**Documento de arquitectura**

Índice

1. **Introducción**
2. **Arquitectura motor e implementación en android y PC**
3. **Arquitectura lógica e implementación**
4. **Sistema de vidas**
5. **Modo historia**
6. **Lectura de archivos**
7. **Guardado y escritura de archivos**
8. **Anuncios (sólo en Android)**
9. **Notificaciones (solo en Android)**
10. **Sensores (solo en Android)**

**Modo historia (con temáticas y niveles desbloqueables )**

1. **Introducción**

Respecto a la anterior práctica y la retroalimentación de la misma se han realizado los siguientes cambios:

* Cambio del sistema de Canvas [de porcentajes a posiciones reales]
* Métodos innecesarios como *getGraphics()* de *IFont*
* Modo de pantalla completa (sin el banner que viene por defecto con el título)

1. **Arquitectura motor e implementación en android y PC**
2. **Arquitectura lógica e implementación**
3. **Lectura de archivos**

Uno de los nuevos sistemas de ambos motores es el *FileManager* (definido como *IFileManager*, *AndroidFileManager* y *PCFileManager* respectivamente) que básicamente funciona como gestor de archivos, se encarga de abrir y leer archivos.

***//TODO: GRAFICO CON HERENCIA***

Métodos:

* ***InputStream openInputFile(String path) throws FileNotFoundException, IOException:*** Abre un archivo de entrada (a escribir) que se encuentra en la ruta *path.*
* ***OutputStream openOutputFile(String path):*** Abre un archivo de salida (a leer) que se encuentra en la ruta *path.*
* ***String readFile(String path):*** Devuelve todo el contenido de un archivo

1. **Modo historia**

Este modo de juego consta de una serie de niveles predefinidos los cuales se leen de archivos localizados en el directorio *Assets/levels* agrupados en 6 temáticas diferentes y 12 niveles cada uno:

* Bosque
* Jungla
* Mar
* Asia
* Juegos
* Ciudad

Estas temáticas, son desbloqueables cuando se resuelven los niveles de la temática anterior a excepción de la primera que está desbloqueada por defecto, lo mismo sucede con los niveles.

La carga de estos archivos la gestiona la clase *GameManager* que en general se encarga de toda la carga de archivos, así como último nivel desbloqueado, temática …

En cuanto a la implementación, se modificó la clase *Table* añadiendo nuevos métodos para distinguir si este era predefinido (en el método *Table.fromFile()* ) o aleatorio, se podría haber usado herencia para esta distinción pero los cambios eran muy ligeros

1. **Guardado y escritura de archivos**

***//TODO: GRAFICO CON HERENCIA Y EXPLICAR LO DEL SHA CON LA SALT***

En cuanto al guardado

1. **Anuncios (sólo en Android)**

En cuanto a los anuncios lo controla la clase *AndroidAdSystem*, cuya función es inicializar el sistema de anuncios

Esta se comunica con el Engine mediante IAdSystem, aparte de eso cuenta con una serie de clases privadas para manejar los callbacks y el hilo de ejecucion de mostrar el anuncio bonificado

1. **Notificaciones (solo en Android)**

Al principio, se implementó el sistema de notificaciones con la plataforma de *Firebase*, sin embargo,se cambió a un sistema de notificaciones local, este funciona mediante el *NotificationHandler* que tiene un ArrayList de notificaciones pendientes definidas como *Notification* y el canal de las mismas, estas se lanzan una vez se sale del juego mediante el *NotificationWorker* que se encarga de crear la notificación en sí con los datos de la clase *Notification*.

***//TODO: GRAFICO CON HERENCIA***

1. **Sensores (solo en Android)**

En cuanto a la implementación de los sensores, la lleva la clase *Sensors* que básicamente consiste en un sistema de listener llamados *ShakeListener* y un SensorManager que revisa en nuestro caso, el acelerómetro del móvil; cuando se detecte que se ha agitado y el tablero que te encuentras es uno aleatorio, cambiará sus soluciones, creando así un tablero diferente.

***//TODO: GRAFICO CON HERENCIA***

1. **Arquitectura lógica**