Generación procedural en los videojuegos

*Memoria del trabajo de fin de grado*

Alejandro Oceja Trueba

*Dirigido por*

Domingo Gómez Pérez

# Capítulo 1: Introducción

## 1.1 Motivación

Uno de los mayores costes, tanto económicos como temporales, dentro de la industria de la creatividad y el entretenimiento digital es la creación de modelados tridimensionales. Además, a causa tanto de la incesante mejora de las capacidades de renderizado de los sistemas de entretenimiento como del aumento de la demanda por parte de los consumidores de productos de mayor complejidad y envergadura, las empresas del sector requieren de una gran cantidad de empleados y horas de trabajo, viéndose obligadas a reducir el presupuesto en otras áreas, como el desarrollo de mecánicas jugables mas avanzadas o una historia más compleja y elaborada. La generación procedural de contenido (PCG) se refiere a la creación algorítmica de contenido en un videojuego con una cantidad limitada, o indirecta de interacción por parte del usuario [1].

Esto permite la generación automática o semiautomática de contenido, permitiendo así la liberación de parte de los recursos destinados a estas tareas para utilizarlos en otras áreas de igual importancia.

Esto afecta aun mas a las empresas pequeñas del sector, las denominadas empresas indie, que se componen de un numero reducido de personas y cuentan con presupuestos bajos, con lo cual suelen optar por diseñar productos con gráficos pobres o menos cuidados, centrándose mas en otras mecánicas. La PCG es de especial interés para este tipo de empresas, pues les permitirá poder generar modelados de calidad sin la inversión de recursos que solo es posible para las grandes empresas del sector.

## 1.2 Background

La utilización de la generación procedural de contenido en los videojuegos existe desde los años 70, cuando se lanzó al mercado *Beneath the Apple Manor*, el primer videojuego en implementar tecnología de generación procedural para sus escenarios.

En esta época, la poca cantidad de memoria en los computadores obligaba a los creadores de videojuegos a limitar sus creaciones a pequeños escenarios, por lo que la PCG era una opción interesante para poder ampliar el contenido de los videojuegos sin necesidad de utilizar mas memoria. Posteriormente en esta época se lanza al mercado *Rogue*, un videojuego similar a *Beneath the Apple Manor*, que también utilizaba la generación procedural de contenido, y que consiguió mas fama que el primero, dando además nombre a un genero de videojuegos, el *roguelike*. En estos videojuegos, el jugador debe recorrer mazmorras que se generan de manera procedural al comenzar la partida, de forma que, en cada partida, el escenario recorrido es diferente, logrando así una cantidad de contenido imposible de alcanzar de manera manual.

Posteriormente se crearon otros videojuegos que implementaron la PCG en diferentes aspectos, por ejemplo, el videojuego de 1996 *Diablo* utiliza la PCG para generar los nombres y las estadísticas de los objetos y los enemigos que el personaje puede encontrar dentro del juego. Esto es también implementado en el videojuego *Borderlands*, en el cual existen 17,750,000 armas diferentes, todas ellas generadas utilizando algoritmos de PCG.

Otro aspecto de los videojuegos en el cual se emplea la PCG es la generación del mundo del juego. El género *sandbox* permite al jugador interactuar con el mundo del juego con total libertad, planteando al jugador objetivos básicos, como sobrevivir, o encontrar recursos. Un videojuego de esta categoría es el popular videojuego *Minecraft*, donde se utilizan técnicas de PCG para generar un mundo infinito, así como el contenido de este.

Similares a *Minecraft* podemos encontrar otros juegos que también utilizan algoritmos para generar mapas de gran tamaño, como *Terraria*, o *Civilization*, pero quizás el mayor exponente del uso de esta tecnología en los videojuegos es *No Mans Sky*, un videojuego ambientado en un universo de ciencia ficción en el que el jugador puede explorar numerosas galaxias, visitando sus planetas. Cada galaxia cuenta con millones de sistemas, y cada sistema cuenta con un numero de planetas orbitando una estrella. Además, cada planeta cuenta con su propia fauna y flora, toda ella generada de forma procedural, desde los modelados tridimensionales hasta las animaciones de cada criatura, incluso la música de el videojuego es generada utilizando la PCG.

Todo este contenido hubiese sido imposible de crear a mano, independientemente del tamaño del equipo de desarrollo encargado de la tarea.

## 1.3 Objetivo

El objetivo principal de este trabajo es aportar técnicas que permitan automatizar el desarrollo de videojuegos, más concretamente, el diseño y generación modelados tridimensionales de calidad y realismo aceptables, centrando el trabajo en la creación de una herramienta para la generación procedural de mapas realistas, además de la investigación e implementación de algunos algoritmos utilizados en la PCG.

Partiendo de ahí, nos planteamos los siguientes objetivos específicos:

Estudio de los algoritmos más comúnmente utilizados en la generación de contenido procedural, así como la implementación de algunos de estos utilizando el motor gráfico Unity.

Crear una plataforma que permita la generación de mapas tridimensionales realistas, con un nivel de detalle variable, y de forma procedural, similar al programa *World generator.*

Conseguir que los mapas generados sean interesantes desde el punto de vista del jugador, además de extenderse indefinidamente.

## Trabajo relacionado

Para analizar el estado del arte de este proyecto, tenemos que tener en cuenta 3 aspectos fundamentales:

La generación procedural de contenido para videojuegos.

La generación de contenido para videojuegos mediante algoritmos evolutivos.

Las diferentes implementaciones de IAs existentes.

### Generación Procedural

Generación procedural es el método de creación de contenidos a través de algoritmos, en oposición a un método de creación manual, y se aplica tanto en simulaciones gráficas por computadora como en videojuegos, instalaciones, programación y en música. Los fractales son un ejemplo de animación mediante generación procedural: funciones matemáticas gráficas repetidas hasta el infinito. El sonido digital también se puede generar

10 de forma procedural, un uso que se ha desarrollado mucho para la música

electrónica.

En videojuegos e instalaciones la generación procedural se utiliza en función de que el contenido se genere en la computadora que los contiene, en tiempo real, y no de manera previa y renderizado en paquetes gráficos predefinidos. Se utiliza principalmente para la generación de ambientes y mapas, aunque también se aplica para IA y jugabilidad. Se utiliza para generar de manera rápida y detallada espacios infinitos y patrones de singularidad en objetos, simulaciones y personajes.

También se utiliza para la creación de sistemas de partículas para agua, fuego y gases; o para generar un sinfín de actores digitales únicos y diferentes como reemplazo de extras. Sin embargo, no se suele utilizar para el contenido final, el cual en general se muestra ya

preestablecido. En (Gamasutra, 2014) se muestra una técnica para generar mazmorras aleatorias.

Los pasos para ello son:



Figura 2.1: Escenario creado mediante Generación Procedural

Generar una serie de salas con una determinada anchura y altura y

colocarlas aleatoriamente dentro de un círculo.

Separar las diferentes salas aplicando físicas de colisión, hasta que no se superpongan.

Después, para escoger qué salas son las principales, se tendrán en cuenta aquellas que estén por encima de 1.25 en el ratio ancho/alto. Tomar los puntos medios de las salas principales y aplicarles el Procedimiento de Delaunay (Departamento de Matemática Aplicada, 2015)

para obtener un grafo a partir de los triángulos obtenidos. A partir del grafo, se debe obtener el árbol de expansión mínimo (Algorithms y more, 2012). Esto hará que todas las habitaciones principales sean accesibles, pero que no estén conectadas directamente.

Finalmente, se añaden los pasillos a la mazmorra. Si los nodos están cerca horizontalmente, se añade una línea horizontal. Lo mismo para los nodos cercanos verticalmente. Si no es así, se añaden dos líneas que formen una L.

Técnicas Evolutivas

Existen proyectos que, mediante gramáticas evolutivas, generan historias de fondo que aportan consistencia a la experiencia de juego, con la intención de evitar que éstas deban ser ideadas por desarrolladores de forma manual (García-Ortega y García-Sanchez, 2014).

Existen herramientas que permiten utilizar algoritmos genéticos para generar mazmorras en base a unos parámetros de diseño (Font, Izquierdo, Manrique y Togelius, 2016), que tienen que venir definidos previamente.

En este sentido cabe reseñar el ejemplo explicado en (Togelius, Preuss, Beume, Wessing, Hagelbäck, Yannakakis y Grappiolo, 2013), que cuenta cómo se pueden desarrollar funciones heurísticas que miden propiedades de los mapas y comprueban si afectan a la experiencia de juego. Se diseñaron dos representaciones diferentes de mapas, una para un juego de estrategia genérico y otro para el videojuego Starcraft. Mediante este método

se puede automatizar completamente la generación de mapas o como herramienta de apoyo para diseñadores humanos. Otro ejemplo más es el referido en (Pérez, Togelius, Samothrakis, Rohlfshagen y Lucas, 2013) que analiza 3 subcomponentes diferentes en el

proceso de generación de mapas y propone soluciones a todos ellos.

Los mapas se representan mediante 2 conceptos distintos, la geometría y la disposición

del contenido. La geometría se lleva a cabo tanto en el exterior como en el interior conectando segmentos de mapas pre-hechos para formar mapas más grandes y completos. La disposición del contenido a través del mapa se determina mediante el uso de características de la geometría como el uso de una Red de Producción de Patrones de Composición. Por último están las preferencias del jugador para diseño del contenido, que se capturan y se utilizan en un marco de sistema recomendador. Todas las soluciones se

combinan y se prueban en el videojuego de acción y disparos Angry Bots (Unity, 2011) y se evalúan en un experimento a gran escala. Existe un precedente de uso de AGs para evolucionar el comportamiento de los bots -enemigos jugadores controlados por el ordenador- en un videojuego más o menos actual, concretamente el Unreal Tournament 2004 (Bullen y Katchabaw, 2004).Los resultados demostraron una mejora considerable

en el rendimiento de los bots evolucionados, con respecto al grupo de control. En otra ocasión se usó programación genética para la evolución de estrategias defensivas. En (Jackson, 2005) se utiliza la programación genética para evolucionar estrategias en un simple juego que emula los invasores del espacio.



Figura 2.2: Emulador de Space Invaders

Para la función de evaluación se realizaron un conjunto de pruebas. En cada una, el defensor se colocaba en la parte inferior del mapa, mientras que los oponentes se colocaban en cualquiera de las filas superiores, desplazándose inicialmente en una dirección elegida al azar. Después, irían descendiendo de izquierda a derecha, bajando cuando llegasen a los límites laterales del tablero y lanzando bombas de vez en cuando. Se utilizaron los operadores genéticos más comunes.

Se demostró que mediante programación genética es posible desarrollar

estrategias de defensa para hacer frente a juegos de una determinada

dificultad. Esas estrategias incorporan muchos de los aspectos de la evasión,

la búsqueda y la conducta de orientación que se encuentran en jugadores

humanos.

# Capítulo 2: Teoría de la PCG

En la generación de contenido procedural, existen numerosos métodos para abordar los problemas que estos sistemas plantean. Togelius et al. presentó en su estudio sobre la generación procedural de contenido distintas aproximaciones, que permitiesen emplazar cualquier aplicación de PCG en un punto entre dos extremos.

## 2.1.1 Online vs offline

Aquí se distingue la generación de contenido en función de si se realiza durante la ejecución del juego, *online*, por ejemplo, en el videojuego *Left 4 Dead*, la generación de enemigos en el mapa se realiza de forma procedural, variando en función de las acciones de los jugadores, o de si se realiza previamente a la ejecución del juego, o incluso durante el desarrollo de este. Como ejemplo aquí podríamos tomar una herramienta que generase mapas o modelos, sobre los que después trabajase un diseñador, para agregarlos a un juego, o una herramienta de creación de mapas dentro de un juego de estrategia en tiempo real.

Como ventaja principal de este tipo de generación *offline* tenemos que permite la creación de contenidos de alta complejidad, además de permitir un mayor control sobre el contenido final que el usuario experimentará. Como contraposición, la generación *online* permite al jugador disfrutar de una experiencia mucho mas dinámica, incluso adaptada al usuario, aprendiendo de sus decisiones y presentándole una experiencia más disfrutable.

## 2.1.2 Necesaria vs opcional

Como segunda distinción para la generación de contenido procedural, tenemos el tipo de contenido *necesario* y el *opcional.*

El contenido *necesario* es todo aquel que el jugador requiere para poder progresar a través del juego, como los mapas, mazmorras, reglas de la jugabilidad, etc., mientras que el contenido *opcional* es aquel contenido que no es necesario para la correcta progresión en juego, y por lo tanto el jugador puede escoger evitarlo, por ejemplo, armas u objetos, o lugares secundarios, a los que el jugador puede decidir no entrar, sin que ello afecte a su experiencia final.

La principal diferencia entre ambos contenidos es que el contenido *necesario* debe ser siempre correcto, o al menos tener un alto nivel de éxito, ya que generar una mazmorra que el jugador no pudiese recorrer debido a una geometría extraña, o un enemigo demasiado difícil de vencer, harían imposible para el jugador progresar en el juego.

En cuanto al contenido *opcional*, no importa si algunas veces se generase contenido de este tipo, como objetos sin utilidad o una mazmorra secundaria sin sentido, ya que el jugador siempre podrá escoger no utilizar dicho objeto o no entrar en la mazmorra.

Como ejemplo de contenido *opcional* tenemos el videojuego *Borderlands*, en el cual se generan armas de forma procedural, pero cuyo algoritmo genera en algunos casos armas que son inútiles, o simplemente inferiores a las que el jugador posee, y sin embargo esto no afecta en ningún momento a la experiencia del jugador de forma negativa, mientras que en el videojuego *Gran Turismo 5*, donde se generan de manera procedural las pistas de rally, la generación de una pista imposible de transitar para el jugador significaría un grave problema de diseño.

## 2.1.3 Aleatorio vs parametrizado

Otra distinción importante de la PCG es cuanto control se puede ejercer sobre el contenido que se va a generar. Un algoritmo puede simplemente recibir como parámetro una semilla numérica aleatoria a partir de la cual generar una mazmorra, o recibir un vector de parámetros con el numero de habitaciones, longitud de los pasillos, numero de enemigos, etc.

## 2.1.4 Estocástico vs determinista

Aquí nos referimos al grado de aleatoriedad del proceso de generación, a la cantidad de variación entre el resultado de la ejecución de un algoritmo con unos parámetros y el de la ejecución posterior del mismo algoritmo con los mismos parámetros.

Un algoritmo determinista siempre producirá los mismos resultados dados los mismos parámetros, mientras que uno estocástico variará el resultado generado.

## 2.1.5 Constructivo vs generación y prueba

En los algoritmos constructivos, el contenido se genera en una sola ejecución del mismo, como en los juegos *roguelike*, mientras que, en la generación y prueba, se genera todo el contenido, o una parte del mismo, y después se evalúa para comprobar si el resultado es lo suficientemente satisfactorio, repitiendo la generación en el caso de no serlo.

## 2.5.6 Generación automática vs generación mixta

Normalmente, en la mayoría de los algoritmos de PCG, el algoritmo recibe unos parámetros de entrada por parte del usuario, y a partir de estos, genera el contenido procedural, sin necesidad de más intervención por parte del usuario. Sin embargo, además de esta forma de generación automática, existe un tipo de generación mixta, donde algoritmo y diseñador cooperan para producir el contenido, bien el diseñador creando un nivel inacabado, para que el algoritmo rellene posteriormente los huecos de forma consistente, o bien el algoritmo genera parte del contenido, que después el diseñador puede modificar y terminar de forma más detallada.