Universidad de Huelva

ESCUELTA TÉCNICA SUPERIOR DE INGENIERÍA

DE LA UNIVERSIDAD DE HUELVA

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Generación de Mazmorras. Análisis e investigación del problema. Acercamiento apoyado en búsqueda.

Autor:

Alejandro Seguí Díaz

Tutores:

Gonzalo A. Aranda Corral

Daniel Márquez Quintanilla

Huelva, 30 de junio de 2015.

Curso académico 2014/15.

Índice general

Índice de figuras

Introducción

En la actualidad, los videojuegos han conseguido posicionarse en un mercado líder indiscutible a nivel internacional. En el ámbito del entretenimiento, la industria cinematográfica está dando paso a los videojuegos, que avanzan a pasos de gigante. Muchas grandes personalidades del cine se dirigen a los creadores de videojuegos para continuar su carrera. Un ejemplo claro de una persona adelantada para su época en este sentido, es George Lucas ([1]), autor de la famosa saga Star Wars. George Lucas se introdujo en el mercado de los videojuegos en el 1986 con el título Labyrinth, dando paso posteriormente a otros títulos muy sonados y de culto como Monkey Island o Grimm Fandango.

Un videojuego, al igual que una película, puede contar una historia. Aún así, existen diferentes géneros de videojuegos, que van desde simulación de juegos de mesa, donde evidentemente no existe historia, o está intrínseca en el origen del mismo juego, hasta las denominadas aventuras gráficas, donde el enfoque está en la parte argumental. Es por ello que la industria del cine deja paso a los videojuegos de manera tan rápida.

Mapas y escenarios.

Muchos géneros de videojuegos plantean su argumento en un escenario donde se da rienda suelta a la capacidad perceptiva del jugador, dando lugar en mayor o menor medida a que complete la historia con su imaginación. En este escenario, el jugador desarrollará las acciones que se le ofrezcan según el tipo de juego. Así, podrá completar la historia, permitiendo además en muchos casos que las acciones influyan en el desarrollo argumental del videojuego. Tipos de juegos que requieren mapas son juegos de acción, estrategia o rol entre otros.

El tipo de escenario que podemos encontrarnos en un juego es de lo más variopinto. A la hora de plantear la elaboración de un juego, se tiene en cuenta la elección del tipo de escenario. Muchas cuestiones sobre las que se basan estas elecciones radican sobre la complejidad de optimización del mismo:

- Si el escenario es demasiado grande, no nos interesa renderizar el escenario completo, sino solamente la parte visible.
- Lo mismo pasa con las físicas. No nos interesa comprobar colisiones con elementos que es obvio que no van a colisionar.
- El *nivel de detalle* (Level of detail por el inglés [2]) es otro aspecto a tener en cuenta. Si una geometría está demasiado lejana del jugador, no necesitamos renderizarla exactamente como es. Este aspecto es relevante a los juegos 3D principalmente.
- La división o no del escenario por zonas. Esto puede evitarnos la obligatoria presencia en memoria de un mapa completo, pudiéndolo dividir en partes para ahorrar en el preciado recurso que es la memoria. Es un aspecto a tener en cuenta sobre todo cuando se elaboran juegos para videoconsolas, las cuales suelen contar con una memoria muy limitada.

Así, se han elegido dos criterios principales como forma de clasificar los tipos de escenario:

- Organización del escenario. Este criterio se refiere a la existencia de cortes en el desarrollo del juego para la carga de las distintas partes del mundo en el que se desarrolla el juego. Algunos tipos son:
  - Escenarios totalmente continuos continuos representando un mundo totalmente abierto. Ejemplos son GTAV o Minecraft.
  - Dividido por zonas donde no existen cortes en la misma zona. Un ejemplo es Borderlands.
  - Dividido por zonas donde pueden existir cortes en la misma para interiores. Un ejemplo es Skyrim.
- Composición del escenario. Nos referimos a si la geometría empleada ayuda de forma implícita al manejo a nivel computacional del escenario. Algunos tipos son:

- Escenarios en 2D discretizados en *tiles*. Básicamente, la geometría está representada por una matriz de dos dimensiones, donde cada casilla de la matriz está relacionada con una textura, siendo todas las texturas del mismo tamaño para la misma matriz. Este tipo de escenarios son muy fácilmente optimizables. Ejemplos de este tipo es, por ejemplo, Final Fantasy VI.
- Una extensión a 3D de los escenarios de tiles son los escenarios compuestos por *voxels* [3]. Estos han sido recientemente popularizados por el famoso título Minecraft, donde el escenario está gestionado por un *motor de voxels*.
- Escenarios con geometría deliberada. Es decir, la geometría (ya sea 2D o 3D), no presenta ninguna propensión a ser optimizada. Este tipo de escenarios, suelen optimizarse con el uso de *quadtrees* [4] (para 2D) u *octrees* [5] (para 3D)
- Escenarios hechos con los denominados *brushes* [6]. Esta técnica se utiliza para 3D, y consiste en el uso de geometrías convexas para componer un escenario. Ejemplos famosos de este tipo de escenarios es *Quake* [7].

Una vez expuesta los criterios para clasificar los tipos de mapas, remarcar que hay casos en los que los desarrolladores convienen formas nuevas de organizar el mapa debido a necesidades particulares, por lo que los ejemplos expuestos en cada criterio no completan de ninguna manera las múltiples categorías para cada uno.

Para el desarrollo de este proyecto y atendiendo a las especificaciones presentadas en el capítulo 2, abarcaremos *mapas de tiles* para el género de juego *roguelike* [8]. Este género se caracteriza precisamente por mapas generados de forma procedimental. Cada nivel es entendido como una planta, donde el jugador tiene que llegar desde donde aparece hasta un tile considerado como final para poder continuar hasta la siguiente planta. Así, el jugador va avanzando niveles hasta que llega al último y completa el juego al finalizar el mismo.

Motivación y Objetivos.

Como se ha mencionado previamente, la industria de los videojuegos avanza a pasos de gigante. Es por ello que las compañías invierten cada

vez más en videojuegos, siendo a veces el coste en marketing superior al de desarrollo [9]. Existe un amplio espectro de puestos encontrados en este campo [10] [11], entre los cuales existen

- *Game programmer*. Personal dedicado a la programación de las mecánicas del juego, eventos y otros relacionados con el propio juego
- *Core programmer*. Personal dedicado a la elaboración del motor que será la base sobre la que funcionará el juego. Normalmente se subdivide en subroles como *graphics programmer* o *physics programmer*. En muchos casos, se parte de un motor ya elaborado sobre el que se realizan las modificaciones necesarias para el juego en cuestión.
- *Game designer*. Dedicados a la elaboración de las mecánicas del juego, así como el plot del mismo.
- *Level designer*. Personal dedicado exclusivamente a la elaboración de escenarios empleando programas externos o el mismo motor que se esté utilizando, si éste posee dichas capacidades.
- *AI programmer*. Dedicado a la programación de los distintos sistemas de inteligencia artificial existentes en el juego.
- *Muchos más...*

En este listado, solo hemos tenido en cuenta algunos de los puestos del aspecto de desarrollo, pero quedaría añadir muchos más como diseñadores gráficos y apartado de marketing.

Destacar que el *level designer*, pese a que no se encarga de la construcción de las herramientas para crear el escenario, debe conocer las bases del tipo de escenario que se empleará en el juego, de forma que adapte sus técnicas de diseño al tipo de escenario.

Recientemente, y debido a las cada vez más crecientes facilidades para crear un videojuego, han surgido los llamados *equipos indies* [**?**], caracterizados principalmente por tener un bajo presupuesto y personal. Normalmente suelen empezar con un presupuesto nulo, pero en algunos casos llegan a triunfar de manera inesperada incluso por los mismos desarrolladores. Ejemplos son *Hotline Miami*, *Minecraft* o *Risk of Rain*.

Los equipos con un presupuesto considerable (coloquialmente denominados AAA), no tienen problemas a la hora de contratar *diseñadores de*

niveles, ya que disponen de una gran cantidad de dinero para depositar en los distintos roles necesarios para un juego. Aún así, en combinación con un generador de mapas, un *diseñador de niveles* puede desarrollar ideas que den un resultado muy bueno. Ejemplos de esto son *Diablo* o *Torchlight*, donde los mapas son generados automáticamente partiendo de patrones elaborados a mano por *diseñadores de niveles*.

Así, el la motivación de este proyecto reside en dos ideas principales.

- Ahorro de coste y tiempo para equipos indies, que no pueden permitirse el lujo de gastar en personal exclusivo para el diseño de niveles.
- Adición de variedad a los escenarios de juegos, permitiendo incluso a un equipo de desarrolladores AAA

Solución Propuesta.

La solución que se propone es un sistema capaz de generar escenarios de manera automática con una mínima interacción (o incluso nula si se desea) de los diseñadores. Esto además, conlleva dinamismo en los escenarios que se podrán jugar, de forma que de partida a partida, la distribución del escenario será completamente distinta.

La elaboración de un generador de escenarios, une las competencias de dos roles en el desarrollo de videojuegos:

- *Diseñador de niveles.* Se necesitan conocimientos sobre la composición de los escenarios del juego, además de las propiedades concretas en cuanto a los criterios mencionados anteriormente.
- *Programador de IA*. La creación del sistema que genere los escenarios es un trabajo que compete a este rol, ya que estamos tratando de resolver un problema donde las posibilidades de resolución son casi infinitas.

Finalmente, se elaborará el sistema de forma que se emplea los conceptos de una búsqueda para el mismo.

Estructura de la memoria.

Esta memoria se estructura en varios capítulos, con la siguiente distribución de los temas trabajados:

- Capítulo 1. Nunc viverra volutpat bibendum. Nunc augue orci, tempus nec interdum sit amet, ornare id elit. Integer congue risus vitae ipsum pharetra rutrum. Curabitur mollis sagittis pretium. Etiam a lacus sed mauris rhoncus ullamcorper non sit amet felis.
- Capítulo 2. Nunc viverra volutpat bibendum. Nunc augue orci, tempus nec interdum sit amet, ornare id elit. Integer congue risus vitae ipsum pharetra rutrum. Curabitur mollis sagittis pretium. Etiam a lacus sed mauris rhoncus ullamcorper non sit amet felis.
- Capítulo 3. Nunc viverra volutpat bibendum. Nunc augue orci, tempus nec interdum sit amet, ornare id elit. Integer congue risus vitae ipsum pharetra rutrum. Curabitur mollis sagittis pretium. Etiam a lacus sed mauris rhoncus ullamcorper non sit amet felis.
- Capítulo 4. Nunc viverra volutpat bibendum. Nunc augue orci, tempus nec interdum sit amet, ornare id elit. Integer congue risus vitae ipsum pharetra rutrum. Curabitur mollis sagittis pretium. Etiam a lacus sed mauris rhoncus ullamcorper non sit amet felis.
- Capítulo 5. Nunc viverra volutpat bibendum. Nunc augue orci, tempus nec interdum sit amet, ornare id elit. Integer congue risus vitae ipsum pharetra rutrum. Curabitur mollis sagittis pretium. Etiam a lacus sed mauris rhoncus ullamcorper non sit amet felis.
- Capítulo 6. Nunc viverra volutpat bibendum. Nunc augue orci, tempus nec interdum sit amet, ornare id elit. Integer congue risus vitae ipsum pharetra rutrum. Curabitur mollis sagittis pretium. Etiam a lacus sed mauris rhoncus ullamcorper non sit amet felis.
- Capítulo 7. Nunc viverra volutpat bibendum. Nunc augue orci, tempus nec interdum sit amet, ornare id elit. Integer congue risus vitae ipsum pharetra rutrum. Curabitur mollis sagittis pretium. Etiam a lacus sed mauris rhoncus ullamcorper non sit amet felis.
- Capítulo **??**. Nunc viverra volutpat bibendum. Nunc augue orci, tempus nec interdum sit amet, ornare id elit. Integer congue risus vitae

ipsum pharetra rutrum. Curabitur mollis sagittis pretium. Etiam a lacus sed mauris rhoncus ullamcorper non sit amet felis.

- Bibliografía.

Capítulo 1

Estado del arte.

En este capítulo, analizaremos el ámbito de la generación procedimental de contenido para videojuegos. Veremos las distintas subdisciplinas presentes, así como los problemas que resuelven. Se hará hincapié en la generación de escenarios, ya que es el campo que compete al proyecto.

1.1. Generación de contenido procedimental.

Más conocida por su nombre en inglés (Procedural Content Generation), se refiere a la disciplina de generar contenido partiendo de algoritmos, en lugar de hacerlo manualmente.

Tuvo sus inicios en la subcultura informática llamada *Demoscene* [12]. Este movimiento tuvo sus inicios a finales de los años 70 y principios de los 80, y continúa a día de hoy. Consiste en crear contenido visual y sonoro de forma programada, ya sea en parte o en su totalidad, y tenía como uno de sus objetivos escudriñar al máximo las limitadas capacidades de los ordenadores de la época.

Los *sceners* de este campo, conseguían generar contenidos que eran impensables para la época, como por ejemplo escenas en 3D cuando a OpenGL aún le quedaban un par de décadas para aparecer. Por su habilidad, la mayoria de los demosceners terminaban trabajando para empresas de videojuegos de la época, cuando aún no había tantas facilidades a la hora de crearlos.

Existen diversas subdisciplinas en las que se aplica el concepto de generación procedimental de contenido:

- *Escenarios*.
- Texturas [13]. Un ejemplo de ello es la generación de texturas que imitan el mármol o la madera. En algunos casos se emplean autómatas celulares.
- Geometría. Generación de follaje, o árboles empleando L-Systems. [13]
- Mecánicas. El juego Left4Dead y su secuela, emplean un sistema procedimental para gestionar los momentos de tensión y la dificultad de las situaciones según las acciones que han ido tomando los jugadores.

Otra clasificación de los métodos procedimentales es según el momento de ejecución del procedimiento. Si el procedimiento es ejecutado antes del lanzamiento del juego, o después con la particularidad de que se realiza en servidores ajenos al jugador, lo denominaremos *offline*. Sin embargo, si el procesamiento se realiza en el sistema en que se está ejecutando el juego, lo llamaremos *online*.

1.2. Generación de mapas.

La generación de escenarios se considera un concepto muy amplio, ya que según el contexto del juego, puede variar bastante. Por ejemplo, en un juego ambientado en el espacio exterior, el escenario puede entenderse como una galaxia completa, como es el caso de *Elite: Dangerous*.

Otro tipo de modelo a usar en la generación de escenarios es el ruido Perlin. Éste puede ser muy útil en conjunto con el concepto de mapas de alturas para generar terrenos al aire libre montañosos [14]. También se ha empleado el *algoritmo de Voronoi* para la generación de terrenos [15].

Algún ejemplo más histórico de generación de escenarios pueden ser los múltiples algoritmos de generación de laberintos que se conocen [16].

Debido a que los métodos anteriores son demasiado genéricos, una opción muy popular es idear y construir un generador específico para el juego en cuestión que se esté desarrollando. Un ejemplo de ello es *TinyKeep*, cuyo autor describe a grandes rasgos en [17] las fases y el desarrollo del algoritmo que ha creado. Esta opción es la que se desarrollará en este proyecto, procurando mantener un nivel de genericidad y capacidad de personalización del sistema.

Capítulo 2

Especificaciones.

En este capítulo definiremos las especificaciones y directrices tanto del tipo de juego como del sistema de generación. Dichas directrices que delimitarán el sistema a crear, han sido establecidas por la empresa indie de videojuegos *TheGameKitchen*, y el sistema será empleado en un futuro título de la misma.

2.1. Tipo de juego.

El género de juego al que nos enfocaremos, será del tipo *roguelike* [8], cuyas características principales son:

- *Generación de mazmorras*. Cada vez que el jugador inicia una partida, la experiencia será ligeramente distinta.
- *Importancia considerable a la exploración*. El hecho de que las mazmorras no sean siempre iguales, incita al jugador a tener que invertir tiempo en explorar para poder encontrar la salida.
- *Desarrollo del juego por plantas*. El objetivo del jugador suele ser llegar a una habitación considerada como final, donde puede elegir pasar a una siguiente planta, o investigar un poco más en la presente.
- *Dificultad progresiva*. Cada planta, tendrá una dificultad ligeramente mayor a la de la anterior hasta llegar a la última.

- *Muerte permanente*. Una vez que el jugador muere, no hay manera de cargar la partida. La unica opción es comenzar de nuevo.

Históricamente, el género *roguelike* se identificaba además por otro tipo de características, como la mecánica por turnos o el énfasis en jugabilidad y desinterés en los gráficos, pero con el tiempo, el género se ha ido abriendo paso a una definición más genérica. Debido a ésto, hoy en día existen desde *shooters* considerados *roguelike*, como por ejemplo *Tower of Guns* o *Paranautical Activity*, hasta *plataformas*, como *Risk of Rain* o *Spelunky*.

En concreto, el juego estará representado en un mapa de tiles.

2.2. Directrices para la generación.

Capítulo 3

Representación.

3.1. Topología.

3.2. Habitaciones.

3.2.1. Puertas potenciales.

3.2.2. Prefabs.

3.2.3. Instancias.

3.2.4. Mapa.

Capítulo 4

Movimientos.

4.1. Cómputo de posibles movimientos.

4.2. Conexiones entre mapa y habitaciones.

Capítulo 5

Sistema de generación.

5.1. Interfaz de construcción.

5.2. Guardado de movimientos

5.3. Interfaz aleatoria.

5.4. Interfaz basada en búsqueda.

Capítulo 6

Experimentación.

6.1. Seccion31.

6.2. Seccion32.

Capítulo 7

Trabajo futuro.

7.1. Seccion31.

7.2. Seccion32.

Bibliografía

[1] LucasArts Wikipedia page

[2] Techopedia LOD page

[3] Voxel wikipedia

[4] Quadtree wikipedia

[5] Octree wikipedia

[6] Brush wikipedia

[7] Quake wikipedia

[8] RogueLike wikipedia

[9] List of most expensive video games to develop.

[10] Gamedev Job Roles at CreativeSkillSet

[11] Video Game Development Wikipedia page

[12] DemoScene wikipedia

[13] http://www.amazon.com/Texturing-Modeling-Third-Edition-Procedural/dp/1558608486

[14] http://libnoise.sourceforge.net/tutorials/tutorial3.html

[15] http://www-cs-students.stanford.edu/ amitp/game-programming/polygon-map-generation/

[16] http://weblog.jamisbuck.org/2011/2/7/maze-generation-algorithm-recap

[17] Explicación del algoritmo de generación de mazmorras empleado en TinyKeep.