Historias de usuario

GITHUB: https://github.com/JVISERASS/Proyecto-Final-Unity

Coche de carreras

- Como coche de carreras, quiero poder moverme para correr la carrera
- Como coche de carreras, quiero que una cámara me siga para mostrarme por la pantalla
- Como coche de carreras, quiero poder detectar colisiones con otros coches o con los límites de la pista para simular un accidente.
- Como coche de carreras, quiero tener un sonido de motor personalizado para mejorar la inmersión en el juego

Coches adversarios

- Como coche adversario quiero moverme por la pista para ganar al coche de carreras
- Como coche adversario, quiero poder detectar las curvas y adaptarme para no salirme de la pista.

Data Manager

- Como Data Manager, quiero registrar las estadísticas de cada carrera (tiempo, posición final) para hacer un seguimiento del rendimiento del jugador.
- Como Data Manager, quiero almacenar los datos de tuning del jugador para poder usarlos en carreras futuras.
- Como jugador, quiero ver mi puntuación al final de la partida

Menú Principal

- Como jugador, quiero acceder a un menú principal para poder iniciar las carreras o salir del juego

State Machine

- Como jugador, quiero que el juego se reinicie automáticamente tras terminar una partida para jugar otra vez sin problemas.
- Como jugador, quiero que el juego me muestre si he ganado o perdido la carrera.

Game Manager

- Como Game Manager, quiero manejar el flujo del juego para coordinar su ejecución

Tuning

- Cómo Tuning quiero poder modificar los atributos del vehículo, para que el jugador pueda personalizarlo

Análisis de las historias de Usuario

A continuación, se obtienen las tareas, requisitos y objetivos de las diferentes clases a partir de las historias de usuario.

1. Coche de Carreras
Objetivo: Facilitar la experiencia de juego permitiendo que el coche de carreras realice acciones esenciales como moverse, interactuar con el entorno, y mejorar la inmersión a través del sonido.

Requisitos:

- El coche de carreras debe poder moverse y responder a los controles del jugador.
- Se necesita una cámara que siga al coche de carreras durante la carrera.
- Detección de colisiones con otros coches y los límites de la pista.
- Sonido de motor personalizado para el coche del jugador.

Tareas:

- Movimiento: Implementar la lógica de control del coche de carreras, incluyendo aceleración, frenado y dirección.
- Cámara: Configurar una cámara que siga al coche de carreras en tiempo real.
- Colisiones: Desarrollar un sistema de detección de colisiones que maneje la interacción con otros coches y la pista.
- Sonido: Añadir un sonido de motor personalizado que se reproduzca mientras el coche esté en movimiento.

2. Coches Adversarios
Objetivo: Implementar la IA de los coches adversarios para competir contra el jugador en la carrera.

Requisitos:

- Los coches adversarios deben poder moverse automáticamente por la pista con el objetivo de ganar la carrera.

Tareas:

- Movimiento de IA: Implementar un controlador de movimiento de IA para que los coches adversarios se desplacen por la pista.

3. Data Manager
- Objetivo: Registrar y almacenar los datos importantes de la carrera y de la personalización del coche.

Requisitos:

- Registrar estadísticas de cada carrera.
- Almacenar los datos de tuning del coche del jugador para su reutilización en carreras futuras.

Tareas:

- Registro de estadísticas: Crear una función para almacenar el tiempo y la posición final de cada carrera.
- Almacenamiento de tuning: Desarrollar la lógica para guardar la configuración de tuning del coche del jugador.

4. Menú Principal

Objetivo: Proporcionar al jugador una interfaz desde donde pueda acceder a las opciones de iniciar una carrera o salir del juego.

Requisitos:

- Acceso a un menú principal al iniciar el juego.
- Opciones para iniciar una carrera o salir del juego desde el menú.
- Opciones para hacer Tuning al coche

Tareas:

- Diseño de interfaz de menú: Crear el diseño visual del menú principal.
- Lógica de navegación: Implementar la lógica que permita al jugador iniciar una carrera o salir del juego al seleccionar las opciones correspondientes en el menú.
- Tuning: Crear un apartado que permita al usuario tunear su vehículo

5. State Machine

Objetivo: Gestionar los diferentes estados del juego.

Requisitos:

- El juego debe reiniciarse automáticamente tras finalizar una partida.
- El jugador debe recibir un mensaje indicando si ha ganado o perdido la carrera.

Tareas:

- Gestión de estado de finalización: Programar el estado de fin de partida que permita reiniciar el juego.
- Mostrar estado de victoria o derrota: Crear una interfaz que muestre el resultado al jugador al finalizar la carrera.

6. Game Manager
Objetivo: Controlar y coordinar el flujo del juego.

Requisitos:

- Gestionar la ejecución del juego (pausa, reanudación y finalización.)

Tareas:

- Flujo de juego: Implementar la lógica para manejar el flujo de juego, activando y desactivando los estados y escenas correspondientes.
- Coordinación de subsistemas: Configurar el GameManager para interactuar con otros sistemas como DataManager, SceneManager y StateMachine para mantener la coherencia del juego.

7. Tuning
Objetivo: Permitir que el jugador personalice los atributos de su vehículo.
Requisitos:

- Modificar los atributos del coche para reflejar la personalización elegida por el jugador.

Tareas:

- Atributos de tuning: Crear un sistema de personalización de atributos, como velocidad, manejo, aceleración o temas estéticos.
- Almacenamiento de configuración: Hacer que DataManager almacene los cambios realizados para que se apliquen en carreras futuras.

8. Score Manager
Objetivo: Llevar un seguimiento de la puntuación del jugador.

Requisitos:
- Mostrar la puntuación final del jugador después de cada partida.
- Almacenar la puntuación final.

Tareas:
- Puntuación: Crear un sistema de puntuación
- Almacenamiento: Almacenar la puntuación
- Histórico: Mostrar un histórico de las puntuaciones del jugador

9. Cámara
Objetivo: Seguir al coche de carreras y mostrarlo por pantalla

Requisitos:

- Seguir al coche de carreras
- Varias perspectivas (1ª persona, 3ª persona)

Tareas:
- Seguimiento: Crear un sistema de seguimiento
- Perspectivas: Configurar varias perspectivas

Patrones de diseño utilizados en el diagrama UML:

Singleton
Se va a usar el patrón singleton en GameManager, SoundManager, SceneManager, DataManage y ScoreManager ya que son clases con una funcionalidad concreta que solo van a tener una instancia en el juego.

Factory
Se va a usar el patrón de diseño Factory (UserRaceCarFactory, CompetitorCarFactory, VehicleFactory) para instanciar vehículos. Esto es útil de cara al tuning y a la variedad de vehículos, sin tener que crear muchas clases adicionales. Tambien se ha creado un TuningFactory para poder crear mejoras al coche, tanto esteticas como dinamicas. La decision de usar un Factory es compleja pero nos otorga gran modularidad y mantenimiento a largo plazo, por lo que si queremos meter en un futuro algun tipo de tuning especial, nos sera mas sencillo. Tambien se piensa en los desarrolladores de mods, no necesitan tocar el código base. Pueden añadir fácilmente nuevas opciones sin preocuparse por compatibilidad con otros módulos existentes.

Observer
Patrón que permite que un objeto notifique a otros de cambios en su estado. Se podría usar en Events.

Facade
La clase GameManager actúa como una fachada de SoundManager, SceneManager, DataManager, StateMachine y ScoreManager