Proyecto #1 – TRON

Autor: Bolaños Zamora Sebastián

Curso: Algoritmos Y Estructuras De Datos

Profesor: Jose Isaac Ramirez Herrera

Introducción

El proyecto pretende implementar un juego basado en el clásico "Tron", utilizando estructuras de datos lineales. El juego consiste en un entorno bidimensional, donde las motos dejan una estela detrás de ellas. Los jugadores deben evitar colisionar la estela de los enemigos, o las paredes del entorno.

La implementación de este juego se ha realizado en C#, empleando diversas estructuras de datos como listas enlazadas, pilas y colas para manejar las estelas de las motos, los ítems y poderes disponibles en el juego, así como para gestionar los movimientos y comportamientos de los enemigos controlados por la inteligencia artificial.

El proyecto no solo se enfoca en la creación del juego desde un punto de vista gráfico, sino también en la correcta aplicación de conceptos avanzados de estructuras de datos, lo que permite una mejor optimización y manejo de los recursos durante la ejecución del juego.

Tabla de contenidos

Breve Descripción del Problema

El problema a resolver en este proyecto consiste en la implementación de un juego de Tron que debe manejar múltiples elementos en tiempo real, garantizando la fluidez y la correcta interacción entre ellos. Los principales desafíos incluyen:

- **Gestión de Estelas**: Cada moto deja una estela a medida que se mueve por el entorno. Estas estelas deben ser gestionadas como listas enlazadas, permitiendo su crecimiento y contracción según las reglas del juego.
- **Movimiento de Motos y Enemigos**: Las motos, tanto la del jugador como las controladas por la inteligencia artificial (enemigos), deben moverse de manera independiente, siguiendo reglas específicas de movimiento y evitando colisiones. Cada moto debe manejar sus propios recursos y comportamientos, incluyendo la ejecución de poderes especiales.
- **Manejo de Ítems y Poderes**: Los ítems y poderes que aparecen en el juego deben ser manejados mediante estructuras de datos adecuadas. Los ítems se gestionan con una cola, aplicándose automáticamente cuando se recogen, mientras que los poderes se gestionan con una pila.

Descripción de la solución

Requerimiento 001: Moto de Luz como Lista Enlazada

- **Descripción**: La moto se implementó como una lista enlazada simple, donde cada nodo representa una celda ocupada por la estela.
- **Alternativas Consideradas**: Se consideró separar la moto de la estela.
- **Limitaciones**: La gestión de memoria se vuelve crítica a medida que la estela crece.
- **Problemas Encontrados**: La gestión de la estela para que siguiera el movimiento de la moto.

Requerimiento 002: Atributos de la Moto

- **Descripción**: La moto tiene velocidad, tamaño de estela, y combustible. Estos atributos son variables controladas por métodos específicos.
- **Alternativas Consideradas**: Usar constantes en lugar de variables dinámicas.
- **Limitaciones**: Solo se puede apreciar un poder a la vez
- **Problemas Encontrados**: La interacción de la moto con los objetos del mapa se volvió muy tediosa por la rigidez de tipado de C#.

Requerimiento 003: Moto al ser destruida

- **Descripción**: Cuando una moto es destruida esta suelta aleatoriamente por el grid los poderes que tenía en el inventario.
- **Alternativas Consideradas**: Generar nuevos poderes a partir de los que había en la lista.
- **Limitaciones**: La interacción de los enemigos con esta función.
- **Problemas Encontrados**: Los poderes se estaban generando sobre objetos previamente creados en el grid.

Requerimiento 004: Pila de poderes

- **Descripción**: Los poderes están guardados en una lista de tipo pila, estos se ejecutan al presionar la tecla “e” y se cambia su orden con la tecla “r”.
- **Alternativas Consideradas**:
- **Limitaciones**: La ejecución de 2 poderes al mismo tiempo no se nota visualmente debido a que solo se ve uno de los dos que se están ejecutando.
- **Problemas Encontrados**: Como identificar qué tipo de poder se quería ejecutar.

Requerimiento 005: Cola de ítems

- **Descripción**: Los ítems están guardados en una lista de tipo cola, estos se ejecutan automáticamente mediante un bucle de un segundo.
- **Alternativas Consideradas**:
- **Limitaciones**: El bucle esta siempre corriendo de fondo, aunque la cola no tenga elementos.
- **Problemas Encontrados**: Como identificar qué tipo de ítem se quería ejecutar.

Requerimiento 006: Destrucción de moto

- **Descripción:** Cuando la moto se mueve a un nodo el cual está ocupado por una estela, otra moto, el borde del grid, si se queda sin combustible o ejecuta una bomba esta se destruye. La moto esta en un bucle infinito en el cual se mueve automáticamente hacia el siguiente nodo, el cual el jugador puede cambiar, pero no detener.

- **Alternativas Consideradas**: Dejar que el jugador los mueva manualmente.
- **Limitaciones**: Los enemigos tienen el mismo método de destrucción que la moto.
- **Problemas Encontrados**: La interacción de la moto con ella misma.

Requerimiento 007: Grid

- **Descripción**: Para el grid se tienen 3 elementos, el nodo, el cual tiene los datos de cada celda del grid; la lista enlazada, esta conecta todos los nodos en 4 direcciones y por último el grid como tal, este se encarga de la utilizar lo anterior para crear un grid del tamaño deseado.
- **Alternativas Consideradas**: Hacer que los nodos tuvieran datos genéricos.
- **Limitaciones**: Los nodos no tienen datos genéricos, por lo que sus variables son fuertemente tipadas.
- **Problemas Encontrados**: La variable “ocupante” la cual almacena el ocupante de cada nodo no se logró hacer genérico.

Requerimiento 007: Grid

- **Descripción**: Para el grid se tienen 3 elementos, el nodo, el cual tiene los datos de cada celda del grid; la lista enlazada, esta conecta todos los nodos en 4 direcciones y por último el grid como tal, este se encarga de la utilizar lo anterior para crear un grid del tamaño deseado.
- **Alternativas Consideradas**: Hacer que los nodos tuvieran datos genéricos.
- **Limitaciones**: El grid se puede volver muy pesado.
- **Problemas Encontrados**: La variable “ocupante” la cual almacena el ocupante de cada nodo no se logró hacer genérico.

Requerimiento 008: Ítems y poderes

- **Descripción**: Se creo una super clase de métodos para manejar las imágenes de estos objetos. Los ítems y poderes están divididos en 2 clases distintas para poder manejarlos por separado.
- **Alternativas Consideradas**: Hacer los ítems y poderes en una sola clase.
- **Limitaciones**: El método de identificación de ítems y poderes.
- **Problemas Encontrados**: La moto no ejecutaba correctamente el ítem o poder que había recogido debido a que no los lograba identificar.

Requerimiento 009: Los enemigos

- **Descripción**: Estos se heredan la clase moto, pero se les modifico el movimiento para que sean autónomos.
- **Alternativas Consideradas**: Crear una clase separada para los enemigos.
- **Limitaciones**: La IA de los enemigos no es nada avanzada.
- **Problemas Encontrados**: Los enemigos y la estela se compartían la misma estela.

Diseño general