Videojuegos para Dispositivos Móviles

Práctica 1: Nonograma

Sam Blázquez Martín
Miguel Hernández García
Juan Diego Mendoza Reyes

Arquitectura de la solución

La solución está formada por 6 módulos. 4 de ellos son específicos de Android o de PC, y 2 módulos son comunes a las dos plataformas.

App y AppDesktop

Empezamos hablando de los módulos App y AppDesktop: Estos módulos son los responsables de la inicialización de la ejecución tanto en Android como en PC, respectivamente. Sus únicas funciones son la inicialización del motor y la inicialización del gestor de escenas con la correspondiente escena inicial.

Lógica

Después tenemos al módulo de Lógica, donde se encuentran las distintas escenas que heredan de la interfaz Scene, también hay tres clases específicas de la lógica para guardar coordenadas de procesado de input y la creación del tablero.

Las escenas del juego cuentan cada una con sus métodos init, update, render y handleInput correspondientes los cuales son llamados por el engine cuando el bucle del juego lo requiera. Decidimos implementar una función handleInput para separar el procesado del input y que fuera llamada por el método update cuando este lo requiriera.

Todas las escenas funcionan a través de un SceneManager que va poniendo las escenas o borrándolas y llamando a sus respectivos métodos dentro del engine mencionado anteriormente.

En el caso de la escena inicial 'MainMenuScene', aquí cargamos todos los recursos (sonidos, fuentes e imágenes) ya que esta escena es la primera y nunca se borra, y guardamos todos esos valores en el engine para no tener que crearlos en cada escena y evitar así código duplicado.

En la escena 'MyScene' que contiene toda la lógica del juego la creación del tablero se realiza con 60% de probabilidad de que la casilla sea azul y 40% que no para evitar

casillas de Si/No/Si/No y que sea más fácil de adivinar para el jugador. Para mostrar los valores en pantalla para dar pistas al jugador, los números que dan pistas en las filas se van añadiendo en función creas el propio tablero, y las columnas se realizan con programación dinámica para que el coste en espacio sea lineal en vez de cuadrático. Por último, tengo una serie de casos base establecidos para que el tablero tenga más sentido:

Permito que las columnas puedan estar enteras seleccionadas, pero no se lo permito a las filas para evitar crear cubos completos y que el juego tenga figuras más irregulares.

Y claramente no permito filas y/o columnas completamente vacías, pues no tiene ningún sentido.

La clase Cell representa una única casilla del tablero y contiene métodos que facilitan su renderizado específico, el procesamiento al recibir algún tipo de input y la información de su estado respecto al juego(Correcto/Erróneo). A la hora de pulsar el botón de Comprobar, en vez de tener que recorrer la matriz entera cada vez que pulses el botón para revisar que celda está bien y cual no y tener otro coste en espacio cuadrático, lo que hacemos es que la propia celda guarde un estado de si está bien seleccionada o no una vez la pulses, actualizando el valor en tiempo real, evitando así tantos recorridos innecesarios. Esta implementación no tiene una diferencia muy notoria en nuestra práctica, pero tiene un gran impacto en tableros a gran escala de 100x100, pues al actualizar la celda nada más pulsarla no tiene que recorrer toda la matriz cada vez que quieras comprobar si te has equivocado.

La clase Interactive es una clase específica de la que heredan los elementos para los que se procesa input. De esta clase padre derivan las celdas y botones del juego, ya que requieren de un input para realizar una acción.

Interfaces(Engine)

Implementa las distintas interfaces utilizadas en la solución, esto nos permite la utilización de distintas clases desde la lógica, abstrayéndonos del proceso específico de cada plataforma. Las interfaces son las siguientes:

- Engine
- IAudio
- IEventHandler
- IFont
- IGraphics
- IImage
- Input
- ISceneMngr
- ISound
- IState
- Scene

La interfaz IState no hemos necesitado utilizarla en la solución debido a la implementación de ISceneMngr que maneja mediante un Stack la escena actual, el resto de escenas se guardan, pero no son actualizadas.

Aparte hay una clase Vector2D se encuentra en el módulo Interfaces(Engine), pero no es una interfaz, sino que es una clase utilizada para las coordenadas del input y es posicionada en este módulo porque es común a la lógica y los engine.

EngineAndroid y EngineDesktop

Estos módulos contienen los hilos del juego y se encargan del renderizado, reproducción del sonido y de recibir el input.

En primer lugar tenemos el Engine, el cual inicializa tanto al render, el eventHandler, el input y el manejador del audio. Hereda de runnable, ya que contiene la hebra principal del juego, esta hebra realiza el bucle principal del juego, que incluye renderizado y actualización de la escena. La escena a renderizar y actualizar es la indicada por el SceneManager.

La clase Audio se encarga de guardar los distintos sonidos y gestionarlos.

La clase Font guarda información sobre una fuente utilizada en el renderizado de los textos. También carga el recurso en su inicialización.

La clase Image guarda información sobre una imagen. Al igual que el Font, también carga el recurso de imagen en su inicialización.

La clase Render se encarga del renderizado de distintas formas como texto y cuadrados a petición del engine. También inicializa el canvas y lo prepara todo para que se pueda dibujar en Graphics2D. Aparte tiene los métodos necesarios para el reescalado de cada motor.

La clase SceneMngr contiene un stack con las escenas del juego. También tiene métodos update, render y handleInput que son llamados por otras clases cuando sea necesario. Solo la escena que está encima del stack es procesada.

La clase Sound se encarga de reproducir un sonido especificado desde la lógica. Esta clase es llamada desde el Audio, pues es el encargado de todo lo del sonido.

La clase Input permite la captura de mouse o táctil según la app. Aquí se redefinen los diferentes tipos de acción del input y guarda las coordenadas del input recibido.

La clase IEventHandler es una clase simple, que permite junto a la clase Input, saber qué tipo de Evento se ha realizado. Luego en el HandleInput de cada escena se procesan las coordenadas iniciales con el evento declarado y se procesan con la escala del juego para obtener el input exacto del objeto Interactive correspondiente.

Opcionales realizados

El tablero no tiene por que tener el mismo número de columnas que de filas, sino que se adapta a cualquier numero de filas y columnas.