Instituto Tecnológico de Costa Rica

Ingeniería en Computadores

Algoritmos y estructuras de datos 1

Grupo 02

Primer proyecto II semestre

Documentación

“Tron”

Link de repositorio de github: https://github.com/Sofiitaax/Juego-tron.git

Profesor: Leonardo Araya Martínez

Estudiante: Fiorela Sofía González Rubí

Carné: 2024211034

Año: 2024

Tabla de contenido

Introducción

Este proyecto tiene como objetivo principal implementar una solución a un problema utilizando estructuras de datos lineales, como listas enlazadas, pilas y colas, en el desarrollo de un juego inspirado en *Tron*. En este juego, los jugadores controlan motos de luz en un mapa, navegando en cuatro direcciones posibles y dejando tras de sí una estela destructiva. Si una moto cruza la estela de otra, se destruye, lo que añade un desafío estratégico a la mecánica del juego.

La solución propuesta se centra en la implementación de estructuras de datos eficientes que permitan gestionar los elementos del juego, como las estelas, los movimientos de las motos y las colisiones. Además, se fomenta la creatividad mediante el análisis y diseño de algoritmos para resolver problemas clave del juego. El uso de diagramas de clases UML ayuda a modelar el diseño orientado a objetos, mientras que la aplicación de patrones de diseño asegura una solución modular y escalable.

La documentación que sigue abordará los aspectos clave del desarrollo, incluyendo una descripción del problema, una descripción detallada de cada requerimiento, las decisiones de diseño, las alternativas consideradas, y los retos encontrados durante la implementación.

Descripción del problema

El proyecto consiste en la creación de un juego inspirado en *Tron*, donde el jugador controla una moto de luz que se desplaza en un mapa cuadriculado. A medida que las motos se mueven, dejan una estela de luz detrás de ellas que actúa como una barrera destructiva. Si una moto toca una estela, ya sea la suya o la de otro jugador, se destruye. El objetivo es evitar colisiones y sobrevivir mientras se navega por el mapa, recolectando ítems y poderes que ayudan a llegar al gane.

La implementación del juego utiliza estructuras de datos lineales, como listas enlazadas, pilas y colas, para gestionar las diferentes dinámicas del juego. Estas estructuras se emplean para almacenar información clave, como los movimientos y las

posiciones de las motos, los ítems que aparecen en el mapa, y las estelas que dejan las motos a su paso. Las listas permiten gestionar las posiciones y trayectorias de las motos, las pilas manejan los poderes que los jugadores pueden acumular y utilizar, mientras que las colas se emplean para gestionar los recursos como el combustible y los ítems recogidos.

Este enfoque no solo permite implementar el juego de manera eficiente, sino también desarrollar un marco flexible que facilita la manipulación de datos durante el transcurso del juego. Además, la solución incluye la integración de patrones de diseño que mejoran la modularidad y escalabilidad del proyecto, asegurando que futuras mejoras y modificaciones puedan implementarse fácilmente.

Descripción de la solución

Detalles de la implementación de cada uno de los requisitos del proyecto, incluyendo las alternativas consideradas, las limitaciones y los problemas encontrados durante el desarrollo del juego.

Implementación de las motos de luz.

Cada moto se representa como una lista enlazada simple, en la que cada nodo almacena la posición de la moto en el mapa y una referencia al siguiente nodo en la estela. A medida que la moto se mueve, se agrega una nueva posición al frente de la lista y se elimina la última posición, simulando el comportamiento de una oruga. El tamaño de la estela está inicialmente fijado en 3, pero puede crecer si se recogen ítems que lo incrementan.

Una alternativa fue usar una matriz para almacenar las posiciones de las motos en lugar de listas enlazadas. Sin embargo, las listas enlazadas proporcionan mayor flexibilidad para modificar dinámicamente el tamaño de la estela sin necesidad de reservar o liberar grandes bloques de memoria y era lo que se solicitaba en los requerimientos.

Una limitación es que, dado que la estela se gestiona como una lista, la eliminación de nodos es una operación que puede aumentar el consumo de recursos si la estela crece considerablemente.

Implementación de los atributos de las motos: Velocidad, tamaño de la estela, combustible, ítems y poderes.

Velocidad: La velocidad de la moto se genera aleatoriamente al inicio del juego con un valor entre 1 y 10.
Tamaño de la estela: Se inicializa con 3 nodos, pero puede aumentar al recoger ítems que incrementan su longitud con un valor aleatorio entre 1 y 10.

Combustible: Cada moto tiene un valor máximo de combustible de 100, que se reduce a medida que la moto se mueve. El consumo de combustible es de una unidad de combustible por cada 5 celdas recorridas.
Ítems: Se almacenan en una cola y se aplican automáticamente en el orden en que se recogieron, priorizando el combustible.
Poderes: Se almacenan en una pila, permitiendo al jugador reorganizar el orden en que se aplican y ejecutarlos. Tanto la organización como el uso de poder se realiza manualmente por medio de botones integrados en la interfaz.

Hubo dificultades para implementar la sincronización entre el movimiento de la moto y el consumo de combustible, ya que el conteo de celdas no se realizaba de forma correcta.

Implementación del manejo de la distribución de ítems/poderes no utilizados cuando la moto se destruye.

Cuando una moto se destruye, los ítems y poderes no utilizados se dispersan en el mapa en posiciones aleatorias, permitiendo que otros jugadores los recojan. Para manejar esto, se limpian la pila de poderes y la cola de ítems y un con un ramdon se colocan nuevamente en el mapa.

La dispersión aleatoria de fue compleja de lograr observar ya que es muy complejo que cuando una moto se destruya tenga ítems disponibles en la cola.

El mayor problema fue lograr la distribución de los ítems y poderes ya que si se observaba como se eliminaban de la pila, pero no aparecían visualmente en el mapa así que se necesitó abordarlo haciendo uso de la lógica previa de generación de ítems y poderes en el mapa.

Implementación de aplicación de poderes y reorganización de la pila.

Los poderes se almacenan en una pila y se visualizan en la interfaz del jugador. Este puede reorganizar los poderes para aplicar el que más le convenga implementado haciendo con pop, se pasan los elementos temporalmente o una lista y el poder que se sacó se agrega al final de la pila y se regresan. El jugador puede presionar un botón para ejecutar el poder que esté en el tope de la pila, lo que permite controlar sobre el uso de estos recursos de forma estratégica.

Implementación de destrucción de motos por colisiones y falta de combustible

Las motos se destruyen si cruzan una estela, chocan con otra moto o se quedan sin combustible. Las colisiones se detectan al comparar la posición de la moto con las posiciones de las estelas.

Implementación de ítems y poderes aleatorios en la cuadrícula

Los ítems y poderes se generan aleatoriamente en la malla del mapa con cierta probabilidad y cada cierto intervalo de tiempo. Los ítems incluyen celdas de combustible, ítems que aumentan la longitud de la estela, y bombas que destruyen al jugador que las recoge. Los poderes, el escudo y la hipervelocidad, tienen efectos temporales sobre la moto visualmente.

Se consideró la posibilidad de que los ítems se generaran en lugares fijos, pero se decidió que la generación aleatoria sería más interesante para el jugador, al crear un entorno de juego menos predecible.

Fue complejo lograr que se visualizaran las imágenes en el mapa y también que a la hora de recogerlas desaparecieran.

Diseño general (diagrama de clases UML)