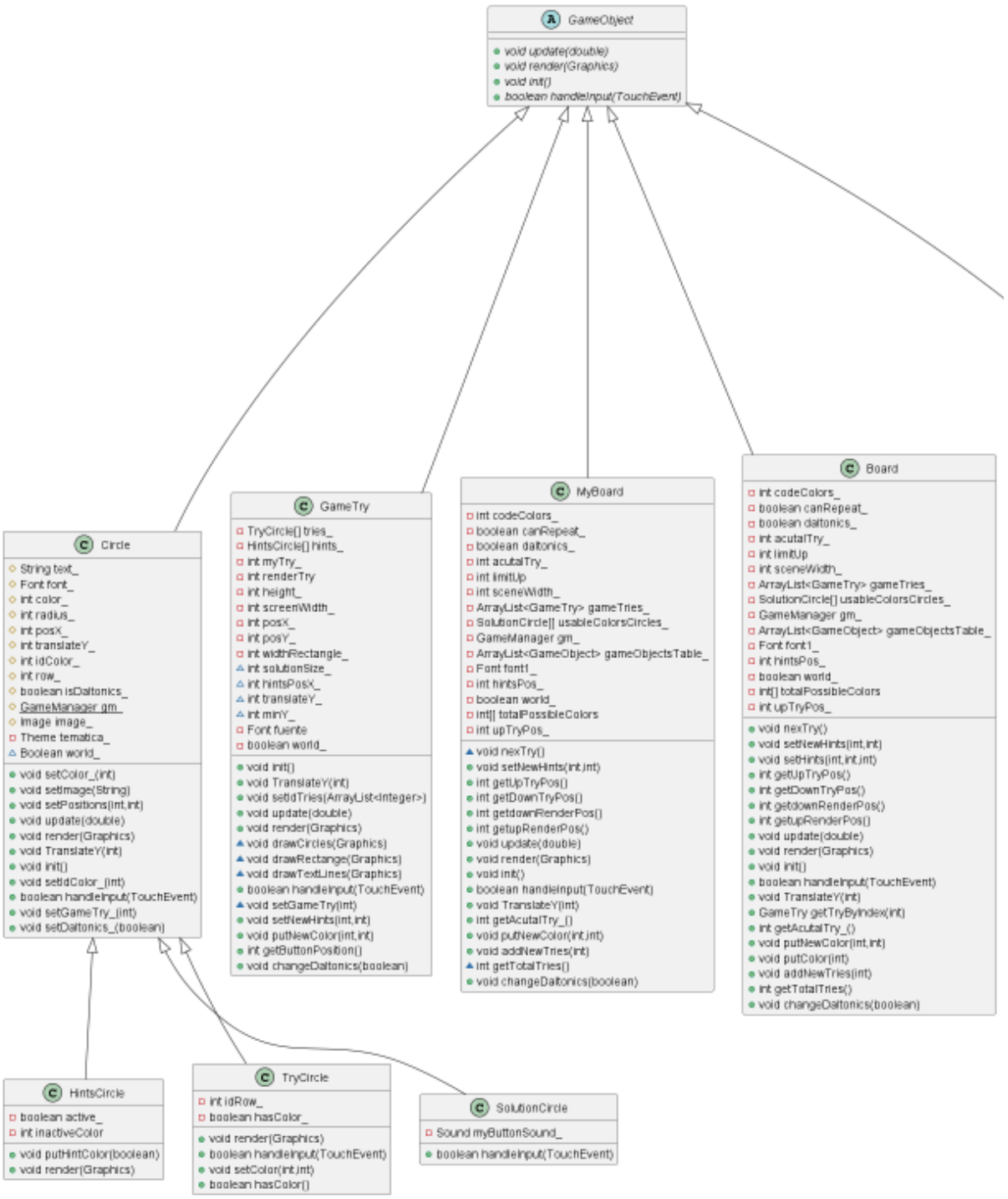
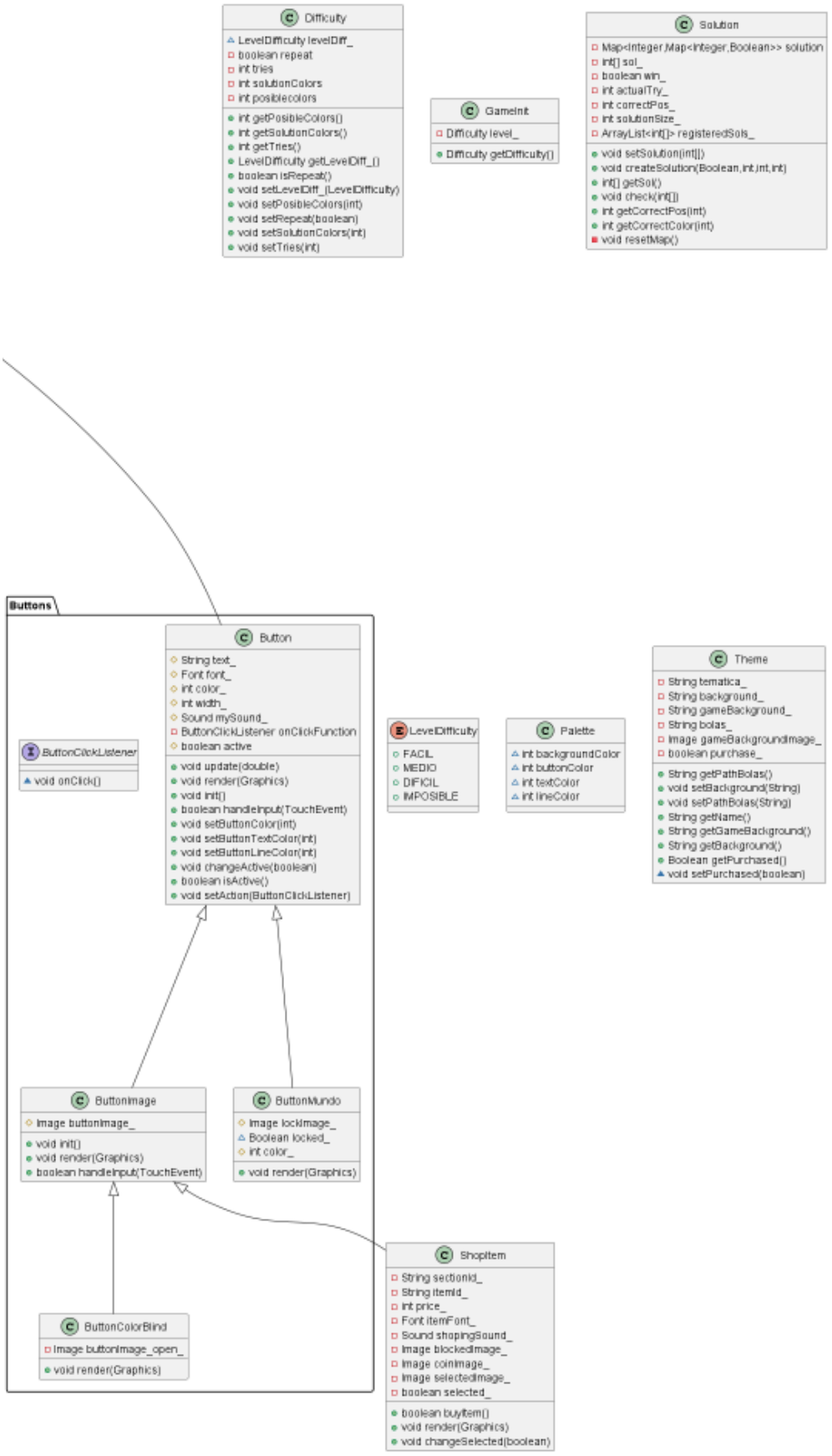
Miriam Martín Sánchez, Paula Morillas Alonso, Laura Gómez Bodego

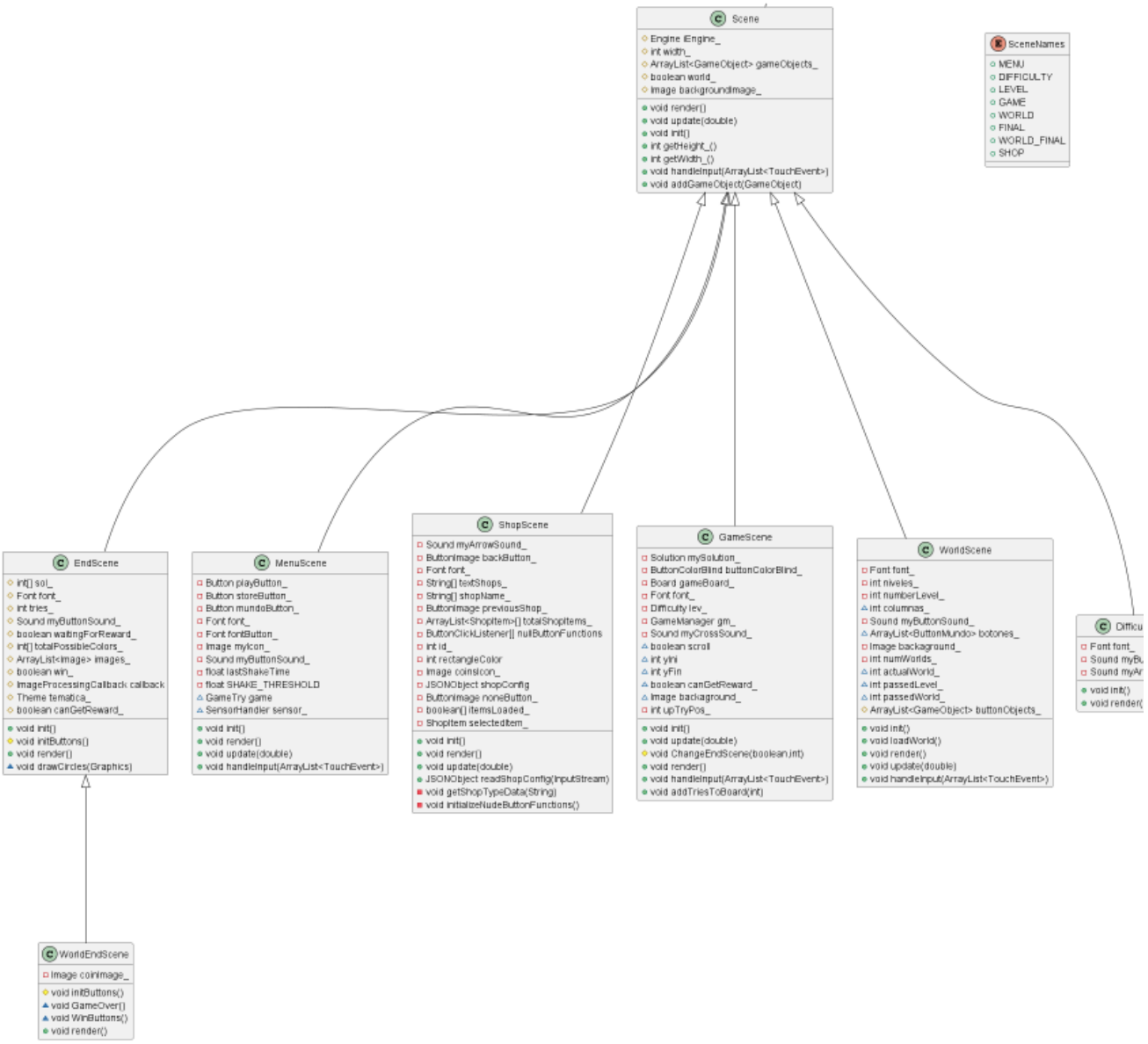
MÓVILES PRACTICA 2

Arquitectura de clases

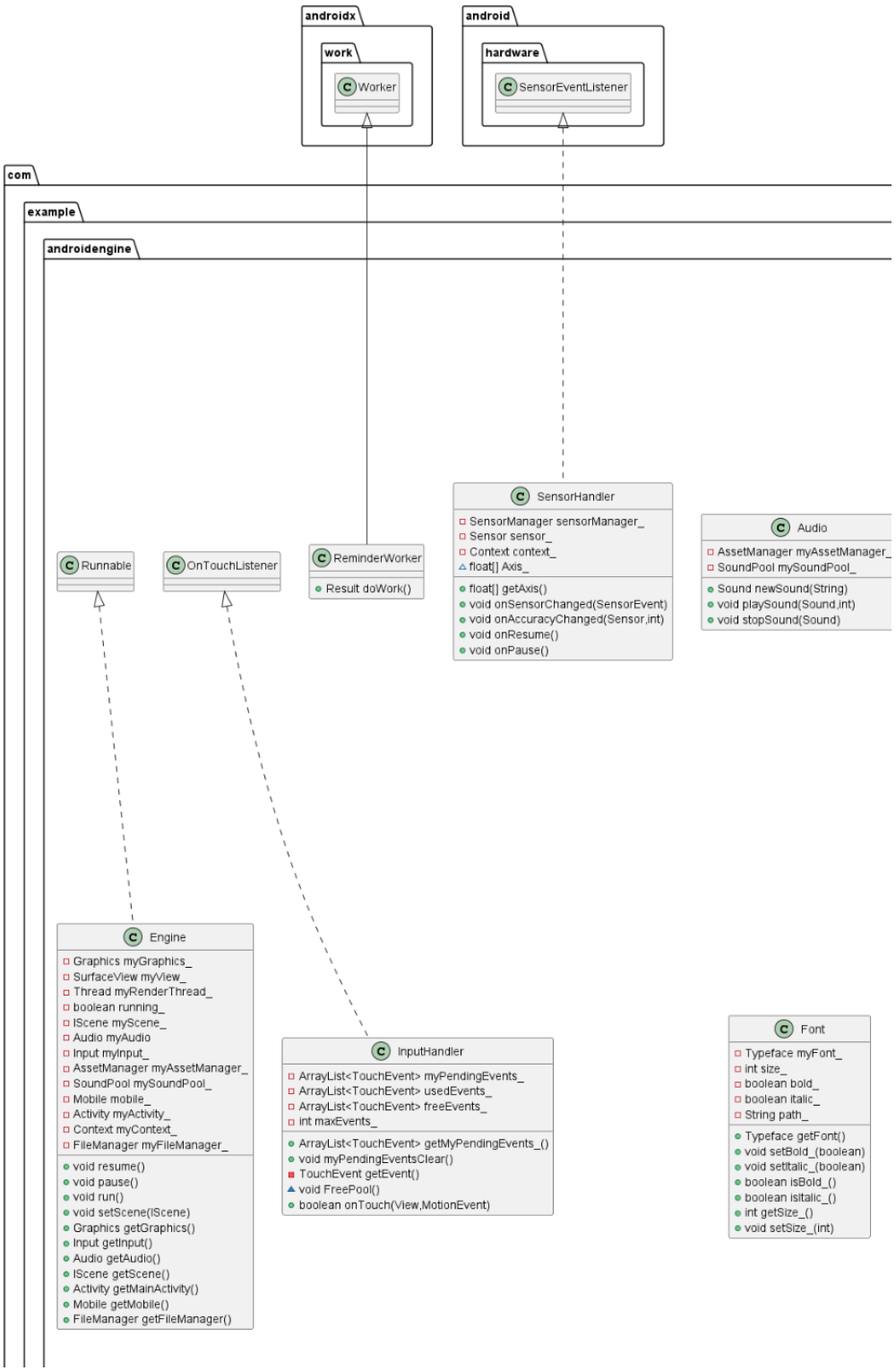
Al eliminar la parte correspondiente al desarrollo para escritorio, el proyecto se ha reducido a un módulo de aplicación y un módulo de librerías de android. Este primero, contiene la lógica del juego y se estructura de la siguiente manera:

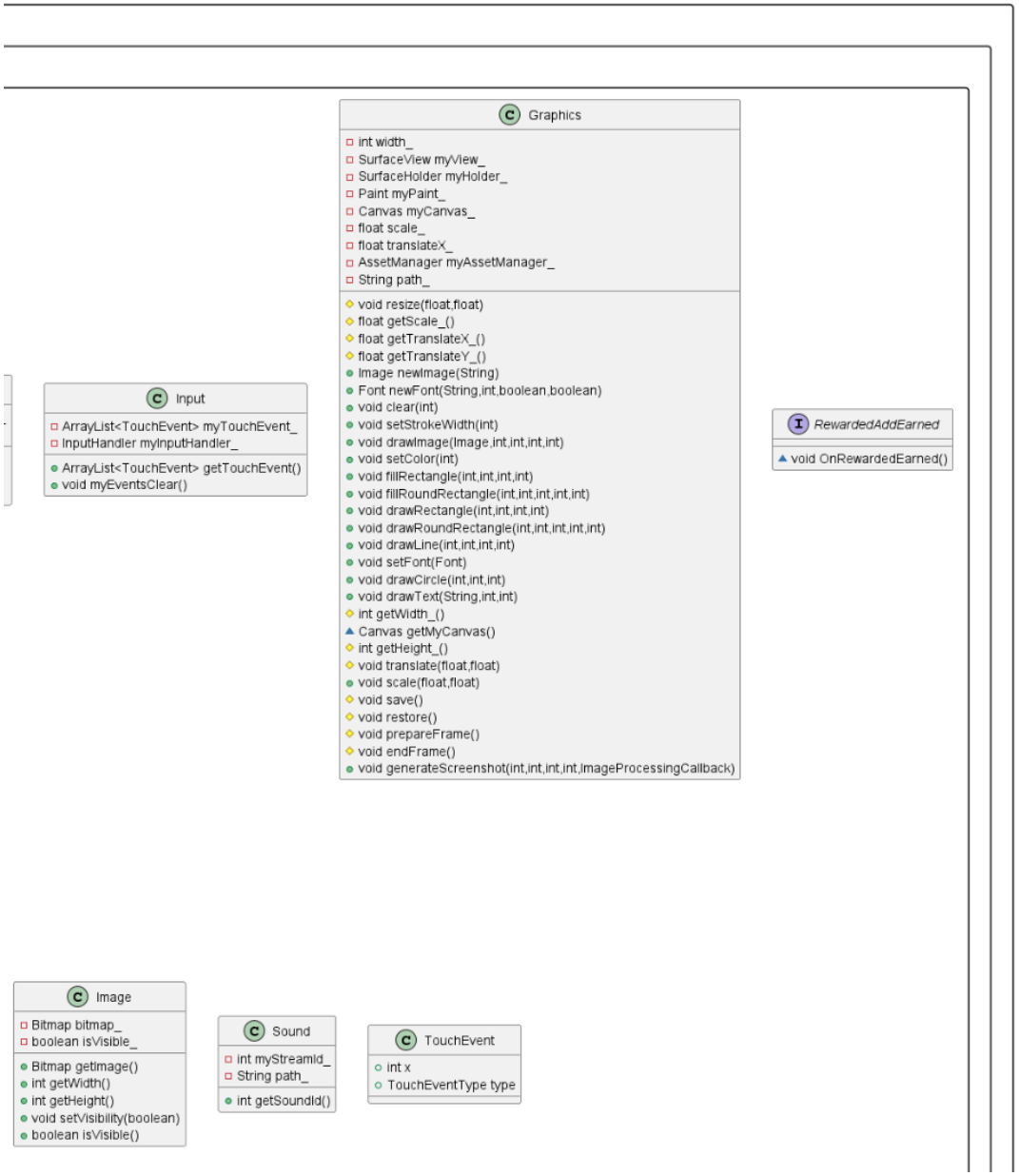


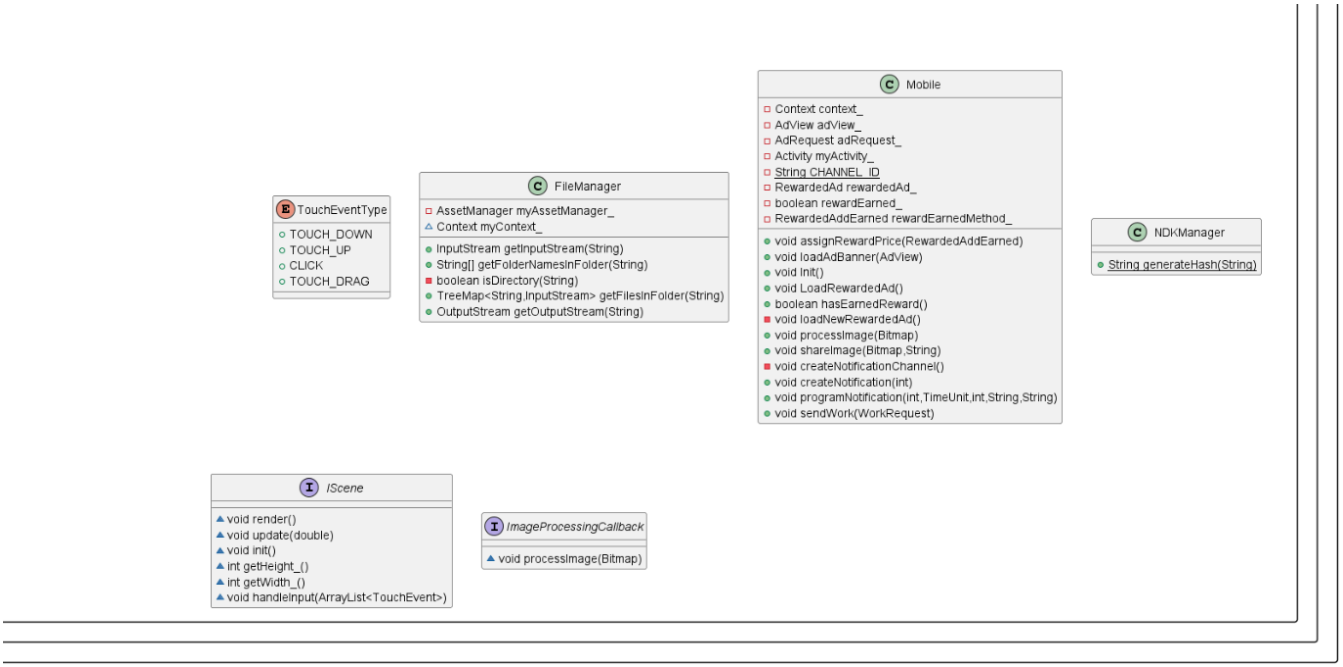

Difficulty
LevelDifficulty levelDiff_
boolean repeat
int tries
int solutionColors
int posiblecolors
int getPosibleColors()
int getSolutionColors()
int getTries()
LevelDifficulty getLevelDiff_()
boolean isRepeat()
void setLevelDiff_(LevelDifficulty)
void setPosibleColors(int)
void setRepeat(boolean)
void setSolutionColors(int)
void setTries(int)
GameInit
Difficulty level_
Difficulty getDifficulty()
Solution
Map<Integer,Map<Integer,Boolean>> solution
int[] sol_
boolean win_
int actualTry_
int correctPos_
int solutionSize_
ArrayList<int[]> registeredSols_
void setSolution(int[])
void createSolution(Boolean,int,int,int)
int[] getSol()
void check(int[])
int getCorrectPos(int)
int getCorrectColor(int)
void resetMap()
Buttons
Button
String text_
Font font_
int color_
int width_
Sound mySound_
ButtonClickListener onClickFunction
boolean active
void update(double)
void render(Graphics)
void init()
boolean handleInput(TouchEvent)
void setButtonColor(int)
void setButtonTextColor(int)
void setButtonLineColor(int)
void changeActive(boolean)
boolean isActive()
void setAction(ButtonClickListener)
ButtonClickListener
void onClick()
LevelDifficulty
FACIL
MEDIO
DIFICIL
IMPOSIBLE
Palette
int backgroundColor
int buttonColor
int textColor
int lineColor
Theme
String tematica_
String background_
String gameBackground_
String bolas_
Image gameBackgroundImage_
boolean purchase_
String getPathBolas()
void setBackground(String)
void setPathBolas(String)
String getName()
String getGameBackground()
String getBackground()
Boolean getPurchased()
void setPurchased(boolean)
ButtonImage
Image buttonImage_
void init()
void render(Graphics)
boolean handleInput(TouchEvent)
ButtonMundo
Image lockImage_
Boolean locked_
int color_
void render(Graphics)
ShopItem
String sectionId_
String itemId_
int price_
Font itemFont_
Sound shopingSound_
Image blockedImage_
Image coinImage_
Image selectedImage_
boolean selected_
boolean buyItem()
void render(Graphics)
void changeSelected(boolean)
ButtonColorBlind
Image buttonImage_open_
void render(Graphics)



El módulo donde gestionamos la parte del motor, se estructura de la siguiente forma:


androidx
work
Worker
android
hardware
SensorEventListener
com
example
androidengine
SensorHandler
SensorManager sensorManager_
Sensor sensor_
Context context_
float[] Axis_
float[] getAxis()
void onSensorChanged(SensorEvent)
void onAccuracyChanged(Sensor,int)
void onResume()
void onPause()
Audio
AssetManager myAssetManager_
SoundPool mySoundPool_
Sound newSound(String)
void playSound(Sound,int)
void stopSound(Sound)
Runnable
OnTouchListener
ReminderWorker
Result doWork()
Engine
Graphics myGraphics_
SurfaceView myView_
Thread myRenderThread_
boolean running_
IScene myScene_
Audio myAudio
Input myInput_
AssetManager myAssetManager_
SoundPool mySoundPool_
Mobile mobile_
Activity myActivity_
Context myContext_
FileManager myFileManager_
void resume()
void pause()
void run()
void setScene(IScene)
Graphics getGraphics()
Input getInput()
Audio getAudio()
IScene getScene()
Activity getMainActivity()
Mobile getMobile()
FileManager getFileManager()
InputHandler
ArrayList<TouchEvent> myPendingEvents_
ArrayList<TouchEvent> usedEvents_
ArrayList<TouchEvent> freeEvents_
int maxEvents_
ArrayList<TouchEvent> getMyPendingEvents_()
void myPendingEventsClear()
TouchEvent getEvent()
void FreePool()
boolean onTouch(View,MotionEvent)
Font
Typeface myFont_
int size_
boolean bold_
boolean italic_
String path_
Typeface getFont()
void setBold_(boolean)
void setItalic_(boolean)
boolean isBold_()
boolean isItalic_()
int getSize_()
void setSize_(int)


Graphics
int width_
SurfaceView myView_
SurfaceHolder myHolder_
Paint myPaint_
Canvas myCanvas_
float scale_
float translateX_
AssetManager myAssetManager_
String path_
void resize(float,float)
float getScale_()
float getTranslateX_()
float getTranslateY_()
Image newImage(String)
Font newFont(String,int,boolean,boolean)
void clear(int)
void setStrokeWidth(int)
void drawImage(Image,int,int,int,int)
void setColor(int)
void fillRectangle(int,int,int,int)
void fillRoundRectangle(int,int,int,int,int)
void drawRectangle(int,int,int,int)
void drawRoundRectangle(int,int,int,int,int)
void drawLine(int,int,int,int)
void setFont(Font)
void drawCircle(int,int,int)
void drawText(String,int,int)
int getWidth_()
Canvas getMyCanvas()
int getHeight_()
void translate(float,float)
void scale(float,float)
void save()
void restore()
void prepareFrame()
void endFrame()
void generateScreenshot(int,int,int,int,imageProcessingCallback)
Input
ArrayList<TouchEvent> myTouchEvent_
InputHandler myInputHandler_
ArrayList<TouchEvent> getTouchEvent()
void myEventsClear()
RewardedAddEarned
void OnRewardedEarned()
Image
Bitmap bitmap_
boolean isVisible_
Bitmap getImage()
int getWidth()
int getHeight()
void setVisibility(boolean)
boolean isVisible()
Sound
int myStreamId_
String path_
int getSoundId()
TouchEvent
int x
TouchEventType type



Recompensas que puede conseguir el jugador

- Cuando pierdes una partida en mastermind, aparece un botón +2 intentos . Al ser pulsado te llevará a un anuncio y cuando este se acabe, el jugador tendrá dos intentos más.
- Si se gana una partida el jugador recibirá una recompensa de 10 monedas.
- Si en mastermind entras desde una notificación el jugador recibirá 5 monedas más.

Notificaciones

En la clase Mobile de “Android Engine” hay un método llamado `programNotification()` que programa una notificación dado un tiempo,titulo,icono…En el onPause de MainActivity llama a este método(que lanzará una aplicación 45 segundos después de ser cerrada la app)

Sensores

Se ha implementado un sensor que es un acelerómetro. En la escena Menú (la primera que aparece,en otras escenas no hará este comportamiento) si se agita el móvil se reproducirá un sonido de maracas.

Persistencia y guardado en archivos

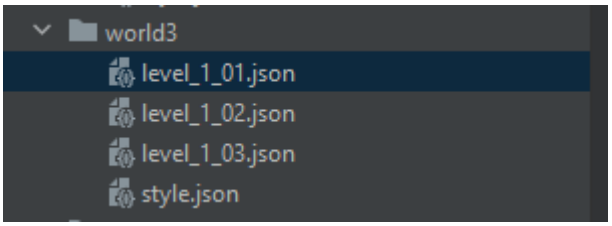
Todos los datos necesarios para guardar el progreso del jugador (mundo y nivel en el que se encuentra, items desbloqueados en la tienda, monedas, etc.) se encuentran en un archivo “saved_data.json” en el que se escribe al pausar el juego y del que se leen los datos al reanudarlo. El archivo se almacena en modo privado, en el contexto de la actividad.

Comportamientos inesperados y su motivo

Al leer el archivo de guardado de una partida, se compara su hash con el hash generado al guardar la partida (“hash_data.json”) Cuando un jugador intente cambiar los datos de guardado de su partida, cambie el valor que cambie en este archivo, su número de monedas en la tienda pasará a ser *-100* como penalización. De esta forma, no podrá comprar nuevos ítems hasta ganar varias partidas.

Cómo agregar mundos

En la carpeta Levels de assets se pueden agregar carpetas,con este estilo de formato

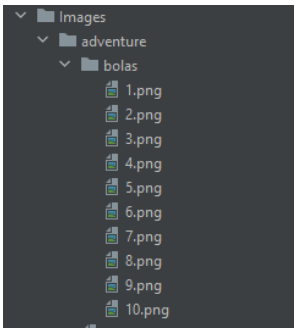


En ellas debe haber un style.json

```
{
 "style": "ADVENTURE",
 "background": "adventure/foreground.jpg",
 "gameBackground":"adventure/foreground.jpg",
 "bolas": "adventure/bolas/"

}
```

"bolas" es la carpeta donde se encuentran las imágenes de las bolas todas con formato 1.png ,para que se lean estas imágenes deben ser números. Si se elimina alguna de estas imágenes se pondrán los círculos por defecto,y si se pone otro formato de imagen no se mostrará en el juego.



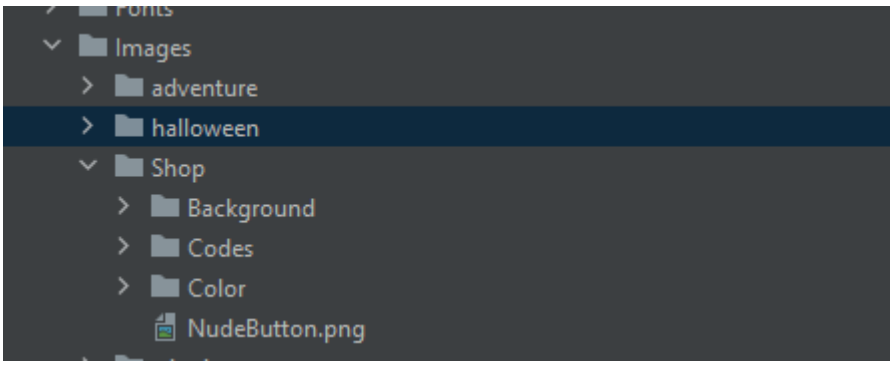
Para meter niveles se debe incluir json con este formato

```
{
 "codeSize": 2,
 "codeOpt": 4,
 "repeat": false,
 "attempts": 4
}
```

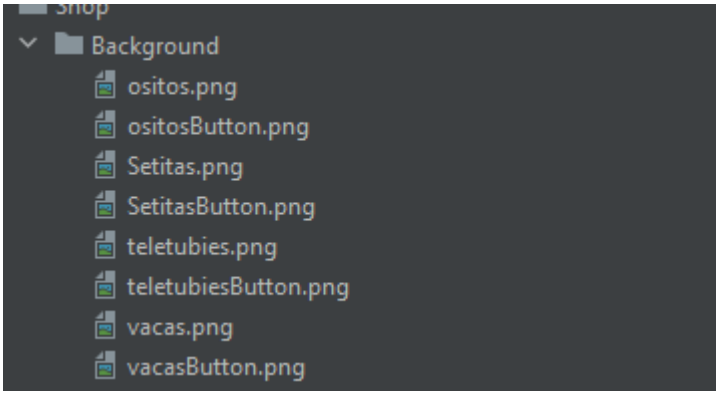
Toda la lectura de los mundos se hace en la carpeta GameLogic/Utils/LevelReader
Esta lectura se hace una única vez cuando se lanza la app.

Cómo agregar items a la tienda

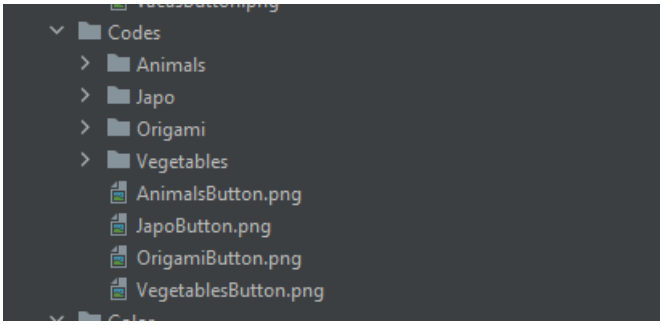
Dentro de la carpeta Shop en assets/Images hay tres carpetas: Background, Codes y Color.
Todas las imágenes deben estar en formato .png.

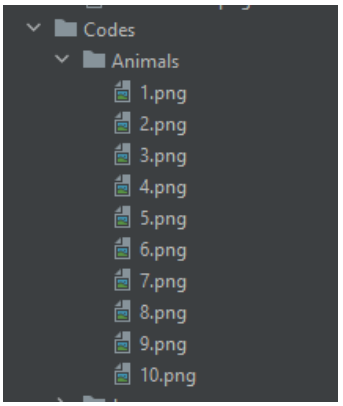


Para meter un nuevo fondo hay que introducir la imagen en la carpeta “Background” y un botón que será el que de acceso a este fondo desde la escena. El botón y el fondo deben tener el mismo nombre, pero el botón tiene que tener añadida la palabra “Button” después del nombre.

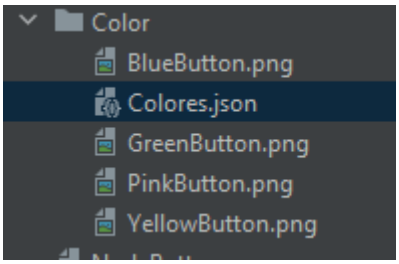


Para meter un nuevo grupo de códigos hay que crear una carpeta dentro de la carpeta “Codes”, con el nombre del paquete de códigos, y añadir la imagen botón (fuera de la carpeta concreta, en Codes) con el mismo nombre que la carpeta de los códigos a los que dará acceso, añadiendo el sufijo “Button”. Dentro de cada carpeta de códigos las imágenes deben estar numeradas del 1 al 10.





Para añadir una nueva paleta de colores hay que añadir la imagen del botón en la carpeta “Color” con el nombre de la paleta seguido de “Button”. Dentro del archivo “Colores.json” hay que añadir una nueva sección con el nombre de la paleta, que dentro tendrá un atributo “palette”, que es un array de hexadecimales con los 4 colores deseados para la paleta.



Cada elemento que se añade en la tienda debe incluirse en el archivo “Shops.json”, dentro de la carpeta assets/Shop. Debe añadirse en el array “ButtonsImages” en la sección correspondiente a cada tienda, con el nombre común a la imagen y botón. En el array “precios” hay que añadir el precio asignado a ese ítem de la tienda (los precios se corresponden a cada ítem que está en su misma posición del array “ButtonsImages”).

Cada sección de la tienda se lee solamente la primera vez que se accede a esta. Cada vez que se accede a una sección de la tienda, se comprueba si ya está cargada, si no es así, la lee del archivo, y si no, carga los ítems ya guardados.