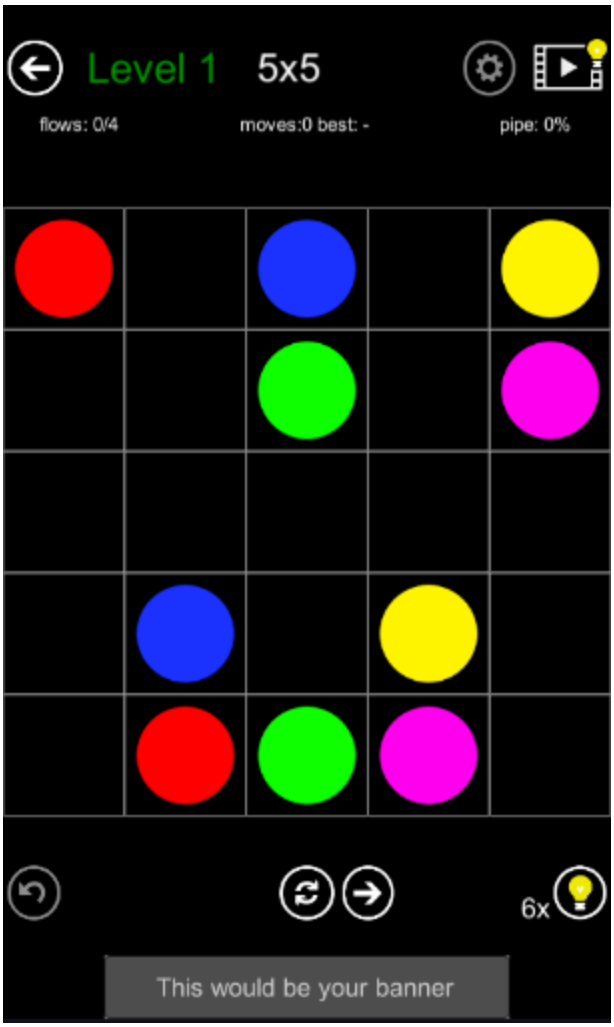
Videojuegos para Dispositivos Móviles

David Czepiel, Pablo Villapún, Sandra Modragón

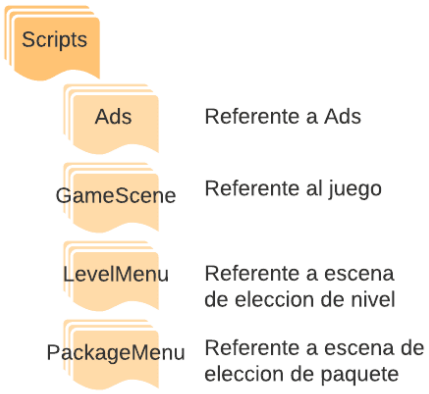
PRÁCTICA 2 – Flow Free



ESTRUCTURA DEL PROYECTO

La estructura del proyecto se divide en **4 grandes carpetas: Data** (que contiene todos los recursos necesarios para el juego), **Prefabs** (que guarda los prefabs correspondientes), **Scenes** (guarda las escenas) y **scr** (donde encontramos el código fuente).

En **Data** encontramos: **Animations**, **Fonts**, **Images** y **LevelData**. Este último guarda la **información referente a los 3 tipos de ScriptableObjects**, las categorías, los paquetes y las skins (pieles). Además en **LevelData se almacenan los .txt** del juego original (**TxtData**)



En scr encontramos dos sub-carpetas: la que define los **ScriptableObjects** y la que define los **Scripts del juego**. En los Scripts, se subcategoriza por el uso de los scripts en el juego: GameScene contiene el código utilizado y referente a la escena del juego (GUIManager, LevelManager, BoardManager…); Ads contiene los archivos de código referente a Unity Ads; LevelMenu y PackageMenu contienen código utilizado en las escenas referentes (LevelSelectionMenu y PackageSelectionMenu).

COMPONENTES Y CLASES

Referente a la escena del juego

El funcionamiento de la escena del juego viene dado de la siguiente manera. El **GameManager** tiene **guardado el nivel** que se ha elegido. El **Level Manage**r, en el Start, pregunta al GameManager por el archivo asociado al nivel, y a partir de ahí **recoge la información para pasárselo a los distintos managers**. El **BoardManager**, por su parte **instancia todos los Prefabs** de Tile dando comienzo a la escena del juego.

En ejecución, el **input se recogerá en el InputManager**, comunicándoselo al **BoardManager** para que lo **procese** y este último **avise** al **LevelManager** del porcentaje de flows completadas.

El LevelManager realizará la comprobación de si se ha pasado o no de nivel. En caso de pasarse avisará a GUI para mostrar el panel final.

- **BoardManager**: Manager que administra todo lo relacionado con el tablero y el mostrado de este en pantalla. Contiene **métodos para la creación del tablero** (prepareBoard, createBoard, applyHint()...), y de **procesado de input** (processInput, checkTileType, checkEventType, ...). Además **contiene una matriz de Tile** y una **Lista de Pipes** para **guardar el estado del tablero**.

  Por cada evento envía información a LevelManager.

  - **Tile:** Componente que **representa una casilla** en la partida. Maneja **información del tipo** de tile en lógica y se **adecua visualmente** en la escena dependiendo de la información que tenga.
  - **Pipe**: Clase encargada de la gestión de la lógica de una tubería en una partida. Guarda el estado actual, el anterior y como debería de ser la solución de la tubería. Contiene métodos para la gestión de cortes en el juego.
  - **TouchFeedback:** controla el círculo de feedback por el que el input (dedo o mouse) está pasando.

- **LevelManager**: Clase que gestiona la comunicación entre el board, GameManager y UI.

  Este Manager es el que avisa a: BoardManager para que inicie la creación del board en la escena, habiendo obtenido información del GameManager; GUIManager para que empiece mostrando la información inicial del nivel; InputManager para enviarle información para la futura transformación de coordenadas.

  Cuando un evento se ha registrado en el BoardManager, éste avisa a LevelManager con el fin de enviar la información necesaria al resto de managers. Un ejemplo de esto sería cada vez que se añade un tile a una tubería, teniendo que modificar la GUI para mostrar la información de cómo va el board.

  - **Map**: Clase que guarda la información de un archivo de mapa. Contiene métodos para leer el archivo de texto y getters para dar la información necesaria.
  - **PlaceBoardInScreen**: Clase encargada de posicionar y escalar el objeto BoardManager para ajustarse a la pantalla.
  - **CameraPlacer**: Encargada de posicionar la cámara para visualizar el (0,0) a la esquina inferior izquierda. Esto es útil para que todas las coordenadas sean positivas y faciliten los cálculos a la hora de posicionar Board y realizar el input.

- **GameManager**: Clase que guarda la información necesaria entre escena y escena. Además se encarga del guardado y administración del progreso del jugador.
    - **ProgressSerialization:** Clase estática que contiene métodos para guardar y cargar el progreso del jugador. Este archivo también contiene la definición de las clases que guardan la información necesaria para serializar el progreso.
- **GUIManager:** Manager encargado de la gestión y actualización de la GUI durante la ejecución del juego.
    - **ButtonBehaviour**: componente que contiene diferentes métodos para que sean llamados desde distintos botones.
- **InputManager**: Manager que controla la recogida del input y su transformación. El input es quien avisa al BoardManager de que un evento ha ocurrido, de tal forma que solo se hace update visual si un evento ha ocurrido.

Referentes a la escena de elección de paquete:

- **CategoryManagement:** Clase encargada de **gestionar la creación de los diferentes elementos** presentes en pantalla en la escena de selección de categoría. Esta clase toma del GameManager la información sobre las diferentes categorías disponibles, y para cada una de ellas genera los elementos que permitan que estas se visualicen en pantalla. **Para cada una de las categorías, genera un texto** en la UI con el título de la categoría, **junto con una serie de botones que representan los diferentes paquetes** de niveles que tiene cada categoría. Estos últimos contienen 2 textos: uno que muestra el título del paquete de niveles y otro que muestra el número de niveles que el jugador ha pasado en dicho paquete. Esta clase, aparte de generar estos elementos se encarga de personalizar su apariencia según la información de la categoría: por ejemplo si la categoría almacena el color naranja, tanto el fondo del título de la categoría como los títulos de los botones se mostrarán naranjas. Estos objetos que genera vienen dados por prefabs, los cuales añade al canvas que va a hacer scroll, a la vez que aumenta el tamaño del canvas para que su scroll se ajuste, estos prefabs:
    - **Categoría Niveles:** es un prefab que contiene un vertical layout group, es el contenedor de todos los elementos que representan una categoría (título y botones)
    - **Title prefab:** prefab que representa el título de cada una de las categorías (un texto más un fondo), está controlado por el script **TitleBehaviour,** el cual se encarga de personalizar tanto el texto del título como el fondo con la información dada por la categoría
    - **BotonPaqueteNiveles:** prefab que contiene un botón y dos textos. Este prefab está controlado y gestionado por el script **ButtonPackageSelectionBehavoiur** el cual se encarga de modificar tanto el título (junto con el color) del paquete de

niveles que representa este botón como el texto que representa el número de niveles completados en el paquete. Este botón es el que al ser pulsado le indica al gamemanager qué paquete debe mostrar al pasar a la escena de selección de nivel.

Referentes a la escena de elección de nivel

- **LevelDisplayManagement**: clase encargada de **gestionar la creación de los diferentes elementos que se mostrarán en pantalla en la escena de selección de nivel**, esta clase toma del GameManager la información sobre el paquete de niveles que hemos seleccionado y generar los botones que permitirán acceder a cada uno de los niveles del paquete, estos botones son generados en grupos de 30, los cuales se caracterizan por estar administrados por un prefab que contiene un **vertical layout group** (para organizar el grupo), que contendrá tantos los botones del grupo como el título del mismo y un prefab con un **horizontal layout group** (para organizar las filas de botones, las cuales tienen 5 botones), aparte de generar estos elementos, esta clase también se encarga de alterar la apariencia del título que aparece en la parte superior de la pantalla en esta escena (tanto con el nombre del paquete como con el color asignado). Los prefabs con los que trabaja esta clase para organizar las escenas son los siguientes:
    - **VerticalLayoutPrefab**: prefab que nada más que contiene un componente de tipo vertical layout group, su función es almacenar tanto el título como los 30 botones que forman cada uno de los subgrupos del paquete de niveles.
    - **HorizontalLayoutPrefab**: prefab que nada más que contiene un horizontal layout group, su función es almacenar los botones que conforman cada una de las filas de cada uno de los subgrupos del paquete (5 botones por fila).
    - **LevelGroupTitle**: prefab que representa el título de cada uno de los subgrupos de niveles dentro del paquete, está controlado por el script **TitleBehavoiur** para cambiar al título y apariencia indicados por la información proporcionada por el GameManager.
    - **LevelButtonPrefab**: prefab utilizado para **representar cada uno de los niveles de un paquete**, aparte de un botón contiene diferentes elementos para su representación en pantalla, como un texto que indica el nivel que representa y diferentes imágenes tanto para su representación como para indicar si ya ha sido completado. Este prefab es controlado por el script **LevelButtonBehaviour** el cual tiene como función recibir la información que debe almacenar cada botón, como el número del nivel que representa o si ha sido completado o no. **Se encarga tanto de alterar la apariencia del botón según la información mencionada como de informarle al GameManager** qué nivel tenemos que cargar y jugar a la hora de ser pulsado.

Referentes a los Ads

Cabe mencionar que se intentó que **AdsManager** controlara la carga y muestra de los distintos tipos de banners. Sin embargo y dados los bugs de la actualización no conseguimos que funcionara correctamente.

Al parecer Unity cada vez que se muestra un anuncio (del tipo que sea) se añade a una "lista"(o estructura que lo controle) para después llamar a **OnUnityAdsShowComplete**. Esto genera confusión debido a que si, por ejemplo, se muestra un anuncio de tipo **RewardedAds** y después otro de tipo **InterstitialAd** se llamará al método de **showComplete** de todos los anuncios añadidos aunque alguno de ellos no haya sido el anuncio que se haya terminado. Es incluso así, que el propio método contiene el parámetro adUnitId el cual debería de mandar el tipo del anuncio que se ha terminado.

```
public void OnUnityAdsShowComplete(string adUnitId,
UnityAdsShowCompletionState showCompletionState)
```

Y siguiendo el ejemplo de antes **debería** de valer "InterstitialAd_Android"**, sin embargo** vale el del propio anuncio al que se está avisando (RewardedAds)

Esto dificulta mucho que haya un manager que maneje todo esto, por ello tomamos la decisión de separarlo en distintas clases.

- AdsManager: se encarga de inicializar los Ads en Unity. Este componente se encuentra en el objeto GameManager con el fin de que solo se inicialice una única vez.
- BannerAdExample: se encarga de la carga y muestra del banner en escena. Al estar el banner en todas las escenas, existe un prefab AdsManager que contiene este script.
- InterstitialAdExample: se encarga de la carga y muestra del IntersititialAd en escena. Este se llama (al show) al cambiar de nivel.
- RewardedAdsButton se encarga de la carga y muestra del RewardedAds en escena. Se llama al show desde el botón del video de la GUI.