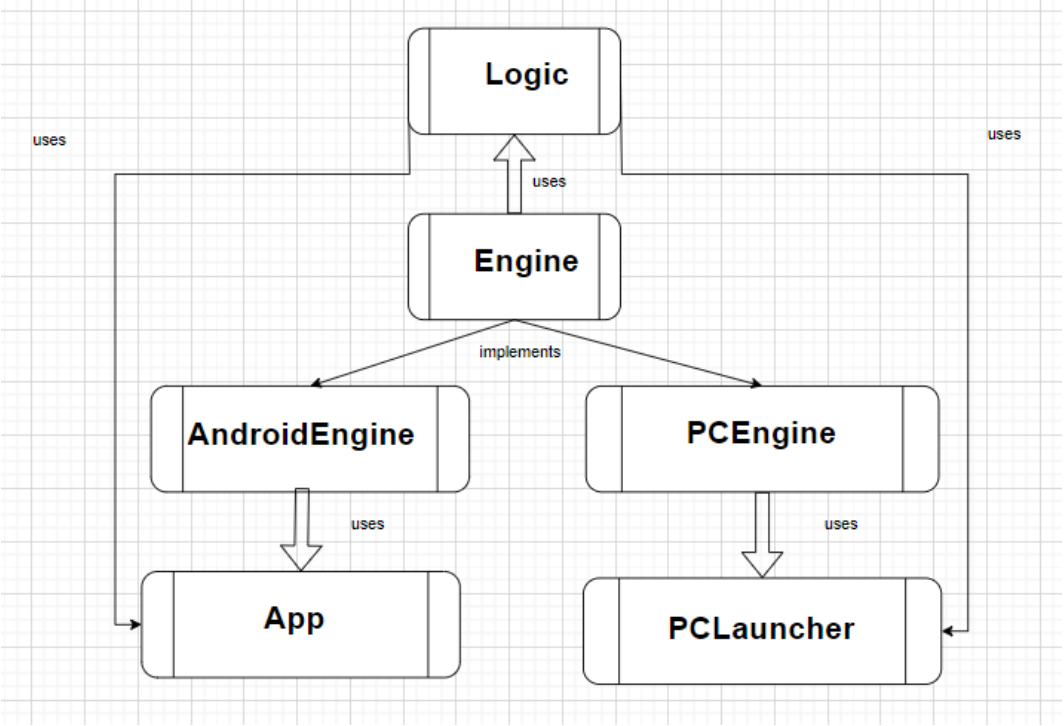
Práctica 1: Nonogramas

1. Introducción

Esta práctica se compone en un total de 6 módulos distintos, dos de ellos siendo launchers para diferentes plataformas (PC y Android respectivamente), tres ellos siendo el motor básico y sus adaptaciones para cada una de las versiones, y por último tenemos la lógica del juego en sí.



En esta imagen se puede observar los diferentes módulos y la comunicación entre ellos, como se puede ver la lógica se comunica con los launchers pero es para después definirse en los diferentes motores como el "juego" que se corre como se ve en la imagen de abajo.

```
engine = new AndroidEngine(renderView, context: this);
engine.getGraphics().setResolution( width: 400, height: 600);

Logic logic = new Logic(engine);
engine.setLogic(logic);
```

Fragmento de código del launcher de Android

A continuación se explicará en detalle el contenido de cada módulo así como las clases que los forman.

2. Módulos e implementación

2.1. Lógica

Este módulo como se ha explicado anteriormente, contiene la definición del juego en sí, así como las diferentes escenas del mismo

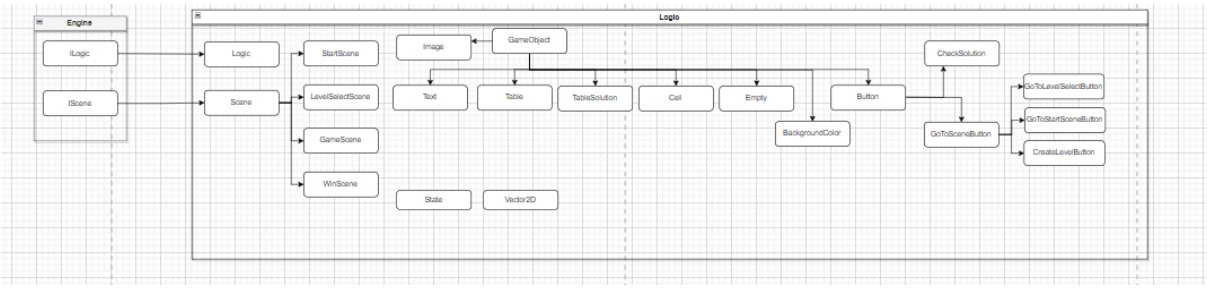


Diagrama con la herencia de las clases que están en el módulo de lógica

Como se puede observar en la imagen, la arquitectura del juego es muy básica usando *GameObjects* con herencia a cada uno de los elementos del juego (tablero, botones, casillas…) todo esto encapsulado en las distintas escenas del juego, que a su vez se llamarán en la lógica, corriendo solamente la escena que necesitemos.

2.2. Engine

Este módulo conforma de un motor básico para correr un juego(definido en la interfaz ILogic), este se compone en su mayoría de interfaces que se definen después en los respectivos motores para las respectivas plataformas.

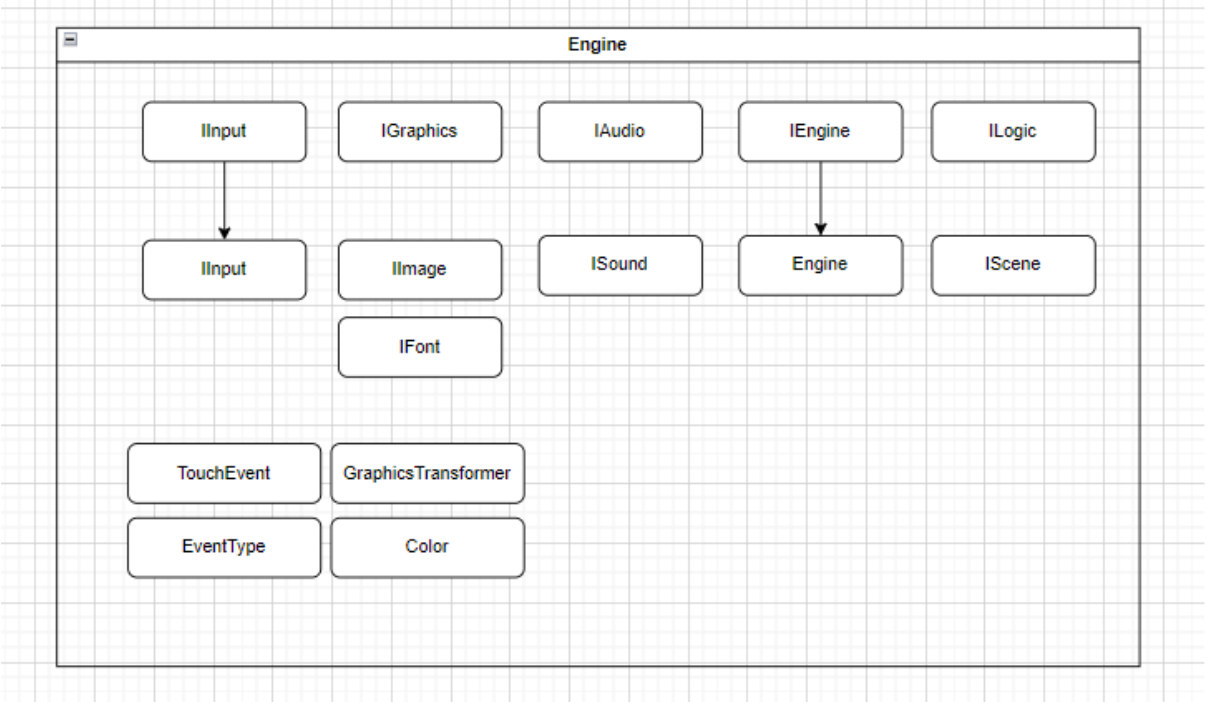


Diagrama con la herencia de las clases que están en el módulo de engine

El motor se compone de varios sistemas que llevan diferentes funciones:

→ **IGraphics:** Contiene métodos para manejar los gráficos dibujando diferentes elementos, imágenes(definida en la interfaz *IImage*) y textos con diferentes fuentes (definida en la interfaz *IFont*), con distintos colores (definida en la clase *Color*).
→**IAudio:** Contiene métodos para manejar el audio, es decir, reproducir sonidos(definidos en la interfaz *ISound*)
→**Input:** Maneja el input, contiene una lista de eventos(definidos en la clase *TouchEvent,* estos contienen una posición y un tipo de evento(EventType)).
→**IILogic:** Interfaz que define el juego a correr en el motor.
→**Engine:** Core del módulo, contiene métodos para definir los distintos sistemas que se usarán en el bucle principal de la aplicación.

2.3. AndroidEngine

Este módulo conforma una extensión del anterior pero con adaptaciones para que se pueda correr el juego en *Android*

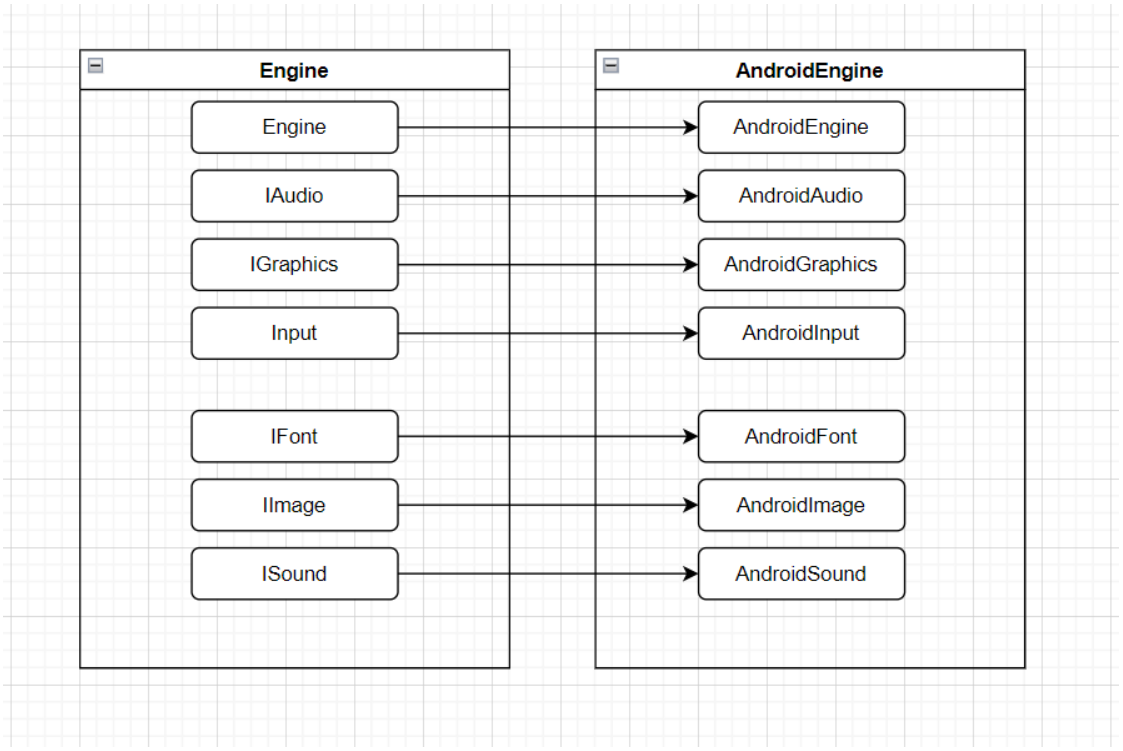


Diagrama con la herencia de las clases que están en el módulo de Android Engine

Como se puede observar todas las clases son una definición de interfaces y clases abstractas del módulo de *Engine* con las diferentes librerías de android.

2.4. PCEngine

Este módulo funciona similar al anterior, aunque en lugar de ampliar esta funcionalidad para *Android*, la amplia para que funcione en *PC.*

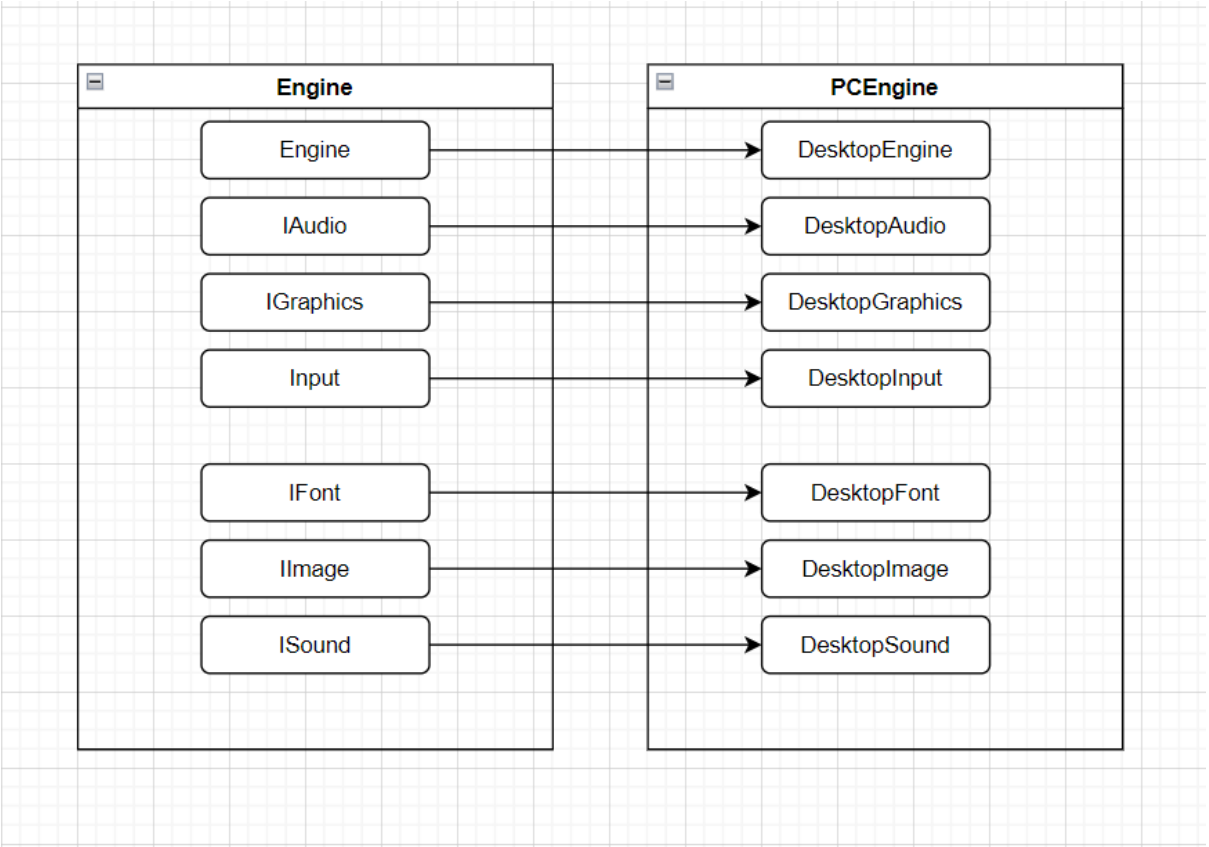


Diagrama con la herencia de las clases que están en el módulo de PC Engine

2.5. Launchers

Estos launchers son dos módulos que únicamente tienen una clase y ambos funcionan de una forma similar:

1. Crean el motor correspondiente (en el caso de Android además se define la SurfaceView que contiene nuestra aplicación)
2. Crean la lógica y esta, se asigna a su motor.
3. Se corre el juego (en el caso de Android, se corre en una hebra de ejecución aparte, en PC se llama directamente al método del bucle principal).