Resumen de la Actividad

Para conseguir realizar los objetivos esenciales se explica a continuación una breve explicación de los elementos creados y los elementos modificados del juego base:

GameFlowManager.cs (Modificado)

El sistema se basa en un patrón de diseño parecido al Singleton, solo que sin las ventajas de este. Ya que esta clase no se mantiene entre escenas. Aun así, el flujo del programa pasa a través de esta super clase que recoger en su método “Start” al resto de clases de jerarquía inferior.

En esta clase se evalúa accediendo al ObjectiveManager a la situación de victoria o derrota del juego. Accediendo a esos instantes se puede hacer aparecer el cartel de “GameOver” creado.

En esta clase también se tiene control de las monedas recolectadas durante la fase, las cuales se recolectan por el Kart a medida que choca con ellas (trigger). Este GameManager tiene control de las monedas recolectadas en partidas anteriores y durante la partida actual, sumándolas y almacenándolas si se han terminado las tres vueltas.

GameOverMenuController.cs (Creado)

Una clase placeholder que se oculta a si mismo en Start para asegurarse que siempre aparece oculto. Puede modificarse para añadir alguna animación cuando aparece o desaparece.

TimeDisplay.cs (Modificado)

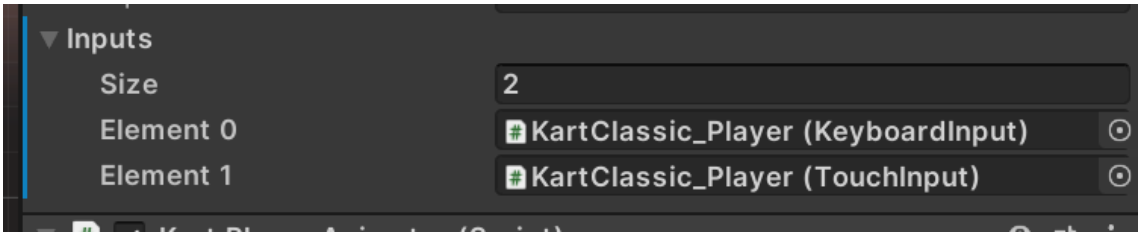
En esta clase se tiene control de los tiempos realizados en cada vuelta, viene perfecto para evaluar la mejor vuelta y si se supera un record mostrarlo por pantalla al realizar una vuelta mejor que la registrada como RECORD. Otra modificación realizada en convertir en Public Static su función que toma un float y retorna una string temporal. Es válida para usarla en la primera escena para mostrar el record con el mismo formato. Esto se hace ya que tanto las monedas como el record se almacenan en int y float respectivamente mediante la clase PlayerPrefs. Almacenar strings directamente consume más recursos que en int o float.

RecordDisplay.cs (Creado)

Record display accede a la variable BestTime almacenada en memoria local por el PlayerPref y lo muestra en pantalla si es un valor mayor que 0. Si no se muestra que no se ha realizado aún un record válido.

ArcadeKart.cs (Modificado)

Se modifica el array de inputs haciéndolo publico para almacenarlos en la clase como componente de forma ordenada.



Se discrimina mediante las siguientes líneas a continuación uno de los controles dependiendo de si el sistema se está buildeando para móvil o para ordenador.

```
185     #if UNITY_ANDROID
186             //Si es Android se usan los controles TOUCH
187             var inputSource = m_Inputs[1];
188     #else
189             //Si es Windows se usa el teclado
190             var inputSource = m_Inputs[0];
191     #endif
```

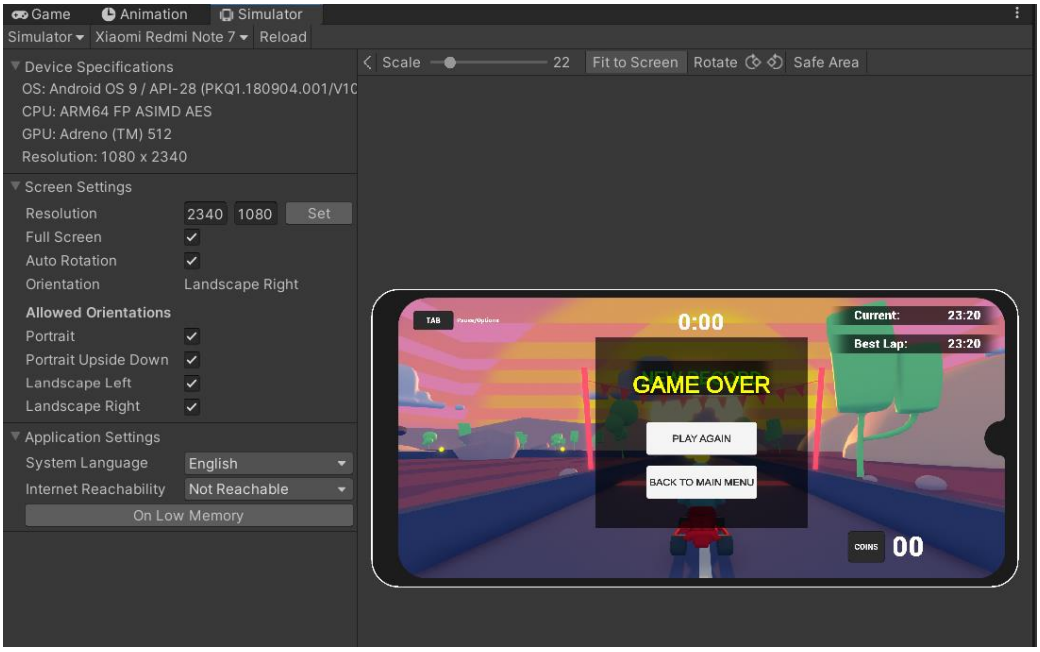
TouchInput.cs (Creado)

Se trata de un análogo a KeyBoardInput ya que hereda de la misma clase “BaseInput”. Solo que los ejes de movimiento dependen del movimiento del dedo en la pantalla táctil. El control se basa en el diseño del videojuego “MarioKart” para móvil. La velocidad se mantiene constante hacia delante y el jugador se limita a realizar movimientos en horizontal para hacer girar el coche. Otra idea seria añadir un sistema que añadiendo un segundo toque al pulsar el coche derrape hasta soltar ese segundo toque en pantalla.

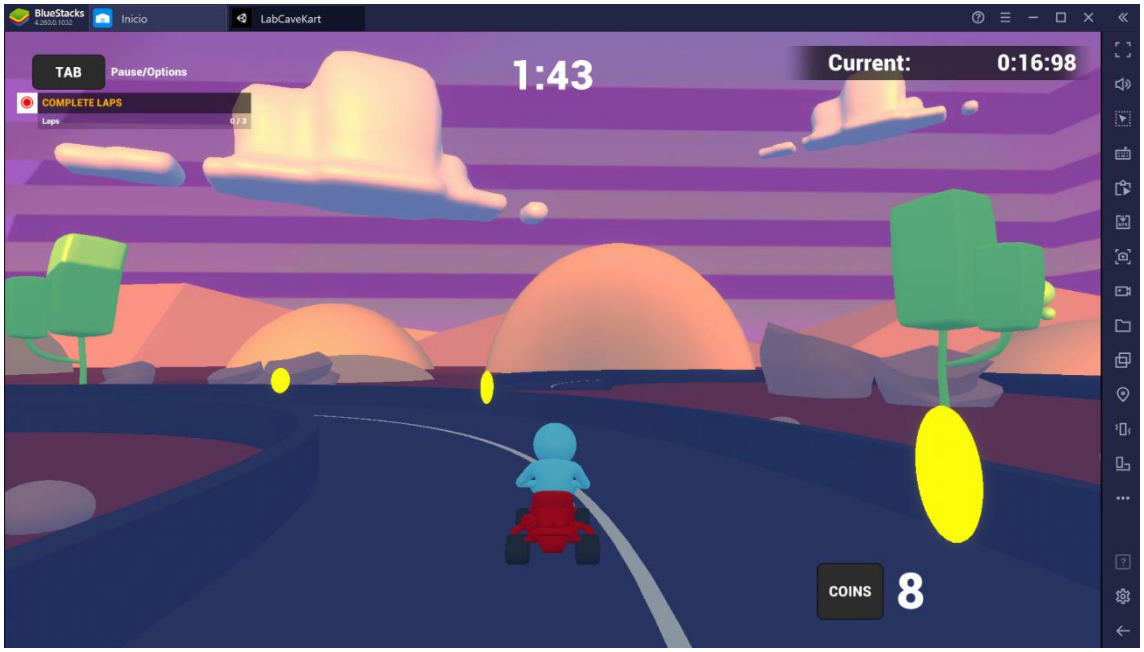
Coin.cs (Creado)

Esta clase controla las monedas. Cuando el coche impacta contra ellas estas tienen en su CallBack: OnTriggerEnter una comprobación de que lo que ha impactado con ellas es el coche mediante Layers que vienen por defecto definidas en los colliders del coche. Al hacerlo, la moneda emite partículas y un sonido de confirmación. Desaparecen aplicando transparencia al material y desactivando su collider. Es mucho más eficaz de esta manera en vez de destruyendo el objeto.

La pantalla es “responsive” en horizontal (LANDSCAPE) al actuar sobre el CanvasScaler de los canvas del Inspector. Esto hace que se respete el tamaño y la distribución a lo ancho. Para asegurarse de que no se invade la “safezone” de la gran mayoría de móviles del mercado se hace uso del package “Device Simulator”, el cual también permite simular los “TouchInput”.



La APK se buildea aplicando compatibilidad con ARM_64. Para testearla se utiliza bluestacks ya que no dispongo de ningún móvil Android.



DISEÑO PARA DISEÑADORES

No me ha dado tiempo a aplicar los cambios necesarios para que el sistema sea fácil de editar por los diseñadores de videojuegos ya que, a mi parecer, debería ser cambiada mucha mecánica y código del sistema de control del Kart y del juego en general. Mi sistema favorito tanto como patrón de diseño como para tener versatilidad de código y de cara a la máxima eficiencia de uso de memoria y recursos es utilizar ScriptableObjects.

Al aplicarlos en la clase Kart se pueden modificar sus campos en tiempo real y permanecerán con ese valor entre escenas y tras el “runtime”. Esto es debido a que son Assets como los materiales, no código serializado.

Al ser assets, ocupan poca memoria ya que no son instancias de la clase y pueden almacenarse y modificarse para crear distintos tipos de Karts o de sistemas de juego. Esto ayuda mucho a los diseñadores y también al programador.

Otro uso muy habitual que utilizo personalmente es como interfaz entre clases. Esto crea una máxima encapsulación (sistemas de eventos, sistema de inputs, mánager de audios, etcétera).

RESPUESTAS A LAS PREGUNTAS REALIZADAS

- **Si en vez de tener un template de Unity hubieras tenido que crear tú el proyecto, cómo lo hubieras organizado a nivel de carpeta y estructura? ¿Qué hubieras cambiado o hecho diferente?**

La estructura de carpeta no esta mal, pero echo de menos reglas de nomenclatura en assets como los prefijos (M_, T_, S_...) o los sufijos (_SO). Los ScriptableObjects los habría introducido en otra carpeta llamada Data donde se incluirían los códigos de los mismos para tener un seguimiento de código correcto. También habría eliminado un nivel jerárquico y habría puesto en contenido de la carpeta "Karting" directamente en Assets.

- **Si se te hubiera pedido almacenar las puntuaciones de manera persistente (en vez de localmente) a nivel de id de usuario, ¿qué servicios hubieras usado?**

En cuanto a nivel de usuario en nube se podrían haber utilizado el sistema GoogleAnalytics, UnityAnalytics, Azure, y cualquier sistema de back. Se podría haber realizado un sistema custom en un servidor local con WordPress en CMS.

- **Si se te hubiera pedido localizar el juego a varios idiomas, ¿cómo lo habrías hecho?**

Existen varios métodos, ahora mismo aplicaría un sistema de eventos por interfaces (ILocalizables) de tal manera que emitiendo un evento desde cualquier sitio se podría realizar la localización en los objetos y textos suscritos al evento. Para guardar los textos se podría hacer con CSV o con XML. Por simplicidad habría usado CSV.

- **¿Dónde y cómo implementarías compras en la aplicación? ¿Qué servicios conoces o utilizarías para ello?**

No se ningún servicio de compras en la aplicación.

- **Si tuvieras que monetizar el juego a través de anuncios, ¿cómo lo harías o qué estrategia seguirías? ¿Qué servicios de anuncios conoces para poder monetizar la aplicación?**

No conozco ningún servicio de anuncios.