Practica 1
Simulador de conducción en línea

Diseño y Desarrollo de Videojuegos - Programación Gráfica

Promoción 2023-2024

Ivan Rodriguez Lozano. Grupo 4.3

Índice

Descripción

Este documento proporciona una memoria detallada del desarrollo de un videojuego que pretende ser un simulador de carreras, con implementación online, destacando los elementos clave del juego, la implementación y los desafíos encontrados.

El juego tiene dos modalidades distintas: una local, en la que el jugador competirá en una carrera contra la inteligencia artificial, y otra en la que se podrá conectar a un servidor para jugar con otros jugadores en un entorno tipo sandbox.

Componentes del juego

Coche jugador
El jugador deberá elegir el coche con el que desea competir, tanto en la modalidad local como en la online. El coche estará equipado con un controlador que utilizará elementos de Unity, como los Wheel Colliders y el Rigidbody, para moverse, además de usar las teclas WASD como método de entrada.

Coche IA
El coche controlado por la IA utiliza el mismo controlador que el del jugador, pero ajustará la aceleración, el frenado y la dirección siguiendo una serie de waypoints colocados en el circuito. Además, contará con un sistema de raycast para evitar obstáculos.

Escena OffLine
Esta es la escena donde el jugador corre contra la IA. Antes de comenzar el jugador elegirá su coche el número de vueltas y el número de oponentes, los cuales, se seleccionarán aleatoriamente de todos los coches que hay.
Una vez le das al play se activa una cuenta atrás y comienza la carrera, que finaliza cuando algún coche gane.

Escena OnLine
En esta escena los jugadores se conectarán a un servidor con una IP y puerto específicos. Una vez dentro se representa a cada jugador en tiempo real en la escena, donde encontrarán diferentes retos y escenarios.

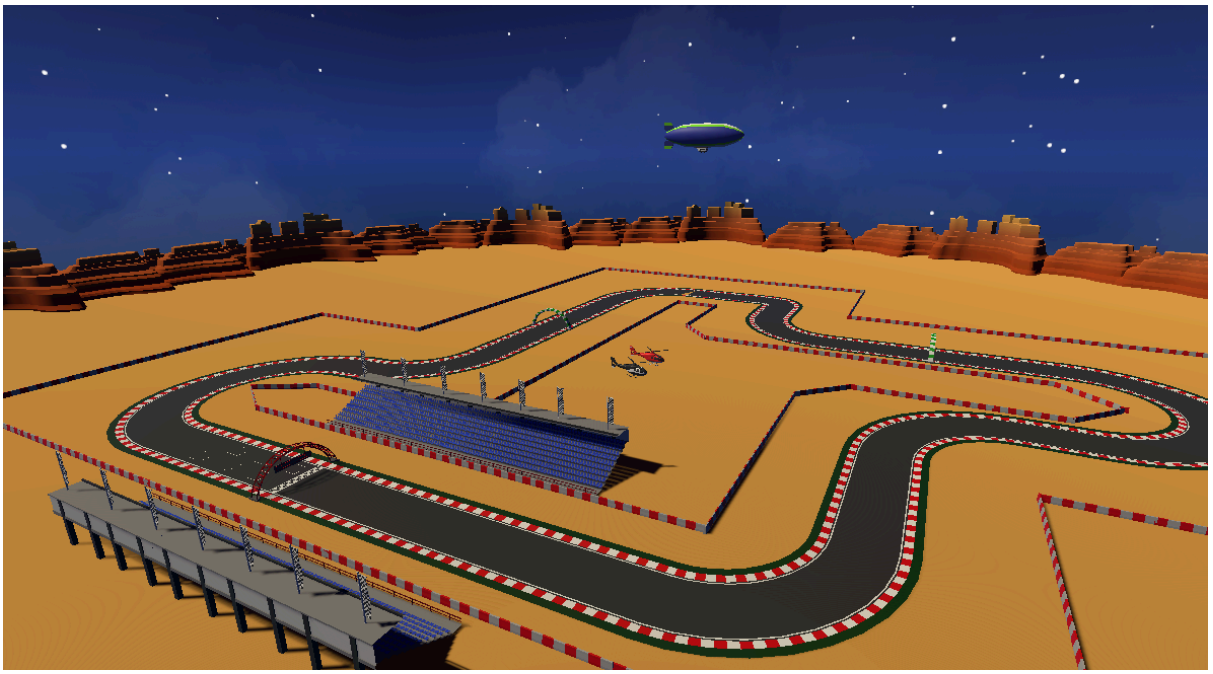
Cámara
La cámara seguirá al coche del jugador en todo momento, dando una perspectiva de 3º persona.

Interfaz
Se podrá navegar entre escenas, gracias a los menús implementados en el juego.

En el menú principal podrás elegir qué modo de juego quieres probar.

En el menú de un jugador, antes de empezar la carrera, podrás elegir el coche que quieras entre los que hay disponibles, elegir el número de vueltas y el número de oponentes. Una vez iniciada la carrera, podrás ver en pantalla la velocidad actual, la posición y las vueltas que llevas.

En el menú online, podrás elegir la IP y el puerto del servidor al que te quieres conectar(la IP y puerto del servidor de la práctica está puesto por defecto) y elegir un nombre de usuario. Una vez conectado al servidor, podrás seleccionar el coche con el que quieres unirte.



Desarrollo

Desarrollo de los tres puntos más importantes y complejos del proyecto.

Controlador de coches

El script utiliza componentes WheelController para manejar las ruedas delanteras y traseras del vehículo, configurando la física del vehículo con rigidbody y ajustando el centro de masa según sea necesario.

Define goTorque, brakeTorque y maxSteerAngle para controlar la potencia del motor, la fuerza de frenado y el ángulo máximo de dirección, respectivamente. maxSpeed limita la velocidad máxima del vehículo.

Se especifica el tipo de tracción del vehículo (CarType) como FWD (tracción delantera), RWD (tracción trasera) o AWD (tracción en las cuatro ruedas).

Gestiona dinámicamente efectos visuales como el humo de las ruedas patinando (smokePrefab) y las marcas de derrape (skidTrailPrefab). Los efectos sonoros como el sonido del motor (engineSound) y el sonido de derrape (skidSound) se activan según el estado del vehículo.

El script maneja el recuento de vueltas (lapCount, lap) y ajusta el factor de posición del vehículo (posFactor) cuando colisiona con objetos específicos (lapsColliders). Permite reiniciar el vehículo (ResetCar()) devolviéndolo a su posición inicial y deteniendo las ruedas.

Implementa un sistema para controlar el deslizamiento de las ruedas (CheckSkid()), ajustando efectos visuales y sonoros en consecuencia. Calcula dinámicamente el sonido del motor basado en la velocidad del vehículo y el cambio de marchas (CalculateEngineSound()).

Permite al usuario reiniciar el vehículo presionando la tecla R (Update()). Además, las funciones ApplyTorque(), ApplyBrake() y ApplySteering() son llamadas por otros scripts o eventos de entrada para controlar el movimiento del vehículo.

Este controlador es la base para el controlador del jugador y el de la IA, por lo que está presente en todos los prefabs de los vehículos.

La implementación ha sido sencilla gracias a las guías del campus, aunque hay alguna implementación propia como el controlador de vueltas. Lo que más tiempo ha llevado ha sido ajustar bien todos los valores en el editor, para que la conducción se sienta realista y a la vez divertida.

Controlador de IA

El script utiliza el script WaypointCircuit, reutilizado del curso anterior, facilita la implementación y define el circuito de waypoints tanto visualmente en el editor como posicionalmente en la escena.
CarController para controlar el vehículo, y transformaciones para establecer puntos objetivo (target) y de frenado (brakeTarget).

En Update(), calcula dinámicamente la posición y orientación del punto objetivo (target) basado en la ubicación actual y velocidad del vehículo. Ajusta continuamente la posición para optimizar la trayectoria según el siguiente punto del circuito y velocidad actual. Define el punto de frenado (brakeTarget) anticipadamente basándose en el ángulo con el siguiente punto, para optimizar la velocidad en curvas y frenadas.

En FixedUpdate(), utiliza rayos (raysSpawn) para detectar obstáculos (obstacleLayer) y otros vehículos (carsLayer) en la trayectoria. Ajusta la velocidad (torque) y dirección (steer) para evitar colisiones y mantener la trayectoria fluida.

En OnDrawGizmos(), visualiza puntos objetivo (target y brakeTarget), la posición del circuito y rayos para detección de obstáculos en el editor.

El script de WaypointCircuit ha facilitado y agilizado mucho la implementación de este controlador, permitiendo recoger datos más precisos, para conseguir un movimiento más fluido y sin tantos errores.
La implementación de la evasión de obstáculos funciona, pero en ciertos puntos frena demasiado el coche o hace giros demasiado bruscos, por lo que esto sería un punto importante a mejorar en futuras actualizaciones del proyecto.

Conexión con el servidor

Incluye parámetros de interfaz de usuario (UI) como TMP_InputField playerName para el nombre del jugador, TMP_InputField serverAddresInput para la dirección del servidor, y TMP_InputField serverPortInput para el puerto del servidor. Gestiona la conexión al servidor con TcpClient client y NetworkStream stream para la comunicación de datos, utilizando un StringBuilder messageBuffer para almacenar mensajes recibidos.
La interfaz gráfica se gestiona con Canvas canvas, y se utilizan Dictionary<string, GameObject> carsOnLobby para los coches en el lobby. También se define un GameObject spawnPoint para la aparición de los coches.

En el método Start(), se establecen valores predeterminados para la dirección y puerto del servidor, luego en el método StartConexion() se establece una conexión TCP con el servidor utilizando la dirección y el puerto proporcionados. Este método se lanza cuando en el menú de la escena online se pulsa el botón ‘conect’

En el método Update(), se actualizan la dirección del servidor y el puerto a partir de los campos de entrada, se leen y procesan mensajes del servidor si hay datos disponibles. Cada jugador, envía la posición y rotación actual del coche al servidor.

El método SendMessageToServer(string message) envía un mensaje concreto al servidor codificado en bytes. El mensaje enviado, se determina con el método UpdateCarMessage() crea un mensaje de actualización con la información del coche del jugador (comando,ID,nombre, posición y rotación) para enviarlo al servidor.

El método ProcessMessage(string message) divide el mensaje recibido en un comando y sus argumentos, procesando comandos como UPDATE_CAR para actualizar la posición y rotación de un coche en el lobby o instanciar uno nuevo si no existe.

El método StartPlaying() instancia el coche seleccionado por el jugador, lo añade al lobby, configura la cámara para seguir al coche del jugador y activa el control del jugador sobre el coche. El método UpdateCar() actualiza la posición y rotación de los coches que estén en la partida mandando mensajes al servidor.

El método OnApplicationQuit() cierra la conexión con el servidor y libera recursos al salir del juego.

El método Pause() invoca el evento de pausa cuando se presiona la tecla Escape.

Este aspecto del proyecto, no ha sido el más costoso, pero si el que más tiempo ha llevado, debido a la poca experiencia relacionada con juegos online en Unity.
El aspecto que más tiempo e investigación ha llevado es el apartado de recibir los mensajes, ya que a veces se mezclaban o llegaban por partes, pero al usar un buffer y un delimitador para procesar mensajes completos se ha logrado evitar que los mensajes se mezclen o que solo llegasen fragmentos de este.

Conclusión

En conclusión, este proyecto ha logrado desarrollar un simulador de carreras con funcionalidades online, cumpliendo con los requisitos establecidos. Los jugadores pueden competir tanto contra la inteligencia artificial en modo local como contra otros jugadores en un entorno multijugador. La implementación de los controladores de coche y de IA, junto con la gestión de la conexión al servidor, ha sido esencial para el correcto funcionamiento del juego. A pesar de algunos desafíos en la integración de mensajes de red, el uso de buffers y delimitadores ha solucionado estos problemas, proporcionando una experiencia de juego fluida y realista. Este proyecto establece una base sólida para futuras mejoras y expansiones en el comportamiento de los coches y la funcionalidad multijugador.