Título

CandyNapper

Grupo

JohnCor

Autores

Carrillo Boj, Carlos; 3lcarry@gmail.com; 659344180
Gonzalez Martín, Adrián; kaseyo23@gmail.com; 635645081
Fernandez Ferrandez, Esteve; eff9@alu.ua.es; 678814149
Pastor Esteve, Gustavo; gustavo.pastor.esteve@gmail.com ; 695551868

1. Introducción

1.1. Presentación

El proyecto será un videojuego de acción en 3º persona con gráficos en 3D.

En el juego, manejaremos a una versión moderna de Hansel y Gretel (Hans y Greta), que, en lugar de en una casa, quedan atrapados en un mundo, SweetWorld, dominado por "La Bruja" y su ejercito de gominolas.

Durante el desarrollo del juego, tendremos que superar una serie de objetivos para conseguir escapar de este mundo. Para ello, controlaremos a uno de los dos personajes pudiendo elegir en tiempo real entre uno y otro. Además deberemos pelear contra el ejercito de gominolas. Ambos personajes tendrán un ataque propio.

En general, el juego seguirá una línea más desenfadada que el estilo clásico de los juegos de acción. Para los gráficos usaremos un estilo no fotorealista, más cercano a los dibujos animados.

1.2. Objetivos

– Crear un Menú inicial, donde puedas decidir entre partida nueva, continuar una anterior, o ajustar algunas opciones básicas. Crear también un menú ingame para guardar la partida.

– Los personajes serán modelos 3D en formato 3DS, diseñados por nosotros.

– Habrán 2 personajes que podrás controlar en todo momento (uno u otro) intercambiándolos cuando lo necesites en el transcurso del juego (sus características están detalladas más abajo).

– Visión del juego desde detrás del personaje principal que estemos controlando en ese momento.

– Habrán 4 tipos de enemigos diferentes (detallados más abajo).

– Hans y Greta, deberán cumplir una serie de objetivos para poder huir de SweetWorld. Implementaremos, al menos, una parte de ellos.

– Implementar un sistema de vida, tanto para los enemigos como para Hans y Greta, basado en: Health Points para los enemigos, y Diabetes Points para Hans y Greta.

– Controlar el movimiento del personaje con el teclado, y su punto de mira con el ratón. Podremos moverlos hacia delante, hacia detrás y con un salto lateral (para esquivar).

– Implementar un HUD que indique la vida restante y un minimapa.

– Integrar un motor de física ya existente, que podamos usar para la detección de colisiones.

– Implementar un sistema, que permita guardar o cargar el estado del juego (mediante ficheros XML). También lo usaremos para cargar y colocar todas las

entidades y el fondo al principio de cada fase.

- Implementar un motor de IA para los enemigos.

Objetivos secundarios:

- Incluir audio y efectos sonoros durante el juego.
- Añadir efectos visuales como humo, brillos, etc.
- Añadir más efectos de física.
- Añadir más fases y dos "jefes finales".
- Hacer el juego multiplataforma (Mac, Windows y Linux).

1.3. Herramientas Hardware y Software

- Doxygen. Para la documentación del código
- Cmake. Para compilar independientemente del IDE o plataforma
- Blender. Para crear los modelos 3D de los personajes
- OpenGL. Para el motor gráfico.
- C++. Como lenguaje principal de programación.
- SFML. Como 'toolkit' para controlar el teclado, ratón, audio y ventana.

1.4. Requerimientos

El juego está orientado a personas mayores de 12 años, ya que contiene escenas de violencia.

Los requisitos, serán un PC, con teclado y ratón y soporte de aceleración 3D con OpenGL. En un principio queremos que sea multiplataforma, por lo que compilará y se ejecutará en Linux, Windows y Mac, siempre que tengan instaladas las herramientas básicas para su compilación (cmake, make, g++, etc).

2. Análisis

2.1. Identificación de las necesidades (captura de requisitos)

2.1.1.Evaluación y Síntesis

La historia del juego empieza una tarde. Hans y Greta, dos hermanos, están pasando una tarde en el bosque. Greta se asoma a un pozo y cae dentro. Cuando Hans se da cuenta, corre a ayudarla pero cae también. Ambos pierden la consciencia y despiertan en SweetWorld.

Antaño, SweetWorld era un sitio feliz, habitado por seres de gominola, hasta que una bruja lo sometió con su ejercito. Ahora nadie puede escapar de SweetWorld (incluidos Hans y Greta).

El videojuego, tendrá un único modo de juego, que sera el modo historia en el cual tienes que ir cumpliendo unos objetivos o fases, para acabar completando la historia. En cada fase, los personajes principales se enfrentarán a los miembros del ejercito de La Bruja, que son seres de gominola. Aparte, en determinadas ocasiones, también serán ayudados por personajes no afines a la bruja.

Los personajes principales son:

- **Hans Grimm**. Es uno de los dos protagonistas, junto con su hermana Greta. Accidentalmente aparece en SweetWorld y deberá derrotar a la Bruja para escapar. Al principio del juego conseguirá una pistola de agua para defenderse.
- **Greta Grimm**. Es la otra protagonista del juego. Junto con su hermano, entra en SweetWorld y con su ayuda deberá huir. Usa un tirachinas como arma.
- **La Bruja**. Con sus poderes hechizó a parte de los habitantes de SweetWorld y formó un ejercito propio. Con él, sometió al resto de habitantes de SweetWorld y se convirtió en una tirana. Su arma principal son sus poderes.
- **El Lugarteniente de La Bruja**. Es el guardaespaldas personal de La Bruja. Su maldad es sólo equivalente a la de La Bruja. Su tamaño es desmesurado y su fuerza es su única arma.
- **Resistencia Anti-Bruja**. Son un grupo de rebeldes, que aun no se han sometido a la bruja. Apenas tienen poder, pero ayudarán en lo que puedan a Hans y a Greta.

Los enemigos básicos (aparte de La Bruja y El Lugarteniente), serán:

- **Soldado de Jengibre.** Es la unidad básica del ejercito. Son muñecos de jengibre que únicamente tendrán un ataque físico con un palo de regaliz.
- **Chicle Boom.** Son bolas de chicle que explotan en cuanto las tocas. Se mueven solas.
- **Fusilero Harbo.** Es una unidad más avanzada. Son ositos de gominola que atacan a distancia lanzando algodón de azúcar (que dejará inmóvil) o bolas de caramelo.
- **Unidad especial O.S.O.** Son las fuerzas especiales del ejercito de La Bruja. Es una combinación de las anteriores, pueden atacar de manera física, inmolandose o a distancia.

2.1.2.Funcionalidades (Requisitos-conviene numerarlos)

Menú principal

Una vez abramos el juego aparecerá un menú que nos dejará elegir entre:

- Iniciar nueva partida: empezaremos una partida de 0, con la posibilidad de guardarla en cualquier momento.
- Continuar partida: nos dejara cargar una partida guardada anteriormente. La elegiremos de una lista.
- Opciones: nos dejará ajustar una serie de opciones:
  - Controles (teclas a las que están asignados)
  - Video (resolución de pantalla)
  - Sonido (volumen de la música, volumen de los sonidos)
- Salir: saldrá del juego.

Pantalla de juego

Tanto si elegimos "Iniciar nueva partida" como "Continuar partida" entraremos en la pantalla de juego. En ella podemos distinguir diversos elementos:

- HUD: nos informará de la vida restante (en nuestro caso sera una barra de diabetes que va aumentando) y nos dejará ver un minimapa, donde aparecerá el objetivo actual y otros elementos del juego.
- Escenario: serán modelos 3D, independientes de las entidades. De hecho, no "interactuan" con ellas.

- Entidades:
  - Objetos: Se diferencian del fondo en algunas características, como por ejemplo, que el resto de entidades pueda colisionar con ellos.
  - Personaje principales: será un modelo en 3D del protagonista que tengamos activo en ese momento.

Nombre	Vida (DP)	Tipo de ataque	Daño de ataque (HP)
Hans	*300*	*Físico* *Físico (espada de madera)*	**10** **30**
Greta	*200*	*Físico* *A distancia (pistola de agua)*	**5** **25**

  - Enemigos:

Nombre	Vida (HP)	Tipo de ataque	Daño de ataque (DP)
Soldado de jengibre	*100*	*Físico*	**5**
Chicle Boom	*50*	*Inmolación*	**25**
Fusilero Harbo	*75*	*A distancia* *Ralentizante*	**10** **Disminuye la velocidad de nuestro personaje en un 50%**
Unidad especial O.S.O.	*200*	*Físico* *A distancia* *Ralentizante*	**10** **15** **Disminuye la velocidad de nuestro personaje en un 70%**

Menú ingame

Si pulsamos en la pantalla de juego la tecla 'Esc', aparecerá el menú ingame, que nos permitirá:

- Continuar partida: volver a la pantalla de juego.
- Guardar partida: guarda la partida actual.
- Volver al menú principal: abandonará la partida actual y volverá al menú inicial del juego.

- Salir: abandonará la partida actual y saldrá del juego. Si no hemos guardado antes, se perderá nuestro progreso.

Pantalla de muerte
Si morimos con uno de los dos personajes, aparecerá la pantalla de muerte, que nos permitirá:

- Volver a ultimo punto de guardado: volverá al instante donde se guardó la partida por última vez.
- Volver a menú principal.
- Salir.

Los diferentes estados del juego los controlaremos mediante una máquina de estados finita.

2.2. Estudio de la viabilidad

2.2.1.Técnica

El lenguaje con el que implementaremos el juego sera C++, dado que usaremos programación orientada a objetos y por su potencia en cuanto al diseño de videojuegos.

Para la parte de la IA, usaremos el motor que implementaremos en la asignatura de Razonamiento, el cual adaptaremos para nuestro juego. En principio, tenemos pensado usar máquinas de estados finitos para controlar los distintos estados del juego (menú, pausa, pantalla de juego, muerte), lógica difusa combinada con maquinas de estados (patrulla, huir, ataque) para controlar el comportamiento de los enemigos y pathfinding para el movimiento de los personajes.

Para la parte gráfica, implementaremos un motor en la asignatura de Gráficos avanzados y animación que no sirva de interfaz para OpenGL. Los gráficos serán 3D y la vista sera en 3ª persona desde detrás del personaje.

Para la física, integraremos un motor 2D ya existente (Box2D, probablemente). En principio, lo usaremos únicamente para la detección de colisiones, pero si tenemos tiempo, controlaremos mas comportamientos físicos.

Respecto al motor de Entrada/Salida, usaremos una librería que nos permita controlar los eventos de teclado y ratón, el audio, y la "ventana" del juego. Para ello, nos planteamos dos posibilidades: la librería GLUT de OpenGL más otra librería de audio, o la SDL, que nos ofrece todo junto (tanto teclado y ratón, como audio y gestión de la "ventana").

2.2.2.Estimación de riesgos

Los principales riesgos que corre nuestro proyecto son de tipo técnico, y aquellos derivados por fallos en el diseño, gestión del trabajo en equipo o planificación.

Respecto a los riesgos técnicos, trabajaremos con tecnologías que no dominamos, como OpenGL, XML (parseo y escritura) o incluso el diseño de modelos 3D en Blender. Lo cual, como mínimo, nos retrasará bastante al principio por temas de documentación.

Por otro lado, nos enfrentaremos a la aplicación más grande que hayamos hecho ninguno (de momento, espero) y a nuestro primer videojuego, con los posibles fallos de diseño (y demás) que eso conlleva. Además, tenemos que desarrollarla en un grupo de 4 personas, lo cual puede ser una ventaja si podemos gestionar bien el trabajo en equipo, o todo lo contrario.

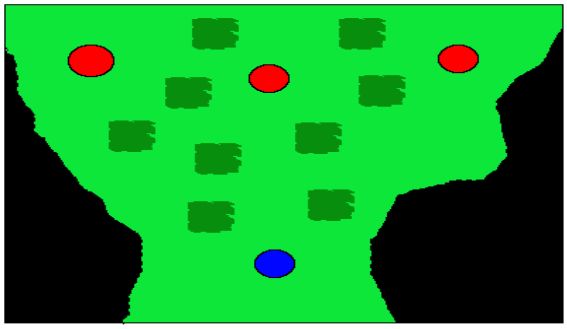
Por último, respecto a la planificación, posiblemente nos hayamos equivocado con el tiempo o la dificultad asignada a cada hito lo cuál retrasará el proyecto y la consecución de las fases que hemos ideado. Todo eso sin contar posibles retrasos ajenos a la planificación, como enfermedades, exámenes, otras practicas, etc.

2.2.3.Planificación temporal

Hito 1 (20/12/2011).

- H1.1 – Implementar la estructura del mapa, para poder ubicar en este el personaje y los enemigos, junto con la estructura básica para manejar las entidades del juego. Las entidades del juego, junto con el fondo, será posible cargarlos desde un fichero XML.
- H1.2 – Aunque el mapa sera en 3D, crear una representación 2D sencilla para para poder realizar pruebas.
- H1.3 – Implementar una máquina de estados para cambiar entre las posibles acciones de los enemigos (stand, patrullar, atacar, huir, perseguir). Se creara la estructura de la máquina de estados y se creara el estado patrullar, y perseguir al enemigo. El resto de estados se irán implementando más adelante.
- H1.4 – Integrar un motor de física, en este hito únicamente la detección de colisiones, quizá se añadan más funcionalidades.
- H1.5 – Añadir al motor de IA, además de la máquina de estados el modulo de pathfinding.
- H1.6 – El motor de IA, además de pathfinding y máquina de estados contendrá un motor de lógica difusa y redes neuronales (que implementaremos en Razonamiento), aunque no los integraremos en nuestro juego.
- H1.7 – Comenzar a crear el motor de entrada/salida. En este hito simplemente manejaremos el teclado, posteriormente incluiremos ratón.
- H1.8 – Crear un entregable, de la representación 2D donde nuestros personajes serán puntos (rojo enemigo, azul nuestro personaje), donde se podrá comprobar el correcto funcionamiento del movimiento de los enemigos (pathfinding) y el cambio de estados de los personajes (máquina de estados). En este entregable, tendremos un menú con dos opciones. Una será para poder introducir diferentes obstaculos (serán

cajas negras por el mapa) y otro para "jugar" y comprobar el correcto funcionamiento.



Hito 2 (20/03/2011).

- H2.1 – Implementar un cargador de modelos 3D, formato 3DS. Además de cargar los modelos, también se le podrán aplicar una serie de texturas.
- H2.2 – Entregable de una interfaz gráfica, en el que podremos probar el correcto funcionamiento del cargador de modelos 3D, pudiendo rotar el modelo y poder aplicarle diversas texturas.
- H2.3 – Crear y cargar el Mapa 3D.
- H2.4 – Implementar el movimiento libre de la cámara por el mapa, mediante el manejo del teclado.
- H2.5 – Crear un editor de mapa, en el cual podremos cargar fondos, y a este ir añadiendo una serie de obstáculos diversos como arboles, casas, y demás objetos que se nos ocurran. Además también podremos indicar la posición original de los enemigos. En si mismo, este editor sera un entregable también.
- H2.6 – Motor del juego, que integre el resto de motores y que haga funcionar correctamente las diferentes partes del juego.
- H2.7 – Cargar nuestro personaje, y implementar que la cámara este constantemente siguiéndole, y que el personaje se pueda mover libremente por el mapa.
- H2.8 – Implementar Ataques de los enemigos.
- H2.9 – Implementar la detección del ratón en el Motor Entrada/Salida.
- H2.10 – Entregable donde podemos ver el mapa con nuestro personaje (en este te dará a elegir cual quieres de los dos, ni aun tendremos funcionando el cambiar personajes), encontrando diferentes enemigos, y poder probar los ataques de los enemigos y nuestros ataques.

Hito 3 (22/05/2011).

- H3.1 – Implementar un motor de optimización entera.
- H3.2 – Crear una interfaz gráfica, para usar el motor de optimización, la cual nos permitirá introducir variables al problema, la función objetivo (a minimizar o maximizar) y una serie de restricciones y sea capaz de darnos la solución optima.
- H3.3 – Implementar el HUD, en el motor gráfico, en el cual veremos un minimapa, la vida que nos queda e información del objetivo actual.
- H3.4 – Implementar sistema de vida de nuestros personajes.
- H3.5 – Entregable en el que veamos el HUD, y en el que se demuestre la implementación de la vida y de los ataques.
- H3.6 – Implementar el cambio de personajes durante el juego.
- H3.7 – Implementar el Menú principal y Menú Ingame.
- H3.8 – Crear una serie de objetivos para seguir la historia.
- H3.9 – Implementar el sistema de cargar/guardar partida.
- H3.10 – Implementar sonido (opcional).
- H3.11 – Añadir efectos visuales (opcional).
- H3.12 – Añadir más efectos físicos (opcional).
- H3.13 – Añadir más fases de juego y jefes finales (opcional).
- H3.14 – Manual del juego orientado al usuario.
- H3.15 – Entregable final en el que podamos ver el juego al completo.

2.2.4.Presupuesto

Por horas:

Descripción	**Horas**	**Puntos**
Hito1		
H1.1 - Estructura Mapa y Leer desde XML	100	20
H1.2 - Representación 2D	50	10
H1.3 - Máquina de Estados	100	20
H1.4 - Pathfinding	125	25
H1.5 - Resto Motor IA	125	25
H1.6 - Motor Físico (Detección Colisiones)	75	15
H1.7 - Motor Entrada/Salida (Comienzo)	50	10
H1.8 - Entregable		-
Total:	625	125
Hito2		
H2.1 - Cargador de Modelos	75	15
H2.2 - Entregable		
H2.3 - Cargar Mapas	75	15
H2.4 – Movimiento Libre de la Cámara	50	10
H2.5 – Editor de Mapas	75	15
H2.6 – Motor del Juego	75	15
H2.7 – Cargar Nuestro Personaje	75	15
H2.8 – Ataques de los Enemigos	75	15
H2.9 – Finalizar Entrada/Salida	75	15
H2.10 - Entregable		
Total:	575	115
Hito3		
H3.1 - Motor Optimización	200	40
H3.2 - Interfaz Motor Optimización	100	20
H3.3 - HUD	75	15
H3.4 – Sistema de Vida	50	10
H3.5 - Entregable		
H3.6 – Cambio de personajes	50	10
H3.7 – Menú Principal y Menú Ingame	75	15
H3.8 – Objetivos	50	10
H3.9 - Cargar/guardar	75	15
H3.10 - Sonido	30	5
H3.11 – Añadir efectos visuales	30	5

H3.12 – Añadir más efectos de Física	30	5
H3.13 – Añadir jefes finales y más fases	30	5
H3.14 - Manual Usuario	20	5
H3.15 – Entregable final		
Total:	815	160
Total Proyecto:	2015	400

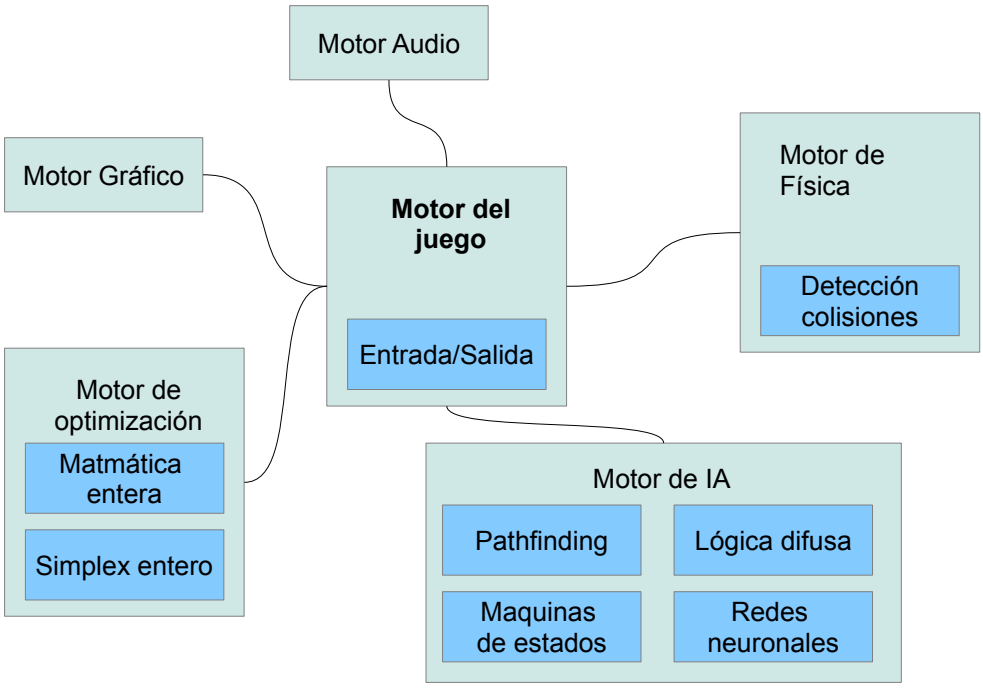
Por asignatura:

Descripción	RAZ	JRV	GAA	MFAC	TOTAL
Hito1					
H1.1 - Estructura Mapa y Leer desde XML			20		20
H1.2 - Representación 2D			10		10
H1.3 - Máquina de Estados	20				20
H1.4 - Pathfinding	25				25
H1.5 - Resto Motor IA	25				25
H1.6 - Motor Físico (Detección Colisiones)		15			15
H1.7 - Motor Entrada/Salida (Comienzo)		10			10
H1.8 - Entregable					
Total:	70	25	30	0	125
Hito2					
H2.1 - Cargador de Modelos			15		15
H2.2 - Entregable					
H2.3 - Cargar Mapas			15		15
H2.4 – Movimiento Libre de la Cámara			10		10
H2.5 – Editor de Mapas			15		15
H2.6 – Motor del Juego		15			15
H2.7 – Cargar Nuestro Personaje		8	7		15
H2.8 – Ataques de los Enemigos	10		5		15
H2.9 – Finalizar Entrada/Salida		15			15
H2.10 - Entregable					
Total:	10	38	67	0	115
Hito3					
H3.1 - Motor Optimización				40	40
H3.2 - Interfaz Motor Optimización				20	20
H3.3 - HUD			15		15

H3.4 – Sistema de Vida		10			10
H3.5 - Entregable					
H3.6 – Cambio de personajes		10			10
H3.7 – Menú Principal y Menú Ingame		10	5		15
H3.8 – Objetivos		10			10
H3.9 - Cargar/guardar		15			15
H3.10 - Sonido		5			5
H3.11 – Añadir efectos visuales			5		5
H3.12 – Añadir más efectos de Física		5			5
H3.13 – Añadir jefes finales y más fases		2	3		5
H3.14 - Manual Usuario		5			5
H3.15 – Entregable final					
Total:	0	72	28	60	160
Total Proyecto:	80	135	125	60	400

3. Diseño

Para el diseño, nos basaremos en el siguiente esquema:



- Motor del juego:
  - Es el "núcleo" de nuestro diseño.
  - Controla el funcionamiento general del juego, así como la maquina de estados que representará el flujo de funcionamiento (Menú principal, Pantalla de Juego, Menú ingame, Muerte).
  - Cargará el "estado" del juego desde un fichero XML.
  - Gestiona todas las entidades que encontremos en el juego (entendemos entidad como cualquier elemento del escenario, sea un árbol o un enemigo).
  - Integra al resto de motores y gestiona su comunicación. Para ello, usa una interfaz intermedia específica para cada módulo (así aumentamos la independencia de cada uno).
  - Se encarga también de la lógica del juego (si el módulo de E/S lee un click, el motor del juego se encarga de "disparar"), incluyendo la "historia".
  - Tiene un módulo de entrada/salida que:
    - Se encarga de gestionar la interacción con el usuario.
    - Implementa el módulo encargado de leer del teclado. En principio solo leeremos las teclas (indicamos las que usaremos por defecto):
      - Movimiento de nuestro personaje ('W', 'A', 'S', 'D').
      - Tecla para apuntar ('Shift').
      - Menú ingame ('Esc').
      - Tecla para el cambio de personajes ('C').
    - Implementa también al módulo encargado de interpretar la entrada del ratón. Con el ratón controlaremos:
      - La cámara. Dado que nuestra perspectiva es desde detrás, es equivalente a la visión (y dirección) de nuestro personaje.
      - La mirilla si la tecla de apuntar está pulsada.
      - La selección de los elementos en el Menú principal y en el ingame.

- Motor Físico:
  - Integraremos un motor de física 2D ya existente (Box2D lo más probable).
  - En principio solo usaremos la detección de colisiones, aunque añadiremos más física si tenemos tiempo.
- Motor Gráfico:
  - Se trata de un motor 3D.
  - Se encargará de gestionar:
    - La iluminación del escenario.
    - La cámara.
    - La carga y dibujo de los modelos 3D y el "fondo".
  - Si tenemos tiempo, implementaremos más efectos visuales en el motor gráfico, como por ejemplo brillos o reflejos.
- Motor IA:
  - Se encarga de representar la Inteligencia Artificial de los enemigos.
  - Implementa un módulo de Pathfinding, encargado del movimiento de los enemigos. Para ello, usará el algoritmo A* (discretizando el mapa).
  - Implementa un módulo de Maquinas de estados, encargada del comportamiento de los enemigos. Para ello, usaremos los siguientes estados:
    - Descanso ('Stand'). No nos busca ni se mueve. De hecho, podríamos acercarnos por la espalda sin que se moviera. Es similar a estar "dormido".
    - Patrulla. Se mueve "patrullando", es decir, buscándonos. En este estado, si nos ve, correrá hacia nosotros para atacarnos. No obstante, aun no está en un estado agresivo.
    - Perseguir. Se moverá hacia nosotros. Si es un enemigo con ataque físico, nos perseguirá hasta estar a nuestro lado. Sin embargo, si es un enemigo con ataque a distancia, nos perseguirá hasta la distancia mínima para atacarnos. En este estado, también llamará al resto de enemigos en un cierto radio (que pasarán también al estado perseguir).
    - Atacar. Ejecutará su ataque contra nosotros hasta que estemos a más distancia de la mínima o alguno de los dos esté muerto.
    - Huir. Si se debilita mucho, el enemigo (salvo que sea de la unidad especial O.S.O.), huirá para pedir ayuda a otros enemigos.
  - Implementar los módulos de lógica difusa y redes neuronales. No obstante, no usaremos estos módulos en nuestro juego.
- Motor Audio:
  - Si tenemos tiempo, implementaremos un motor que se encarge de reproducir audio.
- Motor optimización.
  - Implementa un módulo encargado de resolver problemas de programación lineal entera.
  - En principio, nuestro juego no lo usará.

4. Bibliografía

- http://box2d.org/
- http://www.sfml-dev.org/
- Programming Game AI by Example, Mat Buckland
- http://www.gamedev.net

Proyecto Firmado y Aceptado por

Carrillo Boj, Carlos
Fernández Ferrández, Esteve
González Martín, Adrián
Pastor Esteve, Gustavo