

ITCR

Ingeniería en Computadores

Algoritmos y Estructuras de Datos 1

Documentación Proyecto 1

Estudiante: Jafet Diaz Morales - 2023053249

II Semestre 2024

Tabla de Contenidos

1. Introducción y Breve descripción del problema.....	3
2. Descripción de la solución	4
3. Diagramas UML	5-10

Introducción

En este documento se presenta la implementación del juego Tron en Unity con C# con el fin de hacer un buen uso de estructuras de datos, tales como listas, colas y pilas. Además, el proyecto tiene el fin de fomentar la creatividad y las buenas prácticas de programación (documentar, uso de UML, etc.)

Breve Descripción del Problema

Tron es un juego de carreras de motos de luz con varios añadidos (como items). El mapa consiste en una malla en la que las motos de luz pueden navegar en 4 direcciones posibles sin detenerse. El jugador maneja una moto de luz (utilizando las flechas del teclado) en la malla. Las motos dejan una estela destructiva a su paso. Si otra moto de luz cruza una estela dejada por otra moto de luz, ésta se destruye. Habrá más motos además de la moto del jugador las cuales pueden ser una amenaza para él.

Descripción de la solución

Para realizar las motos se usó una clase llamada Player, la cual tendrá dentro de sus atributos una lista enlazada simple “estela”. Esta lista se irá generando en la pantalla conforme el jugador se mueva, asemejando que lo está siguiendo. El jugador nunca para de moverse sino que con los inputs de las teclas se le cambia la dirección. Los bots son iguales al jugador en excepción de su cambio de dirección, que se hace con un número generado de forma aleatoria cada cierto tiempo. El jugador presentó problemas como la rotación del mismo al moverse. Esto logró arreglarse al verificarse en qué dirección está mirando previo a rotarlo. La estela también presentó dificultades, pero se logró implementar mediante la eliminación del último nodo y agregar uno nuevo.

Para la velocidad de la moto, esta se definió como un atributo con valor aleatorio, la estela como una lista que puede almacenar cierta cantidad de “estelas” (tres espacios), el combustible se implementó como un atributo con un valor flotante. Los ítems y poderes son listas (cola y pila) que se crean por otras clases (un instanciador para ítems y poderes) pero se le asignan como atributos al jugador. Esto fue hecho como una alternativa para no sobrecargar más el código presente en la clase jugador y tener como verificar de forma más sencilla si las estructuras de datos se estaban generando correctamente.

Cuando la moto choca (excepto con los ítems/poderes) o se queda sin combustible se destruye. Por errores con la aparición de los objetos y colisiones las motos enemigas no almacenan objetos. La moto del jugador no lo hace porque el juego ya ha acabado en dado caso.

El jugador puede decidir qué poderes usar. Idealmente los poderes se iban a almacenar en una pila y con una tecla el jugador puede “recorrer” esa pila (en realidad se usa una pila auxiliar para almacenar y obtener el elemento deseado). En esta etapa hubo muchas dificultades, errores de “index error” por ejemplo. Iba a utilizarse un traspaso entre el stack y el stack auxiliar. Sin embargo por estos errores decidió manejarse un orden fijo en el uso de objetos (que depende de las teclas).

Los ítems tienen un número asignado que sirve como dato que los diferencia. Cuando el jugador los recoge, se verifica que número tiene y se aplica el efecto, además se añaden a la cola pero se ejecutan inmediatamente (y se eliminan). Si el combustible está lleno no se aplica el ítem sino que se mantiene en la cola. Tanto ítems como poderes aparecen de forma aleatoria en el mapa pero siempre dentro de los límites del grid.

Se generó una lista genérica y un nodo genérico, sin embargo, para el grid se creó una lista y nodo específicos para su creación. El nodo tiene 4 direcciones y mediante manejo de listas se asignan las direcciones de los nodos para crear la malla. Se inicia el grid y posteriormente se dibuja accediendo a cada nodo específico.

Diagramas UML

A continuación se muestran los diagramas UML con las clases más importantes del proyecto. Hay cosas como el instanciador de enemigos (solo es una función propia de Unity) que no se registra y otras cosas como las colisiones que también las provee Unity

LinkedList <T>	
Atributos	Metodos
SimpleNode <T> head int size	Insertar I Insertar F Eliminar I Eliminar F SimpleNode <T> Get(int x) LinkedList()

Grid Linked List	
Atributos	Metodos
Node head int gridSize	GridLinkedList(int size) Iniciar Grid() Node GetNode(int x, int y)

Cola <T>	
Atributos	
LinkedList<T> list	
Metodos	
cola() enqueue (T data) dequeue () peek()	

Stack <T>	
Atributos	
LinkedList<T> list	
Metodos	
stack() push (T data) pop () peek()	

Player

Atributos

Vector2 gridposition, target position, moveInput

Boolean isFacingUp, isFacingDown, isFacingLeft,
isFacingRight, isMoving

float xInput, yInput, speed, combustible

Rigidbody2D player RB

Stack <T> stack poder, stack auxiliar

LinkedList <T> estelaLuz, sprite Estela

Cola <T> item cola

int estelaSize

Random random

Metodos

Start()

Update()

Direccionar()

FixedUpdate()

Limites()

Generar estela()

CalcularTargetPosition()

OnDrawGizmos()

flip()

OnCollisionEnter2D(Collision2D
collision)

ItemSpawner
Atributos
GameObject item
Vector2 itemPosition
Metodos
Start()
Update()
Instantiate()

GridMap	
Atributos	Metodos
Vector2Int itemPosition	Start()
int gridSize	Update()
GridLinkedList grid,	DibujarGrid()

Poderes Spawner	<T> T[] poder
Atributos	
GameObject poder	GameObject poder
Vector2 poderPosition	
Instantiate()	Instantiate(poder, Stack poder)
Metodos	
Start()	
Update()	

Simple Node	
Atributos	Metodos
T dato	SimpleNode(T dato)
SimpleNode<T> Next	
Node (para el grid)	
Atributos	Metodos
int x, y	
Node Up, down, Right, Left	Node (int x, int y)

Enemy

Atributos

Vector2 grid position, target position, move Input

Boolean isFacingUp, isFacingDown, isFacingLeft,
isFacingRight, isMoving

float combustible, speed, Input

Rigidbody2D enemyRB

Stack <T> stack poder, stack auxiliar

LinkedList <T> estelaLuz, sprite Estela

Cola <T> item cola

int estelaSize

Random random

Metodos

Start()

Generar estela()

Update()

CalcularTargetPosition()

Direccionar()

OnDrawGizmos()

FixedUpdate()

flip()

Limites()

OnCollisionEnter2D(Collision2D
collision)