Universidad Camilo José Cela

Máster en Computación Gráfica, Realidad Virtual y Simulación

Trabajo Fin de Máster

Herramienta de generación de terreno procedural para Unity

Autor: Rafael Polope Contreras

Tutor: Álvaro San Juan Cervera

Septiembre 2023

Resumen:

Este trabajo presenta una herramienta de generación de terrenos procedurales para videojuegos. Utiliza diferentes tipos de ruido para crear terrenos variados y realistas, aplicando algoritmos de erosión para mejorar la apariencia. Además, asigna colores según la altitud del terreno. La herramienta aprovecha la paralelización con Job System y utiliza Niveles de Detalle (LOD) para optimizar el rendimiento del juego.

Uno de los objetivos principales del proyecto es crear terrenos continuos, abordando el problema de las inconsistencias entre las diferentes partes del terreno generado. Además, busca superar los desafíos de rendimiento en tiempo real en la generación de terreno.

Palabras Clave: Generación Procedural, Ruido, Erosión, Colores por Altura, Job System, LOD.

Resumen:

This work presents a procedural terrain generation tool for video games. It uses different types of noise to create diverse and realistic terrains, applying erosion algorithms to enhance their appearance. Additionally, it assigns colors based on the terrain's elevation. The tool leverages parallelization through Job System and utilizes Levels of Detail (LOD) techniques to optimize game performance.

One of the primary goals of the project is to create seamless terrains, addressing the issue of inconsistencies between different terrain parts generated. Furthermore, it aims to overcome real-time performance challenges in terrain generation.

Keywords: Procedural Generation, Noise, Erosion, Color by Height, Job System, LOD.

Agradecimientos:

A Álvaro San Juan, mi tutor en este proyecto, por ofrecerme la oportunidad de desarrollar este proyecto final a pesar de las dificultades presentadas para llevarlo a cabo en último momento.

A todo el equipo de RRHH de U-TAD y de Lãberit, la empresa en la que he estado trabajando durante el curso de este Máster, por haber hecho lo posible para llevar a cabo las prácticas de manera tan entregada y cercana.

A mi familia por soportarme este año especialmente duro, tanto para ellos como para mí, y a mis amigos por tener paciencia y entender el esfuerzo que me ha supuesto la decisión de llevar a cabo este máster y haber estado tan ausente.

Índice general

I. GUÍA DE LA MEMORIA

Este es el prefacio de la memoria. A continuación, se presenta una breve descripción de los capítulos que componen esta memoria:

Capítulo 1. MARCO TEÓRICO. INTRODUCCIÓN. En este capítulo se realiza una introducción a la motivación por la cual se ha llevado a cabo este proyecto, las técnicas más comunes que se utilizan para la generación procedural, el contexto en el que se desarrolla e importancia de este.

Capítulo 2. PLANTEAMIENTO DEL PROBLEMA. En este capítulo se indican los obstáculos que plantea la generación procedural, el manejo de la memoria, de la optimización temporal, la continuidad del terreno, así como el realismo de los resultados.

Capítulo 3. ESTADO DEL ARTE. En este capítulo se analiza el estado del arte de soluciones de generación procedural con objetivos similares a la propuesta de este proyecto tras realizar una revisión bibliográfica.

Capítulo 4. MARCO METODOLÓGICO. MATERIAL Y MÉTODOS. En este capítulo se detalla el diseño y el procedimiento de la herramienta que se desarrolla en este proyecto que permite compararla herramienta frente a otras ya existentes.

Capítulo 5. DESARROLLO. En este capítulo se detalla el desarrollo software de la herramienta, el diseño y la manera en la que se implementa en el motor Unity para ser usado por desarrolladores.

Capítulo 6. RESULTADOS Y DISCUSIÓN. En este capítulo se recogen todos los resultados de las pruebas establecidas para la evaluación de los algoritmos que crean los terrenos de la herramienta. Se describe el conjunto de parámetros empleados y las métricas establecidas para la evaluación de la variable de interés y para la validación de los resultados, tanto visuales como de rendimiento. Se analizan los resultados obtenidos y se identifican los factores y limitaciones que han podido influir en ellos.

Capítulo 7. CONCLUSIONES Y LÍNEAS FUTURAS. Se recogen las conclusiones extraídas a partir del diseño, desarrollo y experimentación del método propuesto para la elaboración de la herramienta. Se describen futuras líneas de trabajo que complementen el desarrollo alcanzado y se indica la aplicación de este proyecto en el ámbito del desarrollo de contenido multimedia.

II. MEMORIA

Capítulo 1

Introducción

1.1. Introducción

La industria de los videojuegos ha sido testigo de una transformación excepcional en los últimos años. La búsqueda constante de experiencias más inmersivas y visualmente impresionantes ha llevado a los desarrolladores a explorar nuevas formas de crear mundos virtuales. En este contexto, la generación procedural de terrenos se ha convertido en una herramienta esencial para satisfacer las demandas de los jugadores cada vez más exigentes y hambrientos de autenticidad.

Este Trabajo de Fin de Máster (TFM) se adentra en el emocionante campo de la generación procedural de terrenos en Unity, uno de los motores de desarrollo de videojuegos más utilizados tanto por la industria como por desarrolladores independientes. El objetivo principal de este proyecto es diseñar y desarrollar una herramienta avanzada que permita a los creadores de juegos generar terrenos de manera eficiente y convincente.

Los datos disponibles, incluidos los proporcionados por fuentes como Eurostat y otros informes, indican un crecimiento constante en la industria de los videojuegos. A medida que la audiencia se expande, también lo hacen sus expectativas. Los jugadores contemporáneos buscan experiencias de juego que sean únicas y estimulantes. En este contexto, la generación procedural de terrenos se plantea como una solución que puede hacer posible la creación de mundos tanto auténticos como sorprendentes.

La historia de la generación procedural de terrenos comenzó con la implementación de algoritmos basados en ruido, entre los que destaca el renombrado Perlin Noise, desarrollado por Ken Perlin en la década de 1980. A pesar de que estos enfoques ofrecieron resultados impresionantes para su época, a menudo carecían de la autenticidad y el realismo que los jugadores modernos esperan en la actualidad. Con el aumento de la demanda de experiencias de juego más inmersivas y visualmente impactantes, se volvió crucial mejorar las técnicas de generación de terrenos.

Un avance destacado en este ámbito fue la introducción del Simplex Noise, una mejora significativa respecto al Perlin Noise, que mantuvo la capacidad de generar terrenos naturales, pero con un rendimiento más eficiente. No obstante, lo que realmente marcó un hito en la generación procedural de terrenos fue la incorporación de algoritmos de erosión. Estos algoritmos simulan procesos geológicos y climáticos, lo que resulta en terrenos con características naturales más convincentes, como montañas, cañones y ríos. Esta adición permitió crear mundos virtuales más realistas y creíbles, lo que contribuyó a elevar la

calidad general de los juegos.

La creciente popularidad de los mundos abiertos en los videojuegos presentó un desafío significativo: generar terrenos continuos y sin interrupciones notables cuando los jugadores exploran los límites del mundo virtual. A medida que los algoritmos mejoraron, los juegos pudieron ofrecer experiencias de juego más fluidas y expansivas, lo que aumentó la inmersión del jugador y su sensación de exploración sin restricciones.

La influencia de la generación procedural de terrenos no se limita únicamente a la creación de paisajes virtuales. Ha dejado una huella indeleble en otros aspectos de la generación procedural en la industria de los videojuegos. Esto incluye la generación de ciudades completas en juegos de mundo abierto, la creación de misiones y contenido diverso que aumenta la rejugabilidad, así como la generación de personajes y criaturas que dan vida a los mundos virtuales. Además, ha influido en la generación de texturas y elementos artísticos, permitiendo un ahorro significativo de tiempo y recursos en el desarrollo de juegos y animaciones. La música y los efectos de sonido también se benefician de la generación procedural, adaptando la banda sonora y la atmósfera del juego en tiempo real para acompañar la acción.

En la última década, hemos sido testigos de un aumento significativo en la adopción de la generación procedural de terrenos en la industria de los videojuegos. Esto marca un cambio notable en la forma en que los desarrolladores abordan la creación de entornos virtuales. Esta adopción creciente se ha visto impulsada por varios factores clave, como la capacidad de generar mundos virtualmente infinitos sin incurrir en costos prohibitivos de almacenamiento. Además, proporciona una experiencia de juego única en cada partida, lo que aumenta la rejugabilidad y la longevidad de un título. Esta estrategia también es esencial en la creación de mundos abiertos sin pantallas de carga notables entre escenarios o biomas, lo que mejora la inmersión del jugador.

La evolución de la generación procedural de terrenos ha estado intrínsecamente ligada a la constante innovación en algoritmos y técnicas. En los últimos años, se han desarrollado y refinado una variedad de enfoques que van desde la generación basada en ruido, como el Perlin Noise y el Simplex Noise, hasta técnicas más avanzadas que incorporan algoritmos de erosión y simulaciones físicas para lograr terrenos aún más realistas.

Uno de los desafíos cruciales en la generación procedural de terrenos, especialmente en entornos de mundo abierto, es la gestión eficiente de los çhunks"de terreno, pequeñas porciones de terreno que se generan y almacenan en memoria para luego ser cargadas y descargadas dinámicamente mientras el jugador se mueve a través del mundo virtual. Este proceso es esencial para mantener un rendimiento óptimo y evitar la carga excesiva de recursos en la memoria del sistema. Además, es fundamental asegurarse de que la transición entre diferentes chunks sea fluida y sin discontinuidades notorias, lo que garantiza una experiencia de juego inmersiva. En el contexto de la generación procedural de terrenos, se ha consolidado una técnica comúnmente empleada denominada "sistema de streaming"de terrenos. Este enfoque se encarga de gestionar la carga y descarga de fragmentos o çhunks"de terreno de manera dinámica en función de diversos factores, como la proximidad del jugador, la dirección de su movimiento y su campo de visión. Uno de los principales objetivos de este sistema es garantizar una transición fluida entre chunks adyacentes, evitando discontinuidades notorias que puedan afectar negativamente a la cohesión y la inmersión en el mundo virtual.

Además, se emplean algoritmos de detección de colisiones y de visibilidad para determinar qué chunks deben ser cargados y renderizados según la posición actual de la cámara

del jugador. Estos algoritmos son cruciales para optimizar el rendimiento del juego, ya que permiten reducir la carga en la unidad central de procesamiento (CPU) y la unidad de procesamiento gráfico (GPU). En consecuencia, solo se procesan y muestran los chunks que son relevantes y visibles desde la perspectiva del jugador en ese momento particular. Esta estrategia de optimización contribuye significativamente a lograr una experiencia de juego fluida y eficiente en términos de recursos.

En lo que respecta al movimiento de la cámara del jugador y su influencia en la generación del terreno, se implementan técnicas de "nivel de detalle"(LOD, por sus siglas en inglés) dinámico. Esta técnica se encarga de ajustar la resolución y el nivel de detalle del terreno en tiempo real en función de la distancia entre la cámara y el terreno circundante. Cuando la cámara se acerca a un chunk de terreno en particular, se aumenta el nivel de detalle, lo que implica mostrar texturas de mayor calidad y una geometría más detallada. Por otro lado, cuando la cámara se aleja, se reduce el nivel de detalle para conservar recursos de hardware y mantener un rendimiento óptimo.

En este contexto, el uso del Job System de Unity ha ganado relevancia. Permite la optimización de procesos intensivos en CPU, como la modificación de mallas de terreno, lo que resulta en una mejora significativa del rendimiento en juegos que implementan generación procedural de terrenos.

1.2. Motivación

El desarrollo de la herramienta de generación procedural de terrenos en Unity se enmarca en un contexto dinámico y desafiante, impulsado por diversas motivaciones que abordan necesidades y aspiraciones cruciales.

La industria de los videojuegos se encuentra en constante evolución, y la búsqueda incesante de experiencias más inmersivas y visualmente impactantes impulsa a los desarrolladores a explorar nuevas formas de crear mundos virtuales. La generación procedural de terrenos se erige como una respuesta a esta demanda creciente. Esta técnica no solo permite satisfacer las expectativas de los jugadores modernos en términos de autenticidad y sorpresa, sino que también agiliza el proceso de desarrollo de videojuegos al proporcionar herramientas eficientes para crear entornos expansivos y convincentes.

La gestión eficiente de chunks de terreno, la optimización del "Job System"de Unity y la elección de algoritmos adecuados son áreas que requieren innovación y soluciones creativas. Esta necesidad de abordar y superar desafíos técnicos en el campo de la generación procedural de terrenos motiva el desarrollo de esta herramienta. La oportunidad de enfrentar estos desafíos y crear una herramienta que marque la diferencia en la industria es una fuerza impulsora clave.

Este proyecto se enmarca en el contexto de una industria de videojuegos en constante cambio y una demanda creciente de experiencias inmersivas. La búsqueda de soluciones técnicas innovadoras y la aspiración de contribuir al crecimiento de la comunidad de desarrolladores son las motivaciones que guían este esfuerzo. La generación procedural de terrenos en Unity tiene el potencial de transformar la forma en que se crean mundos virtuales, y este proyecto se esfuerza por lograr precisamente eso.

Capítulo 2

Planteamiento del problema

2.1. Problemática en la Generación Procedural de Terreno

Este proyecto se enfrenta a varios desafíos críticos en la generación procedural de terrenos en Unity, que requieren soluciones efectivas para mejorar la experiencia del jugador y la eficiencia en el desarrollo de videojuegos.

Uno de los desafíos clave es la gestión eficiente de chunks de terreno en memoria, asegurando una transición suave entre ellos según la proximidad y el movimiento de la cámara del jugador. Esto implica la implementación de un sistema de "streaming"de terrenos que carga y descarga chunks dinámicamente, aplicando técnicas de interpolación y suavización en las fronteras de los chunks para una transición visual coherente.

La integración adecuada del "Job System"de Unity es otro desafío esencial. Este sistema promete mejoras en el rendimiento, pero su coordinación con otros sistemas del juego y la gestión de trabajos en paralelo son aspectos críticos que deben abordarse con precisión.

La optimización de los algoritmos utilizados es fundamental. Se debe garantizar que la generación de terreno sea eficiente en términos de uso de recursos de hardware y tiempo de procesamiento, seleccionando algoritmos apropiados y optimizando su implementación.

Además, la formación de desarrolladores en las técnicas de generación de terreno es un desafío relevante. La creación de recursos educativos y herramientas de aprendizaje ayudará a los desarrolladores a aplicar eficazmente estas técnicas en sus proyectos.

Los objetivos clave del proyecto incluyen la integración efectiva de efectos en las escenas, su versatilidad de uso, facilidad de uso y eficiencia en términos de consumo de memoria y rendimiento, especialmente en algoritmos que pueden ser intensivos en recursos.

Capítulo 3

Objetivos

3.1. Objetivos Generales y Específicos

Este proyecto de generación procedural de terrenos en Unity se enfrenta al desafío de diseñar y desarrollar una herramienta innovadora que aborde las necesidades de la industria de los videojuegos. Los objetivos generales y específicos se centran en la creación de una solución tecnológica efectiva y en la evaluación de su impacto en el proceso de desarrollo de videojuegos. Aquí se desarrollan los objetivos que se deben lograr para llevar a cabo la herramienta conforme se desea:

1. **Diseñar y Desarrollar de la Herramienta:** El objetivo fundamental es crear una solución tecnológica efectiva que permita a los desarrolladores de videojuegos generar terrenos de manera eficiente y convincente. Esta herramienta debe ser capaz de abordar los desafíos técnicos de la generación procedural de terrenos y proporcionar una interfaz intuitiva para los usuarios.

2. **Analizar el Impacto y la Eficiencia de la Herramienta:** Más allá de su desarrollo, es esencial evaluar el aporte real de esta herramienta en el proceso de creación de videojuegos. Esto implica medir la capacidad de la herramienta para optimizar la generación de terrenos y su influencia en la calidad de los juegos resultantes.

    A) **Investigación y Selección de Algoritmos y Técnicas:** Se realizará una investigación exhaustiva para analizar y seleccionar los algoritmos y técnicas más adecuados para la generación procedural de terrenos en Unity.

    B) **Desarrollo de la Interfaz y Experiencia de Usuario:** El diseño de la interfaz de usuario y la experiencia de usuario son elementos cruciales para la herramienta. Se aplicará una estrategia de diseño centrada en el usuario.

    C) **Optimización del Rendimiento:** La generación procedural de terrenos puede ser intensiva en términos de rendimiento. Se establecerán estrategias de optimización para garantizar que la herramienta funcione de manera eficiente en diferentes entornos de desarrollo.

    D) **Evaluación de la Herramienta en un Entorno Real:** Se llevará a cabo un piloto experimental para evaluar la eficacia y eficiencia de la herramienta en un contexto de desarrollo de videojuegos.

Capítulo 4

Estado del arte

4.1. Introducción

La generación procedural de terrenos en la industria de los videojuegos es un campo en constante evolución, impulsado por la creciente demanda de experiencias de juego más inmersivas y visualmente impactantes. En este estado del arte, exploramos a fondo esta técnica, centrándonos en su aplicación en el motor de desarrollo de videojuegos Unity.

La generación procedural de terrenos se ha convertido en una herramienta esencial para los creadores de videojuegos, ya que les permite diseñar mundos expansivos y auténticos que satisfacen las demandas de jugadores cada vez más exigentes. En este proyecto, nos centraremos en el conjunto de técnicas, algoritmos y herramientas que han impulsado campo en la última década. A continuación, se hará un desglose de los puntos en los que consistirá el estado del arte de este proyecto:

4.1.1. Contextualización

Para comprender la importancia de la generación procedural de terrenos, es fundamental contextualizar su evolución a lo largo del tiempo. Comenzando por una breve mirada a su historia, examinamos cómo esta disciplina ha avanzado desde sus modestos comienzos hasta convertirse en un elemento clave en la creación de mundos virtuales de alta calidad.

4.1.2. Técnicas y Algoritmos

Un aspecto fundamental en el estado actual de la generación procedural de terrenos son las técnicas y algoritmos que sustentan su funcionamiento. Exploramos en detalle algunos de los enfoques más influyentes y ampliamente utilizados, desde los algoritmos basados en ruido como el Perlin Noise hasta técnicas más avanzadas que incorporan simulaciones físicas y procesos geológicos para lograr terrenos extremadamente realistas.

4.1.3. Herramientas y Recursos en Unity

Unity, como uno de los motores de desarrollo de videojuegos más populares en la industria, proporciona a los desarrolladores una serie de herramientas y recursos específicos para la generación procedural de terrenos. En esta revisión, examinamos cómo Unity

facilita la creación de terrenos de manera eficiente y convincente, y destacamos algunos de los plugins y extensiones más notables que simplifican aún más este proceso.

4.1.4. Antecedentes

En esta sección, proporcionaremos una breve descripción de los antecedentes relacionados con la generación procedural de terrenos en Unity. exploraremos en detalle las herramientas y recursos disponibles en Unity para la generación procedural de terrenos. Analizaremos diversas soluciones, como plugins, assets y técnicas específicas que facilitan la creación y manipulación de terrenos dentro de la plataforma Unity.

4.1.5. Retos y Futuras Tendencias

A medida que la generación procedural de terrenos se ha convertido en una práctica común, también ha enfrentado desafíos significativos. Exploraremos las dificultades más comunes, como la gestión de chunks de terreno en memoria, la transición fluida entre terrenos generados y los problemas asociados con la dirección y el movimiento de la cámara en juegos de mundo abierto. Además, consideraremos las futuras tendencias y avances que podrían dar forma al desarrollo de esta disciplina en los años venideros.

4.2. Definición del Tema

La generación procedural de terrenos es una disciplina informática que ha desempeñado un papel fundamental en la creación de mundos virtuales, especialmente en la industria de los videojuegos. En esencia, se refiere a la creación automática y algorítmica de entornos de terreno en mundos virtuales.

En el corazón de la generación procedural de terrenos se encuentran los algoritmos matemáticos y computacionales. Estos algoritmos se basan en principios como el ruido, la interpolación y la simulación de procesos naturales para crear terrenos realistas y convincentes.

La generación de terrenos se lleva a cabo mediante la manipulación de datos y la aplicación de fórmulas matemáticas para determinar las alturas y texturas de cada punto del terreno.

La generación procedural de terrenos desempeña un papel crucial en la industria de los videojuegos, donde se utiliza para crear mundos expansivos y auténticos. Los juegos de mundo abierto, en particular, se benefician enormemente de esta técnica, ya que les permite ofrecer vastos entornos sin pantallas de carga notables, lo que mejora la inmersión del jugador.

Pero su influencia se extiende más allá de los videojuegos. Se utiliza en aplicaciones de simulación, como la formación de pilotos, la cartografía digital y la visualización arquitectónica, donde la creación de entornos realistas es esencial.

A medida que la tecnología avanza, los enfoques modernos de generación procedural de terrenos han incorporado algoritmos más avanzados, como los que simulan procesos geológicos y climáticos. Además, se están explorando técnicas que permiten una mayor interacción y personalización de los terrenos por parte de los jugadores.

En resumen, la generación procedural de terrenos es una disciplina que utiliza algoritmos para crear paisajes virtuales de manera automática y coherente. Con un gran impacto en la industria de los videojuegos y se ha expandido a otros campos, a la vez que continúa evolucionando con el avance de la tecnología.

4.3. Historia y Evolución de la Generación Procedural de Terrenos

La generación procedural de terrenos es una técnica que ha desempeñado un papel fundamental en la industria de los videojuegos y otros campos. Su ventaja principal radica en la capacidad de crear mundos y contenidos de manera dinámica, sin necesidad de almacenar grandes cantidades de datos en disco duro. A lo largo de los años, esta técnica ha evolucionado significativamente, impulsada por avances en el hardware y la demanda de mundos más expansivos y realistas.

4.3.1. Década de los 60

Según los resultados de la búsqueda, la generación procedural de terrenos en gráficos por computadora comenzó a aparecer a mediados de la década de 1960

En ese momento, los gráficos por computadora se utilizaban principalmente para fines científicos, de ingeniería e investigativos, pero comenzaron a surgir experimentos artísticos [1]

Las técnicas utilizadas para la generación procedural de terrenos entran en una amplia categoría llamada generación procedural, lo que significa generar algunos objetos o valores específicos mediante un algoritmo [2]

Uno de los primeros ejemplos de generación procedural en gráficos por computadora fue la transformación de declaraciones matemáticas en vectores de herramientas de máquinas en 3D generados por computadora por Douglas T. Ross en 1959[3]

Sin embargo, no fue hasta mediados de la década de 1960 que comenzaron a aparecer experimentos artísticos, especialmente por parte del Dr. Thomas Calvert

En términos de generación de terrenos específicamente, un artículo de G.S. Miller titulado "La definición y representación de mapas de terreno"se presentó en la 13ª Conferencia Anual sobre Gráficos por Computadora y Técnicas Interactivas en 1986 [4]

El artículo discutió el uso de algoritmos fractales para la generación y representación de terrenos. La generación de terrenos procedurales en tiempo real también se discutió en un artículo de 2004 de J. Olsen titulado "Generación de Terrenos Procedurales en Tiempo Real"

4.3.2. Década de los 80

En la década de 1980, la generación procedural de terrenos en gráficos por computadora continuó evolucionando, impulsada por avances en tecnología y la creciente popularidad de los videojuegos. Aunque la información específica sobre esta década es limitada en los resultados de búsqueda, podemos inferir algunos desarrollos basados en el progreso

general de la generación procedural durante este período.

Uno de los primeros ejemplos de generación procedural en videojuegos se puede rastrear hasta el género de juegos de rol de mesa (RPG). Advanced Dungeons & Dragons, el sistema de juego de mesa líder en ese momento, proporcionaba formas para que el "maestro de mazmorras"generara mazmorras y terrenos utilizando tiradas de dados aleatorias y tablas procedimentales de ramificación complejas

Este concepto luego se adaptó a los juegos de computadora, con Strategic Simulations lanzando el Dungeon Master's Assistant, un programa que generaba mazmorras basadas en las tablas publicadas [3]

En el contexto de gráficos por computadora y videojuegos, la generación procedural de terrenos se convirtió en una herramienta valiosa para crear paisajes realistas y diversos. Este enfoque fue particularmente útil para los juegos de mundo abierto que requerían entornos vastos y detallados. El uso de algoritmos para generar terrenos permitió a los desarrolladores reducir la cantidad de trabajo manual y crear paisajes que parecían infinitos en tamaño [4]

Uno de los primeros métodos para la generación procedural de terrenos fue el algoritmo de diamante-cuadrado, una técnica de modelado fractal simple

Este algoritmo permitía la generación de modelos de terreno altamente detallados al subdividir iterativamente un cuadrado y ajustar los valores de altura en cada paso [3]

En la década de 1980, dadp el progreso general en gráficos por computadora y el creciente interés en la generación procedural, produjeron durante este período avances en este campo.

4.3.3. Década de los 2000

En la década de 2000, la generación procedural de terrenos en gráficos por computadora continuó avanzando, impulsada por la creciente popularidad de los videojuegos y la necesidad de entornos más realistas y diversos. El uso de la generación procedural en videojuegos se volvió más extendido durante esta década, con muchos juegos generando aspectos del entorno o personajes no jugadores de manera procedural durante el proceso de desarrollo para ahorrar tiempo en la creación de assets [5]

Uno de los avances destacados en la generación procedural de terrenos durante esta década fue el desarrollo de nuevos algoritmos y técnicas para generar terrenos. Por ejemplo, un nuevo método para la generación procedural de terrenos se presentó en un artículo de 2015 escrito por Christian Schulte titulado .ª Graph-Based Approach to Procedural Terrain"[6]. El artículo describe un proceso de tres pasos para generar terrenos utilizando un enfoque basado en grafos.

Otro desarrollo notable durante esta década fue el uso de la generación procedural como una mecánica de juego, como la creación de nuevos entornos para que el jugador explore. Por ejemplo, los niveles en el juego Spelunky se generan de manera procedural al reorganizar mosaicos prefabricados de geometría en un nivel con una entrada, una salida, un camino resoluble entre los dos y obstáculos en ese camino.

En resumen, la década de 2000 vio un progreso continuo en la generación procedural de terrenos en gráficos por computadora, con avances en algoritmos y técnicas, así como el aumento en el uso de la generación procedural como una mecánica de juego.

4.3.4. Década de los 2010

En la década de 2010, la generación procedural de terrenos en gráficos por computadora continuó avanzando, con el desarrollo de nuevos algoritmos y técnicas para la generación de terrenos. El uso de la generación procedural en videojuegos también siguió creciendo, con muchos juegos utilizando la generación procedural para crear entornos vastos y diversos.

Un ejemplo de un nuevo algoritmo para la generación procedural de terrenos en la década de 2010 es el algoritmo adaptativo y de generación procedural de terrenos con modelos de difusión y ruido de Perlin propuesto en un artículo de 2021 por Zhang et al. [7]. Este algoritmo utiliza modelos generativos basados en difusión para crear terrenos con múltiples niveles de detalle, lo que permite una generación más eficiente de entornos a gran escala.

Otro ejemplo de generación procedural de terrenos en la década de 2010 es la generación procedural de carreteras, propuesta en un artículo de 2010 por Galin [8] . Este método utiliza un algoritmo de ruta más corta anisotrópica ponderada para generar carreteras automáticamente, lo que permite la creación más eficiente de redes de carreteras en entornos a gran escala.

En cuanto a los gráficos, los resultados de la búsqueda incluyen una publicación de blog de Filip Joel sobre la generación procedural de terrenos utilizando Unity3D [9]. El artículo describe un proyecto que utiliza generación de malla y ruido, así como simulación de erosión hidráulica, para crear terrenos realistas en tiempo real.

En resumen, la década de 2010 vio un progreso continuo en la generación procedural de terrenos en gráficos por computadora, con el desarrollo de nuevos algoritmos y técnicas, y el aumento en el uso de la generación procedural en videojuegos para crear entornos vastos y diversos.

4.3.5. El Futuro de la Generación Procedural

El futuro de la generación procedural de terrenos en gráficos por computadora probablemente estará moldeado por avances en redes neuronales, un mayor uso de la generación de assets procedural y la integración de datos del mundo real.

Las redes neuronales tienen el potencial de mejorar el realismo y la complejidad de los terrenos generados de manera procedural. Al utilizar técnicas de transferencia de estilo, los desarrolladores pueden crear formas generales y permitir que la red neuronal agregue detalles que parecen realistas [10]. Este enfoque puede ayudar a superar las limitaciones de simular todos los procesos que crean terrenos reales, como la tectónica de placas y la erosión.

La generación de assets procedural, que utiliza algoritmos para crear automáticamente assets como modelos 3D y texturas, también puede desempeñar un papel significativo en el futuro de la generación de terrenos [11]. Al combinar diversas técnicas de generación procedural sintética con modelos digitales de elevación (DEM) y datos del mundo real, los desarrolladores pueden crear paisajes multibioma con mayor precisión y atractivo visual [12].

La integración de datos del mundo real, como imágenes de satélite y mapas topográficos, puede mejorar aún más el realismo y la precisión de los terrenos generados de manera procedural. Al combinar estas fuentes de datos con técnicas de generación procedural, los

desarrolladores pueden crear entornos más inmersivos y detallados.

En cuanto a hardware y rendimiento, ya se ha explorado el uso de GPU para generar terrenos procedurales complejos a velocidades de fotogramas interactivas [13]. A medida que la tecnología continúa avanzando, podemos esperar más optimizaciones y mejoras en la eficiencia de los algoritmos de generación procedural de terrenos, lo que permitirá entornos aún más detallados y realistas.

En resumen, el futuro de la generación procedural de terrenos en gráficos por computadora probablemente estará caracterizado por la combinación de redes neuronales, generación de assets procedural, integración de datos del mundo real y avances en hardware y rendimiento. Estos desarrollos permitirán a los desarrolladores crear entornos más inmersivos y diversos para videojuegos, simulaciones y otras aplicaciones.

4.4. Técnicas y Algoritmos

La generación procedural de terrenos utiliza algoritmos para crear terrenos en un entorno generado por computadora. Los algoritmos utilizados para la generación de terrenos pueden variar según el resultado deseado, pero algunas de las técnicas más populares incluyen funciones de ruido, fractales y enfoques basados en grafos.

Las funciones de ruido, como el ruido de Perlin, el ruido Simplex o el ruido Voronoi, se utilizan comúnmente en la generación procedural de terrenos. Estas funciones generan valores aleatorios que se pueden usar para crear mapas de alturas, que luego se utilizan para generar el terreno. Al combinar diferentes funciones de ruido, los desarrolladores pueden crear paisajes más complejos y diversos.

Los algoritmos fractales, como el algoritmo diamante-cuadrado, también se utilizan comúnmente en la generación procedural de terrenos. Estos algoritmos utilizan la subdivisión recursiva para generar terrenos, y cada subdivisión agrega más detalle al terreno.

Enfoques basados en grafos, como el método presentado en un artículo de 2015 por Christian Schulte, utilizan grafos para representar el terreno y generan el terreno manipulando el grafo. Este enfoque se puede usar para crear terrenos más complejos y variados, con la capacidad de agregar características como ríos y carreteras.

Otras técnicas utilizadas en la generación procedural de terrenos incluyen la simulación de erosión, que simula los efectos de la erosión causada por el agua y el viento en el terreno, y el uso de datos del mundo real, como imágenes de satélite y mapas topográficos, para crear terrenos más precisos y realistas.

4.4.1. Generación Procedural de Terrenos basada en Funciones de Ruido

La generación procedural de terrenos basada en funciones de ruido es una técnica popular utilizada en gráficos por computadora para crear terrenos en videojuegos, simulaciones y otras aplicaciones. Esta técnica implica el uso de funciones de ruido, como el ruido de Perlin o el ruido Simplex, para generar valores aleatorios que se pueden utilizar para crear mapas de alturas, que luego se utilizan para generar el terreno.

Una de las ventajas de utilizar la generación procedural de terrenos basada en funciones de ruido es que permite la creación de paisajes complejos y diversos con relativamente poco

trabajo manual. Al combinar diferentes funciones de ruido, los desarrolladores pueden crear terrenos con colinas, montañas, valles y otras características.

La generación procedural de terrenos basada en funciones de ruido es una técnica poderosa para crear paisajes complejos y diversos en gráficos por computadora. Al utilizar diferentes tipos de funciones de ruido y combinarlos de diversas formas, los desarrolladores pueden crear terrenos que son tanto realistas como visualmente interesantes.

Existen varios tipos de funciones de ruido que se pueden utilizar para la generación procedural de terrenos. Algunas de las funciones de ruido más comúnmente utilizadas incluyen:

- **Ruido de Perlin:** El ruido de Perlin es un tipo de ruido de gradiente desarrollado por Ken Perlin en la década de 1980. Es un ruido suave y continuo que a menudo se utiliza para generar terrenos de aspecto natural [14].
- **Ruido Simplex:** El ruido Simplex es un tipo de ruido de gradiente desarrollado por Ken Perlin en 2001. Es similar al ruido de Perlin pero es más rápido y tiene una mejor calidad visual [15].
- **Ruido Voronoi:** El ruido Voronoi es un tipo de ruido celular que se genera dividiendo el espacio en celdas basadas en la distancia a un conjunto de puntos. A menudo se utiliza para generar terrenos con características distintas, como acantilados y crestas [16].
- **Ruido de Worley:** El ruido de Worley es otro tipo de ruido celular que se genera dividiendo el espacio en celdas basadas en la distancia a un conjunto de puntos. A menudo se utiliza para generar terrenos con características distintas, como cuevas y túneles [17].
- **Movimiento Browniano Fractal (FBM):** El FBM es una técnica que combina múltiples capas de ruido para crear terrenos más complejos y detallados. A menudo se utiliza en conjunción con otras funciones de ruido, como el ruido de Perlin o el ruido Simplex [18].

4.4.2. Algoritmos basados en fractales para la generación de terrenos procedurales

La generación procedural de terrenos a menudo se basa en algoritmos fractales que permiten crear paisajes realistas y visualmente interesantes. Algunos de los algoritmos fractales más populares para la generación procedural de terrenos incluyen:

- **Algoritmo Diamante-Cuadrado**: El algoritmo Diamante-Cuadrado es un método simple y eficiente para generar terreno fractal. Comienza con una cuadrícula cuadrada y, en cada iteración, divide cada cuadrado en cuatro cuadrados más pequeños. La altura de los nuevos vértices se ajusta en función del promedio de los vértices originales. Este proceso se repite varias veces, lo que da como resultado un terreno con una apariencia fractal [19].
- **Algoritmo de Desplazamiento del Punto Medio**: El algoritmo de Desplazamiento del Punto Medio es otro método simple para generar terreno fractal. Comienza con un segmento de línea y, en cada iteración, divide el segmento en dos partes.

La altura de los nuevos vértices se ajusta mediante un valor de desplazamiento aleatorio. Este proceso se repite, creando un terreno con características fractales [20].

- **Movimiento Browniano Fractal (FBM)**: El Movimiento Browniano Fractal (FBM) es una técnica que combina múltiples capas de funciones de ruido para crear terrenos más complejos y detallados. A menudo se utiliza en conjunto con otros algoritmos fractales, como el Diamante-Cuadrado o el Desplazamiento del Punto Medio, para generar terrenos realistas y visualmente atractivos [21].
- **Técnicas multifractales**: Las técnicas multifractales utilizan diferentes dimensiones fractales para diferentes escalas, lo que permite una representación más precisa del comportamiento del espectro de frecuencias de los paisajes reales. Estas técnicas se pueden utilizar para generar terreno con una apariencia más realista y diversa [22].
- **Enfoques híbridos**: Además de los algoritmos fractales puros, existen enfoques híbridos que combinan técnicas fractales con otros métodos, como simulaciones de erosión o modelos geológicos. Estos enfoques pueden utilizarse para generar terrenos más realistas y visualmente atractivos [23].

Los algoritmos basados en fractales son una poderosa herramienta para la generación procedural de terrenos, permitiendo la creación de paisajes complejos y visualmente interesantes. Al combinar diferentes técnicas fractales e incorporar otros métodos, los desarrolladores pueden crear terrenos que sean tanto realistas como únicos.

4.4.3. Algoritmos de Simulación Física en la Generación Procedural de Terrenos

Los algoritmos de simulación física pueden utilizarse en la generación procedural de terrenos para crear terrenos más realistas y de aspecto natural. Estos algoritmos simulan los efectos de procesos físicos como la erosión, la deposición y la meteorización en el terreno, lo que resulta en un terreno que parece haber sido moldeado por fuerzas naturales. Algunos de los algoritmos de simulación física más comúnmente utilizados en la generación procedural de terrenos incluyen:

- **Algoritmos basados en hidrología**: Los algoritmos basados en hidrología simulan el flujo del agua sobre el terreno, teniendo en cuenta factores como la pendiente, la lluvia y la evaporación. Estos algoritmos pueden utilizarse para crear terrenos con redes de ríos realistas, lagos y otras características de agua [24] [25].
- **Algoritmos de simulación de erosión**: Los algoritmos de simulación de erosión simulan los efectos de la erosión causada por el agua y el viento en el terreno, lo que da como resultado terrenos con características realistas como valles, crestas y cañones. Estos algoritmos pueden utilizarse en conjunto con otras técnicas de generación procedural, como funciones de ruido o fractales, para crear terrenos más realistas y visualmente interesantes [26] [27].
- **Algoritmos de modelado geológico**: Los algoritmos de modelado geológico simulan los procesos geológicos que dan forma al terreno, como la actividad tectónica y las erupciones volcánicas. Estos algoritmos pueden utilizarse para crear terrenos

con características geológicas realistas, como montañas, volcanes y líneas de falla [28] [29].

Los algoritmos de simulación física pueden ser una herramienta poderosa para crear terrenos realistas y visualmente interesantes en la generación procedural. Al simular los efectos de procesos físicos en el terreno, los desarrolladores pueden crear terrenos que parecen haber sido moldeados por fuerzas naturales, lo que resulta en un entorno más inmersivo y creíble.

Además de los ya mencionados, se siguen desarrolando nuevas téncicas y nuevos algoritmos de simulación física para la generación procedural de terrenos se centran en mejorar la realismo, eficiencia y flexibilidad en la creación de terrenos. Algunos de los avances en este campo incluyen:

- **Simulación Física en GPU**: Utilizando la potencia de procesamiento paralelo de las GPUs modernas, los investigadores han desarrollado técnicas para generar terrenos procedurales complejos en tiempo real. Estas técnicas aprovechan las capacidades de las GPUs, como el geometry shader, stream output y el renderizado a texturas 3D, para generar rápidamente grandes bloques de terreno detallado[30].
- **Algoritmos Inspirados en la Hidrología**: Basándose en el concepto de generación de terrenos basada en hidrología, los algoritmos más recientes incorporan modelos de flujo de agua y erosión más realistas. Estos algoritmos simulan los efectos de la lluvia, la evaporación y el transporte de sedimentos, lo que resulta en terrenos con redes fluviales, lagos y otras características acuáticas más precisas[24].
- **Generación de Paisajes Multibioma**: AutoBiomes es un ejemplo de algoritmo de generación procedural que se enfoca en crear paisajes multibioma. Esta técnica combina enfoques sintéticos, basados en física y basados en ejemplos para generar terrenos realistas y visualmente diversos con múltiples biomas distintos[31].
- **Erosión Fluvial Basada en Grafos**: Un reciente artículo propone un algoritmo de generación procedural de terrenos basado en una representación de grafo de erosión fluvial. Este algoritmo ofrece varias mejoras novedosas, incluyendo el uso de un mapa de restricción de altura con dos tipos de fuerzas de restricción localmente definidas, lo que resulta en características de terreno más realistas y detalladas[32].

4.5. Antecedentes en Unity para la Generación Procedural de Terrenos

Introducción:

En el mundo del desarrollo de videojuegos y aplicaciones interactivas, la generación procedural de terrenos se ha convertido en un recurso fundamental para crear mundos virtuales dinámicos y sorprendentes. Unity, una de las plataformas de desarrollo más populares, ofrece una gama diversa de herramientas, activos y recursos que facilitan la creación de terrenos procedurales de alta calidad. En esta sección, exploraremos estas herramientas y recursos, así como algunas de las características clave de los activos disponibles en la Tienda de Assets de Unity. Desde el motor de terreno incorporado hasta los emocionantes activos de generación procedural, descubriremos cómo Unity brinda a los desarrolladores la capacidad de dar vida a mundos virtuales únicos y cautivadores.

4.5.1. Herramientas y Recursos de Unity

Unity proporciona varias herramientas y recursos para la generación procedural de terrenos. A continuación, se presentan algunas de las herramientas y recursos más populares:

- **Motor de Terreno**: El motor de terreno incorporado de Unity permite a los desarrolladores crear y modificar terrenos utilizando una variedad de herramientas, como pinceles, mapas de texturas y mapas de alturas. El motor de terreno puede utilizarse en conjunción con técnicas de generación procedural para crear terrenos más diversos e interesantes [33].
- **Comunidad**: Existen varios tutoriales, cursos, repositorios y recursos disponibles online creados por la basta comunidad de Unity que cubren la generación procedural, incluyendo la generación de terrenos. Estos recursos abordan temas como funciones de ruido, fractales y simulación física, y proporcionan instrucciones paso a paso para crear terrenos procedurales .
- **TerrainGenerator**: TerrainGenerator es una herramienta gratuita y de código abierto para Unity que permite a los desarrolladores crear terrenos procedurales utilizando algoritmos de ruido aleatorio, simulación física y materiales personalizados. La herramienta puede crear una malla de terreno, una malla de agua y colocar objetos de forma aleatoria en una escena[34].
- **Procedural Worlds**: Procedural Worlds es un conjunto de herramientas para Unity que permite a los desarrolladores crear y entregar contenido procedural, incluyendo terrenos, paisajes y mundos. Las herramientas incluyen Gaia Pro, GeNa Pro y otras herramientas galardonadas de creación y mejora de mundos [35].

4.5.2. Plugins y Assets en Unity para la Generación de Terrenos Procedurales

Existen varios plugins y assets en Unity que pueden ayudar en la generación de terrenos procedurales. Algunos de los más populares incluyen:

- **Procedural Terrain Generator**: Este assets, disponible en la tienda de assetss de Unity, permite a los desarrolladores crear terrenos procedurales utilizando una variedad de funciones de ruido, incluidas las funciones de Perlin y Simplex. También incluye características como simulación de erosión y mezcla de texturas [36].
- **Vista 2023 - Procedural Terrain Generator**: Este assets, también disponible en la tienda de assetss de Unity, permite a los desarrolladores crear terrenos procedurales utilizando una variedad de funciones de ruido, incluidas las funciones de Perlin, Simplex y Voronoi. También incluye características como simulación de erosión y mezcla de texturas [37].
- **MapMagic**: MapMagic es una herramienta de generación de terrenos que permite a los desarrolladores crear terrenos procedurales utilizando un sistema basado en nodos. Incluye características como simulación de erosión, mezcla de texturas y generación de biomas[38].

4.5.3. Características del asset Procedural Terrain Generator en la tienda de assets de Unity

El asset Procedural Terrain Generator en la tienda de assets de Unity, desarrollado por Nuance Studios, ofrece varias características para crear terrenos procedurales. Estas características incluyen:

- **Generación Procedural**: El asset permite a los desarrolladores crear terrenos utilizando diversas funciones de ruido, incluyendo ruido de Perlin, ruido de Simplex y ruido de Voronoi. Esto posibilita la generación de paisajes realistas y diversos[36].
- **Personalización**: Los usuarios pueden personalizar fácilmente el terreno ajustando parámetros como escala, frecuencia y amplitud. Esta flexibilidad permite la creación de terrenos únicos y personalizados[36].
- **Simulación de Erosión**: El asset incluye una característica de simulación de erosión incorporada que se puede utilizar para crear lechos de ríos realistas, valles y otras características de terreno relacionadas con la erosión[36].
- **Mezcla de Texturas**: Los desarrolladores pueden mezclar múltiples texturas en el terreno, lo que permite la creación de paisajes más detallados y visualmente atrassets[36].
- **Compatibilidad**: El asset Procedural Terrain Generator es compatible con las versiones de Unity 5.3.4 o superiores [36].

Además del asset Procedural Terrain Generator, existen otros assets disponibles en la tienda de assets de Unity que ofrecen características similares para la generación de terrenos procedurales, como Vista 2023 - Procedural Terrain Generator de Pinwheel Studio[37] y Tellus - Procedural Terrain Generator de Darkcom Dev[39]. Estos assets pueden utilizarse para mejorar las capacidades de generación de terrenos de Unity y acelerar el proceso de desarrollo de juegos.

4.6. Aplicaciones de la Generación Procedural de Terreno

4.6.1. Aplicaciones en Videojuegos

La generación procedural de terreno posee numerosas aplicaciones en la industria de los videojuegos. Algunas de las aplicaciones más destacadas incluyen:

- **Aumento de la Rejugabilidad**: La generación procedural de terreno puede utilizarse para crear variaciones infinitas del entorno de un juego, lo que aumenta la rejugabilidad y mantiene el juego fresco e interesante para los jugadores[40].
- **Ahorro de Tiempo en la Creación de Activos**: Al generar el terreno de forma procedural, los desarrolladores pueden ahorrar tiempo en la creación de activos y centrarse en otros aspectos del desarrollo del juego[41].
- **Creación de Entornos Únicos**: La generación procedural de terreno puede utilizarse para crear entornos únicos y visualmente interesantes que serían difíciles o imposibles de crear manualmente[42].
- **Generación de Jugabilidad Aleatoria**: La generación procedural de terreno puede utilizarse para crear jugabilidad aleatoria, como mapas, niveles, enemigos y armas aleatorias. Esto añade un elemento de imprevisibilidad y desafío al juego[43].
- **Creación de Nuevas Mecánicas de Juego**: La generación procedural de terreno puede utilizarse como una mecánica de juego, como la creación de nuevos entornos para que el jugador explore o la generación de rompecabezas y desafíos[44].

La generación procedural de terreno se ha convertido en una técnica cada vez más popular en el desarrollo de videojuegos, ofreciendo numerosos beneficios como el aumento de la rejugabilidad, el ahorro de tiempo y la creación de entornos únicos. A medida que la tecnología continúa avanzando, podemos esperar ver técnicas de generación de terreno aún más sofisticadas y realistas en los futuros videojuegos.

4.6.2. Aplicaciones en Simulaciones Científicas

La generación procedural de terrenos tiene diversas aplicaciones en simulaciones científicas, particularmente en los campos de la geología, hidrología y ciencias ambientales. Algunas de las aplicaciones destacadas incluyen:

1. **Modelado Geológico**: La generación procedural de terrenos se puede utilizar para crear modelos geológicos realistas en simulaciones científicas. Al simular procesos geológicos como la actividad tectónica y la erosión, los desarrolladores pueden crear terrenos que representen con precisión paisajes del mundo real[45].
2. **Modelado Hidrológico**: La generación procedural de terrenos se puede utilizar para crear modelos hidrológicos en simulaciones científicas. Al simular el flujo de agua sobre el terreno, los desarrolladores pueden crear modelos que representen con precisión sistemas de agua del mundo real, como ríos, lagos y cuencas hidrográficas[46].

3. **Modelado Ambiental**: La generación procedural de terrenos se puede utilizar para crear modelos en simulaciones ambientales, como simulaciones de cambio climático o desastres naturales. Al generar terrenos que representen con precisión entornos del mundo real, los desarrolladores pueden crear simulaciones más precisas y realistas[47].

4. **Compresión de Datos**: La generación procedural de terrenos se puede utilizar para comprimir grandes cantidades de datos de terreno en un archivo de menor tamaño. Al generar terrenos de manera procedural, los desarrolladores pueden crear terrenos sobre la marcha, reduciendo la necesidad de grandes cantidades de datos de terreno pregenerados[48].

La generación procedural de terrenos tiene numerosas aplicaciones en simulaciones científicas, ofreciendo beneficios como mayor precisión, compresión de datos y la capacidad de simular sistemas naturales complejos.

4.6.3. Aplicaciones en Realidad Virtual y Aumentada

La generación procedural de terrenos tiene varias aplicaciones en la realidad virtual (RV) y la realidad aumentada (RA), ofreciendo beneficios como mayor inmersión, interactividad y realismo. Algunas de las aplicaciones destacadas incluyen:

- **Terrenos a escala planetaria en RV**: La generación procedural de terrenos puede utilizarse para crear terrenos a escala planetaria en RV, permitiendo a los usuarios explorar entornos vastos y diversos. Esta técnica se puede utilizar para crear experiencias inmersivas para la educación, el entretenimiento y la investigación científica[49].

- **Generación interactiva de terrenos virtuales utilizando marcadores de RA**: La generación procedural de terrenos puede utilizarse para crear terrenos virtuales interactivos utilizando marcadores de RA. Esta técnica permite a los usuarios crear y modificar terrenos en tiempo real, brindando una experiencia más atractiva e interactiva[50].

- **Generación de RV procedural a partir de espacios físicos 3D reconstruidos**: La generación procedural de terrenos puede utilizarse para generar entornos de RV a partir de espacios físicos 3D reconstruidos. Esta técnica permite a los usuarios explorar entornos del mundo real en RV, brindando una experiencia más inmersiva y realista[51].

- **Generación procedural de escenas de RV**: La generación procedural de terrenos puede utilizarse para generar escenas de RV, como islas u otros entornos. Esta técnica permite a los desarrolladores crear entornos inmersivos y visualmente interesantes que pueden explorarse en RV[52].

- **Experiencias de RV sobre la marcha mientras se camina dentro de grandes entornos de edificios del mundo real desconocidos**: La generación procedural de terrenos puede utilizarse para generar experiencias de RV mientras se camina dentro de grandes entornos de edificios del mundo real desconocidos. Esta técnica permite a los usuarios explorar y navegar por entornos complejos en RV, brindando una experiencia más inmersiva e interactiva[53].

4.6.4. Aplicaciones en Animación y Películas

La generación procedural de terrenos tiene varias aplicaciones en animación y películas, ofreciendo beneficios como mayor eficiencia, flexibilidad y creatividad. Algunas de las aplicaciones destacadas incluyen:

- **Películas animadas generadas proceduralmente**: La generación procedural de terrenos puede utilizarse para crear películas animadas con entornos y personajes generados aleatoriamente. Esta técnica se puede utilizar para crear películas únicas e interesantes visualmente con un esfuerzo manual mínimo[54].
- **Generación procedural de objetos 3D y animaciones**: La generación procedural puede utilizarse para crear objetos 3D y animaciones para películas, como diseños de personajes, animaciones y diálogos de personajes no jugadores. Esta técnica se puede utilizar para crear contenido diverso e interesante con un esfuerzo manual mínimo[55].
- **Creación rápida de espacios visualmente interesantes y precisos**: La generación procedural se utiliza con frecuencia en el cine para crear espacios visualmente interesantes y precisos de forma rápida. Esta técnica se puede utilizar para crear entornos diversos e interesantes que pueden explorarse en películas[56].
- **Creación de cortometrajes animados utilizando generación procedural**: La generación procedural de terrenos puede utilizarse para crear cortometrajes animados utilizando formas generadas por código y campos de distancia. Esta técnica se puede utilizar para crear películas visualmente interesantes y únicas con un tamaño de archivo mínimo[57].

4.7. Desafíos y Tendencias Futuras

La generación procedural de terrenos presenta una serie de desafíos y tendencias que son fundamentales para su evolución y aplicación continua en el desarrollo de juegos y otras áreas. Algunos de estos desafíos y tendencias incluyen:

1. **Consistencia y Coherencia**: Asegurar que el terreno generado sea consistente y coherente en diversas plataformas y dispositivos puede ser un desafío. Técnicas como las funciones de ruido, la síntesis de terreno y la generación de contenido procedural (PCG) pueden ayudar a abordar este desafío[58].
2. **Equilibrio en la Jugabilidad**: Crear un mundo generado que sea justo y siga siendo desafiante puede ser un obstáculo. Equilibrar la dificultad del terreno y los mecanismos de juego es crucial para lograr una experiencia satisfactoria para el jugador [59].
3. **Realismo y Variedad**: Lograr un equilibrio entre el terreno realista y paisajes diversos y visualmente interesantes es un desafío. Las tendencias futuras pueden centrarse en mejorar el realismo y la variedad de los terrenos generados mediante algoritmos y técnicas avanzadas[60].

4. **Optimización de Rendimiento**: Generar terrenos complejos en tiempo real puede ser computacionalmente costoso. Las tendencias futuras pueden implicar la optimización de los algoritmos de generación procedural de terrenos para mejorar el rendimiento y reducir el uso de recursos[61].

5. **Integración con Otros Sistemas de Juego**: La generación procedural de terrenos debe integrarse sin problemas con otros sistemas de juego, como la simulación de física, la inteligencia artificial y el diseño de niveles. Las tendencias futuras pueden implicar el desarrollo de técnicas más avanzadas para integrar la generación procedural de terrenos con estos sistemas[62].

En resumen, la generación procedural de terrenos es una tendencia creciente en el desarrollo de juegos, que ofrece oportunidades para crear mundos de juego más grandes, dinámicos e interesantes. Abordar los desafíos y adoptar las tendencias futuras en este campo conducirá a técnicas de generación de terrenos más avanzadas y realistas.

Capítulo 5

Desarrollo de la solucion

5.1. Análisis

5.1.1. Objetivos de Implementación

Los objetivos de implementación se centran en las metas técnicas y funcionales que se buscan alcanzar en el desarrollo de la herramienta de generación procedural de terrenos en Unity. Estos objetivos se dividen en los siguientes aspectos clave:

1. Diseño de Algoritmos de Generación

El objetivo principal en esta fase de implementación es diseñar algoritmos de generación de terrenos que sean eficientes y capaces de producir resultados convincentes. Esto incluye:

- Investigar y seleccionar algoritmos de generación de terrenos adecuados para el proyecto.
- Diseñar algoritmos que permitan la creación de terrenos realistas y variados.

2. Optimización del Rendimiento

Para garantizar que la herramienta funcione de manera eficiente en diversas plataformas y escenarios de desarrollo, se establecen los siguientes objetivos:

- Optimizar el rendimiento de los algoritmos de generación para maximizar los fotogramas por segundo.
- Implementar estrategias de cálculo paralelo utilizando el sistema de trabajos (Job System) de Unity para acelerar la generación de terrenos.

3. Pruebas y Validación

La validación de la herramienta es fundamental para garantizar su funcionamiento correcto. Los objetivos relacionados con las pruebas son:

- Detectar posibles errores y problemas de rendimiento. Como la consistencia entre chunks vecinos dando lugar a terrenos conitnuos y la generación del terreno continua sin bajadas de fps notables.
- Validar que la herramienta genera terrenos realistas acorde a los parámetros con los que se configura.
- Comprobar que los resultados son visualmente integrables en juegos o proyectos desarrollados en Unity y que permite crear un terreno explorable.

4. Mejoras en Realismo y Diferenciación de Alturas

Además de los objetivos anteriores, se busca mejorar el realismo de los terrenos generados mediante la implementación de algoritmos de erosión. Los objetivos adicionales incluyen:

- Investigar y aplicar algoritmos de erosión para simular procesos geológicos en los terrenos generados.
- Evaluar cómo los algoritmos de erosión mejoran la apariencia y autenticidad de los terrenos.
- Implementar la diferenciación de alturas en los terrenos mediante la asignación de colores según la elevación para una representación visual más rica y comprensible.

5.1.2. Requisitos del Sistema

Requisitos Funcionales

1. **Generación de Terrenos:** El sistema debe ser capaz de generar terrenos de manera procedural en tiempo real, permitiendo a los usuarios especificar parámetros como tamaño, altura, erosión y demás configuraciones de terreno.
2. **Continuidad del terreno:** El terreno no debe presentar discontinuidades y debe generar extensiones de terreno sin que haya una transción notable de uno a otro.
3. **Algoritmos de Generación Configurables:** Los algoritmos de generación utilizados deben ser configurables, lo que permitirá a los usuarios ajustar los detalles de la generación según sus necesidades.
4. **Optimización del Rendimiento:** El sistema debe estar optimizado para garantizar que la generación de terrenos sea eficiente en términos de uso de recursos y tiempos de carga.
5. **Erosión Simulada:** Se deben implementar algoritmos de erosión para simular procesos geológicos y mejorar la apariencia de los terrenos generados.
6. **Diferenciación de Alturas:** El sistema debe asignar colores a diferentes elevaciones del terreno para facilitar la visualización y comprensión de las características del terreno.

Requisitos No Funcionales

1. **Rendimiento:** El sistema debe ser capaz de generar terrenos en tiempo real sin experimentar retrasos notables en la ejecución.

2. **Compatibilidad con Unity:** La herramienta debe integrarse perfectamente con el motor Unity, aprovechando sus capacidades y recursos.

3. **Portabilidad:** El sistema debe ser compatible con múltiples plataformas y versiones de Unity, lo que permite a los desarrolladores utilizarlo en diversos proyectos.

4. **Usabilidad:** El sistema debe ser intuititvo, con parámetros nombrados de manera que no cause confución en el usurio y expresen de manera clara su función.

5. **Realismo Visual:** El sistema debe ser capaz de generar terrenos con realismo visual, incluyendo detalles naturales como montañas, valles.

6. **Diferenciación de Alturas:** La herramienta debe permitir la diferenciación de alturas en el terreno mediante la asignación de colores o texturas específicas para representar diferentes elevaciones, facilitando la visualización y comprensión del terreno generado.

5.1.3. Arquitectura del Sistema

Visión General

La arquitectura del sistema se basa en un enfoque modular que consta de varios componentes interconectados. El sistema se ha diseñado para ser altamente flexible y escalable, permitiendo la generación procedural de terrenos de manera eficiente. La arquitectura se centra en la generación de terrenos y su visualización en tiempo real.

Componentes del Sistema

Los componentes clave del sistema incluyen:

- **MapGenerator:** Responsable de crear el terreno proceduralmente. Utiliza algoritmos de generación de ruido y permite la visualización en el editor de Unity.

- **MapDataGeneratorJob:** Se encarga de la generación de datos del terreno, como el mapa de altura y el mapa de colores, utilizando el Unity Job System.

- **Noise:** Contiene métodos para generar ruido Perlin, Simplex y Voronoi, que se utilizan en la generación del terreno.

- **ErosionJob:** Responsable de tratar cada índice del mapa de alturas generado con MapGenerator mediante algoritmos de erosión para mejorar la apariencia del terreno, creando características como ríos y cañones.

- **MeshDataGeneratorJob:** Genera los datos de la malla del terreno, permitiendo la creación de mallas detalladas y eficientes en cuanto a rendimiento.

- **EndlessTerrain:** Controla la generación continua y la representación de terrenos, generando nuevos trozos de terreno y manejando la gestión de los mismos para lograr un mundo de juego sin fin.
- **TextureGenerator:** Se encarga de generar las texturas del terreno en función del mapa de altura y el gradiente de colores.
- **ConfigSettings:** Almacena las configuraciones del sistema, incluyendo parámetros de generación de terrenos, configuraciones de malla y opciones de visualización.

Esta arquitectura modular y bien definida permite una generación procedural de terrenos flexible y eficaz en Unity.

Diagramas de Arquitectura

A continuación, se presentan diagramas de arquitectura que muestran la estructura y las relaciones entre los componentes del sistema:

Diagrama de Casos de Uso

Para comprender mejor las interacciones entre los usuarios y el sistema, se han creado diagramas de casos de uso. Estos diagramas describen cómo los usuarios interactúan con el sistema y qué funcionalidades están disponibles para ellos.

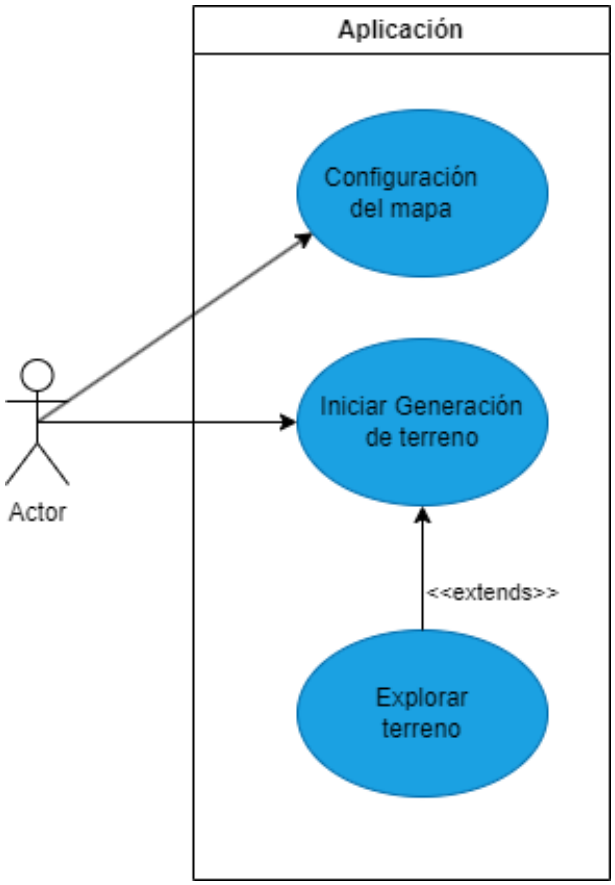


Figura 5.1: Diagrama de Casos de Uso del Proyecto.

El diagrama de secuencia detallará las interacciones entre los componentes del sistema, incluidos el `MapGenerator`, el `MeshGenerator`, el `EndlessTerrain`, y otros, durante la generación y visualización del terreno.

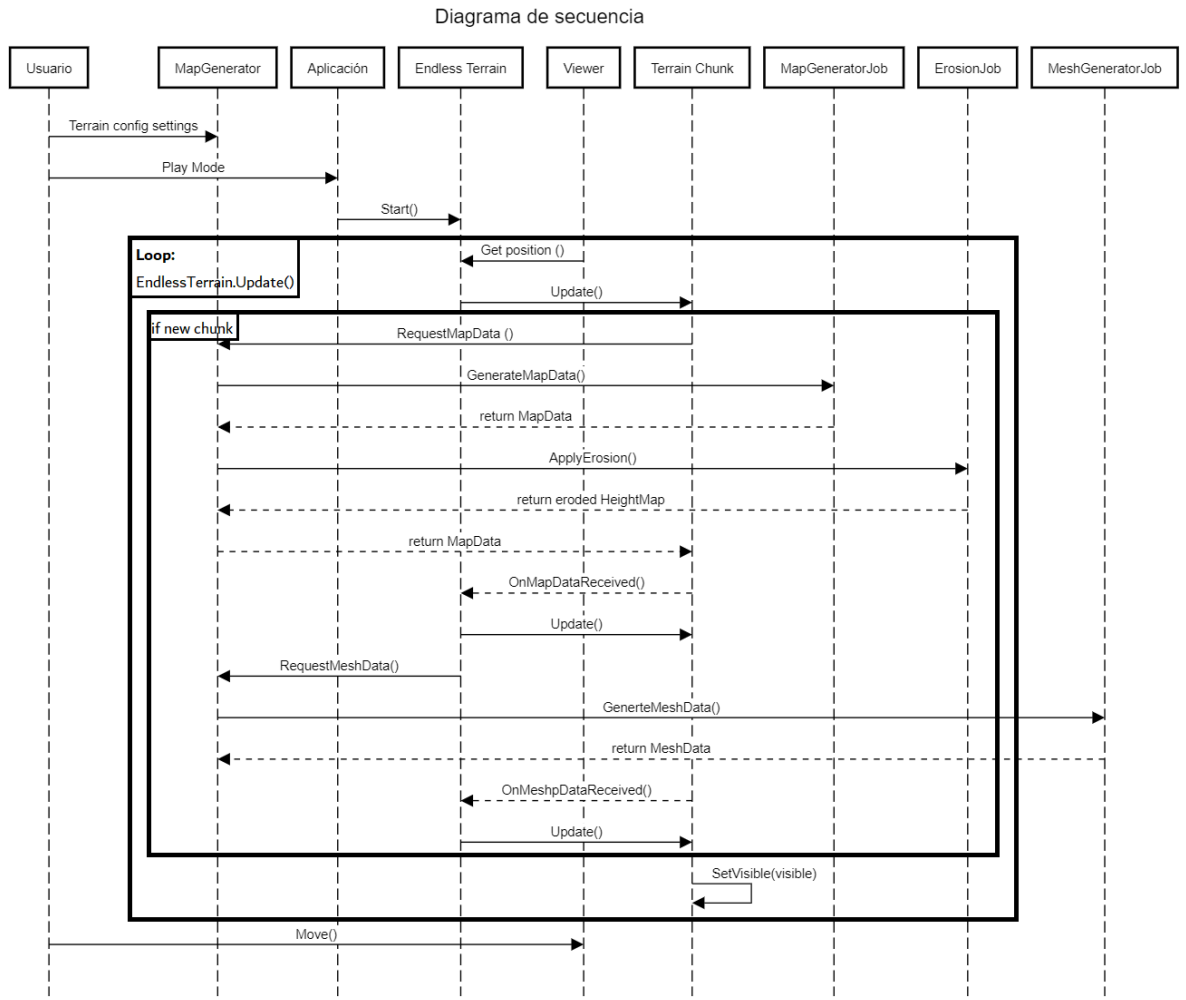


Figura 5.2: Diagrama de Secuencia de Generación de Terreno.

5.1.4. Tecnologías y Herramientas Utilizadas

Elección de Tecnologías

La elección de las tecnologías específicas para este proyecto se basó en los siguientes criterios:

- **Unity 3D:** Se eligió Unity como motor de desarrollo debido a su versatilidad y capacidad para crear aplicaciones interactivas en 3D. Unity proporciona una amplia gama de herramientas y recursos que facilitan el desarrollo de juegos y simulaciones.
- **Unity Job System:** Para optimizar el rendimiento en la generación de terrenos, se utiliza el Unity Job System, que permite la paralelización de tareas en múltiples núcleos de CPU.

- **Burst Compiler:** La herramienta Burst Compiler se utiliza para compilar el código C # en código nativo altamente optimizado, mejorando aún más el rendimiento de la generación de terrenos.
- **Perlin Noise y Simplex Noise:** Se implementan algoritmos de ruido Perlin y Simplex para la generación de terrenos. Estos algoritmos proporcionan resultados realistas y variados.
- **Herramientas de Diseño 3D:** Se utilizan herramientas de diseño 3D, como Blender y Substance Painter, para crear modelos y texturas que se aplicarán al terreno.

Estas tecnologías se eligieron cuidadosamente para garantizar un rendimiento óptimo y una alta calidad en la generación de terrenos en tiempo real.

Lenguajes de Programación

- **C#:** Este proyecto está programado enteramente utilizando C# como lenguaje de programación. C# es el lenguaje que utilizan los componentes Scripts de Unity por lo que está altamente integrado con el motor, y es un lenguaje orientado a objetos que ofrece un alto rendimiento y facilidad de uso.

5.1.5. Herramientas de Desarrollo

Durante el desarrollo de este proyecto, se utilizaron diversas herramientas y software que desempeñaron un papel fundamental en la planificación, implementación y gestión del trabajo. A continuación, se detallan las principales herramientas de desarrollo utilizadas:

- **IDE Principal:** JetBrains Rider fue el IDE principal utilizado para el desarrollo en C#. Rider proporcionó un entorno de desarrollo integrado eficiente y potente para la escritura de código, depuración y pruebas del proyecto.
- **Diagrama de Clases (Visual Studio):** Visual Studio se utilizó específicamente para la creación de diagramas de clases, lo que permitió una representación visual clara de la estructura del proyecto y las relaciones entre las clases.
- **Diagramas (VS Code con el Plugin draw.io):** Visual Studio Code, junto con el plugin draw.io, se utilizó para crear varios tipos de diagramas, incluidos los diagramas de casos de uso y diagramas de actividad. Estas representaciones gráficas ayudaron a comprender y comunicar el flujo de trabajo del sistema.
- **Memoria en LaTeX:** La documentación y memoria del proyecto se crearon utilizando LaTeX, con el editor VS Code.
- **Control de Versiones (Git y GitHub):** Git se utilizó para el control de versiones del código fuente del proyecto. GitHub se empleó como plataforma de alojamiento para el repositorio de Git, lo que facilitó la colaboración en equipo y el seguimiento de cambios.
- **Gestión de Tareas (Trello):** Trello se utilizó para la gestión de tareas y la planificación del proyecto. La herramienta permitió organizar y priorizar tareas, así como realizar un seguimiento del progreso de cada elemento del proyecto.

- **Burst Compiler:** El Burst Compiler se utilizó para compilar el código C# en código nativo altamente optimizado, lo que contribuyó significativamente a mejorar el rendimiento en la generación de terrenos.

Estas herramientas desempeñaron un papel esencial en la realización exitosa del proyecto, proporcionando las capacidades necesarias para el desarrollo, la documentación, la colaboración en equipo y la optimización de rendimiento.

5.1.6. Desafíos y Decisiones de Diseño

Desafíos Técnicos

Durante el diseño y desarrollo del proyecto, se enfrentaron varios desafíos técnicos significativos. Algunos de los desafíos más destacados incluyeron:

- **Optimización de Rendimiento:** Lograr un rendimiento óptimo en la generación procedural de terrenos en tiempo real fue uno de los principales desafíos técnicos. Se implementó el Unity Job System y el Burst Compiler para abordar este desafío.
- **Generación Realista:** La generación de terrenos realistas y variados implicó la implementación de algoritmos de ruido Perlin, Simplex y Voronoi, así como la configuración adecuada de parámetros como escalas y octavas.
- **Erosión y Características Naturales:** Incorporar algoritmos de erosión para simular características naturales como ríos y cañones fue un desafío adicional.
- **Continuidad del terreno:** La continuidad del terreno nuevo que se genera, sin notar la transición entre partes de terreno generadas e integradas al terreno que ya había supuso otro tema técnico que hubo que resolver.
- **Corrección de borde en erosión:** Dado que a eorisón se reliza teneindo en cuenta las alturas de cada fragmetnosde terreno y equilibrándolas, porduce incosistencias con las partes de terreno. Resolver esto fue otro desafío.

Decisiones de Diseño

Las decisiones de diseño desempeñaron un papel fundamental en la arquitectura y funcionalidad del proyecto. Algunas de las decisiones clave incluyeron:

- **Uso del Unity Job System:** Se decidió utilizar el Unity Job System para la paralelización de tareas y optimizar la generación de terrenos, lo que resultó en un rendimiento mejorado.
- **Selección de Algoritmos de Ruido:** La elección de implementar algoritmos de ruido Perlin y Simplex permitió generar terrenos realistas y variados con una apariencia natural.
- **Erosión para Características Naturales:** La incorporación de algoritmos de erosión en el diseño permitió crear características naturales como ríos y cañones, mejorando la apariencia general del terreno.

- **Elección de nave como explorador** La elección de elección de una nave como explorador de terreno se debió a que las irregularidades del terreno para terrenos escarpados podrían complicar la exploración, además de que con un sobrevuelo se pdoría ver mejor el rendimiento de la generació.

Alternativas Consideradas

Antes de tomar las decisiones de diseño finales, se consideraron varias alternativas, incluyendo:

- **Otras Tecnologías de Generación de Terrenos:** Se evaluaron diferentes tecnologías y enfoques para la generación de terrenos, como el uso de mapas de altura pregenerados versus generación procedural en tiempo real.
- **Métodos de Optimización:** Se exploraron diversas técnicas de optimización además del Unity Job System, como el uso de GPU para cálculos intensivos o la paralelización mediante threads manual.
- **Otros Algoritmos de Ruido:** Se investigaron algoritmos de ruido alternativos además de Perlin y Simplex para determinar cuáles producirían los resultados deseados y técnicas de generación, como el algoritmo diamante-cuadrado. Pero se optó por el ruido debido a que facilitaba la consistencia entre lso bordes de partes de terreno nuevas generadas

5.1.7. Planificación del Desarrollo

Metodología de Desarrollo

El proyecto siguió una metodología de desarrollo ágil, lo que permitió una adaptación flexible a medida que se abordaban desafíos técnicos y se tomaban decisiones de diseño. Se realizaron reuniones periódicas de revisión y planificación para ajustar el enfoque según fuera necesario. Cada día se fueron subiendo incrementos de desarrollo al repositorio remoto, gestioando un control de cuáles habían sido las mejroas subidas, cuál era el estado del proyecto y cuáles debían ser los siguientes objetivos.

Gestión de Tiempo

La gestión del tiempo se realizó mediante una planificación detallada en Trello. Creando pilas de tareas "por hacer", .$^{\text{en}}$ desarrollo", "termiandasz "mejorables".

Recursos Necesarios

Los recursos necesarios para llevar a cabo el desarrollo incluyeron:

- Personal de Desarrollo: Para este proyecto el personal fue una única persona que ocupó todos los roles del desarrollo y un product owner que especificaba los requisitos que debía cumplir el proyecto.

- Hardware: Se utilizó un equipo portátil con procesador i5-11400H con 2.7GHz, 16 GB de RAM, 1TB de memoria en disco y tarjeta gráfica NVidia 3060 con 6GB de RAM. El equipo contaba con windows 10 Home como sistema operativo.
- Software: Se requirieron herramientas como Unity3D, JetBrains Rider, Visual Studio, VS Code con el plugin draw.io, LaTeX y el Burst Compiler para el desarrollo y la documentación del proyecto.

La gestión eficaz de estos recursos fue fundamental para el éxito del proyecto.

5.2. Diseño

5.2.1. Diseño de la Arquitectura

Diseño de la Arquitectura del Software

Describe la arquitectura de software que has diseñado, incluyendo patrones arquitectónicos si los has aplicado.

Diagramas de Clases

Incluye diagramas de clases que representen la estructura de clases de tu proyecto y sus relaciones.

5.2.2. Diseño Detallado

Diagramas de Secuencia

Muestra los diagramas de secuencia que ilustran la interacción entre los componentes clave de tu sistema.

Diagramas de Flujo

Incluye diagramas de fujo que representen los flujos de trabajo y procesos en tu aplicación.

5.2.3. Decisiones de Diseño

Decisiones de Diseño Clave

Destaca las decisiones de diseño más importantes que afectaron la arquitectura y funcionalidad de tu proyecto.

Consideraciones de Rendimiento

Explica las consideraciones de rendimiento que tuviste en cuenta en el diseño.

5.2.4. Planificación de Desarrollo

Cronograma de Desarrollo

Incluye el cronograma detallado de desarrollo, indicando hitos y fechas clave.

Asignación de Recursos

Describe cómo asignaste recursos, incluyendo personal y hardware, para llevar a cabo el diseño y desarrollo.

Control de Cambios

Detalla cómo gestionarás los cambios en el diseño a medida que evolucione el proyecto.

5.2.5. Viabilidad

Viabilidad técnica

Viabilidad Legales

Describe cualquier consideración legal relacionada con el diseño, como licencias de software y derechos de autor.

5.3. Implementación

5.3.1. Entorno de Desarrollo

Herramientas de Desarrollo

Enumera las herramientas de desarrollo que has utilizado, incluyendo el entorno de desarrollo integrado (IDE), editores de código, y otros programas relevantes.

Lenguajes de Programación

Describe los lenguajes de programación que has empleado en tu proyecto y su papel en la implementación.

Frameworks y Bibliotecas

Menciona cualquier framework o biblioteca de terceros que hayas utilizado y cómo contribuyeron a la implementación.

5.3.2. Estructura del Código Fuente

Organización de Directorios

Explica la estructura de directorios de tu proyecto y cómo organizaste los archivos de código fuente.

Módulos y Componentes Principales

Identifica los módulos y componentes principales de tu aplicación y cómo se relacionan entre sí.

Flujo de Datos

Describe cómo fluyen los datos a través de tu aplicación, desde la entrada hasta la salida.

5.3.3. Algoritmos y Técnicas

Descripción de Algoritmos Clave

Explica los algoritmos clave que implementaste en tu proyecto y cómo contribuyen a su funcionamiento.

Técnicas de Optimización

Si aplicaste técnicas de optimización, como paralelización o caching, descríbelas y su impacto en el rendimiento.

5.3.4. Desarrollo de Características

Implementación de Funcionalidades

Detalla la implementación de las principales funcionalidades de tu proyecto, destacando aspectos relevantes.

Integración de Componentes

Explica cómo integras los diferentes componentes y módulos en tu aplicación.

5.3.5. Pruebas y Depuración

Pruebas Unitarias

Describe las pruebas unitarias que realizaste y cómo ayudaron a identificar y corregir errores.

Pruebas de Integración

Explica cómo llevaste a cabo las pruebas de integración y qué desafíos enfrentaste.

Depuración y Solución de Problemas

Detalla el proceso de depuración que seguiste para identificar y resolver problemas en el código.

Capítulo 6

Conclusiones

6.1. Revisión de costes

6.2. Conclusiones

6.3. Trabajo futuro

Apéndice A

Apéndice

A.1. Ejemplos del lenguaje de marcado Latex

Ths document is an example of BibTeX using in bibliography management. Three items are cited: *The LaTeX Companion* book [?], the Einstein journal paper [?], and the Donald Knuth's website [?]. The LaTeX related items are [?, ?][1].

Texto en el párrafo 1.

Texto en el párrafo 2.

`Texto` en el párrafo 3.

- Consideración 1
- Consideración 2

1. Punto 1
2. Punto 2

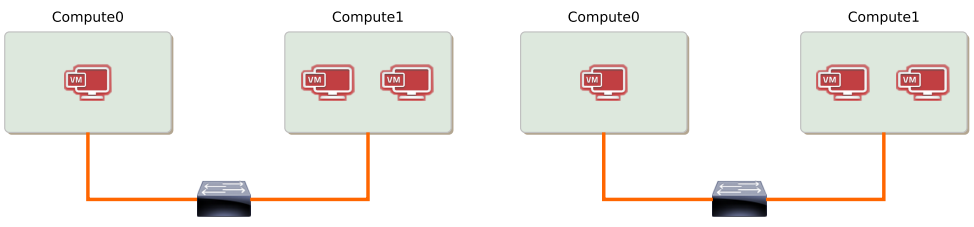
A continuación se muestra una ecuación:

$$\int_0^1 \frac{1}{x^2+1} dx$$

Podemos incluir imágenes en formato: png, pdf o jpg.

En la figura A.1 se muestra un diagrama realizado con https://www.yworks.com/products/yed:

Imagen 1 Imagen 2



[1] Esto está tomado de https://www.overleaf.com/learn/latex/Bibliography_management_with_bibtex

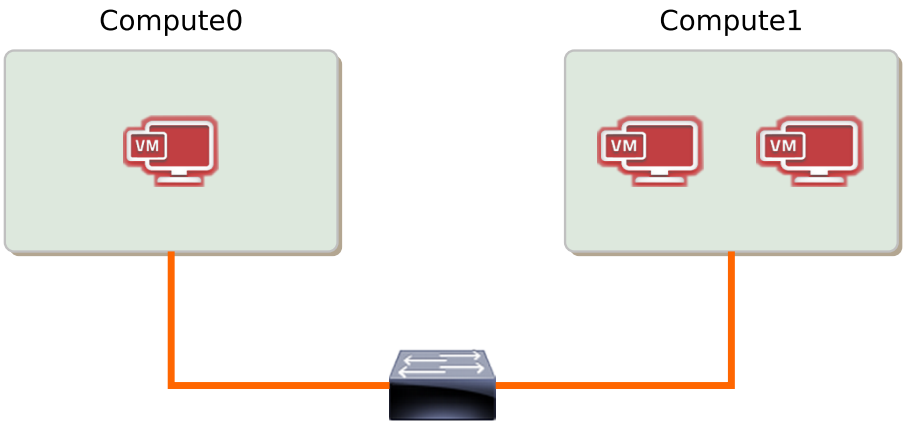


Figura A.1: Esta es una figura que latex decide donde colocar (floating) en el documento.

Este es un ejemplo de una tabla:

Columna 1	Columna 2
1	2

O la misma tabla centrada:

Columna 1	Columna 2
1	2

Para generar el fichero PDF:

```
pdflatex ejemplo-memoria.tex
bibtex ejemplo-memoria
pdflatex ejemplo-memoria.tex
```

También se puede usar `latexmk` que automáticamente regenera la bibliografía.

```
latexmk -pdf ejemplo-memoria.tex
```

Bibliografía

[1] Wikipedia contributors. History of computer animation, 2023.

[2] Jaanus Jaggo Raimond Tunnel and Margus Luik. Computer graphics learning materials, s. f.

[3] Wikipedia contributors. Computer graphics, 2023.

[4] Effelsberg W. Freiknecht J. A survey on the procedural generation of virtual worlds, 2017.

[5] David S Ebert. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.

[6] Vidak Mijailovic. A graph-based approach to procedural terrain, 2015.

[7] Aryamaan Jain, Avinash Sharma, and Rajan. Adaptive & multi-resolution procedural infinite terrain generation with diffusion models and perlin noise. In *Proceedings of the Thirteenth Indian Conference on Computer Vision, Graphics and Image Processing*, ICVGIP '22, New York, NY, USA, 2023. Association for Computing Machinery.

[8] E. Galin, A. Peytavie, N. Maréchal, and E. Guérin. Procedural generation of roads. *Computer Graphics Forum*, 29(2):429–438, 2010.

[9] Alexander Nordh Filip Stal and Joel Weidenmark. Procedural terrain generation, s. f.

[10] Sam Snider-Held. Neural networks and the future of 3d procedural content generation. *Towards Data Science*, 2017.

[11] GarageFarm partner. The power of procedural asset generation and render farms in 3d graphics. *GarageFarm.NET*, s. f.

[12] Roland Fischer, Philipp Dittmann, René Weller, and Gabriel Zachmann. Autobiomes: procedural generation of multi-biome landscapes. *The Visual Computer*, 36(10):2263–2272, 2020.

[13] NVIDIA. Generating complex procedural terrains using gpu. *NVIDIA Developer*, s. f.

[14] Ken Perlin. Improving noise, 2002.

[15] Ken Perlin. Simplex noise demystified, 2001.

[16] François Labelle. Voronoi diagrams and delaunay triangulations. *University of Ottawa*, 2016.

[17] Steve Worley. A cellular texture basis function. *Proceedings of SIGGRAPH '96*, 1996.

[18] Ken Musgrave, Darwyn Peachey, Jim Perlin, and Ken Perlin. *Texturing and Modeling, Third Edition: A Procedural Approach.* Morgan Kaufmann, 2002.

[19] A. Krista Bird B. Thomas Dickerson C. Jessica George. Diamond-square algorithm.

[20] Midpoint displacement algorithm.

[21] Ken Musgrave, Darwyn Peachey, Jim Perlin, and Ken Perlin. *Texturing and Modeling, Third Edition: A Procedural Approach.* Morgan Kaufmann, 2002.

[22] François Labelle. Voronoi diagrams and delaunay triangulations. *University of Ottawa*, 2016.

[23] Hybrid approaches in procedural terrain generation.

[24] Terrain generation using procedural models based on hydrology.

[25] Hydrology-based terrain generation.

[26] Erosion and other algorithms in chunk-based terrain.

[27] Terrain erosion - 3 ways.

[28] Maria Månsson. *Generation of Virtual Terrains for Games.* Linköping University Electronic Press, 2019.

[29] Geological terrain modeling.

[30] Generating complex procedural terrains using gpu.

[31] Claude Côté, Christian Gagné, and Pierre-Luc St-Charles. Autobiomes: A framework for procedural generation of multi-biome landscapes. *Computational Visual Media*, 6(3):309–325, 2020.

[32] Benjamin Marlin and Ali Bakhoda. Procedural terrain generation with graph-based fluvial erosion. *arXiv preprint arXiv:2210.14496*, 2022.

[33] Unity terrain.

[34] Aida Clyens. Terraingenerator.

[35] Anónimo. Procedural worlds.

[36] Renan Oliveira. Procedural terrain generator.

[37] Vista 2023 - procedural terrain generator.

[38] Best terrain generator.

[39] Tellus - procedural terrain generator en la tienda de assets de unity.

[40] Wikipedia contributor. Procedural generation.

[41] List of games using procedural generation.

[42] Procedural generation: An overview.

[43] Clearing up confusion on procedural generation.

[44] Terrain generation.

[45] Markus Gipp, Norman Meuschke, and Andreas Gernandt. Controlled procedural terrain generation using software agents. *Proceedings of the 5th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 407–412, 2013.

[46] Frank Losasso and Hugues Hoppe. Hydrologically accurate terrain generation for interactive simulation. *Proceedings of ACM SIGGRAPH 2004*, pages 359–367, 2004.

[47] Tobias Höllerer, Johannes Lintinen, and Stefan Greuter. Interactive procedural terrain generation. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[48] Adrian Lewis, Alyn Rockwood, Mark Campbell, Patrick Smith, and Brian Wyvill. Interactive terrain modelling using hierarchical procedural models. *Proceedings of the 2000 ACM symposium on Interactive 3D graphics*, pages 41–50, 2000.

[49] Veronica Sundstedt and Gabriel Brostow. Generating planetary-scale terrains in virtual reality. *California State University Theses and Dissertations*, 2014.

[50] Haralambos Papagiannis. Interactive virtual terrain generation using augmented reality markers. *Proceedings of the 2013 ACM international symposium on Pervasive displays*, pages 1–6, 2013.

[51] Sean Fanello, Cem Keskin, Daewon Kim, and Shahram Izadi. Procedural generation of virtual reality content from 3d reconstructed physical space. *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 231–242, 2016.

[52] Kamil Kurek. Locomotion in vr: Procedural generation of a scene. 2017.

[53] Anastasios Roussos, Fotis Liarokapis, Nikolaos Mourkoussis, Margherita Antona, and Constantine Stephanidis. Vroamer: Generating on-the-fly vr experiences while walking inside large, unknown real-world building environments. *International Journal of Human-Computer Interaction*, 32(1):45–64, 2016.

[54] Procedurally generated animated films.

[55] Procedural generation for 3d objects and animations.

[56] Wikipedia contributor. Creating visually interesting and accurate spaces rapidly.

[57] Creating short animated movies using procedural generation.

[58] Ensuring consistency and coherence in procedural terrain.

[59] Balancing gameplay in procedural terrain generation.

[60] Striking a balance between realism and variety in procedural terrain.

[61] Performance optimization in procedural terrain generation.

[62] Integration of procedural terrain generation with other game systems.