Acta de Correcciones al Proyecto de Grado
Ingeniería de Sistemas y Computación

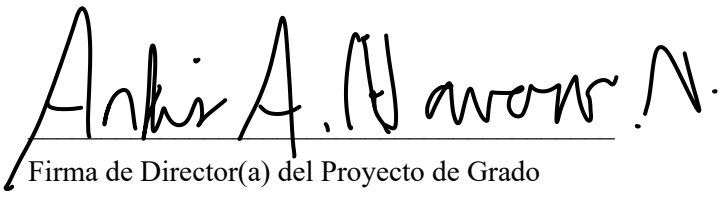
Fecha: Diciembre 9, 2022

Autores: Juan David Gomez Quiñones

Nombre del Proyecto de Grado:
Generador Procedural de Dungeons para Video Juegos 3D

Director:

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.



Firma de Director(a) del Proyecto de Grado

Nota de Aceptación

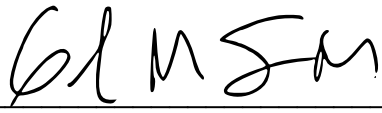
Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero de Sistemas y Computación.

Dr. Camilo Rocha
Decano de la Facultad de Ingeniería

ING. GERARDO MAURICIO SARRIA
Director Carrera Ingeniería Sistemas y Computación.

ING. Andrés Adolfo Navarro Newball
Director(a) Trabajo

ING. David Sebastian Baldeón
Jurado 1



ING. Gerardo Mauricio Sarria
Jurado 2

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias.
Ingeniería de Sistemas y Computación
Trabajo de Grado

Generador Procedural de Dungeons para Video Juegos 3D

Juan David Gomez Quiñones

Director: Dr. Andres Adolfo Navarro Newball

Septiembre 2022

Santiago de Cali, Septiembre 2022.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria Montemiranda
Director Carrera de Ingeniería de Sistemas y Computación.

Cordial Saludo.

Por medio de la presente me permito informarle que el estudiante de Ingeniería de Sistemas, Juan David Gomez Quiñones (cod: 8939024), trabaja bajo mi dirección, en el proyecto de grado titulado "Generador Procedural de Dungeons para Video Juegos 3D ".

Atentamente,

Dr. Andres Adolfo Navarro Newball

Santiago de Cali, Septiembre 2022.

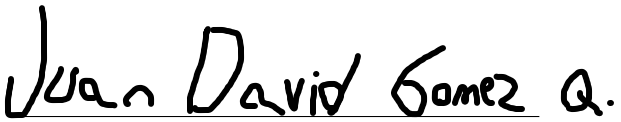
Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria Montemiranda
Director Carrera de Ingeniería de Sistemas y Computación.

Cordial Saludo.

Me permito presentar a su consideración el trabajo de grado titulado "Generador Procedural de Dungeons para Video Juegos 3D " con el fin de cumplir con los requisitos exigidos por la Universidad para optar al título de Ingeniero de Sistemas y Computación.

Al firmar aquí, doy fe que entiendo y conozco las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería y Ciencias aprobadas el 26 de Noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del anteproyecto y del trabajo de grado.

Atentamente,



Juan David Gomez Quiñones
Código: 8939024

Agradecimientos

Agradezco a mi padre Jorge Hernan Gomez y a mi madre Claudia Maria Quiñones por todos los sacrificios que hicieron por mi y que me ayudaron a llegar a este punto de mi vida. También agradezco a mi hermano Simón Gomez por todo su apoyo y motivación.

En segundo lugar doy gracias a mi director de tesis, Andrés Navarro por su guía y acompañamiento durante la elaboración del presente trabajo. Estoy muy agradecido de haber podido realizar este trabajo de grado en un área que me apasiona y de la cual disfruto mucho.

Por ultimo, quiero agradecerle a mis amigos por sus ideas, palabras de animo y compañía a lo largo del desarrollo de este proyecto.

Abstract

The Dungeons are game levels very commonly used in various video game genres, especially in the RPG (Role-playing game) genre. The Dungeons are levels where players can find different types of missions, puzzles, enemies and rewards. Traditionally in the video game development process, the architecture and design of these scenarios are done manually and due to the high demand for this type of scenario within a game, development ends up being too expensive. Due to the above, it is very common to find many linear, monotonous and little entertaining Dungeons with a low probability of replayability. In this work, an algorithm capable of generating a 3D representation of a Dungeon and some of its main elements was developed, taking as input a set of parameters and 3D meshes that allow customizing the Dungeon to the user's needs.

Key Words: Dungeon,Vídeo Games, RPG, 3D Rendering, Random Generation, Parameters,Replayability.

Resumen

Las Dungeons (o mazmorras por su traducción al español) son escenarios de juego muy comúnmente usados en varios géneros de vídeo juegos, sobre todo en el genero RPG (Role-playing game).Las Dungeons son escenarios donde los jugadores pueden encontrar diferentes tipos de misiones, acertijos, enemigos y recompensas. Tradicionalmente en el proceso de desarrollo de vídeo juegos, la arquitectura y diseño de estos escenarios se realizan de forma manual y debido a la alta demanda de este tipo de escenarios dentro de un juego, el desarrollo termina siendo demasiado costoso. Debido a lo anterior, es muy común encontrar muchas Dungeons lineales, monótonas, poco entretenidas y con una baja probabilidad de rejugabilidad. En este trabajo se desarrolló un algoritmo capaz de generar la representación 3D de una Dungeon y algunos de sus elementos principales, tomando como entrada un conjunto de parámetros y mallas 3D que permiten la personalización de la Dungeon a las necesidades del usuario.

Palabras Clave: Dungeon,Vídeo Juegos, RPG, Representacion 3D, Generación Aleatoria, Parámetros,Rejugabilidad.

Índice general

Introducción

Los escenarios en los vídeo juegos son uno de los elementos más importantes dentro de la mayoría de géneros, dado a que son los espacios donde los jugadores experimentan todo lo que un vídeo juego tiene para ofrecer. El tipo de escenario varía mucho dependiendo del tipo de juego y sus características, no es lo mismo comparar un escenario de un vídeo juego de disparos o FPS (First Person Shooter) que el de un juego de aventura RPG (Role-playing game), incluso muchos juegos de un mismo género pueden llegar a tener escenarios completamente distintos. No obstante las Dungeons se han convertido en un tipo de escenario muy común y recurrente en múltiples géneros de vídeo juegos, sobre todo, en el género RPG.

Aunque un vídeo juego puede tener una historia increíble y mecánicas de juego excelentes, la jugabilidad y la experiencia del usuario puede llegar a decaer mucho si el escenario donde se desarrolla el vídeo juego esta mal planteado y es mal desarrollado [1]. Es por eso que muchos desarrolladores tratan de agregar una gran cantidad de escenarios a sus juegos con diseños increíbles e inmersivos que ayuden a explotar todos los aspectos positivos que el juego puede ofrecerle al jugador, sin embargo, ofrecer la cantidad ideal de escenarios con un buen diseño no es tan trivial como suena [2]. Normalmente para desarrollar un conjunto de Dungeons es necesario personal como diseñadores, ingenieros multimedia o de sistemas, tiempo, y por supuesto dinero. Por lo que incluso según las guías oficiales de estrategia de los mejores y mas populares juegos RPG como son "The Elder Scrolls V: Skyrim"[3] o "The Witcher 3: Wild Hunt"[4], se puede evidenciar que la tendencia en desarrollo es hacer un pequeño conjunto de Dungeons principales y complejas, mientras que las demás Dungeons del juego siguen un patrón lineal y simple.

Lo explicado anteriormente trae como consecuencia que los niveles secundarios y con poca complejidad sean muy monótonos, repetitivos y por ende, poco rejugables. Lo cual resulta ser una problemática debido a que afecta significativamente la experiencia del jugador y la cantidad de horas que este invierte en el juego. Es por esto que este proyecto plantea un algoritmo que busca ayudar a los desarrolladores a generar Dungeons complejas e interesantes de forma aleatoria y automática en base a una serie de parámetros que se adaptan a los requisitos del escenario.

Capítulo 1

Descripción del Problema

1.1. Planteamiento del Problema

El diseño de escenarios es uno de los componentes principales en el desarrollo de vídeo juegos, debido a que los escenario son el medio físico en el que los jugadores experimentan todo lo que el juego tiene para ofrecer. Asimismo, con el fin de crear un escenario entretenido, los desarrolladores deben de analizar a fondo los requisitos detrás de este y pensar en las mejores formas en las que desafiar y recompensar a los jugadores sin desbalancear las mecánicas del juego [5].Esto mismo lo manifiesta el desarrollador de Epic Games, J. Brown en su conferencia "The Importance of Everything: Analytics of Map Design", donde menciona que uno de los factores mas importantes para crear un juego exitoso, es lograr escenarios de calidad que sean entretenidos y retengan la atención del jugador [1].

Sin embargo, el proceso de desarrollo de escenarios no es un proceso sencillo y en la mayoría de los casos se realiza de forma manual. Este proceso suele dividirse en múltiples etapas, entre ellas el concepto de arte, borradores, esquemas, renderización, pruebas, corrección de errores y más [6]. A lo anterior se le debe adicionar que para poder crear escenarios de calidad, se requiere de personas especializadas en el diseño de estos, o al menos de personas con conocimientos del tema, por lo que el tiempo y el costo para generar un escenario de vídeo juegos es bastante considerable [7].

Ahora bien, hablando de las Dungeons como un tipo de escenario, se puede decir que la cantidad de Dungeons en un vídeo juego de tipo RPG suele variar según la ambición del proyecto pero casi nunca se cuenta con solo una Dungeon. "The Elder Scrolls V: Skyrim" lanzado en el 2011 por Bethesda Game Studios, y ganador de premios a mejor juego del año, mejor juego de rol entre otros [8], cuenta con cerca de 156 Dungeon jugables, entre mazmorras bajo tierra, cuevas, castillos y más[3].

Si se retoma lo anterior, nos damos cuenta que para garantizar una buena experiencia de juego y crear un juego exitoso, es necesario tener múltiples escenarios que desafíen a los jugadores de la forma adecuada, que no rompan con el balance del juego y que sean lo suficientemente completas e interesantes para garantizar múltiples horas de juego, llamar la atención de los jugadores y qué jueguen en diferentes ocasiones.Todos estos factores hacen que el proceso de planificación y creación de los escenarios consuma bastante tiempo y recursos del proyecto en forma proporcional al tamaño y ambición del vídeo juego. Por lo tanto, una herramienta que ayude a la automatización de varios de los anteriores aspectos, podría ayudar a ahorrar costos y agilizar el proceso de desarrollo.

1.1.1. Formulación

¿Cómo Automatizar el proceso de creación de Dungeons en un escenario de Vídeo Juego?

1.1.2. Sistematización

- ¿Cuáles son los parámetros relevantes para la creación de una Dungeon?
- ¿Como desarrollar un algoritmo que genere de forma aleatoria la arquitectura de una Dungeon en base a los parámetros proporcionados?
- ¿Cómo Construir una representación Visual de la Dungeon a partir de la arquitectura generada?
- ¿Cómo probar la Dungeon generada?

1.2. Objetivos

1.2.1. Objetivo General

El objetivo de este trabajo es diseñar e implementar un algoritmo que permita automatizar el proceso de creación de Dungeons en un escenario de Vídeo Juego.

1.2.2. Objetivos Específicos

- Determinar parámetros relevantes para la creación de una Dungeon.
- Desarrollar un algoritmo que permita generar de forma aleatoria la arquitectura de una Dungeon en base a los parámetros proporcionados.
- Construir una representación Visual de la Dungeon a partir de la arquitectura generada.
- Probar la Dungeon Generada.

1.3. Justificación

El presente proyecto es relevante dentro de la industria de los Vídeo Juegos debido a dos razones importantes. En primer lugar, permitirá automatizar un proceso que durante mucho tiempo se a realizado de forma manual. De esta forma los estudios de Vídeo Juegos (sin importar su tamaño) podrán ahorrar recursos valiosos como lo son el tiempo, el dinero y el personal de desarrollo. En segundo lugar, otorgara una amplia variedad y diversidad a las Dungeons dentro de un los juego, incentivando la rejugabilidad y aumentara las horas que un jugador puede invertir dentro del juego.

En los últimos años la industria de los vídeo juegos a crecido de forma exponencial y hoy en día es una de las industrias con mayor relevancia dentro de la sociedad, siendo una de las industrias

más rentables del mundo. En el 2020 genero aproximadamente 179.7 billones de dólares, quedando por encima de la industria del cine y del deporte (norte americano) juntos en el 2021 [9]. Dentro de los Vídeo Juegos existen diferentes géneros y en definitiva uno de los mas populares es el genero RPG, en el cual las Dungeons son escenarios bastante frecuentes y en muchas ocasiones, parte fundamental de la trama de el Vídeo Juego.

El objetivo de este proyecto es el desarrollar un algoritmo el cual genere la arquitectura base de una Dungeon teniendo en cuenta unas serie de parámetros establecidos, como lo son la longitud de la Dungeon (haciendo referencia a la cantidad de zonas o habitaciones), la cantidad de enemigos y la cantidad de tesoros. Esta arquitectura base que generara el algoritmo servirá como base para que posteriormente el equipo de desarrollo personalice y ambiente la Dungeon dependiendo de la temática de su Vídeo Juego.

Es importante resaltar que desde hace ya un tiempo, existen algoritmos similares y varios estudios han desarrollado su propio generador para su propio vídeo juego, sin embargo estos generadores en su mayoría se quedan mucho en el campo teórico y no fueron diseñados para la implementación dentro de la industria, sino que más bien, como material académico, o fueron diseñados para un solo juego en especifico y son completamente privados del estudio que lo desarrollo. La intención de este proyecto es crear un algoritmo generador que pueda ser usado por aquellos equipos de desarrollo que trabajan sobre el motor de vídeo juegos Ünreal Engine 4z de esta forma no se vean obligados a crear uno propio para un proyecto en especifico.

1.4. Delimitaciones y Alcances

- El objetivo de este proyecto no es desplazar el factor creativo en el diseño de los entornos de los Vídeo Juegos, ya que esto es un factor importante que determina la esencia y el éxito de un este, por lo tanto no se decorar ni se amueblan las Dungeons Generadas. Sino que más bien, se entrega un modelo de la Dungeon que puede ser personalizado y usado como base para modificaciones futuras.
- Aunque la generación de la Dungeon es aleatoria, la generación de cada modulo dentro de esta no lo es. En su lugar se usan módulos que representan habitaciones o sub zonas de la Dungeon. Estos módulos son definidos por el usuario y son usados para construir la Dungeon y son donde se posicionan los enemigos y recompensan.
- El algoritmo esta implementado en el motor de vídeo juegos de Epic Games Unreal Engine 4, sin embargo es posible implementar el algoritmo en otros motores como lo son Unity o Godot Engine.
- El algoritmo espera las siguientes entradas:
    - Conjunto de mallas 3D que se usaran como base para construir la arquitectura de la Dungeon.

- Longitud de la Dungeon.
- Conjunto de actores que representaran las recompensas en la Dungeon.
- Cantidad de recompensas, bonificaciones y su forma de distribución en la Dungeon.
- Conjunto de actores que representaran los enemigos en la Dungeon.
- Modelo de distribución de enemigos (constante o adaptativo). Un modelo constante espera la cantidad de enemigos a distribuir, mientras que un modelo adaptativo espera la dificultad, un APL (Average Party Level) y un multiplicador de la cantidad base de enemigos.

1.4.1. Entregables

- Documento del Proyecto.

Capítulo 2

Marco de Referencia

2.1. Áreas Temáticas

- *Theory of computation →Theory and algorithms for application domains →Algorithmic game theory and mechanism design →Algorithmic game theory*
- *Theory of computation →Theory and algorithms for application domains →Algorithmic game theory and mechanism design →Solution concepts in game theory*
- *Theory of computation →Theory and algorithms for application domains →Algorithmic game theory and mechanism design →Representations of games and their complexity*
- *Applied computing →Computers in other domains →Personal computers and PC applications →Computer games*
- *Mathematics of computing →Discrete mathematics →Graph theory →Random graphs*
- *Mathematics of computing →Discrete mathematics →Graph theory →Graph algorithms*
- *Mathematics of computing →Discrete mathematics →Graph theory →Paths and connectivity problems*

2.2. Trabajos Relacionados

Con el fin de construir una base sólida para el desarrollo del presente proyecto, se realizó la exploración de trabajos relacionados a la generación de automática de entornos de vídeo juegos. A continuación se presentan los resultados de dicha exploración:

- El vídeo juego "Bloodborne"lanzado en el 2015 y desarrollado por FromSoftware cuenta con un generador de dungeons aleatorias muy llamativo para los jugadores y que resulta ser muy similar al algoritmo de desarrollado en este proyecto. Yamagiwa, productor de Bloodborn menciono en una entrevista con IGN que las Çhalice Dungeons"son escenarios generados de forma procedural siempre cambiantes y que permiten a los jugadores obtener en cada partida una experiencia diferente y refrescante", donde deben pensar en nuevas estrategias y enfrentarse a nuevos desafíos, aumentando enormemente la cantidad de horas que se invierten en el juego [10]. Adicionalmente, la guía de estrategia oficial del juego menciona otras características llamativas de dichas dungeons, como lo es la cantidad de recompensas y enemigos que

siempre es variable entre las dungeons y la existencia de ayudas visuales para guiar al jugador por el camino correcto. La guía de estrategia también menciona que las dungeons se generan a través de módulos predefinidos y se conectan a través de niveles, puertas y pasadizos [11].

- Adams *et al.* [12] Presenta un generador automático de Dungeons usando gramáticas de grafos.Trata de solucionar la problemática del alto costo de desarrollar estos mapas y el problema de la rejugabilidad. Como parte del proyecto se implementa un algoritmo capaz de evaluar el mapa generado y adaptarlo a requisitos particulares. Se lograron alcanzar los objetivos del proyecto y se muestra que el uso de grafos permite modelar escenarios y estructuras complejas. Este proyecto solo desarrolla el modelo de grafos y no una representación 3D visual jugable.

- Barriga [13] Presenta un estudio donde se discuten aspectos importantes del desarrollo de escenarios de vídeo juegos. Se menciona que el costo aproximado del desarrollo tradicional de escenarios y contenido, el cual esta cerca del 30 % y 40 % de los 20 a 150 millones de dolares del presupuesto promedio de un vídeo juego distribuido por una editora importante. Asimismo, se presenta que componentes como los escenarios,los espacios, los sistemas, el diseño del juego, la vegetación y contenido derivado pueden ser generados de forma procedural. Por ultimo se presentan algunos de los métodos mas comunes de generación, como los construidos a base de números aleatorios, a base de machine learning y de métodos basados en búsqueda.

- Bidarra *et al.* [14] Presentan un estudio acerca de los métodos más comunes para la generación procedural de contenido para entornos de vídeo juegos, en específico los entornos de tipo Dungeon. Los autores recompilan prácticas similares entre los modelos de generación, mencionando que la mayoría de generadores primero realizan un modelo representacional para luego definir un método de construcción y por ultimo realizar la representación física de la Dungeon. Adicionalmente, los autores presentan varias categorías de algoritmos de generación como lo son los generados mediante Autómatas Celulares, Algoritmos Genéticos, Basados en Restricciones, de aproximación Híbrida,de gramática generativa, entre otros. Asimismo, para cada categoría de algoritmo los autores mencionan el tipo de enfoque que común mente se toma. Un algoritmo del tipo de gramática generativa suelen tomar enfoques de gramática de grafos, lo que sería una categoría y un enfoque similar al tomado en el presente proyecto.

- Chaimowicz *et al.* [15] Presentan un generador de escenarios 2D para vídeo juegos a través de autómatas celulares que aprovecha las propiedades de los fractales curvos del llenado de espacios. Este proyecto tiene como objetivo solucionar el problema de la rejugabilidad y el costo de la generación de los mapas, mostrando que la actual demanda de escenarios interactivos es muy grande para los grandes estudios y aun mucho más grande para los estudios pequeños. Los resultados de este proyecto fueron publicados en Unity engine.

- Ontañon *et al.* [16] Presentan un algoritmo para generar escenarios de vídeo juegos 2D similares a los de juegos como Super Mario Bros, Loderunner y Kid Icarus. Se utilizaron técnicas de machine learning en modelos de Markov, los cuales son modelos estadísticos que permiten

cambiar un sistema de forma aleatoria donde estados futuros dependen de los estados actuales y no en los pasados. Este proyecto busca generar mapas 2D para juegos de plataforma que no sean dependientes del juego como tal, sino que por el contrario se busca poder generar mapas para una gran variedad de vídeo juegos sin muchas características en común.

- Gemrot *et al.* [17] presentan un algoritmo para solucionar el problema de "demasiada aleatoriedad en la generación de Dungeons", ya que según los autores, muchos escenarios generados de forma aleatoria suelen sentirse demasiado desbalanceados y carentes de una estructura lógica. El algoritmo tiene como entrada un conjunto de estructuras en bloques (figuras poligonales que representan la forma de habitación) y produce una Dungeon que sigue la estructura de un grafo conectado con el nivel especificado. Es decir que el algoritmo logra pasar de la representación en grafo de una Dungeon a una estructura de una Dungeon compuesta por habitaciones modulares con formas poligonales. Adicionalmente permite que los usuarios usen sus propias formas para generar las habitaciones, permite el posicionamiento de puertas personalizadas y corredores entre habitaciones. Finalmente, se presenta que un mismo grafo puede generar estructuras diferentes dependiendo de la forma en la que se conectan.

2.3. Marco Teórico

Para el desarrollo del presente proyecto, se deben de tener conocimientos tanto de ciencias de la computación como de mecánicas de Vídeo Juegos. A continuación se presentan las bases teóricas sobre las que se trabajara en el desarrollo del proyecto.

2.3.1. Dungeons

Mientras que en la vida real el termino Dungeon, mazmorra o calabozo hace referencia a celdas laberínticas comúnmente subterráneas y conectadas por pasillos estrechos, en los vídeo juegos de aventura RPG este término suele ser usado con un poco más de libertad. Bidarra *et al.* manifiesta en "Procedural generation of dungeons"[14] que una Dungeon si bien puede ser fiel a su definición original, también puede tratarse de castillos, cuevas, estructuras abandonadas, pantanos, bosques y muchas más estructuras que dependen de la creatividad del equipo de desarrollo.

Por otra parte, más allá de la geometría las Dungeons suelen ser escenarios con un punto de partida y un punto de finalización normalmente custodiado por un enemigo especial de gran importancia llamado usualmente conocido como jefe. Adicionalmente las Dungeons suelen contener tesoros, acertijos, y personajes no jugadores (NPC por sus siglas en ingles). Usualmente el objetivo de un jugador dentro de una Dungeon es llegar al punto de finalización y derrotar al jefe de la Dungeon.

2.3.2. Dungeons & Dragons

Dungeons & Dragons es un juego de mesa de fantasía actualmente publicado por la compañía Wizards of the Coast, es un juego de tablero con lápiz y papel donde la mayoría de los jugadores toman el rol de un personaje de fantasía y un jugador toma el papel de Maestro de Juego y es el encargado de guiar las dinámicas de cada sección de juego. La importancia de este juego en este proyecto es que Dungeons & Dragons es un juego bien conocido por ser el principio de los juegos RPG modernos y ha venido evolucionando en múltiples ediciones desde la fecha de su lanzamiento en 1974. Debido a la larga historia de este juego, se han creado varios libros guía oficiales para los Maestros de Juego, libros que explican aspectos importantes sobre que modelos seguir para el posicionamiento de enemigos y recompensas para cada tipo de Dungeon [18]. Debido a lo anterior, se tomaron varios modelos de la quinta edición del libro guía para maestros de juego como base para los algoritmos de posicionamiento de entidades diferentes a la geometría de la Dungeon.

2.3.3. Dificultad

La dificultad de un vídeo juego hace referencia a el reto que supone un vídeo juego, o un componente de este a los jugadores y es directamente dependiente de las intenciones de los desarrolladores. Sin embargo Csikszentmihalyi [19] señala que es importante saber medir el nivel de dificultad y propone un canal de flujo en el que la dificultad del juego debería de ser establecida en base a el nivel del desafió y la experiencia del jugador. De esta forma se evitan niveles demasiado difíciles y por ende frustrante al igual que niveles demasiado fáciles y por ende aburridores.

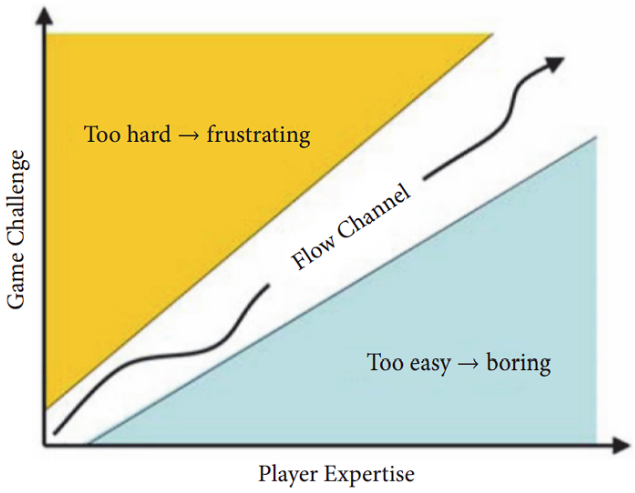


Figura 2.1: Curva de Dificultad propuesta por Csikszentmihalyi

Por lo tanto es común encontrar dos tipos modelos de dificultad en los vídeo juegos:

- Un modelo adaptativo o dinámico que según Zohaib [20] tiene como objetivo es generar siempre escenarios correspondientes al nivel del jugador adaptando las características de los

enemigos y el entrono para poder proporcionar siempre un desafió placentero que no sea ni muy fácil ni muy difícil. Es posible encontrar este tipo de dificultad en juegos como "The elder scrolls Online"donde el nivel de los enemigos en las Dungeons avanza según el nivel del jugador.

- Un modelo constante donde al contrario de la dificultad adaptativa, cada escenario tiene una dificultad predefinida y es fija según los criterios del desarrollador. Si el jugador se encuentra en el momento del juego adecuado podrá encontrar una gran experiencia las primeras veces que juegue la Dungeon ya que un desarrollador definió de forma manual los parámetros de dificultad en base a su experiencia. Sin embargo, los parámetros de la Dungeon no progresan con el jugador, por lo que si este vuelve en una etapa posterior del juego, encontrara un nivel mucho más sencillo. Por otro lado, si el jugador trata de saltar a un nivel muy por encima de su nivel actual encontrara un nivel demasiado difícil y letal [21]. Este modelo se puede ver en juegos "World of Warcraft"donde cada Dungeon está pensado para un momento en específico del juego.

2.3.4. Balance de Elementos

Las Dungeons son entornos llenos de tesoros, enemigos y más elementos, los cuales deben ser posicionados de forma estratégica para garantizar una buena experiencia de juego. Perez [22], muestra múltiples conceptos fundamentales para el posicionamiento de las recompensas dentro de una Dungeon, mostrando que de preferencia deben estar distribuidos por toda la Dungeon para garantizar que los jugadores la exploren. Adicionalmente, las mejores recompensas deben estar posicionadas con los jefes finales o las zonas más complicadas de la Dungeon. Mearls *et al* [18], mencionan aspectos importantes a la hora de posicionar enemigos y la forma en la que escala la dificultad según se avanza en la escenario. Es decir, los jefes y enemigos más difíciles deben estar ubicados en las zonas más lejanas del punto de partida. En base a estas teorías se planea establecer parámetros para la generación aleatoria de la Dungeon.

2.3.5. Motores de Vídeo Juegos

Los motores de vídeo juegos son herramientas muy importantes en la industria de los vídeo juegos en la actualidad, ya que estos se encargan de facilitar la carga de trabajo de los desarrolladores, permitiéndoles implementar funciones de forma inmediata y fácil sin necesidad de hacer todo desde un bajo nivel. Hoy en día casi todos los desarrolladores usan su propio motor de vídeo juegos o hacen uso de un motor de terceros. Para el presente proyecto hizo uso del motor Unreal Engine desarrollado por la compañía Epic Games, ya que hasta el día de hoy Unreal Engine es uno de los motores de terceros más completos y más usados en la industria de los vídeo juegos tanto como para PC como para consolas de Vídeo Juegos [23].

- **Unreal Engine:** Es un conjunto completo de herramientas de desarrollo que facilita la creación y edición de escenarios complejos debido a sus herramientas para edición de terrenos,

cielos, luz ambiental y cuerpos de agua. Adicionalmente, cuenta con herramientas para el desarrollo de animación, simulación, luz, inteligencia artificial, renderización, manejo de colisiones entre otros [24]. Unreal Engine cuenta con dos lenguajes programación para el desarrollo, C++ y una tecnología denominada Blueprint Visual Scripting las cuales pueden ser usadas al mismo tiempo. Asimismo esta característica fue de gran importancia en el desarrollo del presente proyecto, ya que permitió dividir el proyecto en una capa de lógica y otra de renderización, trabajando la lógica de la generación en C++ y el posicionamiento de entidades mediante Blueprints.

2.3.6. Grafo No Dirigido

Un grafo no dirigido es un tipo de grafo $G = (V, E)$, donde V es un conjunto finito de nodos o vértices y E es un conjunto finito de aristas, que relacionan los nodos de forma simétrica y no tienen un sentido definido. En el presente proyecto se usa un grafo no dirigido como representación de una Dungeon, donde V representa el conjunto de habitaciones o sub zonas y E el conjunto de puertas que conecta dos habitaciones de forma bidireccional.

2.3.7. Modelo de Erdös–Rényi:

Roney [25] muestra que el modelo de $Erdos-Rényi$ es uno de los modelos clásicos que permiten la construcción de un grafo aleatorio $G(V,p)$ a través de una distribución de Poisson, partiendo de un conjunto V de vértices. Para todo $V \in G$, se asigna una probabilidad p $\in$ (0,1) tal que para cada V_i,$V_j \in G$, p determinará si existe una conexión entre V_i,V_j, entre mas alto sea el valor de p existirá una mayor probabilidad de conexión entre los dos vértices y aumentara la conectividad del grafo G.

Dado que para este proyecto una Dungeon se modelara a partir de una representación de un grafo donde cada vértice V representa una habitación o sub-zona de la Dungeon y cada arista E representa un camino entre dos vértices V_i,V_j. Se considera que una adaptación del modelo de Erdos Renyi es suficiente para cumplir con los alcances establecidos. No se esperan generar grafos muy extensos con múltiples caminos que escapen de las capacidades del modelo de Erdos-Renyi, ya que un grafo muy extenso y con una gran conectividad puede resultar en una Dungeon demasiado compleja y confusa para los jugadores. En el presente proyecto fue necesario hacer una modificación al algoritmo para garantizar que todos los vértices $V \in G$ tengan al menos una conexión y no tengan más conexiones que el número de puertas máximas definidas por la malla 3D de la habitación. La figura 2.2 muestra el algoritmo de $Erdös - Rényi$.

Algorithm 1: Original ER

Input: v : number of vertices, indexed 0 to $v-1$;
p : inclusion probability
Output: G : Erdős-Rényi graph

```
G = ∅;
for i = 0 to v − 1 do
    for j = 0 to v − 1 do
        Generate a uniform random number θ ∈ [0, 1);
        if θ < p then
            G ← (i, j);
```

Figura 2.2: Algoritmo de Erdos-Renyi [26]

2.3.8. Algoritmos para Recorrer Grafos:

Para poder posicionar las entidades de la Dungeon y garantizar el cumplimiento de sus propiedades y parámetros, es necesario recorrer el grafo que representa a la Dungeon. Es por esto que varios de los algoritmos desarrollados se basan o contienen algoritmos diseñados para recorrer grafos, entre ellos se encuentran los algoritmos DFS y BFS.

- **DFS (Depth-First Search):** También conocido como búsqueda en profundidad por su traducción al español, consiste en recorrer todos los vértices de un grafo *G(V,E)*. Partiendo desde la raíz de G de forma ordenada y en profundidad hasta llegar a un vértice donde no se puedan alcanzar más vértices, al llegar a este punto se procesan los vértices adyacentes no procesados de los vértices ya procesados [27], ver figura 2.3.

```
ITERATIVEDFS(s):
  PUSH(s)
  while the stack is not empty
      v ← POP
      if v is unmarked
          mark v
          for each edge vw
              PUSH(w)
```

Figura 2.3: Algortimo de DFS

- **BFS (Breadth First Search):** También conocido como búsqueda en amplitud por su traducción al español, consiste en explorar todos los vértices de un grafo *G(V,E)*. Partiendo desde la raíz de G y explorando todos los vértices adyacentes. En una siguiente iteración del algoritmo para cada uno de los vértices explorados, se exploran sus vértices adyacentes no explorados [27], ver figura 2.4.

```
BFS(s):
  InitSSSP(s)
  Push(s)
  while the queue is not empty
      u ← Pull( )
      for all edges u→v
          if dist(v) > dist(u) + 1          ⟨⟨if u→v is tense⟩⟩
              dist(v) ← dist(u) + 1         ⟨⟨relax u→v⟩⟩
              pred(v) ← u
              Push(v)
```

Figura 2.4: Algortimo de BFS

2.3.9. Distancias Entre Vértices:

Para poder posicionar los jefes y tesoros dentro de la Dungeon de una forma balanceada, es necesario conocer que tan cerca está un nodo con los elementos mencionados anteriormente de otros nodos importantes como el inicio o el final de la Dungeon. Para esto se planea usar el algoritmo de Dijkstra para grafos no dirigidos. Erickson [27] muestra que dado un grafo *G(V,E)* el algoritmo de Dijkstra permite encontrar el camino mas corto desde un nodo a todos los demás nodos. El algoritmo de Dijkstra funciona de forma similar al algoritmo BFS, solo que en lugar de usar una cola de tipo FIFO (First In First Out), se usa un cola de prioridad donde la clave de un vértice V es su distancia tentativa *dist(V)*, ver figura 2.5.

```
Dijkstra(s):
  InitSSSP(s)
  Insert(s, 0)
  while the priority queue is not empty
      u ← ExtractMin( )
      for all edges u→v
          if u→v is tense
              Relax(u→v)
              if v is in the priority queue
                  DecreaseKey(v, dist(v))
              else
                  Insert(v, dist(v))
```

Figura 2.5: Algortimo de Dijkstra

Capítulo 3

Análisis

Como se mencionó anteriormente, al crear un escenario de vídeo juegos se debe de buscar una forma de diseñar un desafío entretenido para los jugadores. Para esto es necesario tener en cuenta múltiples aspectos, como lo son la geometría, la topología y el balance de los elementos dentro del escenario, sin descuidar también la autoridad que debe de tener el desarrollador sobre el escenario a generar. Al analizar la problemática anterior, se descubre que esta se puede dividir en dos problemáticas diferentes: ¿Cómo definir la geometría y la topología del escenario? y ¿Cómo generar los elementos del escenario? En el presente capitulo se discutirán posibles soluciones para cada una de estas problemáticas.

3.1. Geometría y Topología

Intuitivamente, antes de generar la representación 3D de una Dungeon es necesario pensar en la arquitectura de esta misma y responder a preguntas como: ¿Qué tan larga debe ser?, ¿Qué tan laberíntica debe ser? o ¿Qué diseño debe de seguir?

Si se retoma la definición de una Dungeon como un escenario de vídeo juegos conformado por un conjunto de habitaciones o de sub zonas las cuales están conectadas entre sí, entonces se puede crear una Dungeon a partir de su representación en un grafo bidireccional $G = (V, E)$, siendo G la Dungeon en su totalidad, V el conjunto de habitaciones y E el conjunto de conexiones entre habitaciones. Por lo que una forma de generar la estructura de una Dungeon es mediante un modelo de generación de grafos aleatorios, en el caso de este proyecto, el modelo de *Erdos - Renyi*. De lo anterior se puede concluir que para la generación de la topología de una Dungeon son necesarios los siguientes parámetros:

- **Longitud:** La cantidad de habitaciones, sub zonas o $|V|$.
- **Probabilidad de conexión:** Una probabilidad $1 \geq P \geq 0$, la cual define cuando una pareja de vértices $u, v \in V$ se relacionan de forma bidireccional a través de una arista $e \in E$. Valores de P cercanos a 1 resultan en Dungeons con varios caminos diferentes, mientras que valores cercanos a 0 resultan en Dungeons simple o lineales.

En adición a estos dos parámetros, es importante pensar otro parámetro que le permita al desarrollador establecer un límite a la cantidad de caminos o puertas que tiene cada habitación, lo que también puede entenderse como el límite de aristas por nodo. Esto es necesario debido a que el

modelo de *Erdos - Renyi* permite una cantidad de aristas por nodo de hasta $|V|-1$ [25], lo que significa que un escenarios de una gran longitud podría generar habitaciones con demasiados caminos o puertas diferentes. La generación no regulada de aristas es un resultado indeseado debido a que más allá de las limitaciones físicas de cuantas puertas puede tener una malla 3D de una habitación, puede ser frustrante para un jugador tener muchos caminos distintos a elegir. Al revisar las guías de estrategia de algunos de los juegos RPG más populares hasta la fecha como "The Witcher 3: Wild Hunt"[4] y "The Elder Scrolls V: Skyrim"[3] se puede observar principalmente que el nivel de complejidad depende del juego y que realmente la mayoría de habitaciones o sub zonas de las Dungeons no suelen tener más de 4 o 5 caminos diferentes, ver figuras 3.1 y 3.2.

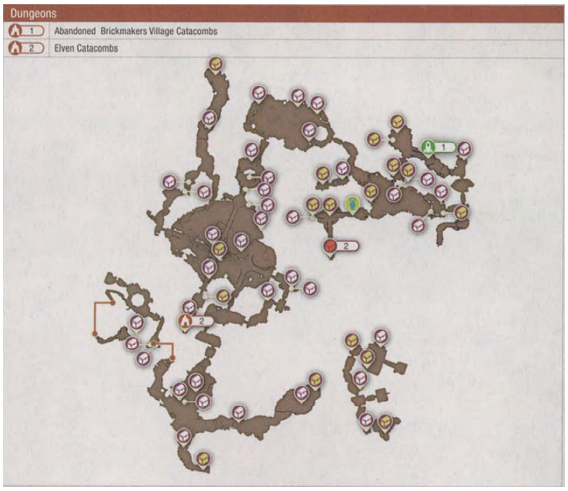


Figura 3.1:
"Elven Catacombs" de "The Witcher 3: Wild Hunt

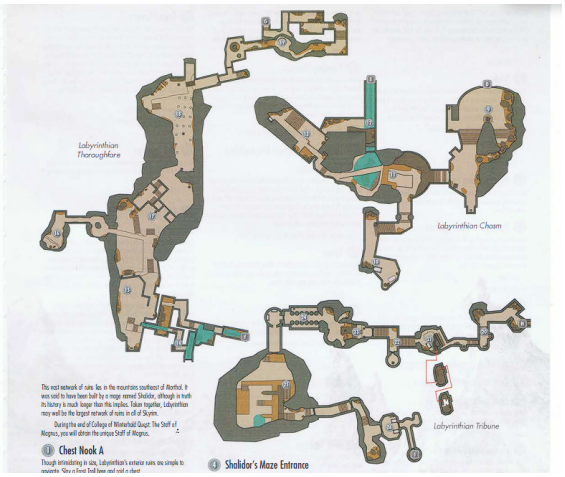


Figura 3.2:
"Labyrinthian" de "The Elder Scrolls V: Skyrim"

Ahora bien, al hablar de cómo será la representación 3D de la Dungeon es importante comprender que no todos los vídeo juegos son iguales y por lo tanto no se puede esperar el mismo tipo de Dungeons en todo los vídeo juegos. El diseño de los escenarios está fuertemente relacionado con la temática del vídeo juego ya que estos son el medio físico por el cual un juego expresa todo lo que tiene para ofrecer, asimismo los desarrolladores siempre están en una búsqueda constante de componentes que ofrezcan una sensación única de inmersión y que hagan que sus escenarios resalten de los escenarios de los demás vídeo juegos [2].

Por lo tanto, el presente proyecto no genera la estructura de cada habitación o sub zona, si no que más bien sigue un modelo similar a la generación de Dungeons procedurales del vídeo juego "Bloodborne"[10]. El desarrollador puede establecer un conjunto de mallas 3D a partir de las cuales son posicionadas y conectadas según la representación del grafo de la Dungeon. De esta forma el desarrollador puede establecer el diseño, la temática y todas las características de diseño del entorno.

El desarrollador no está limitado a seguir un único estilo de Dungeon si no que el mismo puede escoger que tipo de Dungeon generar. Esta aproximación deja dos posibles usos principales para el generador.

- **Generación en Etapa de Desarrollo:** Se genera una Dungeon cuya intención es servir como base a una Dungeon estática la cual puede ser refinada en un futuro y agregada a la documentación del juego.
- **Generación Dentro del Juego:** En esta aproximación el generador está integrado al juego y su función es generar en tiempo real varias versiones de una misma Dungeon con parámetros iguales. De esta forma un jugador podría experimentar desafíos distintos en una nueva iteración de una misma Dungeon. Esta mecánica sería muy útil en juego MMORPG (Masive Multiplayer Online Rol Play Game) donde los jugadores deben repetir varias veces una misma Dungeon para conseguir un ítem con bajas probabilidades o para subir de nivel.

3.2. Posicionamiento de Elementos

En el presente proyecto se trabaja sobre dos de los elementos más importantes de una Dungeon, los enemigos y las recompensas. Sin embargo, al igual que en el análisis de la geometría y la topología es importante comprender que no todas las recompensas y enemigos son lo mismo en todos los vídeo juegos. Aunque la naturaleza del enemigo sea igual en dos juegos distintos, el desafió que representa al jugador no necesariamente es igual. Por ejemplo, no es lo mismo enfrentar un enemigo de tipo “Esqueleto” en un juego como “The Elder Scrolls V: Skyrim” a un “Esqueleto” en un juego como “Dark Souls 3” donde la dificultad suele ser muy superior. Siguiendo lo planteado por Csikszentmihalyi [19] y Zohaib [20] se puede pensar las siguientes dos formas de definir la dificultad y el posicionamiento de los enemigos dentro de la Dungeon.

- **Modelo Constante:** En este modelo el desarrollador ingresa directamente la cantidad de enemigos totales en la Dungeon.
- **Modelo Adaptativo:** En este modelo el desarrollador no ingresa directamente la cantidad de enemigos si no que ingresa parámetros para que el generador decida la cantidad de enemigos y su posición en base a los modelos discutidos en la guía de maestros de juego de ”Dungeons & Dragons”[18]. Se debe ingresar el nivel estimado en el que se encuentra el jugador respecto al nivel máximo disponible en el juego, la dificultad que se espera que la Dungeon represente para el jugador y un multiplicador para controlar la cantidad de enemigos que son posicionados.

Los modelos mencionados anteriormente solo se usan para el posicionamiento de enemigos regulares. Sin embargo, el jefe de la Dungeon se debe de posicionar en la habitación más alejada de la Dungeon con respecto a la habitación de inicio, esto debido a que como su nombre lo dice, es el enemigo más fuerte del escenario y debe ser el desafió final de la Dungeon.

Ahora bien, los tesoros son una parte fundamental en los vídeo juegos de tipo de RPG ya que son una forma directa de recompensar al jugador después de uno o varios desafíos. Aunque al igual que con los enemigos el tipo de tesoros dependa del juego, el nivel y las intenciones de los desarrolladores, casi todos los juegos genero RPG tienen algo en común, una moneda “in game”. Los nombres,valores y representaciones de estas monedas pueden cambiar con los juegos, en “Dark Souls” se manejan las almas, en ”Diablo”se maneja el oro, en “The Legend of Zelda” se manejan las Rupias, etc. Estas monedas suelen servir como moneda de intercambio de bienes mediante los jugadores o NPCs del juego. Las mecánicas de distribución de recompensas se realizan según la guía de maestros de juego de ”Dungeons & Dragons”[18], donde el desarrollador debe introducir los siguientes parámetros:

- **Monto de Recompensa:** Una cantidad aproximada de que tantas recompensas debe recolectar en total el jugador en su travesía por la Dungeon. El desarrollador debe definir este valor debido a que una recompensa de una misma magnitud puede tener dos significados completamente opuestos en juegos distintos. Adicionalmente de que no existe una tasa de cambio entre monedas de cada juego.
- **Monto Correspondiente al Jefe:** Los jefes suelen resguardar la mayor cantidad de las recompensas de una Dungeon, por lo que el desarrollador debe definir qué porcentaje del monto total de recompensas quiere asignarle al jefe de la Dungeon.
- **Bonificación por Dificultad:** Este parámetro determina si el desarrollador desea dar una bonificación adicional por completar los desafíos en una dificultad de juego más alta. Aunque muchos juegos deciden recompensar a los jugadores con mejores recompensas según la dificultad del juego, esto no es necesariamente una regla estricta, por lo que el desarrollador puede definir si otorgar o no la bonificación y en qué medida hacerlo.

Capítulo 4

Diseño

Después de analizar y delimitar los parámetros para la solución de la problemática, se realiza el diseño de un algoritmo capaz de generar una Dungeon con las siguientes entradas y salidas:

4.1. Entradas:

- **Longitud:** De tipo entero, define la cantidad de habitaciones en la Dungeon.
- P**:** De tipo flotante, define la probabilidad para el modelo de $Erdos - Renyi$.
- **Limite de grado:** De tipo entero, define el número máximo de aristas que cada vértice del grafo pude tener.
- **Monto Total De Recompensas:** De tipo entero, define la cantidad de tesoros a distribuir en la Dungeon.
- **Monto Correspondiente al jefe:** De tipo flotante, define qué porcentaje del monto total de recompensas entregara el jefe.
- **Bonificación de recompensa:** De tipo flotante, la bonificación que recibe la recompensa debido al nivel de dificultad.
- **Modelo de Dificultad:** De tipo entero, define si se usa un modelo de dificultad constante o adaptativo, como se puede apreciar en el cuadro 4.1 .

Valor	Modelo
1	Constante
2	Adaptativo

Cuadro 4.1: Modelo de Dificultad

Si el modelo de dificultad seleccionado es de tipo constante, entonces se espera la siguiente entrada adicional:

- **Enemigos Totales:** De tipo entero, define la cantidad de enemigos no jefes de la Dungeon.

Por el contrario, si el modelo de dificultad seleccionado es de tipo adaptativo, entonces se esperan las siguientes entradas adicionales:

- **Dificultad:** De tipo entero, define que tan desafiante es cada enemigo dentro de la Dungeon. Ver cuadro 4.2.

Valor	Dificultad
1	Fácil
2	Medio
3	Desafiante
4	Difícil
5	Épico

Cuadro 4.2: Niveles de dificultad

- **Multiplicador de Enemigos:** De tipo entero, establece un multiplicador a aplicar a la cantidad de enemigos retornados por el modelo de posicionamiento adaptativo.

- **Nivel Máximo:** De tipo entero, define el nivel máximo que un jugador puede obtener en el juego.

- **Average Party Level (APL):** De tipo entero, define el nivel en el que se espera que uno o varios jugadores recorran la Dungeon. El APL también puede traducirse como el promedio de niveles de los jugadores en un equipo.

Cabe resaltar que el algoritmo realmente recibe otras entradas como el conjunto de mallas 3D y actores para construir la representación visual de la Dungeon, Sin embargo, al tratarse de parámetros relacionados con "Unreal Engine 4", se hablara más a fondo de ellos en el capítulo de Implementación 5.

4.2. Salida:

La salida del algoritmo es un grafo el cual representa la Dungeon generada junto a sus atributos y la distribución de enemigos y recompensas.

Según lo discutido en la etapa de análisis, el algoritmo de generación se divide en dos componentes, un componente encargado de generar el grafo que representa la arquitectura de la Dungeon y un componente encargado de definir y posicionar los elementos como las recompensas y los enemigos en la Dungeon.

4.3. Componente de Arquitectura:

Este componente usa las entradas de la longitud, la probabilidad P y el límite de grado para generar una Dungeon cuya representación se expresa en un grafo bidireccional $G = (V, E)$, tal que la longitud de la Dungeon es igual a $|V|$ y cada vértice $v \in V$ tiene como máximo la cantidad de aristas definidas por el parámetro del Límite del Grado. Este componente consta de dos etapas:

- Primero se realiza un algoritmo de $Erdos - Renyi$ para generar la base del grafo de forma aleatoria tomando el vértice 0 como punto de partida de la Dungeon. Cabe destacar que antes de ejecutar el algoritmo de $Erdos - Renyi$ se aumenta el grado del vértice 0 en uno (1) sin establecer una conexión con otro vértice del grafo, esto con la finalidad de representar la puerta o el camino de entrada a la Dungeon.

- En segundo lugar, gracias a las propiedades de conectividad que tiene el modelo de $Erdos - Renyi$ es muy probable obtener un grafo inconexo [25], en el cual no existe un camino que recorra todos los vértices del grafo. Lo que se traduciría como zonas inalcanzables dentro de la Dungeon, como se puede apreciar en la figura 4.1. Por lo tanto, es necesario realizar un algoritmo el cual se encargue de convertir el grafo generado con el modelo de $Erdos - Renyi$ en un grafo conexo.

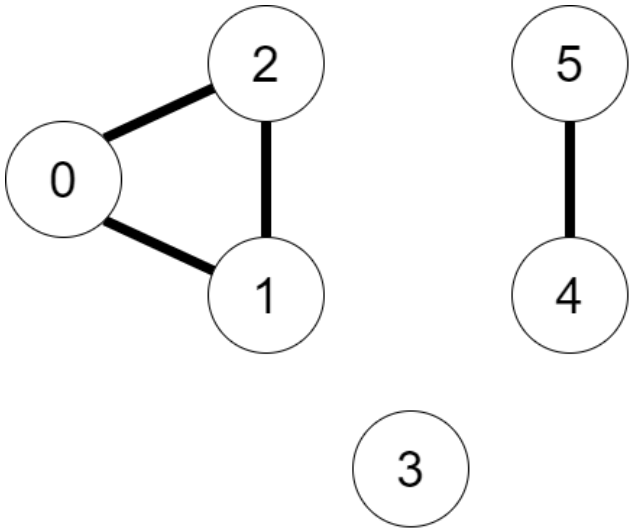


Figura 4.1: Grafo Inconexo

El algoritmo en cuestión parte de una DFS con la finalidad de identificar cuáles son los componentes del grafo, tal como se muestra en la figura 4.2.

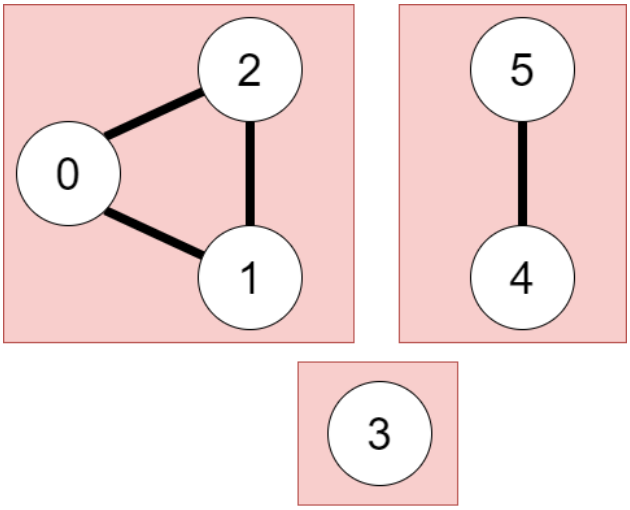


Figura 4.2: Componentes del Grafo

Para luego mediante una función de "linker" establecer una arista entre cada pareja de vértices de componentes distintos $u, w \in V$ tal que u y w tienen el menor grado en la componente, tal como se muestra en la figura 4.3.

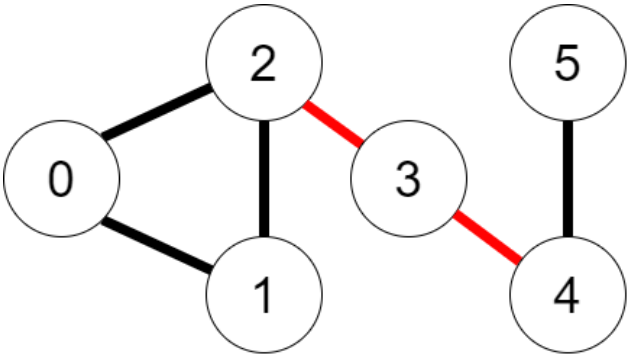


Figura 4.3: Grafo Conexo

4.3.1. Diagrama de flujo:

La figura 4.4 presenta el diagrama de flujo del algoritmo mencionado anteriormente para generar el grafo que representa la arquitectura de la Dungeon.

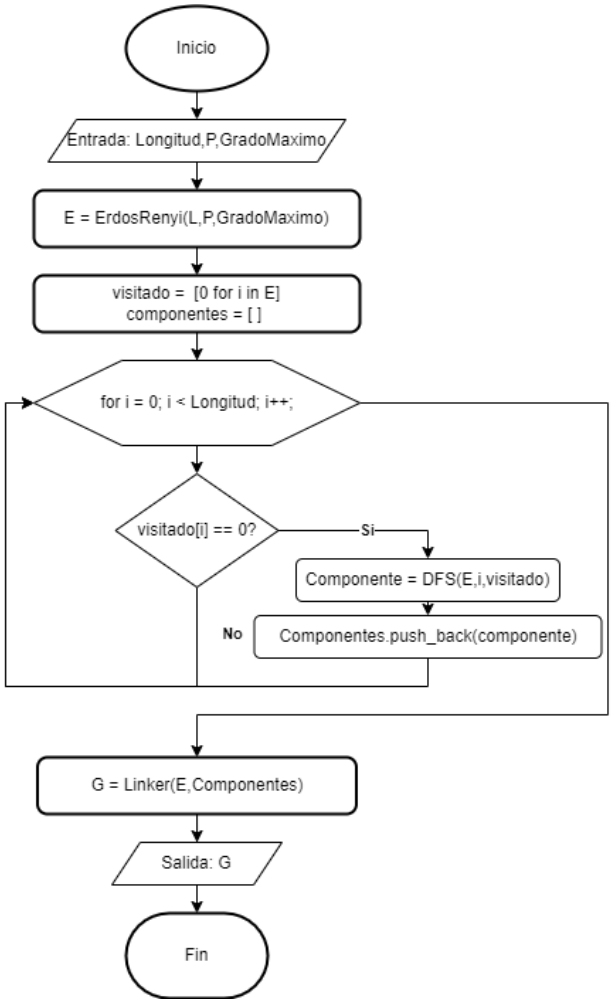


Figura 4.4: Diagrama de flujo del componente de arquitectura

4.4. Componente de Elementos:

4.4.1. Jefe de Dungeon:

Debido a que los enemigos de tipo jefe suelen representar el desafió final de la Dungeon, el algoritmo posiciona al jefe la habitación más lejana al punto de partida de la Dungeon. Es decir que este algoritmo busca el elemento maximal en el conjunto de los caminos más cortos entre todos los vértices respecto al vértice de inicio. Para esto se usa el algoritmo de *Dijkstra* asumiendo que todas las aristas del grafo tienen un peso de uno (1).

4.4.1.1. Diagrama de flujo:

La figura 4.5 presenta el diagrama de flujo del algoritmo mencionado anteriormente para definir el vertice del grafo en el cual se debe de posicionar al jefe de la Dungeon.

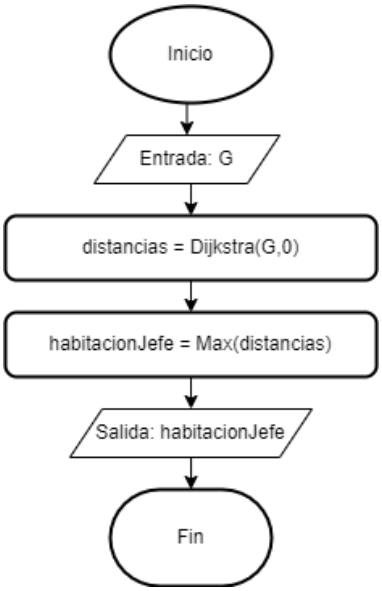


Figura 4.5: Diagrama de flujo para el posicionamiento del jefe

4.4.2. Enemigos Regulares:

Los enemigos regulares se posicionan según el modelo de dificultad seleccionado:

4.4.2.1. Modelo Constante:

Este modelo recibe la cantidad de enemigos totales y su funcionamiento se puede dividir en dos casos:

- Si la cantidad de enemigos totales es menor a la cantidad de habitaciones diferentes a la habitación donde se posiciona el jefe, entonces se posicionan los enemigos de forma aleatoria en habitaciones diferentes asegurando que no se repitan las habitaciones.
- Si la cantidad de enemigos es mayor a la cantidad de habitaciones disponibles entonces se divide el número de enemigos entre el número de habitaciones y se posiciona esa cantidad de enemigos por habitación. En el caso de que la división de enemigos entre habitaciones no de un valor exacto, se calcula el excedente de enemigos y estos se colocan de forma aleatoria entre las habitaciones (asegurándose de no repetir habitación).

La figura 4.6 presenta el diagrama de flujo del algoritmo mencionado anteriormente para el posicionamiento constante de enemigos en la Dungeon.

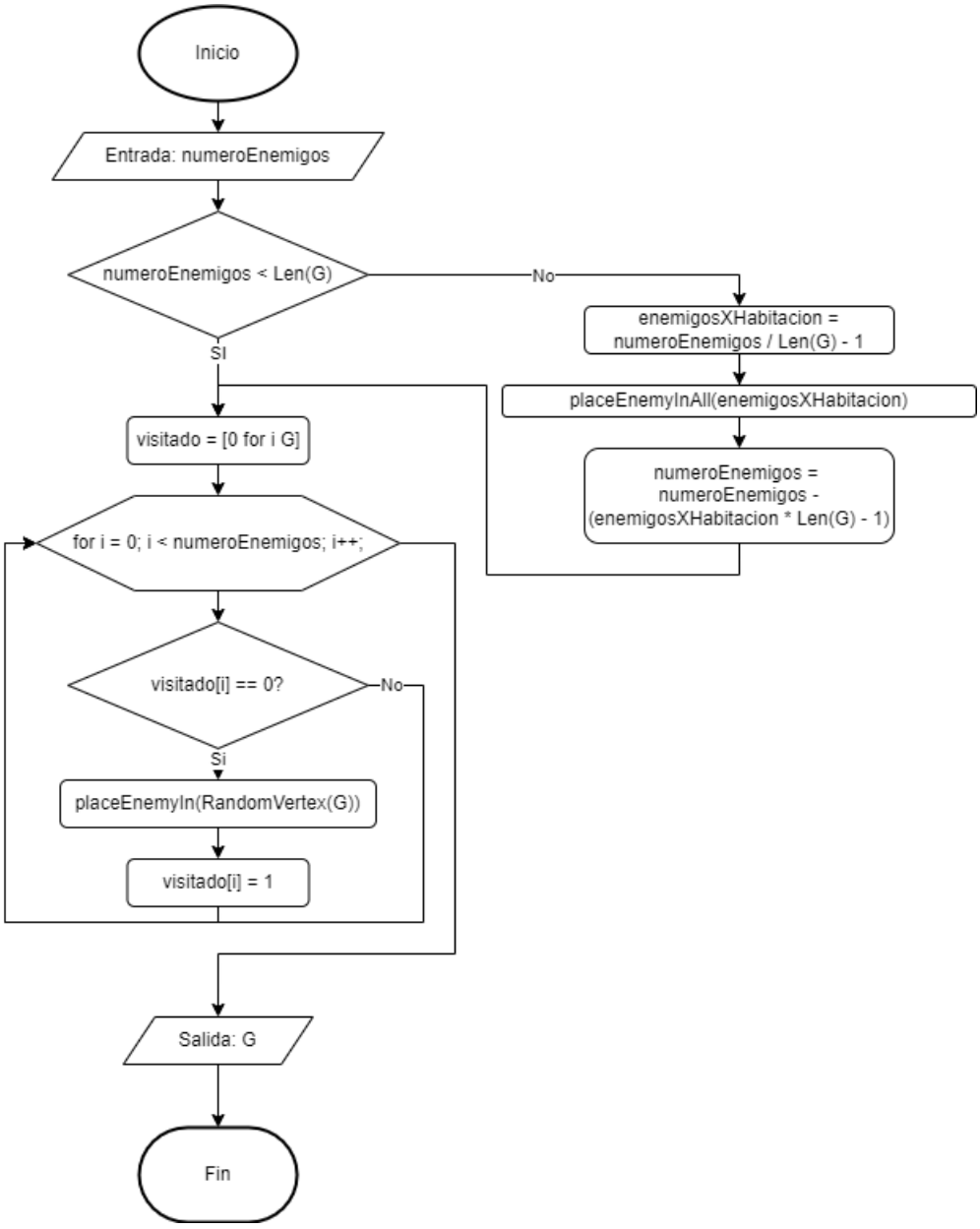


Figura 4.6: Diagrama de flujo para el posicionamiento constante de enemigos

4.4.2.2. Modelo Adaptativo:

Este modelo se basa en los modelos de posicionamiento de enemigos para Dungeons & Dragons [18]. El algoritmo recibe la dificultad, el nivel máximo del juego, el APL y un multiplicador k de enemigos y define cuantos enemigos deben ser posicionados de entre los siguientes cuadros que contemplan tanto la dificultad seleccionada como el progreso del jugador. Dado que la dificultad

hace referencia a que tan desafiante es cada enemigo para un jugador y que entre más alto las dificultades más altas posicionan menos enemigos.

- Si el APL se encuentra entre el primer 5 % del nivel máximo del juego, significa que se trata de jugadores nuevos por lo que se posiciona la cantidad de enemigos por habitación según el cuadro 4.3.

Dificultad	Enemigos por Habitación
Fácil	$2 \times k$
Medio	$1 \times k$
Desafiante	$1 \times k$
Difícil	$1 \times k$
Épico	$0{,}5 \times k$

Cuadro 4.3: Modelo para Jugadores Nuevos, $APL \leq 5\,\%$ nivel máximo.

- Si el APL se encuentra entre el 6 % y el 20 % del nivel máximo del juego, significa que se trata de jugadores con una experiencia intermedia por lo que se posiciona la cantidad de enemigos por habitación según el cuadro 4.4.

Dificultad	Enemigos por Habitación
Fácil	$2 \times k$
Medio	$2 \times k$
Desafiante	$1 \times k$
Difícil	$0{,}5 \times k$
Épico	$0{,}25 \times k$

Cuadro 4.4: Modelo para Jugadores Intermedios, 6 % nivel máximo $\leq APL \leq$ 20 % nivel máximo.

- Si el APL se encuentra entre el 21 % y el 100 % del nivel máximo del juego, significa que se trata de jugadores experimentados por lo que se posiciona la cantidad de enemigos por habitación según el cuadro 4.5.

Dificultad	Enemigos por Habitación
Fácil	$4 \times k$
Medio	$2 \times k$
Desafiante	$1 \times k$
Difícil	$0{,}5 \times k$
Épico	$0{,}25 \times k$

Cuadro 4.5: Modelo para Jugadores Experimentados, 21 % nivel máximo $\leq APL \leq$ 100 % nivel máximo.

Una vez definidos cuantos enemigos se deben posicionar por habitación, se procede de forma similar al modelo estático donde se posicionan los enemigos por habitación y en caso de no ser un valor exacto se calcula el excedente y se posiciona en habitaciones aleatorias (asegurándose de no repetir habitación).

4.4.3. Recompensas:

Las recompensas se entregan a los jugadores después de superar un desafió y se dividen en las recompensas entregadas por derrotar al jede de la Dungeon y las entregadas después de cada enfrentamiento en la Dungeon, por lo que solo las habitaciones que tienen al menos un enemigo son las habitaciones que tiene recompensas. Para distribuir las recompensas el algoritmo primero se calcula el monto de la cantidad de Recompensa del Jefe (ecuación 4.4.3):

$$RecompensadelJefe = \%CorrespondienteAlJefe \times MontoTotalDeRecompensas \tag{4.1}$$

Al substrae la recompensa otorgada por el jefe del monto total se obtiene el $MontoRestante$ el cual debe ser distribuido por cada habitación según su cantidad de enemigos. Se usa un modelo basado en la distribución de recompensas por enfrentamiento manejados en Dungeons & Dragons [18].

El cuadro 4.6 muestra la forma en la que se calculan las recompensas por habitación. Siendo $EnemigosTotales$ la cantidad de enemigos presentes en la Dungeon, $EnemigosHabitacion$ la cantidad de enemigos presentes en una misma habitación y $Bonus$ la bonificación de recompensa la cual es definida por el desarrollador y determina que tanto aumenta la recompensa según el nivel de dificultad.

Dificultad	Recompensa por Habitación
Facil	$(MontoRestante/EnemigosTotales) \times (EnemigosHabitacion)$
Medio	$(MontoRestante/EnemigosTotales) \times (EnemigosHabitacion) \times (1 + (\frac{(750 \times Bonus)}{44000}))$
Desafiante	$(MontoRestante/EnemigosTotales) \times (EnemigosHabitacion) \times (1 + (\frac{(3650 \times Bonus)}{44000}))$
Dificil	$(MontoRestante/EnemigosTotales) \times (EnemigosHabitacion) \times (1 + (\frac{(16500 \times Bonus)}{44000}))$
Epico	$(MontoRestante/EnemigosTotales) \times (EnemigosHabitacion) \times (1 + Bonus)$

Cuadro 4.6: Modelo para calcular recompensas según el nivel de dificultad

Capítulo 5

Implementación

El presente capitulo tiene como objetivo exponer el método empleado para construir la representación 3D mediante el uso del motor de vídeo juegos Unreal Engine 4.

5.1. Arquitectura

La implementación del proyecto consta de dos módulos principales, un módulo de lógica el cual contiene la implementación en C++ de todos los algoritmos discutidos en el capítulo de Diseño 4 y un módulo de visualización el cual se encarga de la lógica y el posicionamiento físico de todos los componentes de la Dungeon mediante la tecnología de Blueprints. Estos módulos pueden comunicarse con facilidad a través del tipo de función "Ufunction" nativa de Unreal Engine 4, la cual permite el paso de datos entre C++ y los Blueprints.

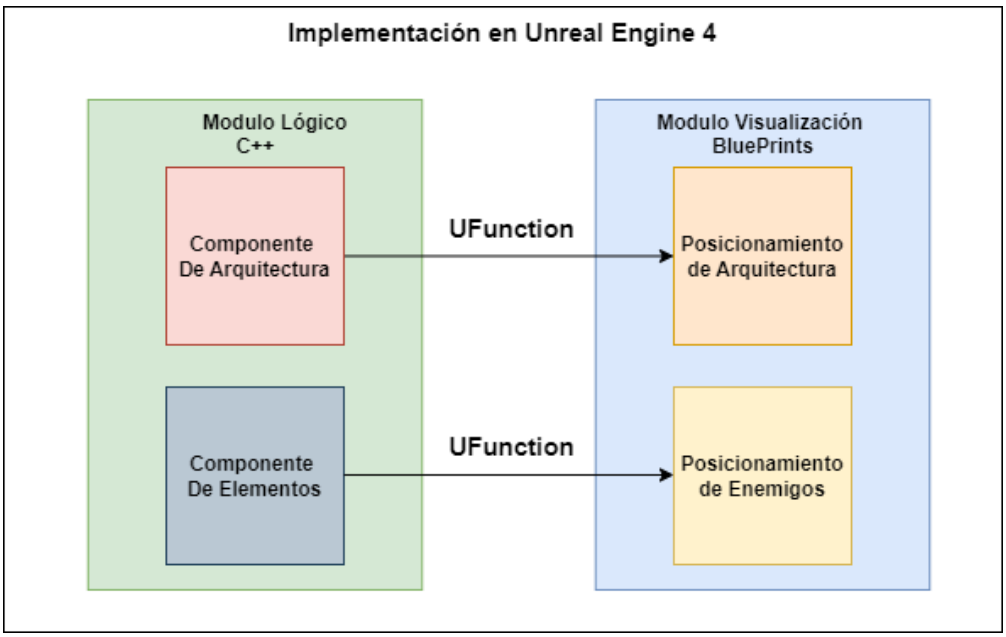


Figura 5.1: Arquitectura de la Implementación

La figura 5.1 muestra la lógica detrás de la implementación del proyecto. La capa de lógica se encarga de crear la representación de la Dungeon y definir todos sus atributos y enviarlos a

la capa de visualización la cual se encarga de posicionar los elementos a través del conjunto de herramientas provistas por Unreal Engine 4. La capa lógica envía a la capa de visualización datos como la cantidad de habitación, cuantas puertas tiene cada habitación, la habitación que contiene el jefe de Dungeon, la cantidad de recompensas, entre otros.

5.2. Nuevos Parametros

Para la construcción visual de la Dungeon es necesario recibir entradas adicionales a las ya mencionadas en el capítulo de Diseño 4:

- **Mallas de Habitaciones:** Conjunto de mallas 3D con las cuales se realiza la construcción visual de la Dungeon. El desarrollador debe de asegurarse de proveer la cantidad de mallas necesarias según el número máximo de puestas establecido. Si el número de puertas máximo es 3, entonces se debe proveer al menos 3 mallas (una de una puerta, una de dos puertas y una de tres puertas).
- **Datos de las Puertas:** Conjunto de coordenadas y rotaciones que especifican donde se encuentra cada puerta en cada Malla de habitación provista en el parámetro anterior. Estas coordenadas son respecto al centro de la malla de la habitación.
- **Actores:** Conjunto de Actores con los cuales representan el Jefe de la Dungeon, los enemigos regulares y las recompensas. Los actores son una clase genérica de Unreal Engine 4 la cual puede tener varios elementos, entre ellos animaciones y código. Por lo que los desarrolladores pueden proveer elementos con su lógica de juego ya desarrollada, como lo puede ser el comportamiento del jefe junto a sus animaciones y atributos.

5.3. Posicionamiento de Entidades:

Una vez creada la representación de la Dungeon a través de un grafo conexo y después de definir los atributos de los elementos de la Dungeon el módulo de Visualización se encarga de construir la representación 3D de la Dungeon tomando de forma aleatoria elementos del conjunto de mallas 3D provistas por el desarrollador que concuerden con la representación en el grafo

Como se mencionó anteriormente, los desarrolladores deben de proveer la cantidad de mallas 3D necesarias según la cantidad máxima de puertas por habitación definida en los parámetros. Esto con el fin de crear una matriz de mallas 3D que sirva como banco de modelos para la generación de la Dungeon. Como se puede observar en la figura 5.2, dada la matriz M de magnitud $R \times C$, todos los elementos de una misma fila r tienen r cantidad de puertas. Por lo que si el algoritmo debe crear una habitación con 3 puertas entonces se tomara un elemento aleatorio de la matriz cuya fila sea igual a 3. Por lo tanto, se puede concluir que las matrices con un mayor número de columnas,

es decir, una mayor cantidad de mallas 3D diferentes con una misma cantidad de puertas, pueden generar Dungeons más diversas.

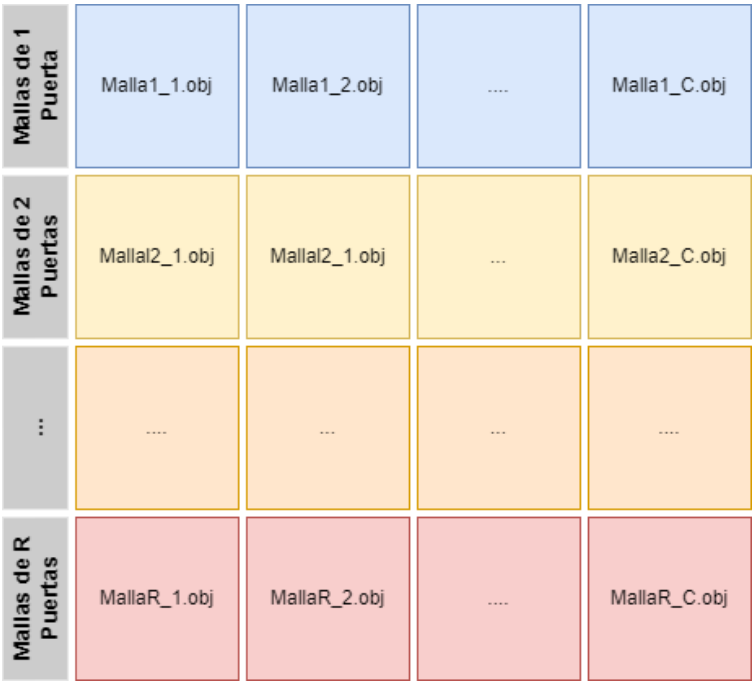


Figura 5.2: Banco de Mallas 3D

Cada habitación se posiciona sobre el eje X a una distancia definida de las otras habitaciones y los elementos de cada habitación como sus enemigos o recompensas (representadas como cofres) se posicionan en las coordenadas de cada habitación según junto con los atributos que indique la capa lógica del proyecto.

5.4. Conexión Entre Habitaciones

Como se mencionó en la sección anterior, las mallas de las habitaciones son posicionadas a lo largo del eje X a una distancia definida de las otras habitaciones, por lo que no hay una conexión física "puerta con puerta" entre las habitaciones. La forma en la que se conectan las habitaciones entre si es por medio de actores de teleportación los cuales son posicionados en la posición de cada puerta de la Dungeon. Cada actor de teletransportacion tiene como destino una habitación que está conectada con la habitación dueña del actor según la representación del grafo. Estos actores, al entrar en contacto con el jugador, lo mueven de forma instantánea a la ubicación de destino. Para el presente proyecto se contemplaron el uso los siguientes actores.

5.4.1. Actor con Cámara al Destino

El objetivo principal de este actor es engañar los sentidos del jugador para que sienta que realmente las habitaciones de la Dungeon están conectadas de forma física. El actor consta de una pareja de teletransportadores donde cada uno cuenta con una cámara virtual y un plano sin colisiones. La idea es que el plano proyecte lo capturado por la cámara virtual del otro teletransportador, algo similar a lo que se hace en los juegos de la saga "Portal". De esta forma el jugador siempre está viendo la ubicación de destino y al entrar en contacto con el plano será trasportador a la posición del otro transportador, dando la sensación que las habitaciones están conectadas de forma física. Tal como se puede apreciar en la figura 5.3, el jugador pude ver sin alteraciones lo que hay en el lugar de destino.

Figura 5.3:
Actor con vista al destino

5.4.2. Actor con Pantalla de Carga

Este actor consta de una malla 3D de una puerta o un objeto similar, el cual al entrar en contacto con el jugador muestra una pantalla de carga de unos pocos segundo de duración y mueve al jugador a la posición de destino. Al igual que con el actor anterior, la teletransportacion se hace de forma inmediata pero la pantalla de carga ayuda a mitigar la sensación de que el jugador está siendo teletransportado y logra crear una sensación de un movimiento natural de una habitación a otra.

Capítulo 6

Resultados

6.1. Primera Prueba:

Esta prueba tiene como objetivo exponer las capacidades de generación de Dungeons distintas a partir de un mismo conjunto de entradas:

Entrada	Valor
Longitud	10
P	0.5
Límite de Grado	5
Monto Total de Recompensas	100000
Monto Correspondiente al Jefe	0.30
Modelo de Dificultad	1
Enemigos Totales	15

Cuadro 6.1: Conjunto de entradas para la primera prueba

En la figura 6.1 se puede observar el grafo de la Dungeon generado en base a las entradas del cuadro 6.1, donde el jefe está posicionado en el vértice más lejano a la entrada. Es necesario en cuenta la diversidad de habitaciones en la representación 3D de esta Dungeon ya que al usar la matriz que organiza el banco de mallas 3D (figura 5.2), no solo los vértices con diferente grado son representados por mallas 3D distintas, sino que también vértices con grados iguales pueden llegar a tener mallas distintas entre sí. El cuadro 6.2 muestra la distribución de enemigos y recompensas en cada habitación de la Dungeon, mientras que la figura 6.1 muestra el grafo resultante y la figura 6.2 muestra la malla 3D que se utilizó para la primera habitación de la Dungeon.

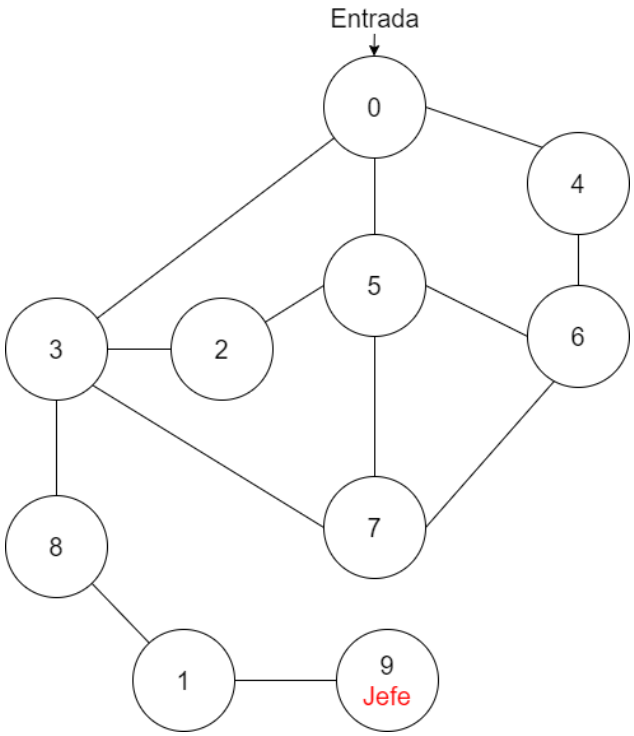


Figura 6.1: Grafo de la Dungeon

Habitación	Enemigos	Recompensa
0	2	9332
1	2	9332
2	2	9332
3	1	4666
4	1	4666
5	2	9332
6	1	4666
7	2	9332
8	2	9332
9	Jefe	30000

Cuadro 6.2: Atributos por Habitación

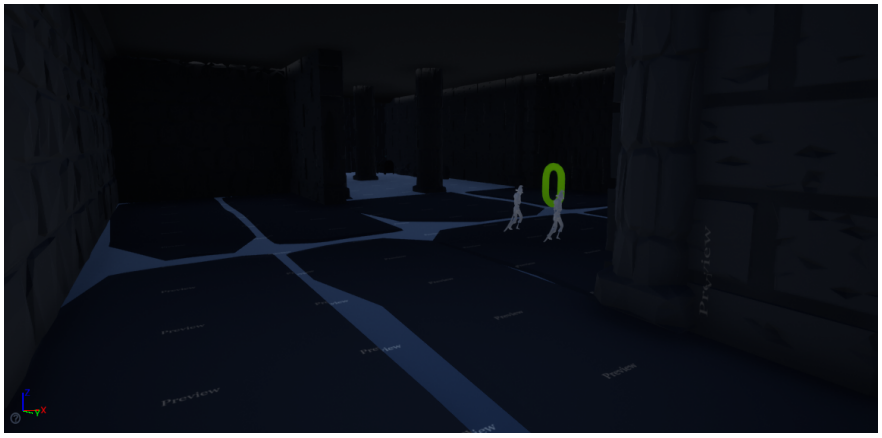


Figura 6.2: Primera habitación de la Dungeon

A continuación se presentan los resultados de una segunda generación de Dungeon a partir de las mismas entradas que recibió la generación anterior en el cuadro 6.1, mostrando que el generador es capaz de crear Dungeons completamente diferentes a través de los mismos parámetros de entrada. El cuadro 6.3 muestra la distribución de enemigos y recompensas en cada habitación de la Dungeon, mientras que la figura 6.3 muestra el grafo resultante y la figura 6.4 muestra la malla 3D que se utilizo para la primera habitación de la Dungeon, la cual es diferente a la malla 3D posicionada en la generación anterior (ver figura 6.2).

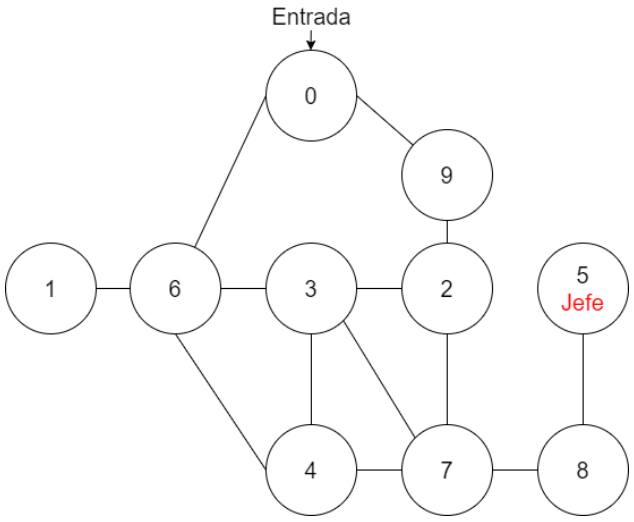


Figura 6.3: Grafo de la Dungeon

Habitación	Enemigos	Recompensa
0	1	4666
1	2	9332
2	2	9332
3	2	9332
4	1	4666
5	Jefe	30000
6	1	4666
7	2	9332
8	2	9332
9	2	9332

Cuadro 6.3: Atributos por Habitación

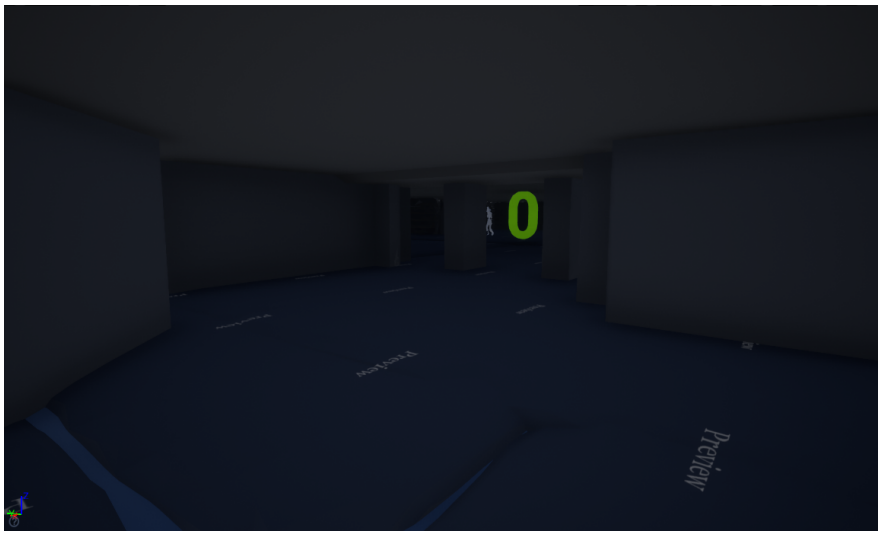


Figura 6.4: Primera habitación de la Dungeon

6.2. Segunda Prueba:

Esta prueba tiene como objetivo exponer las capacidades de generación y posicionamiento de enemigos de forma adaptativa, donde se ingresan parámetros para jugadores nuevos y poco experimentados, buscando generar una Dungeon fácil de jugar y poco laberíntica. Para esto se maneja una probabilidad P cercana a 0 (ver cuadro 6.4).Los resultados de esta prueba pueden apreciarse en la figura 6.5 la cual muestra el grafo resultante y el cuadro 6.5 el cual muestra la distribución de recompensas y enemigos en las habitaciones de la Dungeon.

Entrada	Valor
Longitud	10
P	0.2
Límite de Grado	5
Monto Total de Recompensas	100000
Monto Correspondiente al Jefe	0.30
Modelo de Dificultad	2
Bonificación de Recompensa	1
Dificultad	1
Multiplicador de Enemigos	1
Nivel Máximo	100
APL	1

Cuadro 6.4: Conjunto de entradas para la segunda prueba

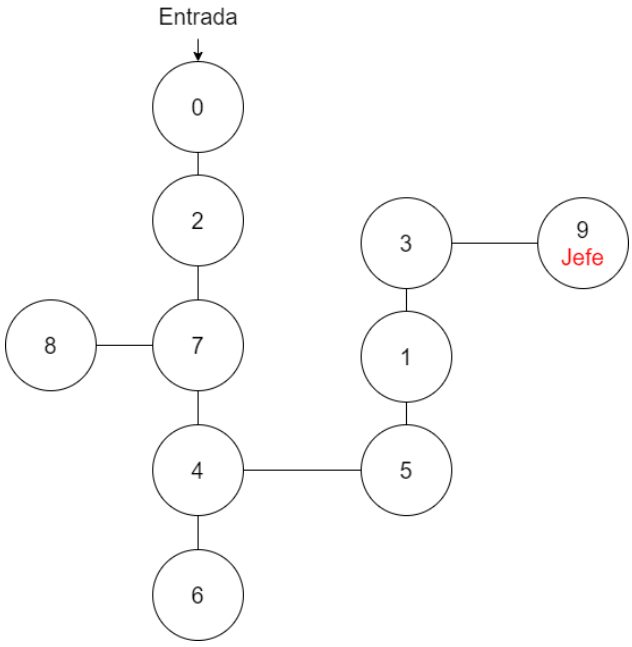


Figura 6.5: Grafo de la Dungeon

Habitación	Enemigos	Recompensa
0	2	7777
1	2	7777
2	2	7777
3	2	7777
4	2	7777
5	2	7777
6	2	7777
7	2	7777
8	2	7777
9	Jefe	30000

Cuadro 6.5: Atributos por Habitación

De la prueba anterior se puede observar que se genera una Dungeon simple y con una cantidad de enemigos constante para las habitaciones, esto debido a que los parámetros ingresados son para generar una Dungeon de baja dificultad para las primeras etapas de juego, por lo que la Dungeon tiende a ser lineal y los enemigos no representan un desafió para el jugador.

6.3. Tercera Prueba

Esta prueba tiene como objetivo exponer las capacidades de generación y posicionamiento de enemigos de forma adaptativa, donde se ingresan parámetros para jugadores experimentados y que están jugando en la dificultad de “Epico”, ya que se buscan generar una Dungeon desafiante y laberíntica, se maneja una probabilidad P cercana a 1 y un APL igual al nivel máximo del juego (ver cuadro 6.6). Los resultados de esta prueba pueden apreciarse en la figura 6.6 la cual muestra el grafo resultante y el cuadro 6.7 el cual muestra la distribución de recompensas y enemigos en las habitaciones de la Dungeon.

Entrada	Valor
Longitud	10
P	0.7
Límite de Grado	5
Monto Total de Recompensas	100000
Monto Correspondiente al Jefe	0.30
Modelo de Dificultad	2
Bonificación de Recompensa	1
Dificultad	5
Multiplicador de Enemigos	2
Nivel Máximo	100
APL	100

Cuadro 6.6: Conjunto de entradas para la segunda prueba

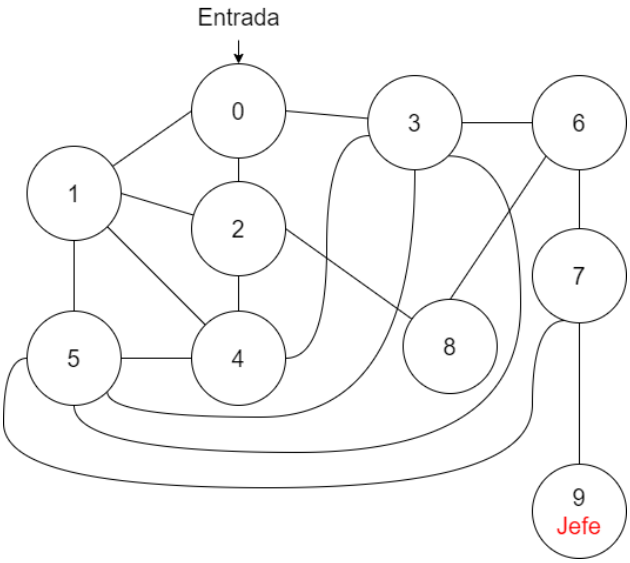


Figura 6.6: Grafo de la Dungeon

Habitación	Enemigos	Recompensa
0	1	28000
1	1	28000
2	0	0
3	1	28000
4	0	0
5	1	28000
6	0	0
7	1	28000
8	0	0
9	Jefe	30000

Cuadro 6.7: Atributos por Habitación

Se puede observar que se genera una Dungeon bastante compleja con muchas puertas y pasadizos los cuales le aportan dificultad al escenario. Adicionalmente como se trata de jugadores en el nivel máximo y jugando en una dificultad de épico, se espera que cada enemigo de la Dungeon represente un gran desafió, incluso en algunos casos, un desafió similar al jefe final. Asimismo, al tratarse de enemigos tan difíciles, el generador aplica la bonificación por dificultad, otorgando el doble de recompensas por cada enemigo. El desarrollador puede modificar la cantidad de enemigos en la Dungeon a través de la entrada de "Multiplicador de Enemigos", por lo que puede hacer que los enfrentamientos sean mucho más frecuentes o por el contrario, mucho más escasos. Esto depende de la noción de un enemigo de dificultad máxima de cada desarrollador.

Capítulo 7

Trabajo Futuro

7.1. Posicionamiento Avanzado de Elementos

El presente proyecto se enfocó en la lógica del cómo definir cuantos elementos hay que posicionar y en que habitaciones de la Dungeon posicionarlos, sin embargo el posicionamiento físico está delimitado al centro de la habitación y sus alrededores. Es posible realizar un posicionamiento avanzado de los enemigos y las recompensas según la geometría de la habitación, calculando cuales zonas de la habitación son aptas para el posicionamiento de los elementos, usando así todo el volumen de la habitación como posible zona de aparición y evitando así el posicionamiento fuera de la habitación y colisiones indeseadas con otros objetos como una pared, una columna o un actor.

La idea detrás de este algoritmo es crear una matriz tridimensional y extender un cubo sobre la malla de la habitación, donde cada posición de la matriz representa un sector del cubo. Posteriormente a través de las herramientas de detección de colisiones de Unreal Engine se marca en la matriz todos los sectores donde existe una colisión entre el cubo y la malla 3D de la habitación. Después de este proceso, los sectores de la matriz sin marcar serán las posiciones posibles en las que se puede posicionar un enemigo o una recompensa.

7.2. Uso de Teletransportadores con Cámara

Debido al tamaño del proyecto, se optó por crear un prototipo funcional de los actores teletransportadores con cámara al punto de destino, mencionados en la sección de Actor con Cámara al Destino en el capítulo de implementación 5.4. Sin embargo en la implementación final del proyecto se hizo uso de los Actor con Pantalla de Carga que cumplen con la misma función de conectar las habitaciones de la Dungeon. Aunque ambos actores de teletransportacion cumplen su función, los actores con cámara amplían las posibilidades de generación. Esto debido a que no necesitan de una malla 3D que indique que están ahí, solo necesitan el plano sin colisiones que proyecta el punto de destino, por lo que con ellos se pueden generar Dungeons con arquitecturas donde las puertas no sean objetos comunes, como lo son pantanos, bosques o cuevas.

Capítulo 8

Conclusiones

El presente proyecto logra crear un generador de escenarios de tipo Dungeons para vídeo juegos del genero RPG (Rol-Playin Game), capaz de definir aspectos como la topología y las mecánicas de recompensas y posicionamiento de enemigos. El generador es lo suficientemente flexible como para permitir la generación de Dungeons según las necesidades de los diferentes vídeo juegos y equipos de desarrollo, siendo capaz de generar tanto Dungeons simples y especiales para los jugadores nuevos en zonas iniciales del juego, como Dungeons laberínticas y difíciles que le significaran un desafió hasta para los jugadores más experimentados.

Este proyecto logra establece la topología y arquitectura de la Dungeon a partir de un grafo aleatorio creado con el algoritmo de $Erdos - Renyi$. Así como logra definir una forma balanceada de posicionar recompensas y enemigos en base las propiedades del grafo aleatorio y a modelos similares establecidos para el manejo de tesoros y amenazas en el juego "Dungeons & Dragons", uno de los padres de los juegos de rol y aventura de la actualidad. Adicionalmente, a partir del uso de las herramientas de desarrollo del motor de vídeo juegos Unreal Engine 4, se logra la construcción jugadle de una Dungeon en un mundo virtual 3D.

Debido a que este trabajo ayuda a automatizar la generación de Dungeons de diferentes tamaños y características, el costo de desarrollo por cada Dungeon se vería disminuido y permitiría a los desarrolladores implementar Dungeons complejas y entretenidas con una mayor frecuencia dentro de su vídeo juegos. Por lo que no solo ayudaría a disminuir los costos en el desarrollo del vídeo juego, sino que también le otorgaría a los jugadores experiencias mucho más completas y dinámicas. Así mismo, al poder crear varias topologías y definir varias formas de distribuir los enemigos y recompensas para una misma Dungeon, se le ofrece a los jugadores la oportunidad volver a jugar una Dungeon como si fuera la primera vez, aumentando el nivel de diversión y por ende la cantidad de horas invertidas en el juego.

Bibliografía

[1] J. Brown, "The importance of everything: Analytics of map design." https://www.youtube.com/watch?v=Ij2nmt3EuJY&ab_channel=GDC, 2014.

[2] D. Cox, "Interior design and environment art: Mastering space, mastering place."

[3] D. S. J. Hodgson and S. Stratton, *The Elder Scrolls Skyrim Official Strategy Guide*. PRIMA Games, 1st ed., 2011.

[4] D. J. Hodgson, *The Witcher 3: Wild Hunt: Prima Official Game Guide*. PRIMA Games, 1st ed., 2015.

[5] K. Boudreau, *Game Level Design fernando emmanuel Related papers Bet ween Play and Design: T he Emergence of Hybrid-Ident it y in Single-Player Videogames*. Course Technology Press, 1st ed., 2005.

[6] P. Sampaio, A. Baffa, B. Feijó, F. Feijó, and M. Lana, "A fast approach for automatic generation of populated maps with seed and difficulty control," 2017.

[7] A. Gellel and P. Sweetser, "A hybrid approach to procedural generation of roguelike video game levels," Association for Computing Machinery, 9 2020.

[8] IMDb, "The elder scrolls v: Skyrim awards."

[9] W. Witkowski, "Videogames are a bigger industry than movies and north american sports combined, thanks to the pandemic," 1 2021.

[10] M. Yamagiwa, "Bloodborne chalice dungeons explained." https://www.youtube.com/watch?v=VrOPyuQh_MM&ab_channel=IGN, 2015.

[11] B. Byrne, *BloodBorne Collector's Edition Guide*. Future Press, 1st ed., 2015.

[12] D. Adams and M. Mendler, "Automatic generation of dungeons for computer games," 2002.

[13] N. A. Barriga, "A short introduction to procedural content generation algorithms forvideogames," 2018.

[14] R. V. D. Linden, R. Lopes, and R. Bidarra, "Procedural generation of dungeons," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, pp. 78–89, 2014.

[15] L. Chaimowicz and Y. P. A. Macedo, "Improving procedural 2d map generation based on multi-layered cellular automata and hilbert curves," 2017.

[16] S. Snodgrass and S. Ontã, "A hierarchical mdmc approach to 2d video game map generation," 2017.

[17] O. Nepožitek and J. Gemrot, "Fast configurable tile-based dungeon level generator," 2018.

[18] M. Mearls, J. Crawford, C. Perkins, J. Wyatt, R. J. Schwalb, R. Thompson, P. Lee, S. F. Gray, M. Carter, C. Sims, J. C. Wilkes, G. Bilsland, and I. W. of the Coast, *Dungeon master's guide.* 2014.

[19] M. Csikszentmihalyi, "Te psychology of optimal experice," *Harper Row*, 2009.

[20] M. Zohaib, "Dynamic difficulty adjustment (dda) in computer games: A review," 2018.

[21] D. Soham, "What is video game difficulty and how does it work?," 2021.

[22] D. Perez, *Beginning RPG Maker MV.* Apress, 2016.

[23] E. Christopoulou and S. Xinogalos, "Overview and comparative analysis of game engines for desktop and mobile devices," *International Journal of Serious Games*, vol. 4, 12 2017.

[24] E. Games, "Unreal engine."

[25] L. G. Rooney, "Random graph models and matchings," 2019.

[26] S. Nobari, X. Lu, P. Karras, and S. Bressan, "Fast random graph generation," 2011.

[27] J. Erickson, *Algorithms.* 1st ed., 2019.