Diseño de nivel en videojuegos cargados en web

Leticia López Ortega

09 de Junio de 2015

Contents

List of Figures

List of Tables

1 Introducción

1.1 Introducción

En el presente proyecto se demuestra la efectividad de máquina de estados orientadas a videojuegos, mostrando la amplia funcionalidad de esta técnica y sus mejoras en cuanto a refactorización de código. El juego tiene un modelo de comportamiento en el que pasa por diferentes estados en los que se determinara funcionalidades propias de cada uno.

1.2 ¿En qu consiste el juego?

Es un videojuego 2D de naves tipo arcade, en el que se proporciona al jugador la opción de elegir sus habilidades principales, contando inicialmente con solo cinco puntos de habilidad, tras elegir el jugador se encuentra en un mundo abierto en el que puede viajar con su nave espacial a diferentes planetas, en dichos planetas habrá un enemigo al cual habrá que derrotar, una vez derrotado se recompensa al jugador con un punto de habilidad que podrá gastar en la habilidad que desee, y seguir así invadiendo planetas, hasta acabar con todos los enemigos, aparte de la recompensa de un punto de habilidad por haber derrotado a un enemigo cada 5 enemigos derrotados la nave del jugador se actualizara, haciendo notar al jugador un avance en el juego y una recompensa por sus habilidades.

2 Estado del arte

2.1 HTML5

Es la versión 5 de HTML (HyperText Markup Language). Es un lenguaje basado en etiquetas, que permite la elaboración de páginas web, definiendo su contenido (imágenes, texto, videos...) en la versión 5 se incluyen nuevas mejoras.

2.1.1 Canvas

Es un elemento de HTML5 que permite generar gráficos estáticos o dinámicos, accediendo a él mediante código en JavaScript.

2.2 CSS3

Es la versión 3 de CSS (Cascading Style Sheets) es un lenguaje que sirve para crear el diseño de un código HTML, sosteniendo la idea de separar el diseño de una página web de su estructura.

2.3 JavaScript

Es un lenguaje de programación interpretado (quiere decir que no necesita compilación) Es un lenguaje orientado a objetos, que se basa en prototipos, es débilmente tipado y dinámico. Normalmente se usa para programar del lado cliente, pero también puede usarse para programar del lado servidor, gracias a Node.js. Comúnmente se usa para añadir funcionalidad a las páginas web, dicho código se escribe en un archivo independiente y se enlaza con la página web definiendo en el código HTML las etiquetas ¡script¿¡/script¿ y entre medias la ruta del fichero.

2.4 lluviaProject

Lluvia es un DSL (Domain Specific Language) que se refiere a una especificación de un lenguaje de programación, en este caso JavaScript, proveyendo al lenguaje de más funcionalidades de las que posee el lenguaje nativo. Ha sido utilizado para desarrollar la mayor parte del software de este proyecto.

2.4.1 Device

Los Device son objetos independientes capaces de comunicarse entre ellos mediante eventos.

2.4.2 States

Los States son automatas que proporcionan determinadas funcionalidades en cada instante

2.4.3 Gates

Son puertas de entrada/salida por las que pasan los eventos con los cuales se comunican los Device.

2.4.4 Boids

Son una inteligencia artificial, en la que cada boid es programado con un comportamiento independiente del resto de boids, creando de esta forma una inteligencia artificial de un conjunto de comportamientos definidos.

3 Arquitectura de la aplicación

3.1 Clases

Una clase es un modelo o plantilla diseñado para crear objetos, las clases poseen atributos y métodos y los objetos los heredan. Cuando un objeto se crea a partir de una clase se llama instanciación de la clase.

3.1.1 PointDealer

Es una clase que hereda de la clase Device, cuya funcionalidad es repartir puntos de habilidad al jugador, con un máximo determinado. Las habilidades en las que puede repartir los puntos son "damage", "resistance" y "speed". Esta clase tiene una interfaz con la que se comunica en todo momento el jugador. Esta provista de 2 botones (button_plus y button_minus) con los que el jugador puede aumentar o diminuir los puntos de cada habilidad, esto se comunica al Device con el uso de Gates. Posee otro botón (button_play) el cual tiene la funcionalidad de enviar al jugador al siguiente Device, únicamente si el jugador ha gastado todos los puntos de habilidad.

3.1.2 Space

Es una clase que hereda de la clase Device, en Space se crean veinte Gates siendo cada uno de ellos un planeta diferente al que el jugador podrá viajar, pero no todos ellos se muestran en el canvas, Space posee cuatro botones cada uno en un margen del canvas, sirviendo al jugador la funcionalidad de desplazarse por el universo en busca de diferentes planetas, siendo elección libre para el jugador cual escoger.

3.1.3 Planet

Es una clase que hereda de la clase Device, en el que se desarrolla la pelea contra el jugador, aquí se crean dos boids (Player y Enemy). En Planet tambien se define el comportamiento de ataque y movimiento de los bois, ya que Planet es el creador de los boids y el mundo en el que existen.

3.1.4 Player

Es una clase que hereda de la clase Boid. Este boid esta totalmente controlado por el jugador mediante el teclado, no tiene ningun comportamiento programado, pero si algunos de los atributos, como la vida, la posicion y el sprite. Los atributos no programados que podra escoger el jugador son, damage, resistance y speed, se eligen al inicio del juego en el Device pointDealer.

3.1.5 Enemy

Es una clase que hereda de la clase Boid, creando un array de Boids que tienen comportamientos definidos, los cuales son, moverse y atacar. Cada enemigo es diferente dependiendo del planeta al que viaje el jugador, los atributos que cambian son, la vida, el sprite, damage, resistance y speed. El unico atributo no variable que posee Enemy es la posición.

4 Desarrollo de la aplicación

4.1 Entorno utilizado

Un entorno de desarrollo puede ser una o varia aplicaciones, que se compone de un editor de texto, un depurador y un compilador, en este caso como el lenguaje utilizado ha sido JavaScript no ha sido necesaria la utilización de un compilador.

4.1.1 SublimeText 3

Es un editor de texto o editor de código fuente y ha sido utilizado para crear todo el código fuente de la aplicación.

4.1.2 Mozilla Firefox

Es un navegador web libre y de código abierto. Su consola JavaScript ha sido utilizada para comprobar los errores en el código fuente y su herramienta de depuración incorporada ha sido de gran ayuda a la hora de realizar las tareas de testing y debug.

4.1.3 PhotoShop CS6

Es un editor de gráficos, con el cual se ha desarrollado gran parte de los diseños del videojuego.

4.1.4 GitHub

Es un control de versiones on-line, basado en Git, desde el cual se descarga lluviaProject y ha sido muy útil para ordenar el código y poder portarlo de un ordenador a otro.

5 Diseño

5.1 Diseño de interfaz

La interface está diseñada con HTML y CSS siendo esta una interfaz sencilla e intuitiva para el jugador. La primera pantalla se compone de 3 y barras y 3 botones, de los cuales la funcionalidad está hecha con JavaScript. La siguiente pantalla se denominaría como el entorno abierto y de libre elección del videojuego, que consiste en una simulación del espacio exterior con diversos planetas a los que el jugador puede viajar.

5.2 Diseño de mundo

Es un diseño de mundo abierto, lo cual quiere decir que el juego ofrece al jugador la posibilidad de moverse libremente por este y de elegir a donde ir.

5.3 Diseño de Enemy y Player

Estos diseños están basados en naves espaciales, pintadas en 2D.

5.4 Diseño de nivel

Es el diseño de escenario o misiones del videojuego, lo cual se basa no solo en el aspecto grafico sino también en la dificultad. Se dice que un juego tiene un buen diseño de nivel cuando el jugador nota que el juego avanza y la dificultad cada vez es más elevada. En el presente proyecto el aspecto gráfico no se ha desarrollado mucho, queriendo darle un aspecto simple y minimalista, sin embargo el desarrollo del diseño de nivel en cuanto al aumento de dificultad en cada etapa del juego ha sido programado meticulosamente.

5.4.1 Objetivo y recompensa

El objetivo del juego es destruir a todos los alienígenas invasores de los planetas. La recompensa es proporcionada al jugador cada vez que este consigue derrotar a un enemigo, dandole un punto de habilidad mas para aumentar sus estadisticas y haciendole posible la victoria en la siguiente batalla.

5.4.2 Dificultad

El orden de dificultad es el orden numérico de los planetas, siendo el planeta 1 el que posee al enemigo más débil y el planeta 20 el que posee el enemigo más poderoso. Al ser un mundo abierto y el jugador no tener consciencia sobre el orden de los planetas, hace que la dificultad del juego aumente y posiblemente también la curiosidad de saber que pasara en el siguiente planeta al que viaje.

6 Conclusiones

El objetivo de este proyecto ha sido crear una pequeña muestra de lo que se puede hacer usando las máquinas de estados para desarrollar videojuegos.

Las conclusiones obtenidas han sido:

- La eficacia de las máquinas de estados para videojuegos, es muy potente y recomendable, aun siendo más dificultosa que un método basado en condiciones, se consigue optimizar el código y el tiempo dedicado.
- La clase pointDealer puede usarse para muchas más aplicaciones orientadas a videojuegos, ya que es un repartidor, podría distribuir no solo puntos, sino comida, oro, etc.
- El uso de la clase boids en la programación de la inteligencia artificial de los enemigos en videojuegos evita la duplicidad de código.
- El desarrollo de autómatas requiere de una programación delicada y el uso de constantes pruebas para comprobar errores.

Optimizaciones del proyecto:

- Implementar un nuevo modo de juego, en el que puedan jugar dos jugadores, en un 1 vs 1, haciendo uso de Node.js.
- Mejorar el diseño gráfico haciéndolo más apetecible para el público.
- Implementar sonidos y músicas dotando al videojuego de realismo.
- Crear un apartado de personalización, para que el jugador pueda customizar su avatar como desee.

Posibles aplicaciones. Podrían usarse algunas de las clases para la creación de nuevos videojuegos, por ejemplo Player, Enemy o PointDealer.

7 Bibliografía

La información obtenida para el desarrollo de este proyecto ha sido:

- lluviaProject Wiki
- "Proyecto de Simulación Comportamental en Entornos con Tasas de Estrés Variable". José Moreno Casero e Iván Bravo del Valle.
- Proyecto

A Apendice A

A.1 Métodos de PointDealer

- get_back_from: Permite disminuir los puntos de habilidad, de la habilidad en la cual se esté pulsando el "button_minus".
- transfer_to: Realiza la tarea opuesta a get_back_from, pues si este quita puntos, transfer_to los suma hasta llegar al máximo de puntos permitido.
- render: Cambia las imágenes de la barra de puntos de cada habilidad dependiendo del botón que pulse el jugador. Dando a conocer al usuario los puntos que esta agregando o quitando en cada momentos y en cada habilidad.
- attend_show_skills: Permite que Point Dealer se muestre al iniciar el juego.

A.2 Métodos de Space

- attend_show_space: Permite que Space se muestre cuando el jugador a pulsado el botónbutton_play de Point Dealer.
- up: Proporciona la funcionalidad del botón situado encima del canvas para mover el mapa de los planetas hacia arriba.
- right: Proporciona la funcionalidad del botón situado a la derecha del canvas para mover el mapa de los planetas hacia la derecha.
- down: Proporciona la funcionalidad del botón situado debajo del canvas para mover el mapa de los planetas hacia abajo.
- left: Proporciona la funcionalidad del botón situado a la izquierda del canvas para mover el mapa de los planetas hacia la izquierda.

A.3 Métodos de Planet

- initialize: Inicializa a Player y a Enemy para que se creen en Planet.
- attend_show_planet: Hace que el planeta se muestre, con una alerta que indica el número del planeta que es.
- has_born: Permite que el boid se cree.
- draw: Permite dibujar el boid.
- new_boid: Crea una nuevo boid.
- new_boid_of: Crea un boid a partir de una clase, previamente definida.
- get_boids:
- repaint(ctx): Pinta el canvas de negro.
- resize: Permite que el jugador pueda poner el juego en apantalla completa usando el botón "button_screen"

A.4 Métodos de Player

- random(max): Escoge un numero aleatorio para empezar a pinta esrellas en el canvas de Planet.
- move: Hace que se mueva Player, los disparos que lanza y las estrellas del canvas.
- paint(ctx): Pinta a Player, a los disparos, a las estrellas y los FPS.
- draw: Ejecuta el método repaint(ctx), declarado en Planet, después el método move y después el método paint.

A.5 Métodos de Enemy

- move_enemy
- draw

B Apendice B

B.1 Diagramas de clases

B.1.1 Diagrama de clases de Devices

B.1.2 Diagrama de clases de Boids

C Apendice C

C.1 Interfaz de los Devices

C.2 Boids

C.2.1 Sprites de Player

C.2.2 Sprites de Enemy