SWITCH DASH

MEMORIA TÉCNICA

Sergio Gavilán Fernández *sgavil01@ucm.es*

Alejandro Villar Rubio *alvill04@ucm.es*

Módulos

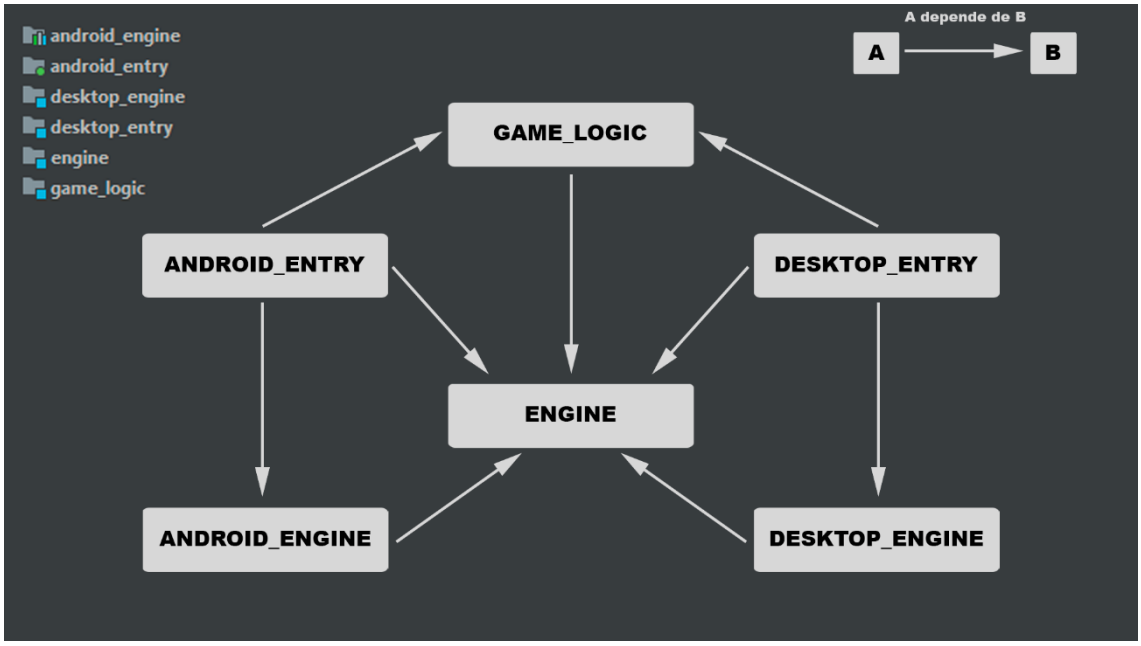
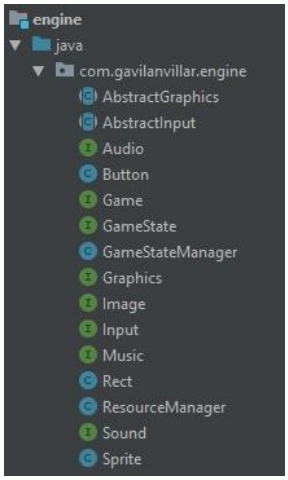


Ilustración 1. Grafo de Dependencia de Módulos

Arquitectura

Engine

Módulo que contiene las interfaces y clases abstractas que implementarán y heredarán, respectivamente, las clases creadas en aquellos módulos que dependerán de este mismo.

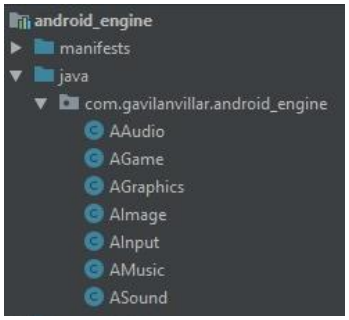


- **AbstractGraphics** *(clase abstracta)***.** Implementa la interfaz *Graphics.* Contiene los métodos genéricos para el escalado de esta interfaz.
- **AbstractInput** *(clase abstracta)***.** Implementa la interfaz *Input* y se encarga de añadir eventos a la lista de TouchEvents. Además, devuelve la lista manteniendo la concurrencia en la aplicación al realizar ambas acciones.
- **Audio** *(interfaz)***.** Contiene los métodos necesarios para crear sonidos y silenciar el audio del juego.
- **Button.** Hace referencia a un botón de la aplicación, es representado por un *Sprite*. También puede saber si se ha pulsado sobre ese botón.
- **Game** *(interfaz)***.** Las implementaciones de esta interfaz son las que manejan el **bucle principal del juego**. Mantiene las instancias de *GameStateManager*, *Graphics*, *Input* y *Audio*. Contiene el método para poner la pantalla completa (solo funcional en PC) y libera los recursos al cerrar la aplicación.
- **GameState** *(interfaz)***.** Representa un estado del juego, es decir, la lógica.
- **GameStateManager.** Gestor que controla el estado de juego actual, permitiendo cambiarlo.
- **Graphics** *(interfaz)***.** Interfaz que será implementada por *AbstractGraphics* para el escalado y por las respectivas clases de *Graphics* de cada plataforma para la carga de imágenes y pintado.
- **Image** *(interfaz)***.** Usada para representar las imágenes cargadas de disco y almacena información de estas, como el tamaño.
- **Input** *(interfaz)***.** Proporciona las funcionalidades de entrada básicas.
- **Music** *(interfaz)***.** Contiene los métodos para gestionar música(pausar, reproducir, mutear…).
- **Rect.** Contiene los parámetros de coordenadas que forman un rectángulo como puede ser una imagen.
- **ResourceManager.** Carga los recursos que se encuentran en el directorio raíz. Esta carga solo se realiza una vez, cada estado tiene una instancia de este gestor y coge los recursos que necesita.
- **Sound** *(interfaz)***.** Contiene los métodos para gestionar un sonido. (pausar, reproducir, mutear…).
- **Sprite.** Contiene la información sobre *Image* y *Rect* fuente y destino de una imagen. Es el paso intermedio entre *Image* y *AbstractGraphics*

Android Entry

Contenedor de la clase *AndroidEntry* que es el punto de entrada de Android. Es el *Activity* principal de la aplicación, que hereda de la clase *AppCompatActivity*.

Android Engine

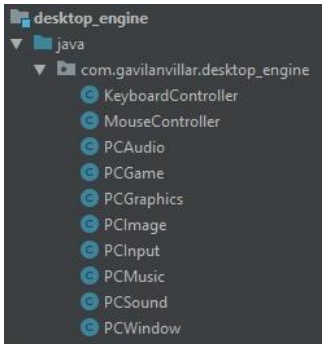


Módulo con las clases que implementan y heredan las contenidas en el módulo *Engine* para la plataforma de Android.

Desktop Entry

Contenedor de la clase *DesktopEntry*, punto de entrada de la plataforma PC. Inicializa la ventana.

Desktop Engine

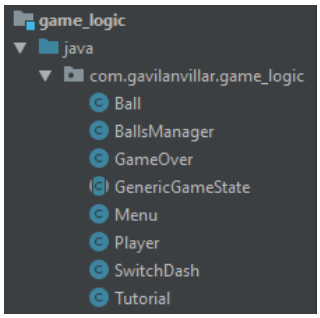


Módulo con las clases que implementan y heredan las contenidas en el módulo *Engine* para la plataforma de PC. Sobre todas las clases destacan tres:

- **PCWindow.** Hereda de JFrame, por lo que se usa para inicializar la ventana y obtener el "bufferStrategy".
- **MouseController / KeyboardController.** Implementa las interfaces *MouseListener, MouseMotionListener* y *KeyboardListener* pertenecientes a Java.

Game Logic

Módulo que contiene todo el apartado de la lógica de la aplicación.



- **GenericGameState** *(clase abstracta)***.** Sirve como estado genérico del juego implementado comportamientos compartidos entre estos.
- **Menu / Tutorial / SwitchDash / GameOver.** Son los diferentes estados del juego.
- **Player / Ball.** Clases que hacen referencia a las entidades formadas por el "jugador" y las "bolas".
- **BallsManager.** Gestiona las bolas usando un "pool".

Partes opcionales

- **Efectos de sonido.** Se pueden cargar y reproducir sonidos y música desde Android y PC, en PC las clases de sonido y música son idénticas ya que en java no hace falta hacer distinción entre sonido y música (en android si) pero se ha hecho de esta manera para mantener la capa de abstracción y permitir la futura implementación de otras posibles plataformas que hagan la distinción entre sonido y música.
- **Pantalla completa.** Funcionalidad solo disponible en PC. Se activa y desactiva con la tecla "F". Está implementado en los "handleEvents" de los GameState. Se comprueba que la tecla pulsada es la "F" que tiene una id de 70, por lo que creemos que nunca se debería activar en Android en el caso de darle un uso normal a la aplicación.
- **Teclado.** Está implementado en la clase *KeyboardListener*. Estos eventos se crean igual que con el ratón, por lo que se gestiona de la misma manera en los estados.