Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores

Curso: CE-1103

Algoritmos y estructuras de datos I

Primer Proyecto

Tron

Profesor:

Leonardo Araya Martínez

Estudiante:

Dylan Guerrero González - 2022016016

Grupo:

1

II Semestre, 2024

El presente proyecto tiene como objetivo la implementación de las diferentes estructuras de datos lineales estudiadas en el curso mediante el desarrollo de un juego basado en la película Tron. En dicha película los personajes luchan grupalmente en una arena mediante el uso de motos, las cuales son llamadas motos de luz; dichas motos dejan una estela destructiva a su paso y si otro personaje colisiona con ella muere al instante.

El proyecto presentado seguirá una mecánica similar mediante la inclusión de un personaje jugable y 4 bots, los cuales se presentan en una arena de tamaño fijo y deben de buscar sobrevivir el mayor tiempo posible. Además, también se incluirán elementos como ítems los cuales afectan permanente la moto y poderes que afectan temporalmente la moto. Entre las estructuras lineales empleadas se encuentran: listas enlazadas, pilas y colas de prioridad; dichas estructuras serán implementadas enteramente por el estudiante con ayuda de los conocimientos obtenidos en clases y, de ser requerido, ayuda de recursos en línea

En cuanto al desarrollo del proyecto en sí, se empleará el lenguaje de programación C# y la interfaz gráfica se realizará empleando el motor de videojuegos Unity, con relación al control de versiones, aspecto muy importante al momento de realizar proyectos programados, se hará mediante la plataforma GitHub la cual se apoya en Git para realizar dicho control.

Contenido

1. Descripción del problema 4

2. Descripción de la solución 5

2.1 Moto de luz 5

2.2 Atributos 6

2.3 Ítems y poderes sin usar 6

2.4 Aplicación de poderes 6

2.5 Aplicación de ítems 6

2.6 Comportamiento de la moto 6

2.7 Grid o malla 7

2.8 Ítems y poderes 7

2.9 Bots 7

2.10 Programación 8

2.11 Estructuras de datos 8

3. Diseño general 9

1. Descripción del problema

Se busca realizar un juego basándose en la película Tron. Entre las características se encuentran: un personaje jugable, 4 bots, consumibles que pueden afectar temporal o permanentemente a los jugadores y una malla sobre la que se desplazan los jugadores

Las formas de implementación de las diversas características se detallan a continuación: la moto del jugador y los bots deben de ser una lista enlazada simple; los ítems, que afectan permanentemente a los jugadores, como una cola de prioridad; los poderes, que afectan temporalmente, como una pila y la malla debe de ser una lista enlazada con referencias a sus cuatro nodos vecinos (se puede pensar como una lista enlazada múltiple). Todos estos tipos de implementación han de ser programados en su totalidad por el estudiante, sin ayuda de clases propias del lenguaje utilizado. Para la interfaz grafica del proyecto se presenta la posibilidad de ser realizada en Windows Forms, MAUI o Unity, siendo esta última opción la elegida, dicha decisión se explicará más adelante.

2. Descripción de la solución

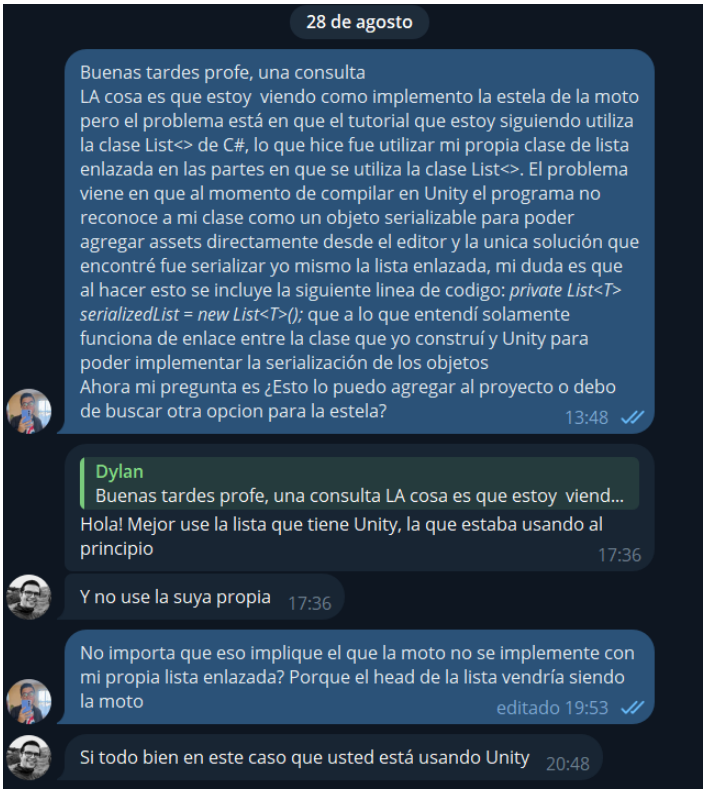
En esta sección se expondrán los aspectos relevantes de cada uno de los requerimientos del proyecto tales como la manera en la que fue realizado, las alternativas que se barajaron, limitaciones de la opción escogida y los problemas relacionados con esta.

2.1 Moto de luz

Fue realizada mediante el uso de la lista por defecto de C#, esto debido a que el tutorial que se siguió requería que los diferentes elementos de la lista pudieran ser accesados desde el editor de Unity y que de esta forma se pudiera ingresar el prefabs correspondientes a la estela y la moto, así como determinar el tamaño inicial de la primera. Para poder realizar dicho cambio se consultó al profesor, como se puede observar en la Figura 1, sobre que posibilidad existía de hacer dicho cambio o si se debía de seguir la instrucción de que esto fuera hecho mediante una lista enlazada simple, consulta a la cual respondió que se realizará la implementación que se expuso al inicio.

Figura 1

Comunicación con el profesor donde se le expone la inquietud sobre la forma de realizar la moto de luz.



2.2 Atributos

Estos se representan como variables propias de la moto las cuales manejan el comportamiento de esta y activan métodos beneficiosos o negativos para el jugador. De todos los atributos descritos en los requerimientos únicamente no fue posible incorporar la velocidad debido a incompatibilidad con la programación del proyecto. En cuanto al tamaño de la estela y el combustible, estos crecen y decrecen respectivamente a la misma velocidad que para este caso se establece en 5 elementos de malla.

2.3 Ítems y poderes sin usar

No fue posible cubrir este requerimiento.

2.4 Aplicación de poderes

El desarrollo de la interfaz relacionada con este aspecto no fue posible, en su lugar se tiene un delay de diez segundos entre la aplicación de un poder y el siguiente.

2.5 Aplicación de ítems

Esta característica se encuentra plenamente funcional. La forma en que funciona se basa en el principio de cola de prioridad, con ayuda de dicho principio se permite que el combustible siempre se *desencole* prioritariamente y de dicha manera sea siempre aplicado de primero sobre los demás; los demás ítems contienen una prioridad más baja y por ende se *desencolaran* con menor prioridad. La única característica que no se encuentra en funciones es la relacionada con la puesta en cola de las celdas de combustible si este valor está en su máximo.

2.6 Comportamiento de la moto

Esto se logró mediante el uso de los componentes `BoxCollider2D` y `RigidBody2D` presentes en Unity los cuales agregan hitbox y físicas a los objetos, de dicha forma se logró que tanto la estela como las motos tengan un área de impacto y un "cuerpo físico", dichas características son registradas por el método `OnTriggerEnter2D()` el cual activa la función relacionada con uno de los tags presentes en los objetos en pantalla. Como recurso adicional también se agregó un muro alrededor de la malla para que así los jugadores no pueden desaparecer del mapa y, de intentar esto, también son destruidos. Para la dirección se utiliza una variable del tipo Vector2 que siempre apunta en una dirección en concreto y

así las motos nunca están detenidas, los cambios de dirección se hacen gracias al método `GetKeyDown` del motor de desarrollo el cual detecta que se presionó una de las teclas de flecha y modifica la dirección de la variable.

En relación con la destrucción de la moto cuando el combustible llega a 0, se hace con ayuda de: la variable asociada al combustible, un método que cuenta los nodos que han sido recorridos, otro que compara si los nodos recorridos son iguales a 5 para rebajar una unidad de combustible y un último método que llama a `Reset()` cuando el parámetro del combustible es igual a 0.

2.7 Grid o malla

La parte más importante del juego junto con la moto. Es creada mediante lo que se puede llamar una lista enlazada cuádruple, esto porque cada nodo tiene referencias a sus cuatro nodos vecinos. Si bien existen otras opciones más sencillas para crear un objeto de este tipo, en este caso al ser una lista enlazada la situación es más complicada pero de igual forma pudo ser realizado con éxito, la parte mas complicada fue a la hora de renderizar el objeto, ya que Unity elige como origen del sistema de coordenadas el centro de la pantalla y por esta razón fue necesario realizar cálculos para corregir el desplazamiento, con ayuda de dichos cálculos se pudo lograr que la malla renderizara en toda la pantalla en lugar de que solo se observara en la porción superior derecha. Para el tamaño se utilizan variables modificables que determinan cuantas filas y columnas deben de existir para luego crear nodos en dicha matriz y por último conectar cada uno de ellos con sus vecinos.

2.8 Ítems y poderes

Cada uno de ellos son prefabs que cuentan con el componente `BoxCollider2D` que ocasiona que al entrar en contacto con el jugador o los bots desaparezcan y vuelvan a aparecer en una posición aleatoria de la malla. Por debajo, el método `OnTriggerEnter2D()` llamada a la función encargada de hacer push o enqueue para almacenar cada uno de ellos en su respectiva estructura y que luego puedan ser utilizados por los jugadores.

2.9 Bots

Estos son una clase que hereda de la clase `Moto` y así permitir reutilizar código a la vez que se ahorra memoria al no crear una nueva clase con métodos redundantes. Gracias a

la herencia se logra que todos los comportamientos sean iguales con excepción de la dirección, en este caso se realiza una sobreescritura (override) al método `Update()` de la clase moto y así se reemplaza la utilización de las teclas por un condicional que recibe un dato aleatorio, en este caso de tipo entero y en un rango de 1 a 4, cada cierto tiempo y basándose en el dato cambia la dirección de moto.

Se intento aplicar otros tipos de comportamiento con la utilización de `NavMesh` que es una herramienta de Unity especializada en el mapeo de obstáculos, sin embargo, por la forma en que la moto fue desarrollada, resultó imposible que esta herramienta funcionara adecuadamente. Otro erro se encuentra en la rotación de la estela, al inicio fue imposible hacerla coincidir con el movimiento de la moto y luego, cuando fue posible que ambos movimientos coincidieran, no se logró que la estela creara un rastro detrás de la moto del bot a pesar de que en el caso del jugador si fue posible.

2.10 Programación

El proyecto fue desarrollo enteramente en C# y para la interfaz gráfica se eligió utilizar el motor de videojuegos Unity. La elección de este motor no fue aleatoria, en su lugar se debió a dos factores: su integración con el lenguaje C#, el cual es un requisito indispensable y la gran cantidad de apoyos que posee en cuanto al desarrollo de videojuegos tales como la presencia de motores de física, la opción de previsualizar el proyecto sin necesidad de realizar una compilación final, ahorrando tiempo y recursos y la presencia de sprites predefinidos que solo requieren arrastrar y/o modificar su tamaño de forma mucho más sencilla y sin necesidad de cambiar variables en el código.

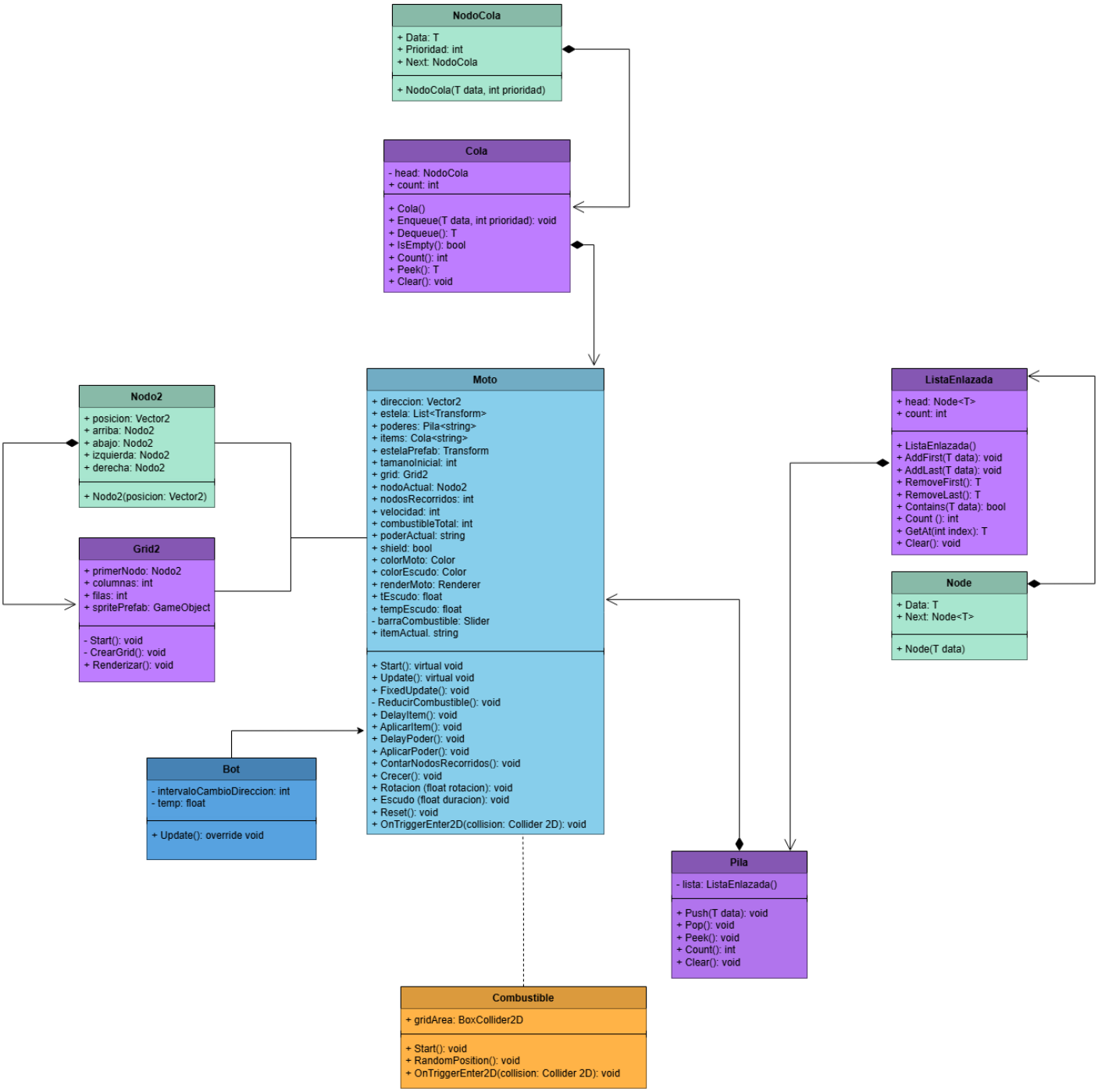
2.11 Estructuras de datos

Como bien se detalla en los requerimientos del proyecto, estas características fueron programadas enteramente por le estudiante con apoyo de los materiales del curso y tutoriales en línea. En la Figura 2 se puede observar el diagrama de clase de cada una de las estructuras solicitadas.

3. Diseño general

Figura 2

Diagrama de las distintas clases presentes en el proyecto



Nota. El archivo también se encuentra en el repositorio del proyecto en formato PNG y SVG