de la pantalla de selección de escenario de simulación seleccionando el escenario

Finalmente, en las siguientes tablas se describen los componentes de los 'GameObject' presentes en la escena:

MainCamera	
Componentes:	
	Camera, capta y visualiza por pantalla los elementos en su campo de visión en el "mundo" 3D.
	GUILayer, componente que hace posible la visualización 2D de los elementos del GUI.
	GUITexture, este componente proporciona la capacidad de cargar imágenes de fondo de forma dinámica. Actúa como "marco" para estas imágenes.
	SimulatorSelect (script): • Muestra los componentes gráficos de interfaz de usuario definidos en el archivo *Languages.xml*. • Carga las imágenes que se cargarán como fondos cuando se seleccione las distintas simulaciones. • Crea y maneja el sistema de menús tipo "drop down" y botones.

Tabla 29 MainCamera – Elemento de la escena ScenarioSelection

TerrainMenu	
	Paisaje mostrado. Está formado por diferentes elementos: el propio 'Terrain', 'ambulancia', 'cones', 'defibrilatos', 'Directional light', 'oxygentube' y 'stabilisers'. **Nota:** para crear este terreno se ha utilizado la herramienta Terrain Toolkit [Six Times Nothing 2009a] (ver Figura 65).

Tabla 30 TerrainMenu – Elemento de la escena ScenarioSelection

4.6.5 Escenario de simulación

Aquí se desarrolla toda la simulación. El "avatar" aparecerá en el escenario y el usuario debe manejarlo mediante los periféricos teclado y ratón.

Señalar que los modelos 3D de los paramédicos pertenecen al paquete de Unity 3 "*Standard Assets/Character Controllers/ 3rd Person Controller*". En la Figura 71 se puede ver un paramédico en una simulación en curso. El modelo de la víctima ha sido creado por Deodato Pechir, profesor del Monterrey Institute of Technology and Higher Education. Una imagen de ésta se puede observar más adelante en la Figura 74.

Figura 71 Vista en fase de edición de la pantalla de simulación

En las siguientes tablas se describen los componentes de todos los 'GameObject' presentes en la escena:

ActionMenuPlugin	
Componentes:	
	ActionMenuWindow (script): • Gestiona el componente gráfico 'menú ventana'. • Crea la ventana. • Cierra la ventana. • Gestiona las acciones disparadas por los botones de la ventana.

Tabla 31 ActionMenuPlugin - Elemento de la escena GeneralScene

HudDialogPlugin	
Componentes:	
	HudDialog (script): • Gestiona el panel de diálogo • Crea el panel. • Cierra el panel.

Tabla 32 hudDialogPlugin - Elemento de la escena GeneralScene

ambulancia	
Componentes:	
	Mesh Collider: • Malla rígida para evitar que otros objetos lo atraviesen. • Proporciona la estructura necesaria para que este GameObject detecte eventos de ratón, tales como pasar el ratón sobre él.
	Ambulance (script): • Gestiona la proximidad del “avatar” a la ambulancia, es decir, si está lo suficientemente próximo significa que ha entrado en esta actividad educativa, la actividad Ambulancia. • El nivel que indica la frontera entre “lejos” y “cerca” la determina la variable “proximity”. • Cuando entra en la región “cerca”, dispara el evento "near_distance". • Cuando entra en la región “lejos”, dispara el evento "far_distance".
	OnMouseOverTarget (script): • Cuando el usuario pasa el ratón sobre este GameObject se dispara un evento *“OnMouseOver”*.

Tabla 33 ambulancia - Elemento de la escena GeneralScene

GUImanager	
Componentes:	
	GUIProgressBar (script): • Muestra mediante un panel en el margen superior izquierdo el progreso en las actividades del escenario de simulación. El formato mostrado es "puntos conseguidos hasta el momento / número máximo de puntos que se pueden conseguir en el escenario". • El valor de la variable "máximo número de puntos" se carga al iniciar la simulación en el script Engine cuando se trata el archivo de contenidos educativos. • La actualización del valor de la variable "puntos conseguidos hasta el momento" se lanza en el script GUIResultPanel.
	GUIResultPanel (script): • Muestra el panel de resultados en pantalla indicando las acciones correctas e incorrectas. • Gestiona las actualizaciones de las puntuaciones generadas por otros scripts. • Actualiza el panel de progreso cuando se modifican las puntuaciones.
	GUIPauseMenu (script): • Pausa el simulador y muestra un menú de opciones de configuración, continuar simulador, y salida de la aplicación.
	GUITimer(script): • Cronómetro de cuenta atrás. • Se muestra en un panel en el ángulo superior derecho. • Cuando entra en el último minuto cambia su apariencia. • Su valor inicial (en segundos) se asigna en la variable *"countDownSeconds"*.

Tabla 34 GUImanager – Elemento de la escena GeneralScene

MainCamera (elemento de las librerías “Standard Assets”)	
Componentes:	
	Camera, capta y visualiza por pantalla los elementos en su campo de visión en el “mundo” 3D.
	GUILayer, añadido a la cámara hace posible la visualización 2D de los elementos del GUI.
	Flare Layer, añadido a la cámara hace posible los efectos de destellos y brillos.
	Audio Listener, capta y reproduce los sonidos que existen a su alrededor.

Tabla 35 MainCamera – Elemento de la escena GeneralScene

Manager	
Componentes:	
	Engine (script): • Proporciona la interfaz principal entre los ficheros externos y el sistema. • Obtiene y gestiona toda la información contenida en el fichero de definición de flujo de estados, *flow.xml*. • Gestiona el sistema de estados y eventos (transiciones entre estados). • Gestiona los scripts habilitados en cada estado. • Coordina los elementos visuales de pantalla (como son paneles y ventanas). • Procesa y gestiona toda la información contenida en el fichero de definición de contenidos educativos.
	MngVictim (script): • Proporciona la interfaz entre el fichero de víctimas y el sistema.

	• Lee el fichero de definición de víctimas XML y lo almacena en una estructura accesible por otros scripts. • Lee el fichero de datos de instrumentos de diagnóstico y lo almacena en una estructura accesible por otros scripts.

Tabla 36 Manager – Elemento de la escena GeneralScene

Paramed	
Componentes:	
	Animation: • Componente que permite animar al "avatar". Las animaciones disponibles son: andar, correr, saltar y esperar, éstas pertenecen al paquete de Unity 3 "*Standard Assets/Character Controllers/ 3rd Person Controller*".
	Character controller: • Componente que permite manejar al "avatar" por el escenario de simulación.
	Third Person Controller (script): • Componente que interpreta las teclas pulsadas en el teclado y las traduce en movimientos del "avatar" y animaciones. • Este script pertenece a la librería "*Standard Assets/Character Controllers/ 3rd Person Controller*".
	Third Person Camera (script): • Componente que utiliza la cámara principal ("Main Camera") para enfocar al "avatar" desde un punto de vista subjetivo. • Este script pertenece a la librería "*Standard Assets/Character Controllers/ 3rd Person Controller*".

Tabla 37 Paramed – Elemento de la escena GeneralScene

Paramedfellow	
Componentes:	
	Animation: • Componente que permite dar animación al "avatar". Las animaciones disponibles son: andar, correr, saltar y esperar, éstas pertenecen al paquete de Unity 3 "Standard Assets/Character Controllers/ 3rd Person Controller".
	Capsule Collider: • Capsula rígida para evitar que otros objetos lo atraviesen. • Proporciona la estructura necesaria para que este GameObject detecte eventos de ratón, tales como pasar el ratón sobre él.
	ParamedActionMenu (script): • Cuando se pasa el ratón por encima muestra el menú de acción del paramédico para terminar el escenario de simulación y pasar a la pantalla de resultado.

Tabla 38 ParamedFellow – Elemento de la escena GeneralScene

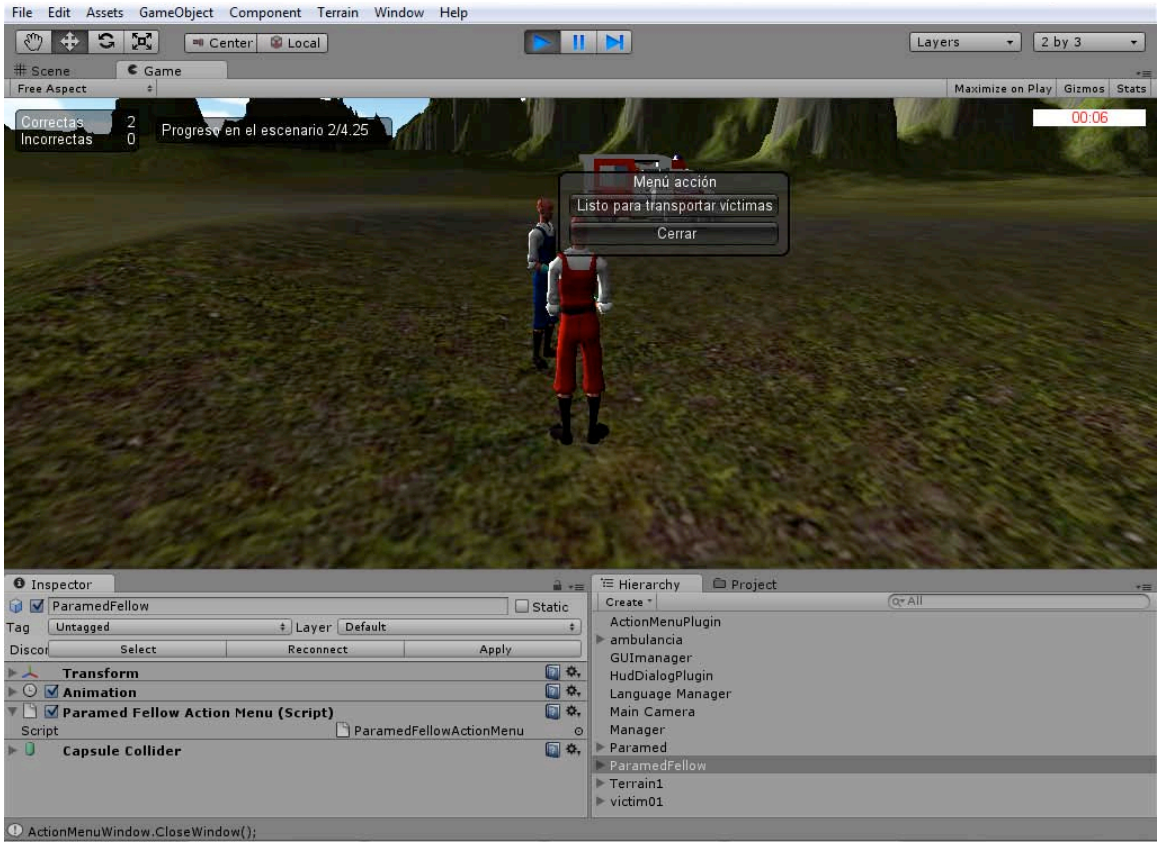

Figura 72 Vista en fase de diseño de ejemplo de interacción entre paramédico y compañero

Terrain	
	Entorno por el que se desarrolla la acción. **Nota:** para crear este terreno se han utilizado las herramientas Terrain Toolkit [Six Times Nothing 2009a], River Toolkit [Six Times Nothing 2009b] y Road/Path Toolkit [Six Times Nothing 2009c] (ver Figura 73).

Tabla 39 Terrain – Elemento de la escena GeneralScene

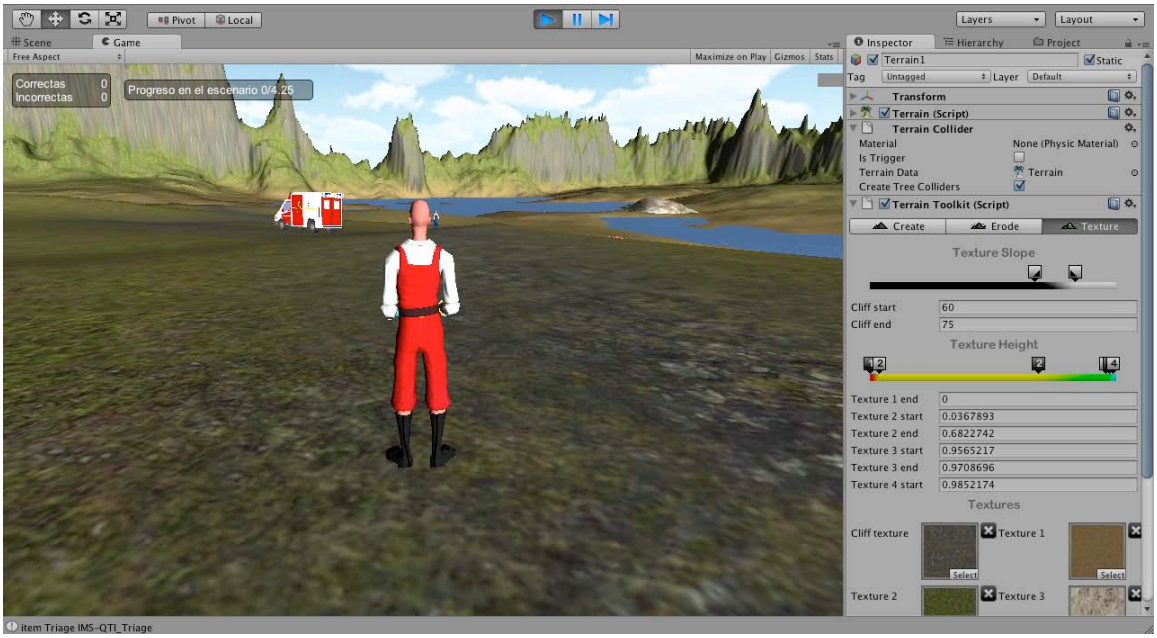

Figura 73 Vista subjetiva del terreno (vista en fase de diseño)

Victim01	
Componentes:	
	Mesh Collider: • Malla rígida para evitar que otros objetos lo atraviesen. • Proporciona la estructura necesaria para que este GamObject detecte eventos de ratón, tales como pasar el ratón sobre la víctima.
	Victim (script): • Gestiona la proximidad del “avatar” a la victima, es decir, si está lo suficientemente próximo significa que ha entrado en esta actividad educativa, la actividad Victima. • El nivel que indica la frontera entre “lejos” y “cerca” la determina la variable“proximity”. • Cuando entra en la región “cerca”, dispara el evento "victim_near_distance". • Cuando entra en la región “lejos”, dispara el evento "victim_far_distance".

	OnMouseOverTarget (script): • Cuando el usuario pasa el ratón sobre este GameObject se dispara un evento *"OnMouseOver"*.
	VictimDiamond (script): • Asigna un color y hace visible el identificador visual de triage con el color deseado: rojo, amarillo, verde, blanco y negro.
	VictimHeartRate (script): • Muestra en pantalla un panel con los valores de las pulsaciones del corazón. El panel está compuesto por un texto que muestra los valores de las pulsaciones, y una imagen que cambia en el tiempo creando una animación. • Los valores de las pulsaciones son tomados de la estructura de datos creada en el script MngVictim. • Reproduce un archivo multimedia simulando el sonido del estetoscopio al tomar las pulsaciones de la víctima. Nota: en la Figura 74 se muestra tanto el panel gráfico como el contenido del componente en el "Inspector", donde se aprecia como el componente tiene una pista de audio.
	VictimVitalsTable (script): • Muestra en pantalla la información recopilada de la víctima.
	Audio Source: • Componente necesario para poder reproducir un archivo de sonido multimedia en la víctima, como por ejemplo el sonido de las pulsaciones del corazón.

Tabla 40 Victim01 – Elemento de la escena GeneralScene

En la siguiente figura (Figura 74) se observa un primer plano de la víctima, que es el utilizado cuando se la está tratando. Además, en esta imagen se está realizando la comprobación de las pulsaciones, caracterizada por la imagen del corazón. Asimismo, en

la imagen está resaltado en el panel de “Inspector” el componente que hace posible esa animación con sonido característico.

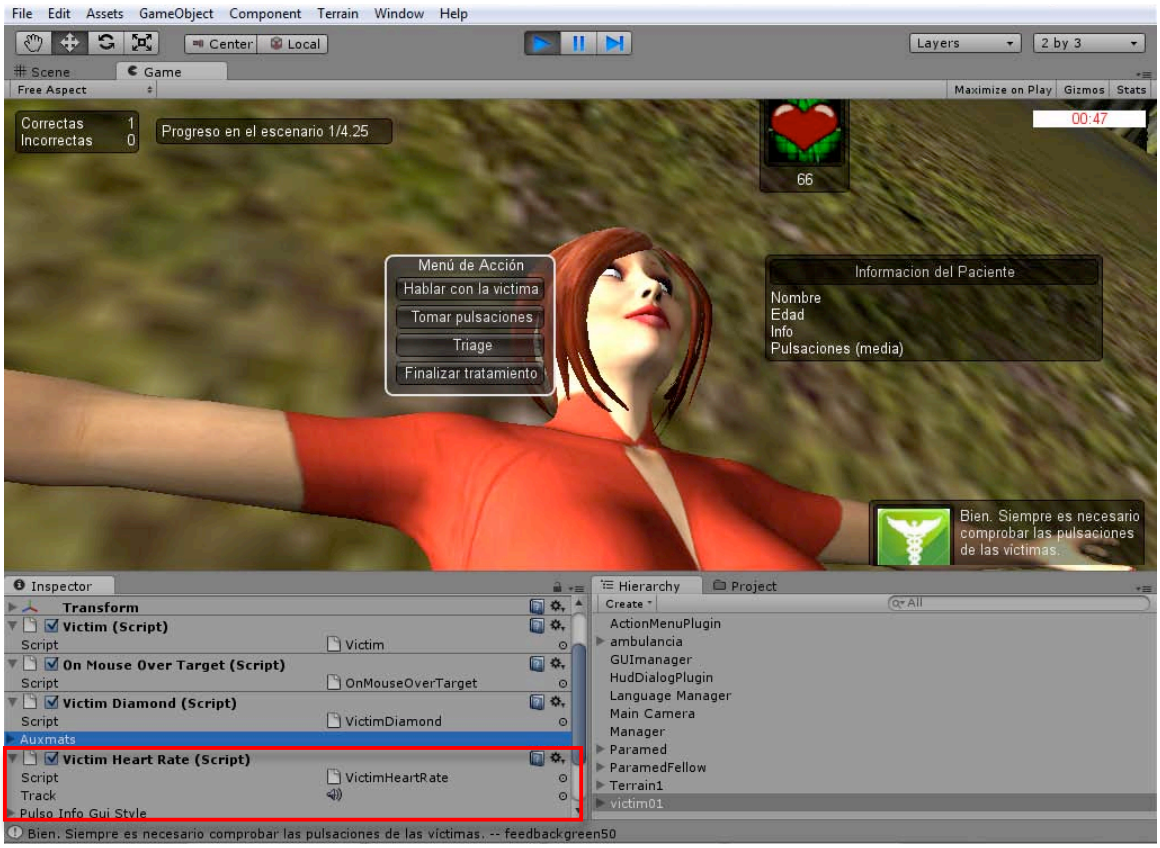

Figura 74 Vista en fase de diseño de la atención a la víctima

En la siguiente figura, Figura 75, se puede ver desde un plano subjetivo del paramédico a la víctima después de habérsele realizado el triage. En este caso se ha clasificado a la víctima con color rojo.

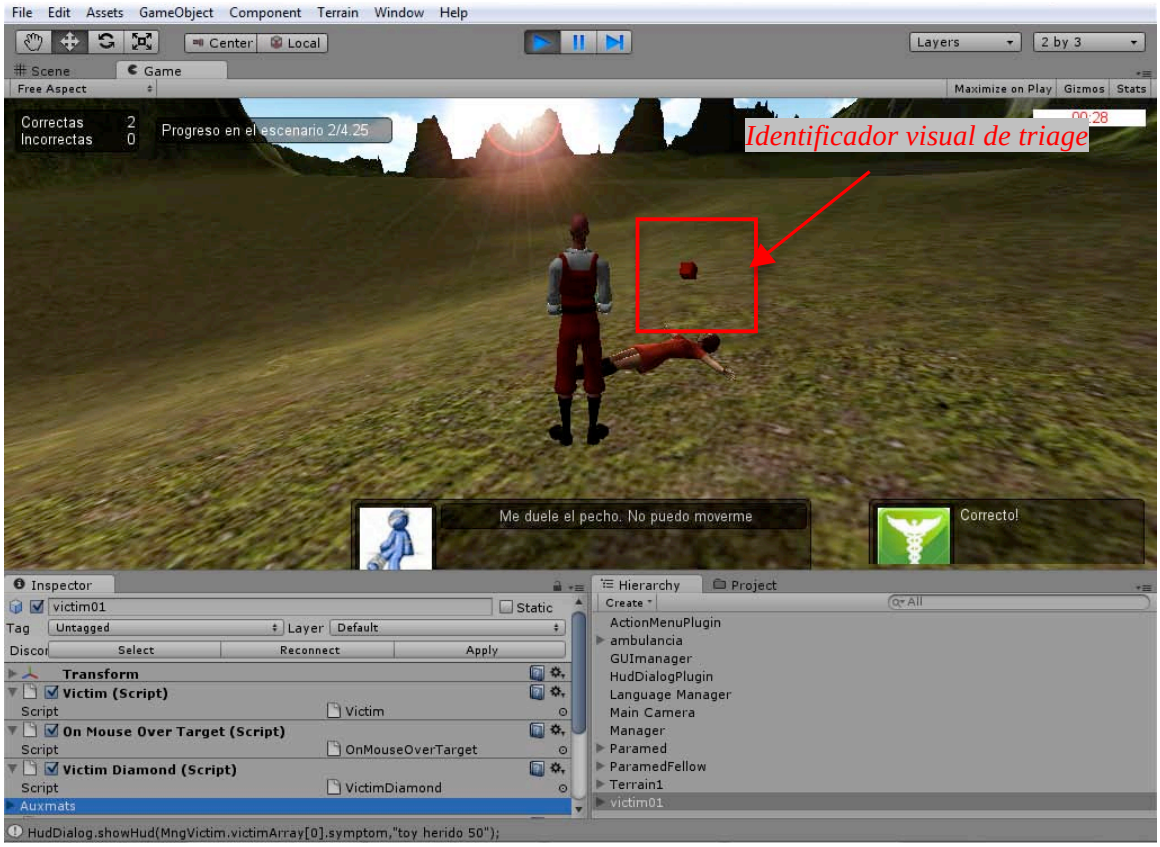

Figura 75 Vista en fase de diseño de la víctima con el identificador de triage

4.6.6 Pantalla de resultados

Tras finalizar la simulación, en esta pantalla el usuario puede comprobar las puntuaciones obtenidas en el simulador.

Se da la opción al usuario de ir a la pantalla de selección de escenario o volver al menú principal de la aplicación.

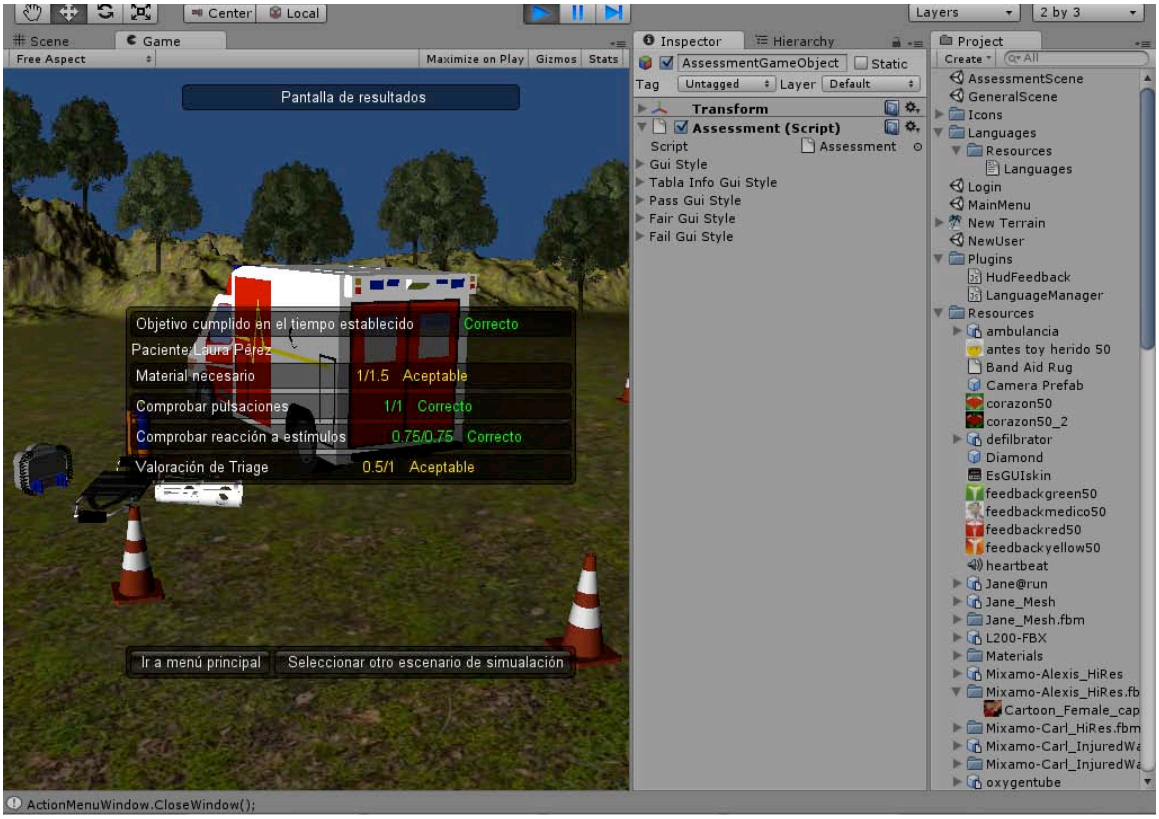

Figura 76 Vista en fase de diseño de la pantalla de resultados

En las siguientes tablas se describen los componentes de los 'GameObject' presentes en la escena:

AssessmentGameOject	
Componentes:	
	Assessment (script): • Muestra los componentes gráficos de la interfaz de usuario definidos en el archivo *Languages.xml*. • Obtiene la puntuación obtenida durante la simulación que está guardada en el script GUIResultPanel y la muestra de forma ordenada. • Escribe la información mostrada en pantalla en un fichero XML externo de resultados, *results.xml*, situado en *"files/results.xml"*.

Tabla 41 AssessmentGameObject – Elemento de la escena AssessmentScene

TerrainMenu (mismo GameObject que en el definido en la Tabla 26)

4.6.7 Carpeta de *Plugins*

Además de los "GameObjects" presentados en los apartados anteriores, y ligados a esas escenas, también están presentes en el proyecto dos scripts en la carpeta *"Plugins"* que se analizan a continuación.

El motivo de que su localización en el proyecto sea diferente, es para conseguir la modularidad de estos componentes (al compilarse con anterioridad a los demás (Unity. Overview: Script compilation)) y de esta forma reutilizarlos sin la necesidad de crear nuevos componentes. Como contrapartida estos scripts no se pueden comunicar con los scripts activos en cada escena, solamente los script habilitados en cada escena pueden comunicarse con los scripts situados en la carpeta *Plugins*, es decir, la comunicación es unidireccional.

HudFeedback (script)	
	Crea el panel de feedback con el texto, imagen, posición y tamaño indicados. Un ejemplo de este panel se puede ver en la Figura 75, donde muestra el mensaje de: "Correcto!".

Tabla 42 HudFeedback - Script situado en la carpeta *Plugins*

LanguageManager (script)	
	Gestiona y almacena información global sobre el idioma seleccionado en la aplicación.

Tabla 43 HudFeedback - Script situado en la carpeta *Plugins*

Capítulo 5

Conclusiones y Trabajo futuro

5.1 Conclusiones

Con la realización de este proyecto se ha conseguido cumplir todos los objetivos funcionales marcados al inicio del mismo así como los objetivos técnicos.

Se ha creado un simulador (implementado en Unity 3D) para la atención de emergencias extra-hospitalarias, flexible y fácil de modificar mediante la creación de nuevos escenarios de simulación. Para ello, se han empleado estándares utilizados en e-learning, SCXML e IMS-QTI Lite, adaptándolos a las necesidades de este proyecto. Además, para ilustrar su funcionalidad se ha creado un caso de ejemplo.

El prototipo desarrollado permite al usuario final modificar los contenidos formativos que aparecen en el simulador así como asignar el contenido auxiliar que se utilizará en la simulación como son archivos de sonido o imágenes. Todas estas

modificaciones se pueden realizar mediante la edición de los ficheros de texto *flow.xml* (que sigue el estándar SCXML) y *eduContent.xml* (que sigue el formato IMS-QTI Lite).

El prototipo permite al desarrollador crear nuevos simuladores de formación con diferente lógica de contenidos e idiomas de una forma rápida y sencilla mediante la modificación del fichero interno *Languages.xml*.

Además se han cumplido también los subobjetivos de crear una aplicación multilingüe (en esta versión sólo bilingüe) y de permitir al tutor el seguimiento de las sesiones de los alumnos mediante la creación de ficheros XML.

En cuanto a la gestión del proyecto, se ha seguido la planificación establecida al principio del proyecto, salvo en ligeras modificaciones debido a que hubo que hacer una presentación de un prototipo intermedio, y por lo tanto se han cumplido los plazos de entrega programados pese a ser muy ajustados. De igual forma la metodología de desarrollo utilizada, Scrum, ha sido una de las claves de éxito para la consecución de este proyecto agilizando el desarrollo e incrementando la eficiencia de trabajo.

Asimismo, la aplicación está disponible en dos plataformas: Windows y Mac OS, extendiendo a casi coste cero de desarrollo e implantación el alcance de la aplicación.

Por otra parte, se proporciona toda la información necesaria para su utilización, así como una detallada documentación para el mantenimiento y ampliación de la aplicación. La información para el desarrollador se encuentra principalmente en los apartados 4.3, 4.4, 4.5 y 4.6 de la memoria y la información para el usuario final en el anexo Manual de usuario.

5.2 Resultados

En cuanto a los resultados obtenidos, la prueba de concepto realizada ha sido útil para divulgar parte de los desarrollos que se hacen en el Departamento de Ingeniería Telemática de la Universidad Carlos III de Madrid y ha sido presentada en diferentes foros a otros grupos de investigación, al instituto de salud Carlos III y a varios colegios que han mostrado su interés en desarrollar herramientas de este tipo para sus programas de formación.

Actualmente el prototipo se encuentra en fase de evaluación y para ello se ha diseñado un cuestionario en el que se pretende evaluar la interfaz de usuario (Figura 78), la utilidad de las opciones ofrecidas (Figura 79), su eficacia como herramienta de formación (Figura 80) y la opinión subjetiva de los encuestados acerca de la aplicación (Figura 81 y Figura 82). Esta encuesta se ha distribuido entre personas de diferentes perfiles:

- Personal sanitario del hospital “La Paz” de Bata (Guinea Ecuatorial) a través de su directora de enfermería.
- Personal docente de la escuela de enfermería de la Universidad Rey Juan Carlos.
- Personal voluntario de Cruz Roja en Aranjuez.
- Personal no sanitario (atención al paciente) del hospital de Fuenlabrada.

En el momento de escribir este documento aún no están los disponibles los resultados del estudio.

A continuación se muestran las diferentes preguntas que aparecen en el cuestionario.

Simergencias: Simulador para la atención de emergencias extra-hospitalarias

El vídeo presentado es una prueba de concepto para el uso de un simulador para la atención a emergencias. El objetivo de este test es valorar la interfaz gráfica y la funcionalidad presentada de cara al desarrollo de futuros simuladores.

Puede ver el vídeo de la aplicación (3minutos) en esta dirección: http://www.it.uc3m.es/mcfp/pfcs/urgencias/DemoUrgencias.mov.

Si su conexión no es muy buena le aconsejamos que lo descargue en su ordenador y lo ejecute desde allí.

*Obligatorio

Centro de procedencia. Puesto que desempeña *

Indique el centro donde desarrolla su actividad relacionada con temas sanitarios y el tipo de labor que realiza

Figura 77 Formulario de evaluación – Presentación

Con las cuestiones que se muestran en la Figura 78 se pretende evaluar la opinión de los usuarios sobre la forma con que el usuario se mueve a través de la aplicación y de la pantalla de simulación así como la interfaz gráfica, incluyendo los elementos visuales como son los paneles de feedback, de diálogo, de puntuación, etc…

Facilidad de uso *

Valore del uno al 5 la facilidad de uso

	1	2	3	4	5	
Difícil (poco intuitivo)	○	○	○	○	○	Muy fácil (muy intuitivo)

Interfaz gráfica *

	1	2	3	4	5	
Me desagrada	○	○	○	○	○	Me encanta

Mecanismos de interacción *

Se utiliza el teclado (flechas) para mover el avatar por el escenario (con mayúsculas para correr) y el ratón para seleccionar opciones de menú o elementos del escenario (ambulancia, víctima, etc.)

	1	2	3	4	5	
No son adecuados	○	○	○	○	○	Son muy adecuados

Figura 78 Formulario de evaluación – Interfaz de usuario

En la Figura 79 se pregunta al usuario acerca de las posibles opciones que se le ofrecen tanto dentro de la aplicación (elegir el género del avatar o la forma de seleccionar el escenario de simulación) como para modificar los contenidos y flujos de simulación en los escenarios.

Figura 79 Formulario de evaluación – Utilidad de las funciones ofrecidas

En la Figura 80 se pregunta sobre la utilidad de la aplicación como herramienta formativa y el uso de entornos 3D frente a otros simuladores con interfaces 2D.

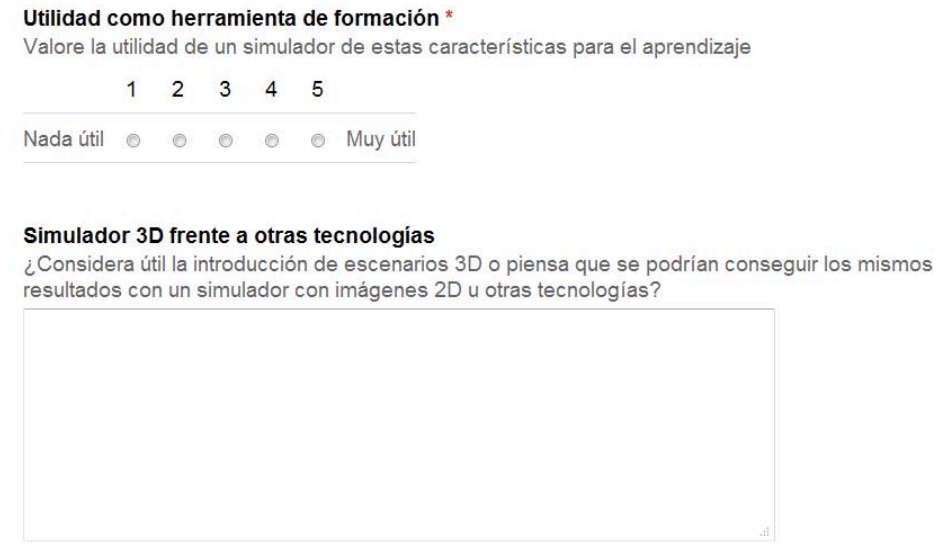

Figura 80 Formulario de evaluación – Utilidad como herramienta de formación

Para finalizar el cuestionario (Figura 81 y Figura 82) se pregunta acerca de la opinión que despierta en los encuestados el simulador y que otra posible funcionalidad les gustaría que se incluyese en las siguientes versiones.

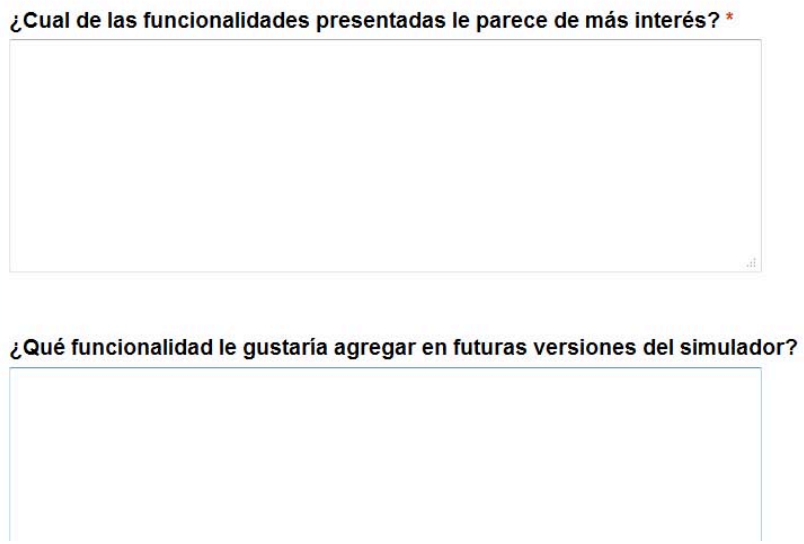

Figura 81 Formulario de evaluación – Funcionalidad más útil y sugerencias

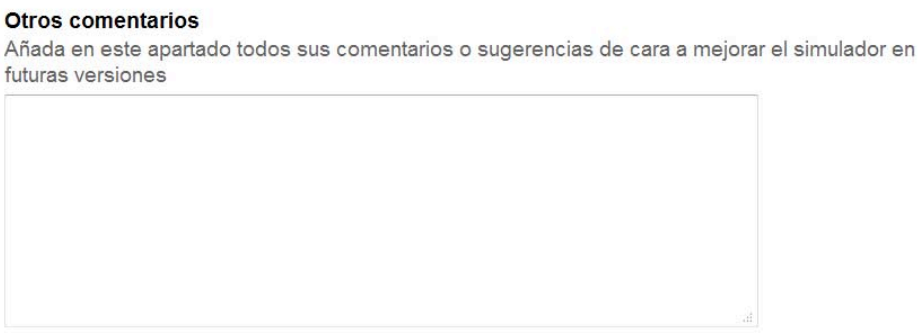

Figura 82 Formulario de evaluación – Otros comentarios

5.3 Líneas de trabajo futuro

Al tratarse este proyecto de una prueba de concepto las opciones de mejoras son muy amplias si lo que se desea es conseguir una aplicación completa. A continuación se van a sugerir las mejoras más prioritarias según la opinión del equipo de desarrollo de este proyecto.

- **Desarrollo de modelos 3D**: Al haberse centrado este proyecto en la arquitectura de la aplicación, no se han podido dedicar los recursos necesarios al desarrollo de modelos 3D de los escenarios de simulación ni de los avatares. Por esta razón, se recomienda contar con un equipo de artistas 3D para futuras versiones.

  Aunque en este proyecto no se han utilizado para la elaboración del prototipo final, durante su desarrollo se han estado investigando los siguientes temas al respecto:

  - Creación de edificios: sobre este tema se ha recuperado información de Goggle-warehouse [Bicer 2008]. Por otro lado, se ha visto la posibilidad de crear edificios a partir de fotografías mediante Goggle SketchUp [Google 2011], trabajo ya realizado por Selene Pinillos Franco (2010) en su proyecto fin de carrera “Virtual CEIP: diseño de un entorno virtual de aprendizaje para un colegio de educación infantil y primaria”.
  - Creación de terrenos: en este proyecto se han implementado dos escenarios (bosque y carretera) para lo cual se han utilizado diferentes plugins (Terrain Toolkit, River Tool y Road/Path Tool) que se han descrito anteriormente en la sección 2.7.2 Extensiones de Unity 3D. Mediante estos mismos plugins se pueden conseguir otros escenarios más detallados que añadirían mucho valor al simulador como producto final.
  - Avatares: Existen aplicaciones como la encontrada en www.evolver.com que permiten crear nuevos avatares e importarlos. En este proyecto solamente se ha trabajado, ligeramente, con las

texturas de avatares ya importados en Unity ya que existen problemas para importar avatares creados con tecnologías como las de evolver.com.

Por otro lado, en la parte funcional también existen varias líneas donde se puede mejorar la aplicación, como por ejemplo:

- **Roles**: sería necesario en un futuro crear dos perfiles diferenciados: alumno y tutor. De esta forma, solamente el tutor tendría acceso a los resultados de los alumnos y para analizarlos y mejorar la enseñanza. Asimismo, los resultados de cada alumno y la información relacionada con sus sesiones podrían almacenarse en una base de datos para facilitar el seguimiento y análisis por el tutor.
- **Ampliación de los escenarios y técnicas de simulación**: en la actualidad ya existe una lista de escenarios y técnicas a desarrollar por parte del hospital "La Paz" de Bata en función del perfil de la persona a formar:
  - Personal no sanitario (policía y bomberos)
  - Auxiliares sanitarios (personal auxiliar de enfermería)
  - Profesionales sanitarios (médicos y enfermeras)

  Este proyecto ya ha tenido este factor en cuenta y la aplicación se ha desarrollado de modo que la creación de estos nuevos escenarios se pueda hacer únicamente incluyendo nuevos modelos 3D y modificando ficheros de texto (de flujo y contenido educativo) pero sin tocar la arquitectura y el motor.
- **Incorporación a otros Proyectos Fin de Carrera**: en la actualidad existe otro proyecto fin de carrera en curso que tiene como objetivo comunicar un simulador desarrollado en Unity 3D con librerías externas que controlan el movimiento de robots de modo que el movimiento de un objeto físico se pueda replicar con total fidelidad en el mundo virtual. Esto permitiría describir el movimiento del médico, paciente, instrumental, etc… Por esta razón podría ser necesario ampliar la funcionalidad de los motores SCXML-UnityScript e IMS-QTI Lite-UnityScript haciendo posible la interpretación de nuevas instrucciones.
- **Aplicación web**: actualmente el prototipo desarrollado se distribuye como una aplicación de escritorio para las plataformas Windows y Mac. En futuras versiones se puede considerar hacer disponible la aplicación a través de un navegador web. Para ello, simplemente se debe construir el proyecto para ser

ejecutado en un navegador web al seleccionar la plataforma, como se muestra en la Figura 83.

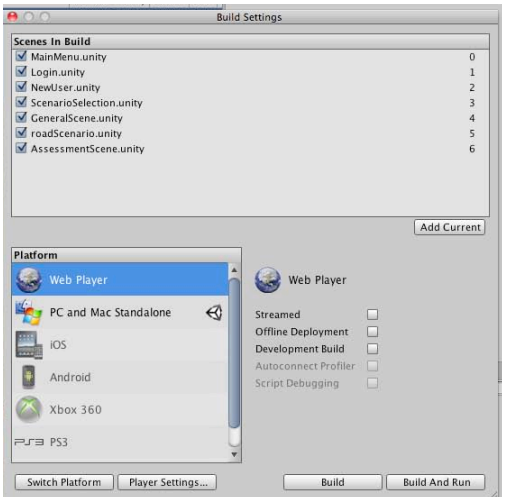

Figura 83 Selección de plataforma al construir un proyecto

Posteriormente sería necesario enlazarlo como contenido de una página web. Para reproducirlo es necesaria la instalación del plugin de Unity en el navegador.

Glosario

Avatar	*Representación gráfica humana*
3D	*3 dimensiones*
Feedback	*Observaciones*
GUI	*“Graphical User Interface”, Interfaz Gráfica de Usuario*
HUD	*“Head-Up Display”, información gráfica mostrada en un nivel de pantalla superior*
IMS	*“Instructional Management Systems”*
IMS-LD	*IMS Learning Design*
IMS-QTI Lite	*Question & Test interoperability*
Script	*Archivo de órdenes o archivo de procesamiento por lotes*
SCXML	*State Chart XML: State Machine Notation for Control Abstraction.*
UI	*“User Interface”, Interfaz de Usuario*
Unity 3D	*Herramienta para el desarrollo de juegos en 2 y 3 dimensiones*
UnityScript	*Adaptación para la herramienta de desarrollo Unity 3D del lenguaje de programación JavaScript*
Wav	*WAVEform audio file format*
XML	*eXtensible Markup Language*

Referencias

[Adobe Flash] Adobe Flash. Página web de Adobe Flash. Disponible [Internet] <http://www.adobe.com/es/products/flash.html?promoid=BPBIP> [Accedido el 1 de junio 2011]

[Agrega 2009] Agrega, 2009. Página web del proyecto Agrega. Disponible [Internet] <http://www.proyectoagrega.es/simuladores/#> [Accedido el 1 de marzo de 2011]

[Bicer 2008] Becir, V. 2008. Hospital 1 floor. Disponible [Internet] <http://sketchup.google.com/3dwarehouse/details?mid=e1effabe682fa3b18b98f3df3988f300> [Accedido el 10 de marzo de 2011]

[Gaudiosi 2008] Gaudiosi J. 2008. *Zero Hour* Trains EMTs With Virtual Worst-Case Scenarios. Wired Magazine: 16.07. Disponible [Internet] <http://www.wired.com/gaming/gamingreviews/multimedia/2008/06/pl_games> [Accedido el 16 de mayo de 2011]

[Google 2011] Google SketchUp 8, 2011. Disponible [Internet] <http://sketchup.google.com/> [Accedido el 10 de junio de 2011]

[IMS 2011] IMS Global Learning Consortium. Disponible [Internet] <http://www.imsglobal.org/> [Accedido el 18 de mayo de 2011]

[IMS 2002] IMS Global Learning Consortium, 2002. IMS Question & Test Interoperability. QTILite Specification. Final Specification Version 1.2. Date: 11 February 2002. Disponible [Internet]

<http://www.imsglobal.org/question/qtiv1p2/imsqti_litev1p2.html> [Accedido el 18 de mayo de 2011]

[IMS 2003] IMS Global Learning Consortium, 2003. IMS Learning Design Best Practice and Implementation Guide. Version 1.0 Final Specification Revision: 20 January 2003. Disponible [Internet] <http://www.imsglobal.org/learningdesign/ldv1p0/imsld_bestv1p0.html> [Accedido el 10 de junio de 2011]

[IMS] IMS Global Learning Consortium, 2006. IMS Question and Test Interoperability Overview. Version 2.1 Public Draft (revision 2) Specification. Disponible [Internet] <http://www.imsglobal.org/question/qtiv2p1pd2/imsqti_oviewv2p1pd2.html> [Accedido el 10 de junio de 2011]

[Mono 2011] Página web principal de Mono. Disponible [Internet] <http://www.go-mono.com/docs/index.aspx> [Accedido el 10 de junio de 2011]

[NEMSPI 2011] NEMSPI, National Emergency Medical Services Preparedness Initiative. Disaster Gaming Zero Hour: America's Medic. Disponible [Internet] <http://inside.gwumc.edu/nemspi/disaster_simulation.htm> [Accedido el 5 de marzo de 2011]

[Pinillos Franco 2010] Pinillos Franco, S. 2010. Virtual CEIP: diseño de un entorno virtual de aprendizaje para un colegio de educación infantil y primaria

[PMBOK 2004] PMBOK, 2004. Project Management Body of Knowledge Third Edition. Project Management Institute 2004

[PMI 1992] PMI, 1992. Project & Program Risk Management: a guide to managing project risks & opportunities; edited by R. Max Wideman ISBN: 1880410060.

[Rins 2010] Rins, E.C. 2010. Los simuladores como herramienta y la cognición distribuida. Disponible [Internet] <http://www.educant.org/node/4526> [Accedido el 8 de junio de 2011]

[Ruiz-Parra *et al.* 2009] Ruiz-Parra, A.I., Angel-Muller, E. y Guevara, O. 2009. Clinical simulation and virtual learning. Complementary technologies for medical education. Rev.fac.med.unal. Jan./March 2009, vol.57, no.1, p.67-79. Available from <http://www.scielo.unal.edu.co/scielo.php?script=sci_arttext&pid=S0120-00112009000100009&lng=en&nrm=iso>. ISSN 0120-0011. [Accedido el 8 de junio de 2011]

[Salas *et al.* 1994] Salas Perea, R. S. y Ardanza Zulueta, P., 1994. La simulación como método de enseñanza y aprendizaje. Centro nacinal de Perfecionamiento Médico. Cuba. Disponible [Internet]

<http://bvs.sld.cu/revistas/ems/vol9_1_95/ems03195.htm> [Accedido el 18 de mayo de 2011]

[Six Times Nothing, 2009a] Six Times Nothing, 2009a. Terrain Toolkit, Disponible [Internet] <http://www.sixtimesnothing.com/terraintoolkit/> [Accedido el 1 de Mayo de 2011]

[Six Times Nothing 2009b] Six Times Nothing, 2009b. River Tool Disponible [Internet] <http://sixtimesnothing.com/river-tool/> [Accedido el 1 de Mayo de 2011]

[Six Times Nothing 2009c] Six Times Nothing, 2009c. Road/Path Tool Disponible [Internet] <http://sixtimesnothing.com/road-path-tool/> [Accedido el 1 de Mayo de 2011]

[Unity Answers 2011a] Unity Answers, 2011. Disponible [Internet] <http://answers.unity3d.com/questions/10608/why-build-a-web-game-using-unity-rather-than-flash.html> [Accedido el 8 de junio de 2011]

[Unity Answers 2011b] Unity Answers, 2011. Disponible [Internet] <http://answers.unity3d.com/> [Accedido el 1 de marzo de 2011]

[Unity 2011a] Unity, 2011. Scripting Overview. Disponible [Internet] <http://unity3d.com/support/documentation/ScriptReference/> [Accedido el 10 de junio de 2011]

[Unity 2011b] Unity, 2011. Página oficial. Disponible [Internet] <http://unity3d.com/> [Accedido el 10 de junio de 2011]

[Unity 2011c] Unity, 2011. Overview: Script compilation (Advanced). Disponible [Internet] <http://unity3d.com/support/documentation/ScriptReference/index.Script_compilation_28Advanced29.html> [Accedido el 26 de mayo de 2011]

[Unity 2011d] Unity, 2011. System requirements. Disponible [Internet] <http://unity3d.com/unity/system-requirements> [Accedido el 11 de mayo de 2011]

[Unity 2011e] Unity, 2011. Best Of All Worlds. Disponible [Internet] <http://unity3d.com/unity/features/scripting> [Accedido el 8 de junio de 2011]

[Unity RF 2011] Página web Unity Real Factory. Disponible [Internet] <http://foros.unity-rf.info/index.php> [Accedido el 1 de marzo de 2011]

[Unreal Technology 2009] Unreal Technology, 2009. Gaudiosi J. Disponible [Internet] <http://www.unrealengine.com/showcase/simulation_training/zero_hour/> [Accedido el 30 de mayo de 2011]

[Virtual Heroes 2007] Virtual Heroes, 2007. Zero Hour America's Medic. National Emergency Medical Services Preparedness Initiative. Disponible [Internet] <http://www.nationalemspreparedness.org/> [Accedido el 2 de marzo de 2011]

[Vergara Olivares et al.] Vergara Olivares, J.M, Buforn Galiana, A., Rodríguez Serrano, C. Triage. Disponible [Internet] <www.sld.cu/galerias/pdf/sitios/pdguanabo/triage.pdf> [Accedido el 1 de abril de 2011]

[W3C 2008] World Wide Web Consortium, 2008. Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation 26 November 2008. Disponible [Internet] <http://www.w3.org/TR/2008/REC-xml-20081126/> [Accedido el 10 de junio de 2011]

[W3C 2011] World Wide Web Consortium, 2011. State Chart XML (SCXML): State Machine Notation for Control Abstraction. W3C Working Draft 26 April 2011. Disponible [Internet] <http://www.w3.org/TR/scxml/> [Accedido el 25 de mayo de 2011]

[Wikipedia 2011a] Wikipedia 2011. Flight simulator. Disponible [Internet] <http://en.wikipedia.org/wiki/Flight_simulator> [Accedido el 1 de junio de 2011]

[Wikipedia 2011b] Wikipedia 2011. IMS Learning Design. Disponible [Internet] <http://en.wikipedia.org/wiki/IMS_Learning_Design> [Accedido el 18 de mayo de 2011]

[Wikipedia 2011c] Wikipedia 2011. SCXML. Disponible [Internet] <http://en.wikipedia.org/wiki/SCXML> [Accedido el 18 de mayo de 2011]

[Wikipedia 2011d] Wikipedia 2011. Triaje. Disponible [Internet] <http://es.wikipedia.org/wiki/Triaje> [Accedido el 18 de mayo de 2011]

[Wikipedia 2011e] Wikipedia 2011. XML Extensible Markup Language. Disponible [Internet] <http://es.wikipedia.org/wiki/Extensible_Markup_Language> [Accedido el 7 de junio de 2011]

Anexos

Manual de usuario

SimEmergencias es un simulador 3D para el aprendizaje del personal de asistencia sanitaria extra-hospitalaria. En el presente manual se explica el funcionamiento general de la aplicación y se recogen algunas ayudas sobre posibles dudas que podrían surgir al usuario durante la simulación.

Requisitos del sistema

Los requisitos del sistema especificados por Unity [Unity 2011] para ejecutar correctamente una aplicación desarrollada en el motor de juegos Unity es la siguiente:

- Sistema operativo: Windows 2000 o posterior; Mac OS X 10.4 o posterior
- Tarjeta gráfica: casi todas las tarjetas gráficas 3D, dependiendo de la complejidad gráfica de la aplicación.

Instrucciones de instalación

El fichero de instalación se encuentran disponibles en la siguiente dirección web:

https://svn.gast.it.uc3m.es/PFCs/SimuladorDeEmergenciasConUnity_2/SimEmergencias

Para acceder a dichos ficheros de instalación es necesario introducir un ‘usuario’ y ‘contraseña’ válidos.

Para instalar la aplicación simplemente hacer doble-click sobre el fichero descargado, *SimEmergencias.zip*. Transcurridos unos segundos aparecerá una carpeta con nombre *SimEmergencias*, dentro de ésta se encuentra el archivo ejecutable y demás ficheros necesarios.

Si se ha descargado la aplicación para la plataforma Mac, el ejecutable se llama *SimEmergencias.app*.

Si se ha descargado la aplicación para la plataforma Windows, el ejecutable *SimEmergencias.exe*.

Nota: los ficheros de la aplicación se encuentran disponibles en el CD-ROM adjunto en este documento.

Arranque de la aplicación

Para lanzar la aplicación simplemente hacer doble click sobre el icono de la aplicación. El icono de la aplicación se muestra en la Ilustración 1.

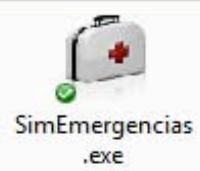

Ilustración 1 Icono de la aplicación

A continuación aparecerá una pantalla para seleccionar la resolución de la pantalla, así como la calidad gráfica. Se recomienda utilizar las opciones por defecto pues esta aplicación ha sido optimizada para una resolución de 1024 x 768. La elección del nivel de calidad gráfica dependerá del equipo (en especial de la tarjeta gráfica) donde el usuario ejecute el simulador).

También se puede seleccionar si la aplicación se ejecutará dentro de una ventana ("Windowed") o bien en modo pantalla completa, esta última opción se activará al quitar la marca de la opción "Windowed".

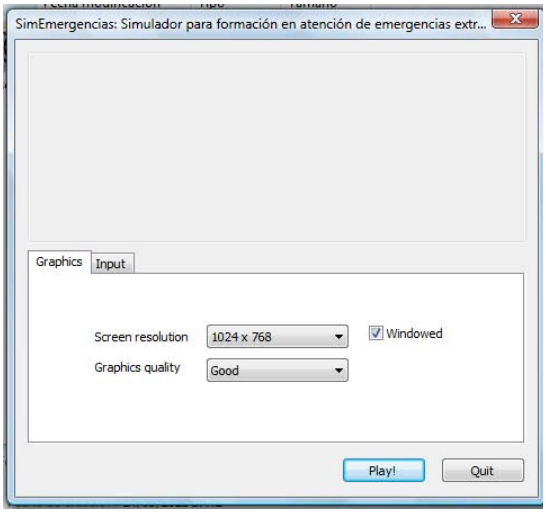

Ilustración 2 Ventana de configuración de resolución y calidad gráfica

Si se quisiese modificar las teclas de entrada con los que el usuario controlará el personaje durante la simulación, habría que hacer "click" en la pestaña "Input", como se muestra en la Ilustración 3.

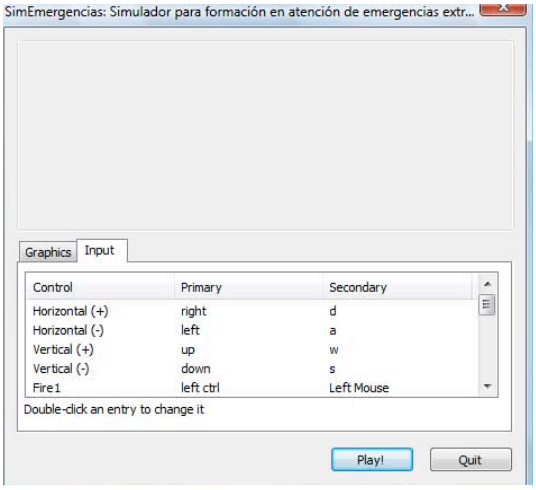

Ilustración 3 Ventana de configuración de teclas de entrada

En cualquiera de los casos, para comenzar la aplicación se debe pulsar el botón "Play!", acto seguido aparecerá una pantalla de transición como la que se muestra a continuación. En caso de querer salir de la aplicación habría que pulsar el botón "Quit".

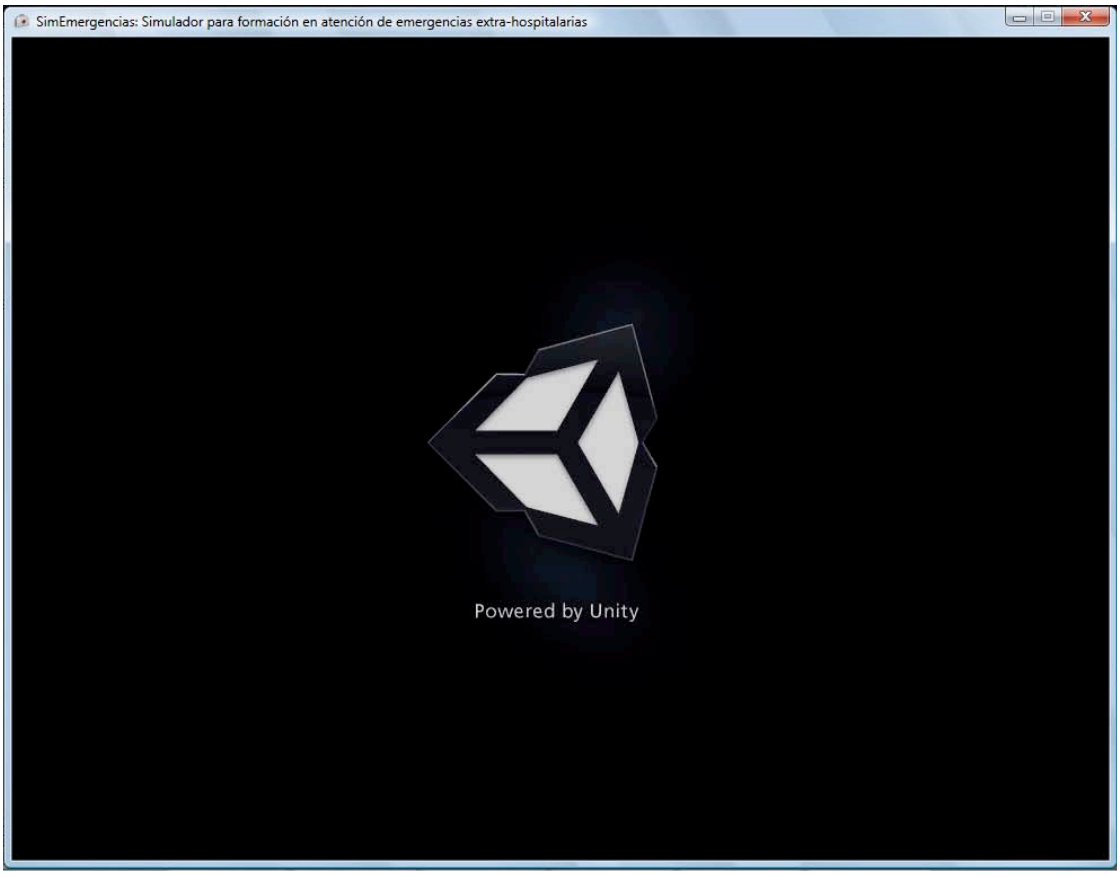

Ilustración 4 Pantalla de carga de la aplicación

Menú principal

Es la primera pantalla que se muestra al iniciar la aplicación. Desde esta pantalla el usuario puede entrar en el simulador mediante su cuenta de usuario (opción “Entrar”); si todavía el usuario no dispone de cuenta puede crear una a través de la opción “Nuevo usuario”; también puede cambiar el idioma de la aplicación, pudiendo elegir entre castellano e inglés, finalmente el usuario puede salir de la aplicación mediante la opción “Salir”.

Ilustración 5 Pantalla del Menú principal

Opción de entrada (login)

En esta pantalla el usuario debe introducir su nombre de usuario y su clave. A continuación debe pulsar “Continuar” para pasar a la pantalla de selección de escenario. Si el usuario pulsa el botón “Atrás” se volverá a la pantalla de menú principal.

Ilustración 6 Pantalla de Login

Opción crear nuevo usuario

En esta pantalla un usuario sin cuenta puede crearse una cuenta nueva. Para ello es necesario introducir los siguientes datos del alumno: nombre y apellidos, nombre de usuario y contraseña (se debe introducir dos veces para asegurar la corrección de ésta).

A continuación debe pulsar “Continuar” para pasar a la pantalla de selección de escenario. Si el usuario pulsa el botón “Atrás” se volverá a la pantalla de menú principal.

Ilustración 7 Pantalla de creación de cuenta de nuevo usuario

Opción idioma

Cambia el idioma de la aplicación, de castellano a inglés, o viceversa.

Ilustración 8 Pantalla de Menú principal en inglés

Opción salir

Se cierra la aplicación.

Escenarios de simulación

El paso previo a comenzar un escenario de simulación es elegir qué escenario o técnica se quiere entrenar, además de elegir el personaje de la simulación.

Menú de selección de escenario de simulación

Llegado el usuario a esta pantalla debe seleccionar el personaje con el que quiere entrar en el simulador como se muestra en la Ilustración 9.

Ilustración 9 Pantalla de selección de escenario de simulación – elección de personaje

Acto seguido se debe seleccionar la simulación a entrenar por “Escenario” o bien por “Técnica”.

La selección por “Escenario” permite al usuario practicar los casos típicos que ocurren en una ubicación concreta, por ejemplo un bosque o una carretera. Si la selección es por “Técnica”, el usuario podrá practicar exclusivamente la técnica seleccionada, por ejemplo shocks, heridas, hemorragias o traumas. Nótese que en esta versión de la aplicación el usuario no podrá optar por esta opción.

Así, si se pulsa en cualquiera de las opciones, “Escenario” o “Técnica”, aparecerá a su derecha una columna de opciones donde poder elegir. En el ejemplo de la Ilustración 10 se ha seleccionado la opción “Escenario” y los escenarios disponibles son: “Bosque” o “Carretera”.

Ilustración 10 Pantalla de selección de escenario de simulación – selección por escenario

En este estado, si el usuario pasa el ratón por cada una de las opciones puede ver una imagen descriptiva del escenario y un pequeño texto explicativo. En la Ilustración 11 se muestra el resultado de pasar el ratón sobre la opción “Bosque”. Nótese cómo la imagen de fondo ha cambiado.

Ilustración 11 Pantalla de selección de escenario de simulación – información sobre escenario Bosque

En la Ilustración 12 se muestra cómo cambiaría la pantalla si el usuario pasase el ratón sobre la opción “Carretera”. Como se puede ver, la imagen de fondo y el texto explicativo han cambiado.

Ilustración 12 Pantalla de selección de escenario de simulación – información sobre escenario Carretera

En el caso de que el usuario estuviese interesado en entrenar una técnica específica, se debe seleccionar la opción técnica y seleccionar una de las opciones de la lista que aparece a la derecha de la pantalla, se muestra un ejemplo a continuación. Como se ha indicado anteriormente, en esta versión de la aplicación no está disponible esta opción.

Ilustración 13 Pantalla de selección de escenario de simulación – selección por técnica

Finalmente, si por ejemplo se selecciona el escenario “Bosque”, la imagen de fondo se cambiará permanentemente por la de la descripción del escenario “Bosque” y ya se podrá pasar al simulador tras hacer “click” en el botón “Ir a simulador”.

Ilustración 14 Pantalla de selección de escenario de simulación – simulación seleccionada

Controles

Controles de teclado

Los controles de teclado por defecto con los que el usuario puede mover al personaje por la pantalla son:

- '←' o 'a' para girar a la izquierda.
- '→' o 'd' para girar a la derecha.
- '↑' o 'w' para avanzar hacia delante.
- '↓' o 's' para avanzar hacia atrás.
- 'Mayús' para mover la personaje más deprisa
- 'Barra espaciadora' para saltar

Controles de ratón

El usuario a través del ratón puede interaccionar con los diferentes objetos del entorno y con los elementos de menú y botones.

Un ejemplo de interacción con los elementos del entorno ocurre cuando se pasa el ratón por ciertos elementos clave, como pueden ser una ambulancia, una víctima o un compañero, como resultado de esta acción aparece, por ejemplo, un menú en pantalla.

Interfaz de usuario

En este apartado se analizarán los elementos de la interfaz de usuario.

Elementos visuales de pantalla o Heads-up Display (HUD)

Durante la simulación, aparecen en la pantalla diferentes elementos y paneles que se explican en las imágenes a continuación.

Ilustración 15 Explicación de elementos visuales de pantalla

1. Panel de puntuación: en este panel se detalla la puntuación indicando el número de opciones correctas e incorrectas.
2. Panel de progreso: en este panel se muestra la puntuación conseguida hasta el momento sobre el máximo posible.
3. Cronómetro de cuenta atrás: muestra el tiempo restante para que finalice el tiempo máximo establecido para resolver una determinada situación.
4. Cuadro de diálogo: muestra una imagen representativa del personaje que está diciendo ese mensaje, y el mensaje en sí.

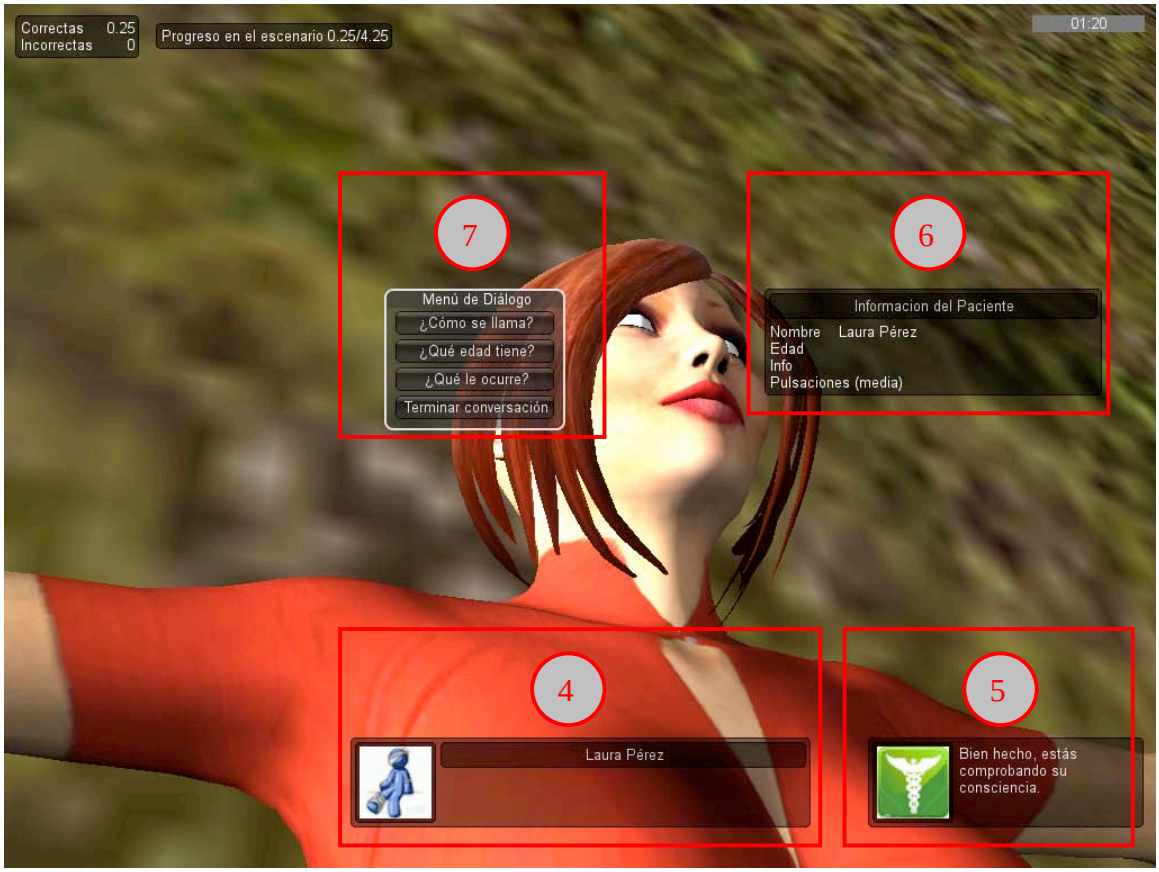

Ilustración 16 Explicación de elementos visuales de pantalla

5. Panel de feedback: en este panel se propone realimentación al usuario por las acciones tomadas y orientaciones sobre la siguiente acción a realizar.
6. Panel de constantes vitales de la víctima: en este cuadro se muestra la información relativa al historial médico de la víctima que se ha conseguido mediante diálogo con ella o usando pruebas diagnósticas u otros instrumentos de valoración (por ejemplo valor medio del pulso).
7. Menú de acción: a través de este tipo de menú el usuario puede interaccionar con el entorno, víctimas, etc… y tomar decisiones. Estos menús están compuestos por un título (situado en la cabecera de la ventana) y de botones donde se muestran las diferentes opciones.

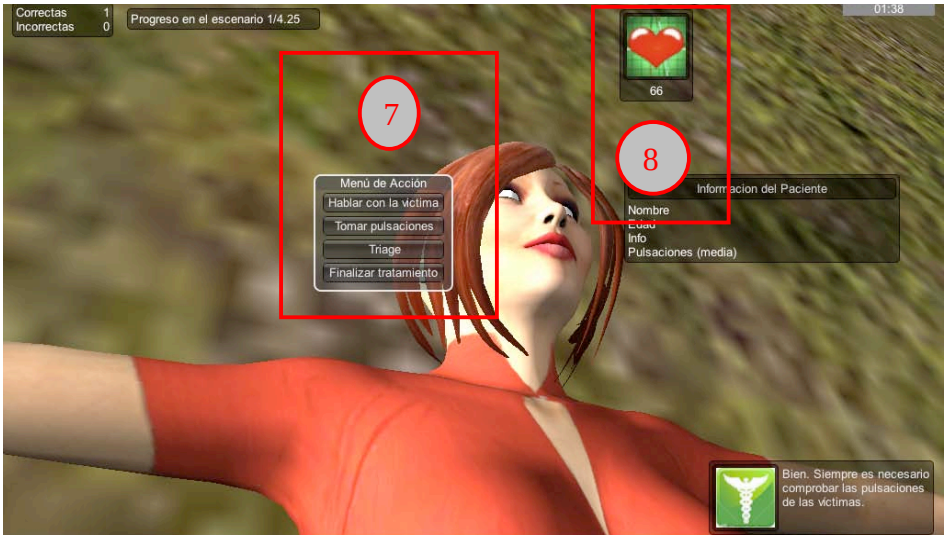

Ilustración 17 Explicación de elementos visuales de pantalla

8. Cuadro de medición de constantes: en este tipo de cuadro se muestra información que se está tomando relativa a las constantes vitales de la víctima, en este caso pulsaciones.

Ilustración 18 Explicación de elementos visuales de pantalla

9. Visualizador de triage: mediante este marcador visual, una vez realizada la primera valoración de la víctima aparecerá en pantalla un icono de color que indica la clasificación de las víctimas atendiendo al código internacional de colores tal y como se indica en la siguiente tabla.

Blanco	cuando la víctima ha fallecido
Negro	cuando las posibilidades de recuperación son nulas
Rojo	cuando el paciente tiene posibilidad de sobrevivir y la actuación médica debe ser inmediata
Amarillo	es un paciente diferible, para ser vigilado mientras se le puede atender
Verde	paciente levemente lesionado, que puede caminar y su traslado no precisa medio especial

Finalización de la simulación

Tras haber realizado todas las tareas que el usuario estime oportunas, para terminar la simulación se ha de interactuar con el paramédico compañero e indicarle que hay que transportar a las víctimas como se muestra en el menú que aparece en la siguiente pantalla.

Ilustración 19 Explicación de elementos visuales de pantalla

Pantalla de resultados

Tras finalizar la simulación se muestra la pantalla de resultados obtenidos. En ella se desglosan las puntuaciones por categorías y se muestra la puntuación conseguida.

Ilustración 20 Pantalla de resultados

Modificación de contenidos

El usuario final en fase de producción puede modificar parte del contenido de la aplicación. Para ello tendrá que proceder a editar ciertos ficheros de texto y modificar su contenido. También en este caso se puede modificar el sonido del corazón.

La siguiente tabla indica los contenidos modificables, el fichero desde el que se puede hacer la modificación y la referencia a la explicación de dicho fichero.

Funcionalidad	**Ruta del fichero a modificar**	**Referencia**
Habilitar la funcionalidad proporcionada por un *script*	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Deshabilitar la funcionalidad proporcionada por un *script*	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Mostrar el panel de diálogo con un texto y una imagen	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Esconder el panel de diálogo	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Añadir condición	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Esconder panel de feedback	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados

Funcionalidad	**Ruta del fichero a modificar**	**Referencia**
Cerrar ventana	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Comenzar la animación multimedia de toma de pulso al paciente	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Ajustar color al identificador de triage	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Modificar u obtener la información mostrada en la tabla de información de la víctima	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Mostrar el contenido indicado del panel de “feedback”	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Mostrar un menú donde se requiere la acción del usuario interaccionando con él	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Definir cual es archivo de contenidos educativos	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados

Funcionalidad	Ruta del fichero a modificar	Referencia
Definir cual es archivo de los contenidos (historial médico) de las víctimas	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Definir cual es el archivo multimedia con el sonido de latidos de corazón	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Definir cual es el archivo donde se encuentran los datos relativos a las pulsaciones de una víctima	*files/flow.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del flujo de estados
Definir ventana de actividad educativa	*files/eduContent.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del contenido educativo
Asignar el título de la ventana de actividad educativa	*files/eduContent.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del contenido educativo
Definir los botones de la ventana de actividad educativa	*files/eduContent.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del contenido educativo
Asignar la puntuación asociado a pulsar los botones de la ventana de actividad educativa	*files/eduContent.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del contenido educativo

Funcionalidad	**Ruta del fichero a modificar**	**Referencia**
Asignar el feedback asociado a pulsar los botones de la ventana de actividad educativa	*files/eduContent.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del contenido educativo
Asignar acciones en el sistema (por ejemplo cerrar una ventana) asociadas a pulsar los botones de la ventana de actividad educativa	*files/eduContent.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de definición del contenido educativo
Definir la información propia de la víctima relacionada con su historia clínica (nº identificador, nombre, edad y síntoma de la víctima)	*files/* Ejemplo: *files/victim.xml*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de víctimas
Incluir ficheros de instrumentos de diagnóstico	*files/heart/* Ejemplo: *files/heart/heart*	4.4.1 Ficheros de contenido del escenario de simulación → Fichero de datos de instrumentos de diagnóstico
Incluir ficheros de diagnóstico multimedia	*files/audio/* Ejemplo*:* *files/audio/heartbeat.wav*	4.4.1 Ficheros de contenido del escenario de simulación → Ficheros para diagnóstico multimedia