Flow Free

Daniel González Cerdeiras
Georgi Mednikov
Eloy Moreno Cortijo

Estructura de archivos

Dentro de la carpeta Assets, que es la única que se ha ampliado con el desarrollo, se separan los archivos dependiendo de su tipo, Packs (el conjunto de scriptable objects que se han creado para guardar los conjuntos de niveles, colores, etc.), Scripts, Prefabs, Scenes y finalmente los Assets originales del juego. Dentro de cada uno de estos, se diferencian por su función, por ejemplo:

Scripts ->

- Ads: Relacionado con el sistema de anuncios.
- Logic: Clases que contienen la lógica del juego (Map, Flow…)
- Managers: Gestionan las distintas escenas.
- Rendering: Animaciones y scripts que modifican el dibujado.
- Structures: Estructuras que se usan en los Scriptable Objects o en el guardado.

Estructura de clases

Se basa en regla general en una jerarquía, con claras excepciones como scripts de animación, que solo se llaman y actúan de forma independiente, avisando de su terminación a través de eventos de Unity.

En lo más alto está el GameManager, que se encarga de gestionar el juego. También están el AdManager y el SaveManager, que se encargan de cargar anuncios y guardar la partida respectivamente. GameManager tiene referencias a estos dos y los llama cuando necesita, y por “cercanía” a él (dado que es un singleton) estos también lo son.

Debajo están los scripts encargados de cada escena, los LevelManagers, que no sobreviven al cambio entre escenas y son controlados por el GameManager. Hay varios LevelManagers con funcionalidades específicas para cada escena, como crear los botones deslizantes en el selector de niveles, etc.

En lo más bajo de la jerarquía está el BoardManager, que informa a la lógica de los eventos y renderiza las consecuencias. Con él está el InputManager, que traduce a posiciones del juego el input. Board Manager interactúa con las clases de lógica que no heredan de Monobehaviour, que son Map (representación lógica del tablero), Flow (representación lógica de los flows) y LogicTile (representación lógica de cada casilla del tablero). Estas clases son la base de la jerarquía y no mandan sobre nadie, solo la evolución del estado del juego para ser dibujado en pantalla.

Funcionalidad

El juego resultante es prácticamente igual al original pero menos logrado estéticamente. Tiene pantalla de inicio, de selección de packs y de niveles, y la escena del juego per sé. Todas las animaciones del juego no relacionadas con la interfaz (como que al empezar un nivel los iconos de conseguir pistas salten) están hechas, incluyendo seleccionar un extremo de una tubería, cortar una tubería, poner una pista en el tablero, el cambio de niveles… Además se ha cuidado el funcionamiento de la lógica de forma que funcione correctamente. Por ejemplo, si se cortan varios caminos a la vez todos retroceden y reproducen su animación, una pista puede cortar caminos, el último camino creado si no se ha completado termina en un círculo pequeñito indicando que está incompleto, etc. En general el juego como tal está muy cerrado, incluso funciona el deshacer movimiento, con el pequeño fallo de que si el movimiento a deshacer cortó flujos estos no se restauran. También creamos puentes en teoría, aunque dejamos de preocuparnos por ellos cuando descubrimos que no hacían falta.

En cuanto a la interfaz del juego, la información se actualiza correctamente y cuando debe en partida, y en general funciona en todas las escenas del juego. Tuvimos el problema que mencionamos en clase de que la parte del canvas donde pone el nivel y el tamaño, debido a un bug de Unity, aparecen mal colocados los elementos hasta que se refresca el rect pasando el ratón por encima del botón, y lo que nos dijiste no funcionaba así que lo situamos algo más separado para que se viera bien. También, en la pantalla de inicio, se puede cambiar la paleta de colores del juego, dándole a Colors y seleccionando uno. Solo hay dos, de forma demostrativa, y cambian la scheme de la selección de niveles y de las tuberías en los niveles.

La implementación de anuncios en el proyecto se ha hecho usando la reducida y poco funcional API de Unity Ads 4.0.0. En concreto, se ha implementado un anuncio de tipo banner en la parte baja de la pantalla y además de que aparecen anuncios intersticiales entre los niveles con una posibilidad del 66%. Los anuncios de premio aparecen cuando el jugador pulsa el botón situado en la parte superior derecha de la pantalla y al terminar el anuncio se premia al jugador con una pista. Cabe destacar que hemos observado comportamiento anómalo en el método Advertisement.Show(string placementId, Listener). Cuando este método es llamado desde Android es necesario darle un Listener cada vez, pues parece que lo borra al terminar el evento. En cambio, en el editor no realiza dicho borrado y por lo tanto añadir listeners cada vez desemboca en que se llame varias veces al evento OnUnityAdsShowComplete(). Creemos que esto es un error por parte de Unity y por ello lo hemos encapsulado en #if.

El guardado de niveles se hace al terminar cada nivel y al cerrar la aplicación. La carga ocurre al iniciar la misma. El formato del archivo es una estructura con un valor para las pistas, la skin seleccionada y la lista de packs y niveles. El guardado de niveles se hace de forma incremental: empieza vacía la lista de packs y niveles y se rellena según el juego carga los niveles y solicita al saveFIle sus datos. De esta manera, es capaz de incorporar contenido descargable y contenidos que faltan directamente sin perder los datos guardados. Además, utiliza un hash para ofuscar los datos guardados y comprobar si están en buen

estado. A la función de hash se le envía el archivo de datos serializado con unos caracteres al inicio y al final para encriptarlo más aún.

Más allá de las features que no se pedían, como el deshacer un movimiento o poder cambiar la paleta de colores, no se ha hecho ningún opcional de los planteados en el enunciado.