TEC | Tecnológico de Costa Rica

Primer Proyecto

Escuela de Ingeniería en Computadores

Algoritmos y Estructuras de Datos I

Profesor:
Leonardo Araya

Estudiante:
Paula Melina Porras Mora | 2023082886

23 de septiembre del 2024

Tabla de contenidos

Introducción

El objetivo del proyecto es desarrollar un juego de carreras de motos con experiencia entretenida para los jugadores, además sirve como ejercicio integral aplicando conceptos básicos de programación. En particular, se centra en el uso de estructuras de datos lineales, el diseño de algoritmos eficientes y la gestión del código fuente mediante GitHub.

Estructuras de Datos Lineales: El juego emplea diversas estructuras de datos lineales, como listas, pilas y colas, para gestionar de manera efectiva los elementos del juego y las interacciones entre ellos. Las listas se utilizan para almacenar y manipular colecciones de objetos como los ítems del juego, las motos y los bots, las pilas se aplican para gestionar eventos en orden inverso, como las acciones de deshacer en el juego, y las colas se emplean para manejar la secuencia de eventos o la cola de acciones de los bots y jugadores.

Diseño de Algoritmos: La implementación del juego requiere el diseño de algoritmos eficientes para manejar diversas funcionalidades, como la actualización de la posición de las motos, la detección de colisiones y la aplicación de los efectos de los ítems recogidos, el diseño de estos algoritmos se enfoca en la eficiencia y en la correcta implementación de la lógica del juego.

Uso de GitHub: GitHub se utiliza como plataforma principal para la gestión del código fuente del proyecto. Con GitHub, se facilita el control de versiones y documentación del progreso del proyecto, la integración con GitHub permite rastrear los cambios en el código, gestionar ramas para nuevas funcionalidades y solucionar problemas, y mantener un historial detallado de las modificaciones realizadas. Además, el uso de GitHub asegura una copia de seguridad centralizada y accesible del proyecto, fomentando las mejores prácticas en el desarrollo de software.

Por lo que en este proyecto no solo se busca desarrollar un juego atractivo y funcional, sino también aplicar y reforzar conocimientos clave en programación, diseño de algoritmos y gestión de proyectos mediante herramientas modernas, la combinación de estas prácticas contribuirá a un desarrollo más organizado y eficiente.

Descripción del problema

El proyecto tiene como objetivo desarrollar un juego basado en carreras de motos, donde los jugadores deben competir en un entorno lleno de obstáculos y potenciadores. Cada moto deja una estela que se extiende de acuerdo con las interacciones del jugador con el entorno (recoger items), el juego incluye items como escudos, bombas y poderes especiales que modifican la velocidad y la invulnerabilidad de los jugadores durante cierto tiempo.

Además del jugador principal, el juego también integra bots, que simulan el comportamiento de otros jugadores y están sujetos a las mismas reglas del juego, estos actúan de manera aleatoria para aportar un mayor desafío a las partidas, garantizando que siempre haya al menos cuatro oponentes controlados por la máquina.

El desarrollo del proyecto se realizó en C# con la interfaz gráfica Unity, para ofrecer una experiencia visual atractiva y fluida.

Descripción de la solución

A lo largo del desarrollo del juego, se implementaron diversas soluciones para abordar los problemas técnicos relacionados con la gestión de elementos del juego, las mecánicas de interacción, y demás. A continuación, se detalla cómo se solucionaron los problemas mediante el uso de estructuras de datos lineales, algoritmos, y la integración con GitHub para la gestión del código.

- Se utilizaron listas para gestionar y almacenar elementos dinámicos del juego, como las motos, los ítems y los bots. Cada vez que un jugador interactúa con un ítem o un bot, las listas permiten añadir o eliminar elementos de manera eficiente.
- Colas se usaron para gestionar las secuencias de eventos en el juego, como la recolección de ítems en el orden en que se presentan. Los ítems, al ser recogidos, son procesados y luego eliminados de la cola de ítems disponibles en el mapa.
- Las pilas se emplearon para implementar un sistema de "deshacer" en ciertas mecánicas del juego.
- Se desarrollaron clases como GameManager, Player, Item, y Bot, que se encargan de coordinar los elementos del juego y utilizar las estructuras de datos mencionadas.
- Se crearon clases específicas para cada ítem del juego,cada una de estas clases incluye atributos que definen su comportamiento (duración, efecto visual, y estado) y métodos que se activan al ser recogidos por los jugadores o bots.
- Los efectos de los ítems se manejan a través de métodos de la clase Item, que se llaman cuando el jugador o bot interactúa con ellos. Las actualizaciones visuales se integraron con el motor gráfico de Unity, mientras que las lógicas de juego se implementaron en C#
- Cada bot tiene una clase propia (BotController), que gestiona sus decisiones y su interacción con el mundo del juego.
- Se utilizó **GitHub** para el control de versiones, a través de este se creó un flujo de trabajo que incluye ramas dedicadas a nuevas funcionalidades y correcciones de errores, lo que permitió el desarrollo simultáneo de varias características sin comprometer la estabilidad del proyecto.
- GridManager gestiona la creación y representación del grid del juego. El grid se implementa utilizando nodos (Nodo2) conectados entre sí, formando una estructura bidimensional.
- Al inicio del juego, se crean los nodos del grid y se almacenan en un diccionario para un acceso rápido a través de coordenadas. Cada nodo está conectado a sus nodos vecinos (arriba, abajo, izquierda, derecha).

- Se utiliza el método `Renderizar` para instanciar visualmente cada nodo en el grid utilizando un prefab.
- Nodo2 representa una celda del grid, conteniendo información sobre su posición en el espacio y referencias a los nodos vecinos. Esta clase es fundamental para definir la estructura del grid y permite que los objetos (como jugadores y bots) naveguen por el mismo.
- PlayerControl: Controla el movimiento de los jugadores en el grid. Los jugadores se desplazan en una dirección continua y pueden cambiar su dirección basándose en la entrada del teclado. La clase maneja la lógica de la estela destructiva que dejan las motos, gestionando la adición y eliminación de segmentos de la estela en función del movimiento. La clase también contiene métodos para recolectar ítems y activar poderes como el escudo y la hiper velocidad.
- BotController: Hereda de `PlayerController` y añade funcionalidades específicas para los bots. Los bots cambian de dirección automáticamente después de un intervalo de tiempo o al chocar con un obstáculo o el borde del grid. Se asegura de que los bots no se salgan del grid verificando la validez de las nuevas direcciones antes de moverse.
- Item: Representa los ítems que pueden ser recolectados por los jugadores o bots. Los ítems se colocan aleatoriamente en el grid al inicio del juego y se reposicionan tras ser recolectados.
- Los poderes aparecen aleatoriamente en el grid y permiten activar habilidades especiales en los jugadores, como el escudo o la velocidad aumentada.

Diagrama UML

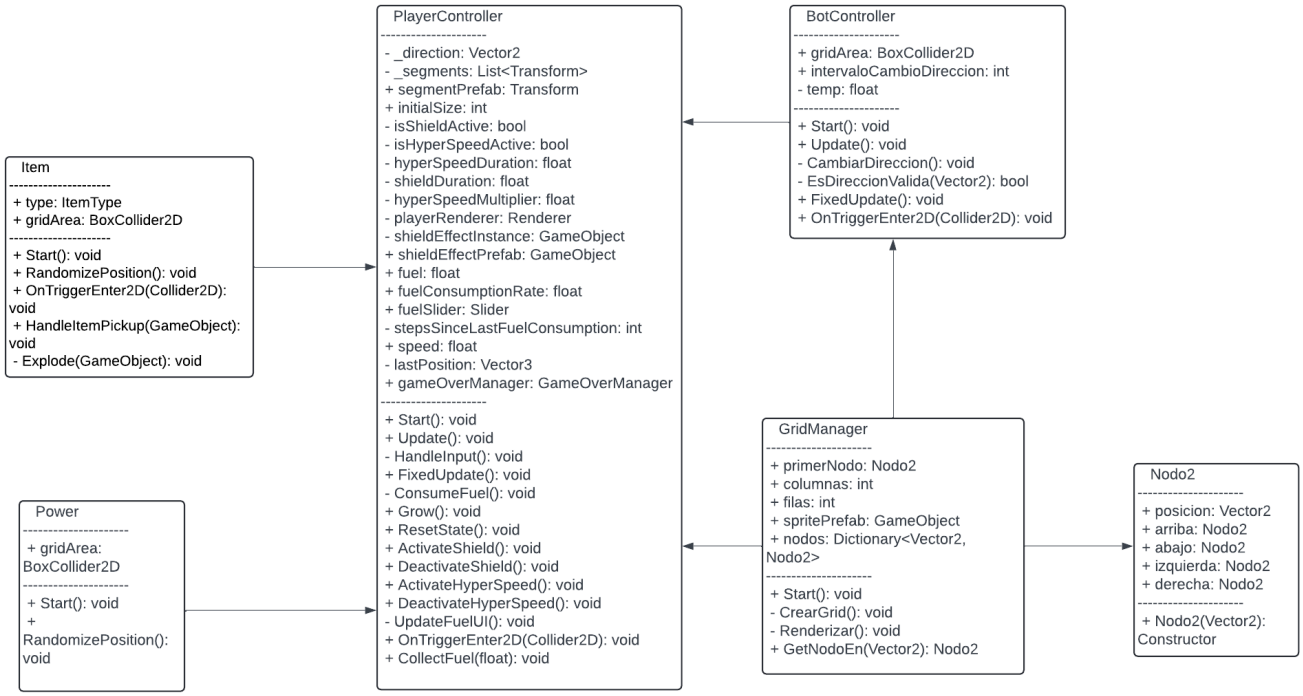

Fig. 1. Diagrama UML

Enlace a GitHub

lyna006/Proyecto_Tron: 2D Game, inspired by Tron movie (github.com)