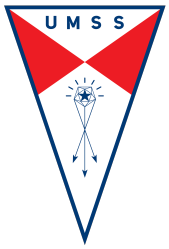
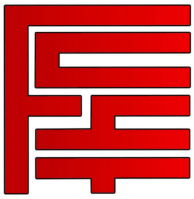


UNIVERSIDAD MAYOR DE SAN SIMÓN
FACULTAD DE CIENCIAS Y TECNOLOGIA
INGENIERÍA INFORMÁTICA



DESARROLLAR UN ASSET PARA FACILITAR LA CREACIÓN PROCEDURAL DE MAPAS TIPO MOSAICO 2D Y 3D A DESARROLLADORES DE VIDEOJUEGOS.

Proyecto de Grado Presentado para optar al Diploma Académico de Licenciatura en Ingeniería Informática

Presentado por: Ríos Cardozo Nicolás Luis

Tutor: Lic. Yony Montoya

COCHABAMBA - BOLIVIA

II, 2025

DEDICATORIA

Dedicado a

AGRADECIMIENTOS

Agradezco a ...

FICHARESUMEN

mi resumen

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO I

INTRODUCCIÓN

El presente proyecto consiste en resolver la necesidad de herramientas mas accesibles en la generación procedural de mapas en el desarrollo de videojuegos

1.1. ANTECEDENTES

El desarrollo de videojuegos es un área en crecimiento mas accesible de ingresar gracias a la facilidad que generan el uso de motores de juegos, y herramientas para estos generados por sus respectivas comunidades.

A pesar de todos los beneficios de usar un motor de videojuegos como base para el desarrollo, hacer videojuegos es actualmente una tarea que demanda de muchos aspectos en los que trabajar por lo que se puede terminar tomando mucho tiempo en terminar de implementar todos los aspectos necesarios que lo involucra

Entre las necesidades mas comunes para un desarrollador de videojuegos es la creación de mapas que puede ser una tarea tardía, por lo que muchos proyecto pequeños optan por la generación automática de estos por diferentes métodos

cita ejemplo (**?**)

1.2. DESCRIPCIÓN DEL PROBLEMA

Un videojuego es un proyecto pesado que incluye muchas áreas a desarrollar entre ellos diseñar el mapa de los niveles así que para aligerar la carga los desarrolladores pueden usar Assets de terceros.

En el desarrollo de videojuegos un Asset son los recursos que se utiliza como modelos 3D, musica, código entre otros. Estos suelen estar publicados para proyectos usando algún motor de videojuegos especifico para ser compatible.

1.2.1. Definición del Problema

Dificultad para la generación de mapas en en videojuegos.

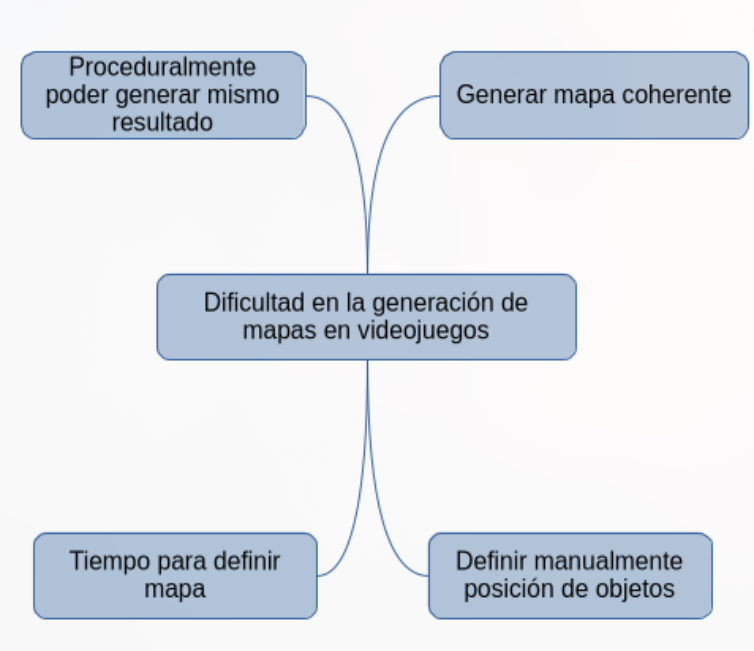


Figura 1.1: Árbol de problemas, centrado el la generación de mapas para videojuegos.

1.3. OBJETIVOS

A continuación se presentan el objetivo general y los objetivos específicos en este proyecto de grado.

1.3.1. Objetivo general

Desarrollar un Asset para facilitar la creación procedural de mapas tipo mosaico 2D y 3D a desarrolladores de videojuegos

1.3.2. Objetivos específicos

1. Investigar algoritmos para la generación de mapas
2. Investigar la técnica de seeding/semillas para añadir control y reproducibilidad de los mapas resultados generados
3. Investigar un motor de videojuegos para implementar un Asset compatible con esa tecnología
4. Implementar la funcionalidad de generación de mapas compatible con las herramientas del motor de videojuegos
5. Definir casos de uso para las pruebas de la funcionalidad
6. Hacer el Asset publico para cualquier desarrollador de videojuegos

1.4. JUSTIFICACIÓN

Como se menciono en puntos anteriores hacer videojuegos es un tarea pesada, entre las necesidades mas comunes para un desarrollador de videojuegos es la creación de mapas, para aligerar tal carga se busca publicar un Asset que ayude a desarrolladores a facilitar esa área del desarrollo de videojuegos.

1.5. LÍMITES Y ALCANCES

El presente trabajo de grado se enfoca en los siguientes aspectos:

-
-
-
-

1.6. METODOLOGÍA DE DESARROLLO

Al momento de avanzar en el procedimiento se definió que el proceso a seguir necesitaría ser un proceso ágil para poder avanzar evitando interrupciones y acomodarse a necesidades surgentes en el desarrollo. Entre los procesos ágiles conocidos se eligió Kanban por el echo de poder amoldarse al trabajo en solitario requerido y ser flexible con flujo de trabajo para evitar posibles atascos. Las tareas principales de investigación que se definió serían las siguientes:

Tabla 1.1: Cronograma de actividades

Nro. Objetivo Específico	Actividades	Recursos Necesarios	Resultados a obtener
1	Investigación de la implementación de algoritmos de generación procedural y seeding		Diseño de algoritmos a usar
2	Investigación del motor a usar y herramientas para generar mapas	lenguaje de programación compatible con el motor	Diseño de uso en el motor y herramientas a usar de este
3	Diseñar y desarrollar biblioteca de generación de mapas	lenguaje de programación compatible con el motor	biblioteca de generación de mapas
4	Desarrollar pruebas para la biblioteca creada	Motor y biblioteca de unit test	conjunto de test de unidad
5	publicar Asset implementado	Motor, biblioteca de generación de mapas	Asset

Gran parte de los puntos no requieren la totalidad de la investigación de puntos anteriores así que se irán generando tareas a cumplir según lo de lo que ya no se tenga bloqueos para su desarrollo.

CAPÍTULO II

MARCO TEÓRICO

En este capítulo se explican los conceptos de los mapas estilo mosaico y las técnicas a usar para poder generarlos procedural-mente, también se profundizara en la tecnología de los motores de videojuegos a usar [1] .

2.1. MAPAS ESTILO MOSAICO O TILEMAPS

Los mapas estilo mosaico o también llamados tilemaps son una técnica común en el desarrollo de videojuegos especialmente en juegos 2D, que consiste en construir el mapa del mundo o nivel de juego a base de pequeñas imágenes con forma usualmente cuadrada a los que se les llaman mosaicos o tiles, los beneficios de usar esto es que no necesitar grandes imágenes que pueden pesar mucho en cambio se construye usando pequeñas imágenes que se pueden repetir varias vecen el diferentes partes del mapa.

Otro beneficio de esto es que se pueden poner en matrices de 2 y 3 dimensiones lo que hace sencillo poder definir las posiciones de los tiles usando algoritmos.

En el caso de mapas 3D se usan modelos 3D o también llamados 3D mesh, en vez de imágenes obteniendo los mismos beneficios.

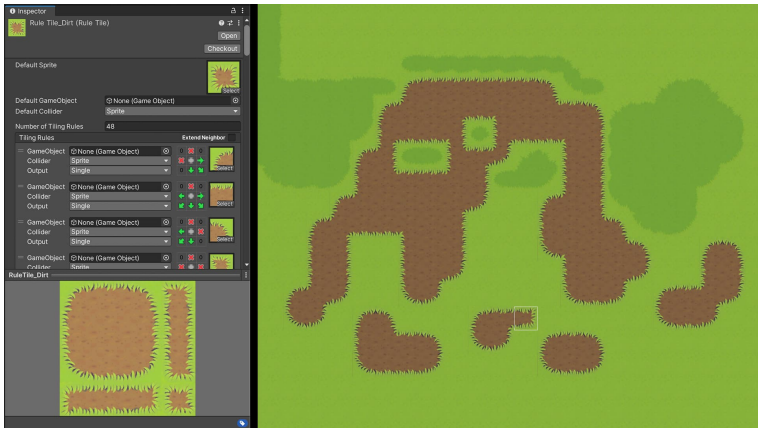


Figura 2.1: herramienta de tilemap de Unity

cada grilla de la matriz del mapa pudiera definirse de forma manual pero esta tarea se volvería muy pesada rápidamente en mapas grandes, por ello se busca implementar una forma de definir la matriz de forma procedural.

[1] footnote: ejemplo

2.2. SCP O PROBLEMAS DE SATISFACCIÓN DE RESTRICCIONES

Para generar un mapa como este es importante que el resultado tenga coherencia, eso significa que hay grillas que si tienen cierto valor deberían de estar rodeadas solo de otros valores coherentes.

Dado lo anterior mencionado generar estos resultado entra en la categoría de problemas de satisfacción de restricciones, estos se definen como problemas matemáticos definidos como un conjunto de objetos tal que su estado debe satisfacer un número de restricciones o limitaciones.

2.3. ALGORITMO AC-3

AC-3 o Arc Consistency 3, es un algoritmo conocido para la solución de problemas de satisfacción de restricciones, es la tercera iteración de esta familia de algoritmos, este en particular desarrollado por Alan Mackworth en 1977

- Se empieza definiendo una lista con 2 valores por cada arista.
- Mientras la lista no este vaciá:
    - remover una arista A a B.
    - Por cada valor del dominio de B:
        - Busca un apoyo a ese valor dada la restricción de la arista
        - Si ningún apoyo es encontrado:
            - Remover el valor del dominio de B
            - Añadir aristas a la lista de B a cada otra variable con la que contenga una restricción aparte de A

2.4. WAVE FUNCTION COLLAPSE

El colapso de la función de onda o wave function collapse es una implementación basada en AC-3 pero inspirado en un concepto de física cuántica con el mismo nombre. Cuando se mide un sistema cuando en superposición de estados su función de onda se colapsa a un solo estado, por lo que se considera el estado es indeterminado hasta que se haga una medición, como referencia un experimento conocidos de este fenómeno es el experimento de la doble rendija.

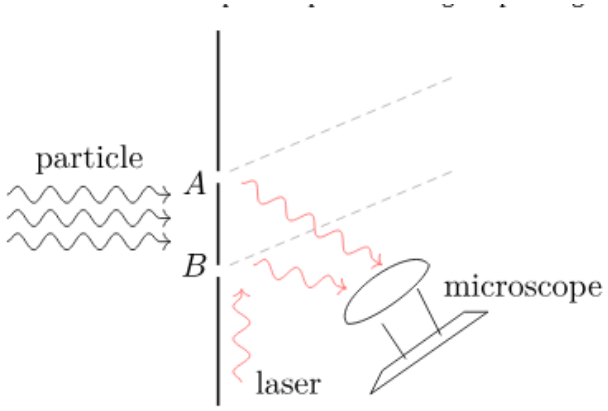


Figura 2.2: experimento de la doble rendija.

La implementación de wave function collapse se usa principalmente para la generación procedural de bitmaps usando un bitmap de ejemplo como entrada

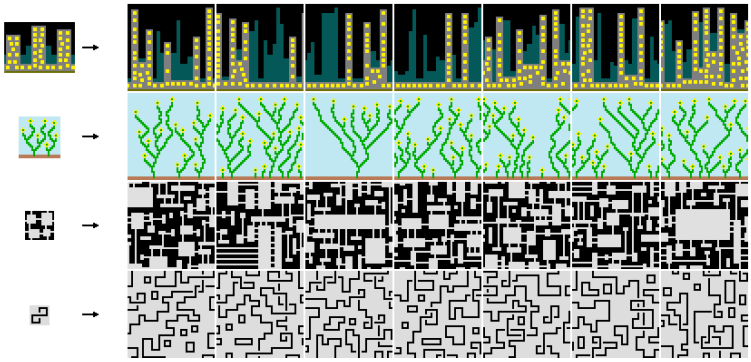


Figura 2.3: wave function collapse ejemplo.

2.4.1. ejemplo subseccion

2.5. MOTORES DE VIDEOJUEGOS

Usar motores de videojuegos como base de un proyecto es actualmente lo mas frecuente en el desarrollo de videojuegos ya que bastante costoso implementar desde 0 todas las funciones que aportan estos.

Por lo que los Assets necesitan ser compatibles con el motor de videojuegos que se elija usar entre ellos las opciones mas conocidas y completas serian Unity y Unreal

Unity y Unreal son motores de videojuegos los cuales han estado dominando la industria por un buen tiempo por lo cual tienen un buen repertorio de Assets publicados dando varias opciones a nuevos desarrolladores, aparte de estos hay varios otros motores entre los cuales uno adquirió popularidad recientemente Godot.

Godot es un motor de videojuegos gratuito de código abierto creado originalmente en Argentina por Ariel Manzur y Juan Linietsky como un proyecto cerrado el cual pasaría a lanzarse como código abierto el 14 de enero de 2014 con licencia MIT Aunque adquirió popularidad rápidamente

el aporte de la comunidad es bastante pequeño comparado con otros motores que llevan mas tiempo encabezando el mercado.

Una gran ventaja de usar Godot por encima de motores como Unity y Unreal es que es completamente gratuito por lo que desarrolladores no necesitan pagar licencias para publicar sus juegos, también que al ser código abierto no esta sujeto a las políticas de ninguna compañía por lo que se puede tener mas libertad creativa.

Otra ventaja de Godot es que al poder usar el mismo lenguaje de programación C# usado en Unity y Unreal estos proyectos serian mas fáciles de exportarse a este nuevo motor de ser necesario y viceversa.

2.6. ESTADO DEL ARTE

2.7. TECNOLOGÍAS

2.7.1. Lenguaje de programación

C#

2.7.2. Godot

Godot

2.7.3.

2.7.4. GIT-GITHUB

2.8. PROCESO DE DESARROLLO

Como se mencionó en el anterior capítulo este proyecto estará desarrollado usando el proceso ágil conocido como Kanban. en los siguientes puntos se profundizará más respecto al proceso y el motivo de su elección

2.8.1. Kanban

2.8.2. Por qué no otros procesos

CAPÍTULO III

MARCO DE APLICACIÓN

3.1. DISEÑO

diseño

3.1.1. subsubtitulo

subdiseño

3.2. IMPLEMENTACIÓN

implementations

3.2.1. Herramientas de software

herramentations

CAPÍTULO IV

DESARROLLO DEL PROYECTO

4.1. DISEÑO DE ARQUITECTURA

Aqui hay un diseño

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. SUBTITULO

Lorem ipsum dolor sit amet

5.1.1. Subsubtitulo

Lorem ipsum dolor sit amet

BIBLIOGRAFÍA