HOW TO MAKE A SCROLLING SCREEN DOCUMENT

In this document, the following topics are going to be explained:

* Introduction of the copter flying game
* Pros and cons of the game
* Requirements for making the scrolling effect
* How to make the effect
* Additional part about the game features
* References

# Introduction Of The Copter Flying Game

The game’s concept is similar to Flappy Bird. User just taps the screen to navigate the helicopter up and down so that it won’t hit the obstacles, which are the cave terrains, the pointy pillars and flying rockets. On the journey, the user also needs to control the helicopter to pick the floating items for obtaining scores, fuel and additional lives. The game has no rounds and it’s over when user plays until he/she lost all the lives or ran out of fuel.

The cave terrains are already set on the screen. The cave pillars will randomly spawn after a random period of time. The rocket, however, will aim at the position of the helicopter to shoot. There will be a short warning before the rocket appears. As time pass, the warning time is shorter till the point that it appears right before the rocket flies in. User can choose the different game speed in the Settings to challenge his/herself. To make the game more difficult, some floating items will spawn near the pillar so user has to be careful.

As the game begin, the user is given 5 lives and 200 fuel points. The objects in the game will adjust player’s scores, lives and fuel points as follow:

* [Floating Item] Star: +10 scores
* [Floating Item] Heart: + 1 life
* [Floating Item] Fuel: +20
* [Obstacle] Cave terrain/ceiling: Game Over
* [Obstacle] Cave pillar: -1 life
* [Obstacle] Flying rocket: -1 life

# Pros and Cons

## Pros

* Player touch the mobile screen to play. There is no need for additional devices.
* Easy to play
* Nice music and sound

## Cons

* Multiplayer feature is not completed
* Score storing on cloud has not perfect. The backend that is used to upload the scores, parse.com, has some system error:

com.mygdx.game.android W/System.err: com.parse.ParseRequest$ParseRequestException: bad json response

It returns a JSON object that could not be read by the android device.

* Only suitable with devices that has the latest SDK version

# Game Requirements

* Android mobile device with SDK version is 23.
* Some images to make the scrolling screen:
  + The upper and lower cave ceilings/terrains

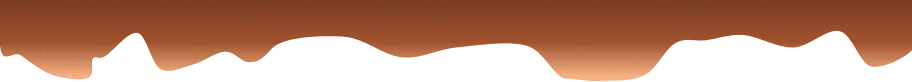


Figure 1: The upper cave ceiling/terrain

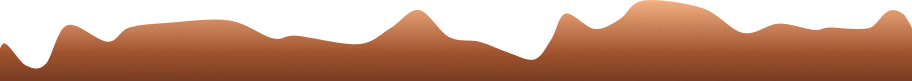


Figure 2: The lower cave ceiling/terrain

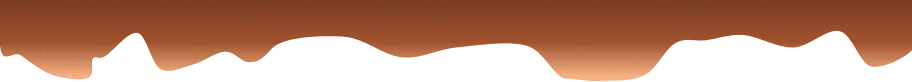
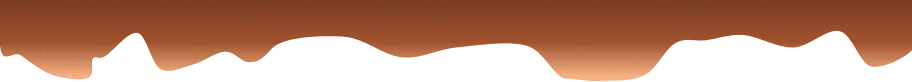
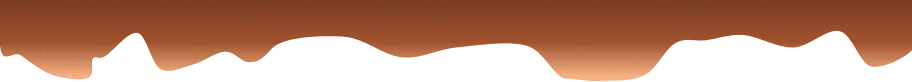
* + The background image

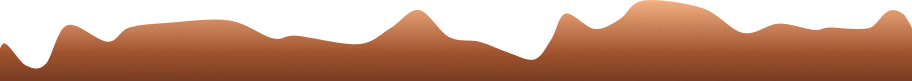
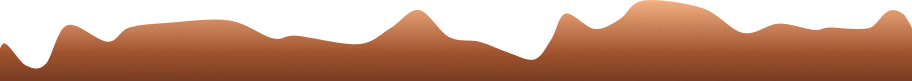
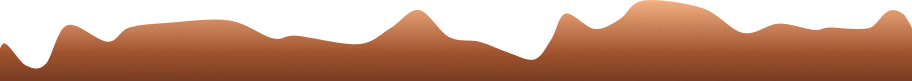


# How To Make The Scrolling Effect For The Background

The basic concept is:

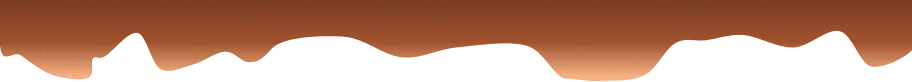
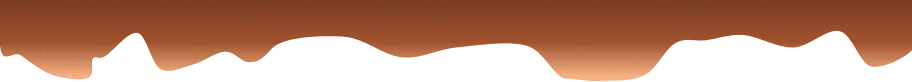
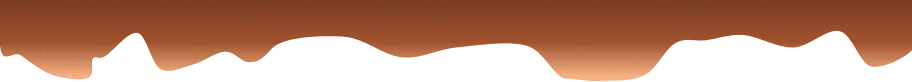
* First, draw 3 images of the upper cave terrain that located next to each other and 3 of the lower cave ceiling in the same way.

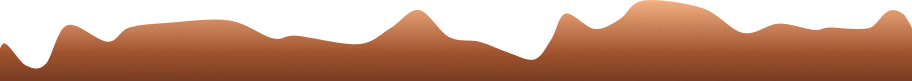
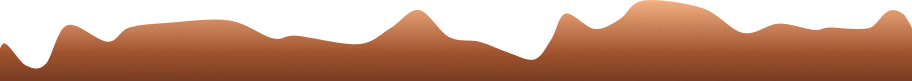
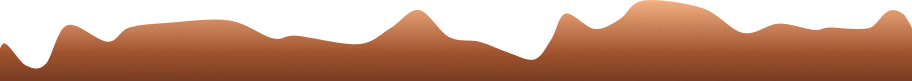




* Then, create a camera with a view port that fits one image of the cave terrain.

**CAMERA VIEW PORT**





* After that, make all these initiative images move gradually to the left. If an image has left the camera’s view port, destroy it and create a new one at the end of the image line.

Step 1:

Make an abstract class called GameObject. This class will defines all the common attributes and actions of a game object. There are many data properties and object actions in the GameObject.class file but the ones you should take note right now are:

* Data properties:
  + **Vector2 pos:** defines the current position of the object
  + **TextureRegion textureRegion or Texture texture:** defines the image of the object. Some objects use TextureRegion, some apply pure Texture.
  + **SpriteBatch sprite:** defines the tool that draws the object
* Behaviours:
  + **update() method:** updates the position of the object.
  + **render() method:** renders the position and image of the object on the game screen.



## Step 2:

Make a class called Cave Object that extends the GameObject class. The interface implemented in this class is used for Game Saving, which is not really matters for this scrolling effect.

In this class, you need to define one more property. That is the type of the CaveObject.

The constructor of this object focuses on 3 data fields, as you see in the picture below:

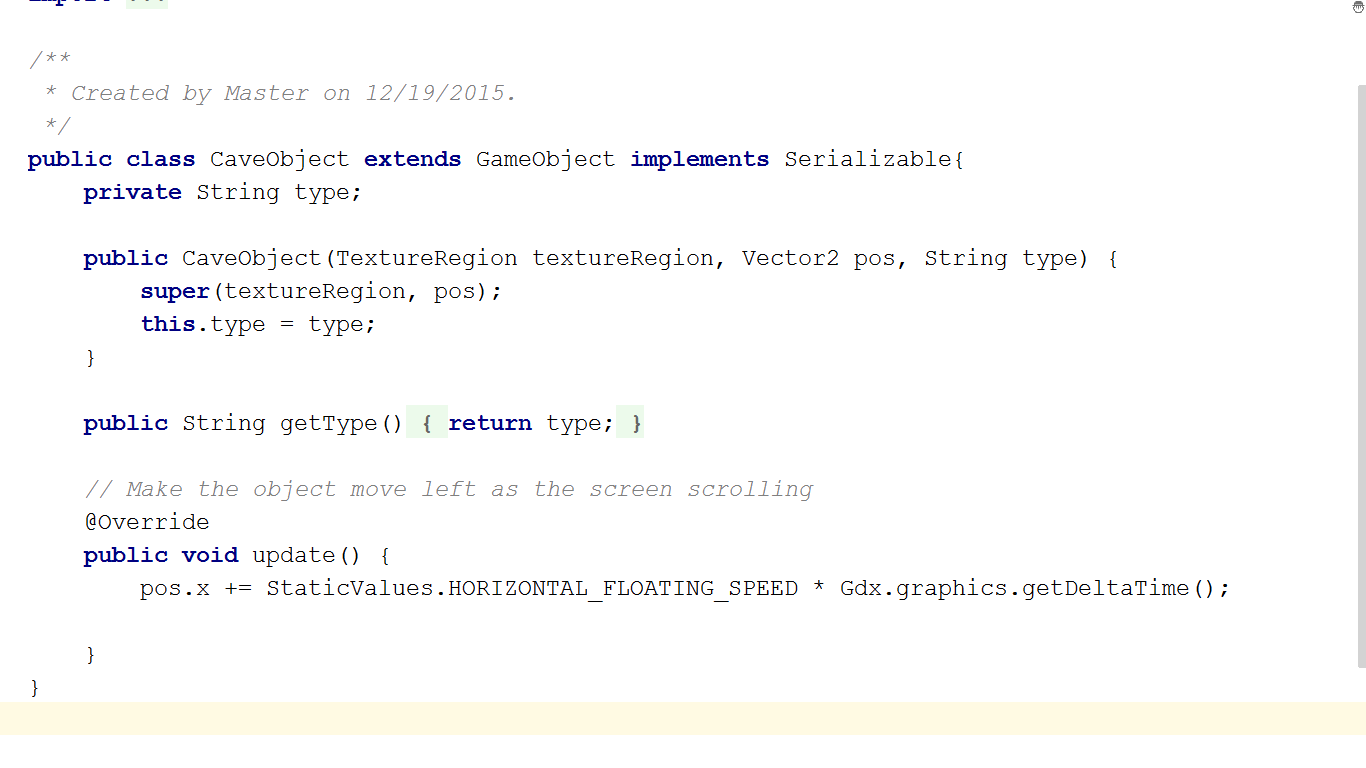
* The image of the CaveObject, defined by **TextureRegion textureRegion**
* The position of the object, defined by **Vector2 pos**
* The type of the object, whether it is the cave terrain or other cave obstacles, defined by **String type**

The **update()** method changes the x-coordinate of the object according to the *delta time*\*. The **HORIZONTAL\_FLOATING\_SPEED** is a static value, defined in StaticValues class, a class that holds all that static properties uses across many classes in the application. The method **Gdx.graphics.getDeltaTime()** gets the delta time value of the screen.

\*\*\* SO WHAT IS DELTA TIME?

According to what Mr.Bose (2014) writes in his book “Libgdx Game Development Essentials”, delta time value represents the time elapsed after the last frame was drawn. On a fast device, this will be less and this value will be more on a slow device, which essentially means that multiplying a value with this gives a result that is the same on both devices.

Therefore, in order for the animation runs smoothly on the device, we multiple the floating speed of the screen with this value to create a frame-based animation, where the animation is directly controlled by the frame rate or FPS. That means it will prevent the situation when the animation runs un-smoothly on slower device due to low frame rate.



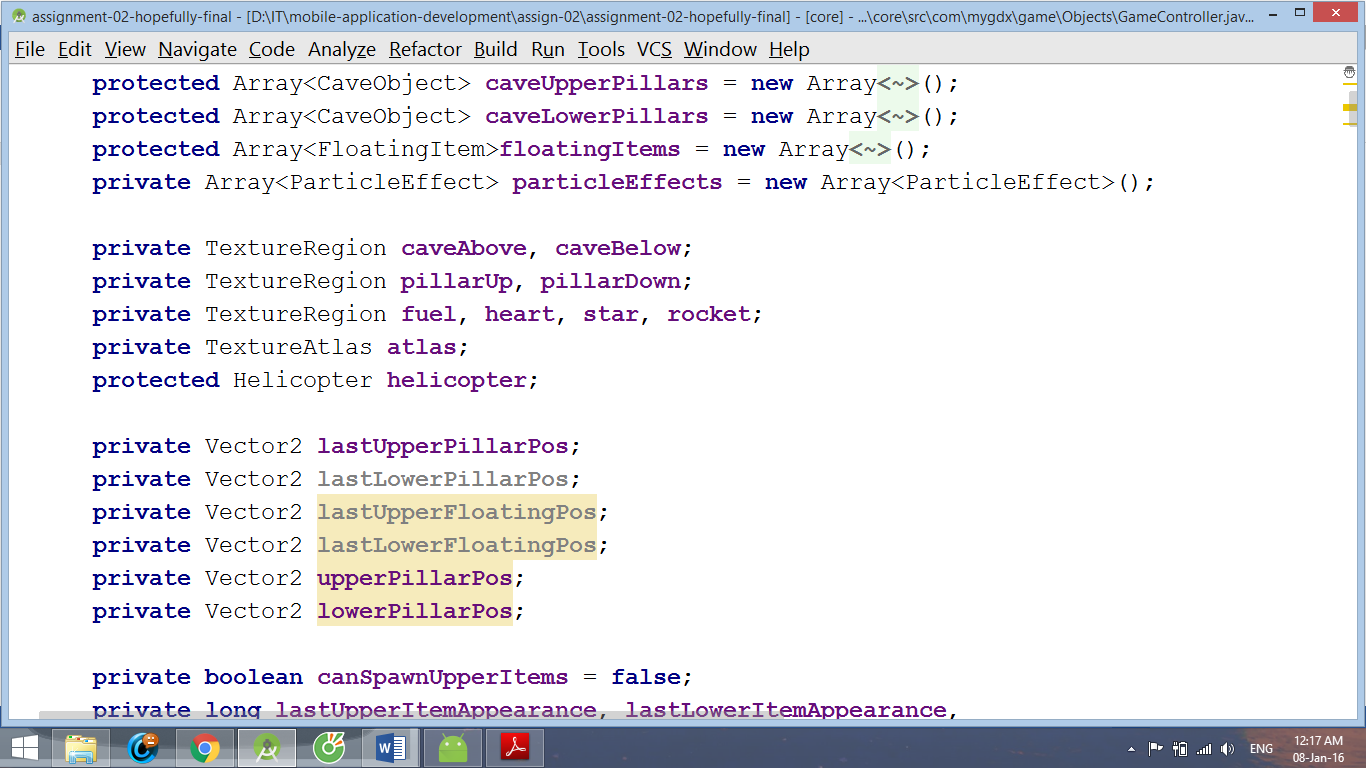
## Step 3:

Create GameController class that stores in the logic of the game. In this class, certain things needs to be done to achieve this effect.

### 3.1. Create an Array of CaveObject objects called caveTerrains

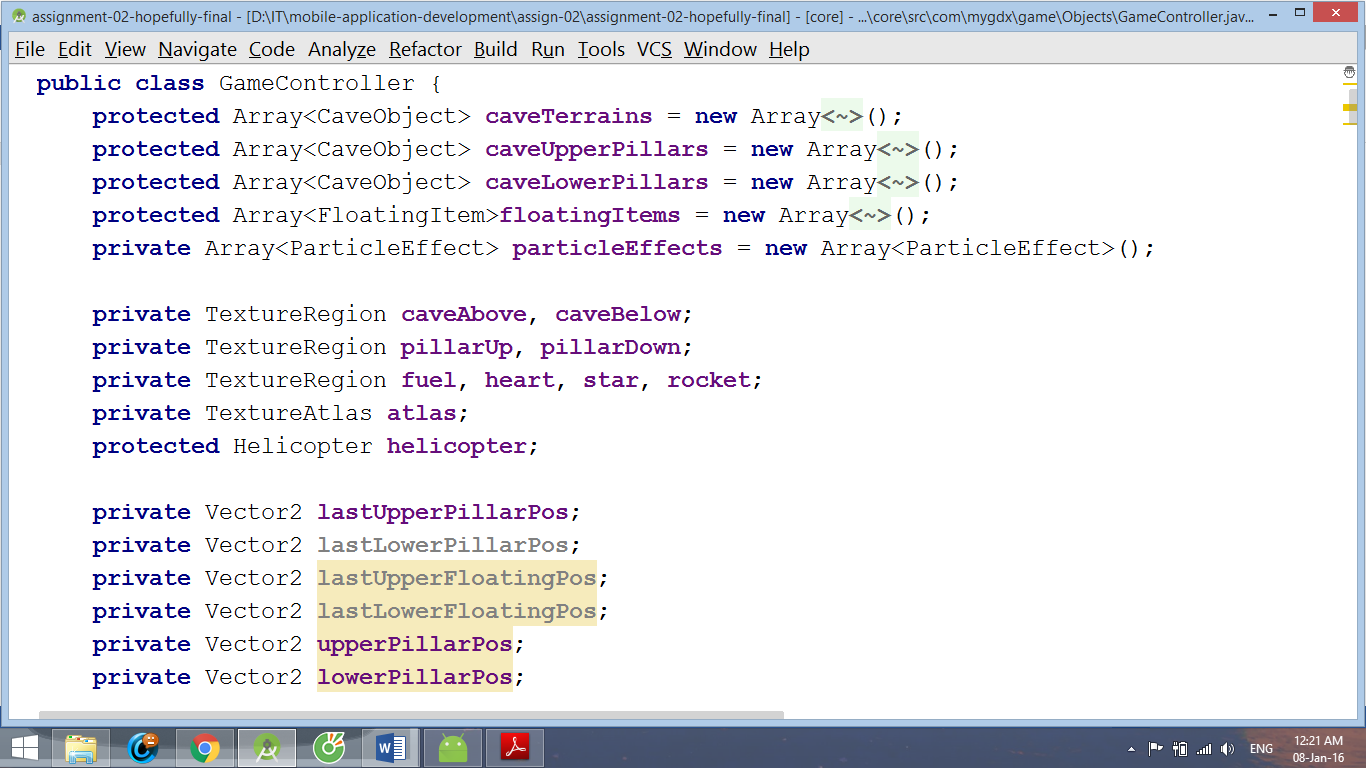


### 3.2. For each cave terrain (upper cave and lower cave), create an image that shows how it looks like on the screen.

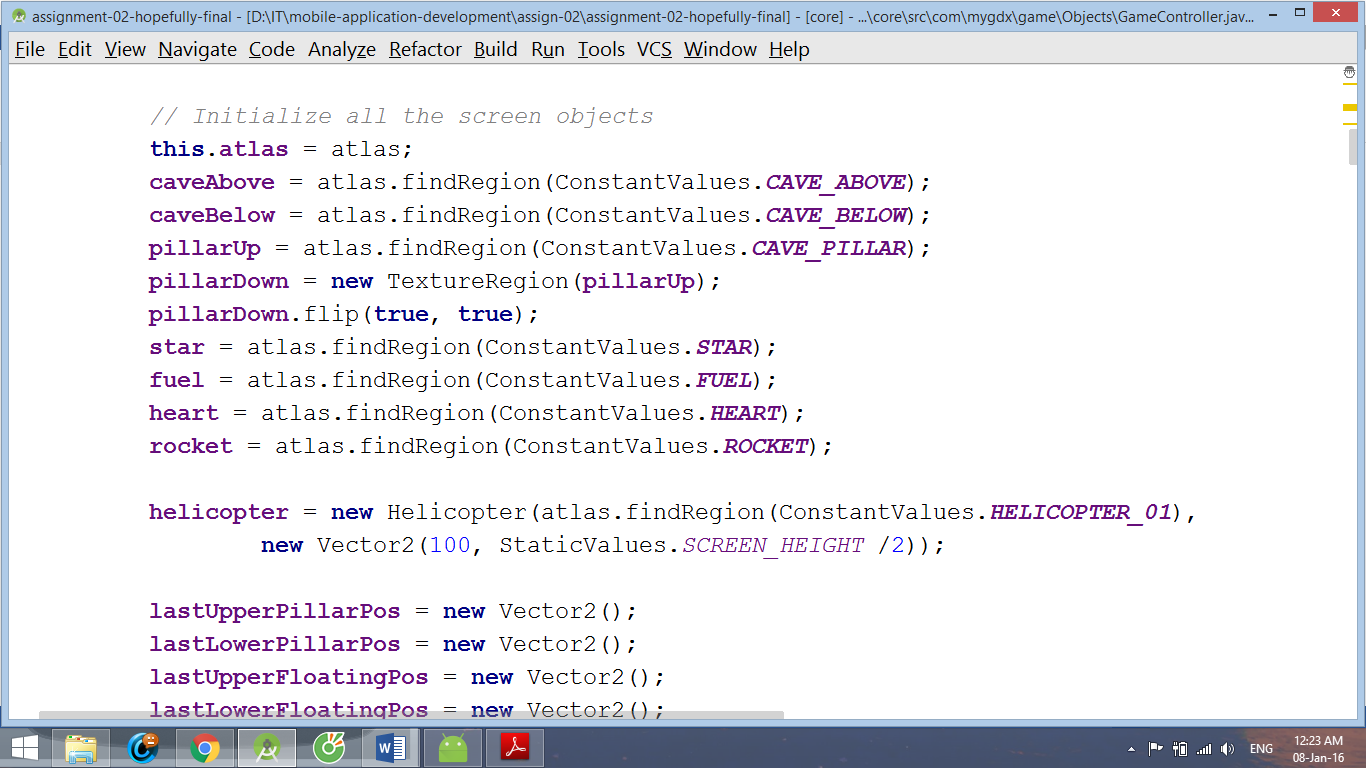


### In the constructor, assign values for these TextureRegion objects. In order to do so, create a TextureAtlas that holds all the game images first. Then, use findRegion() method to extract the texture regions for the caveAbove and caveBelow.

Create a texture atlas that holds all the game images



Create the texture regions for caveAbove and caveBelow.



ConstantValues.CAVE\_ABOVE is a final static String defined in ConstantValues.class to hold the caveAbove’s region name in the texture atlas. The same goes to ConstantValues.CAVE\_BELOW

### Create initScene() method that sets up how the cave terrains will be put when the screen first starts.

In this method, add 3 CaveObject objects that define the upper cave terrain to the caveTerrains Array. They will be set next to one another vertically.

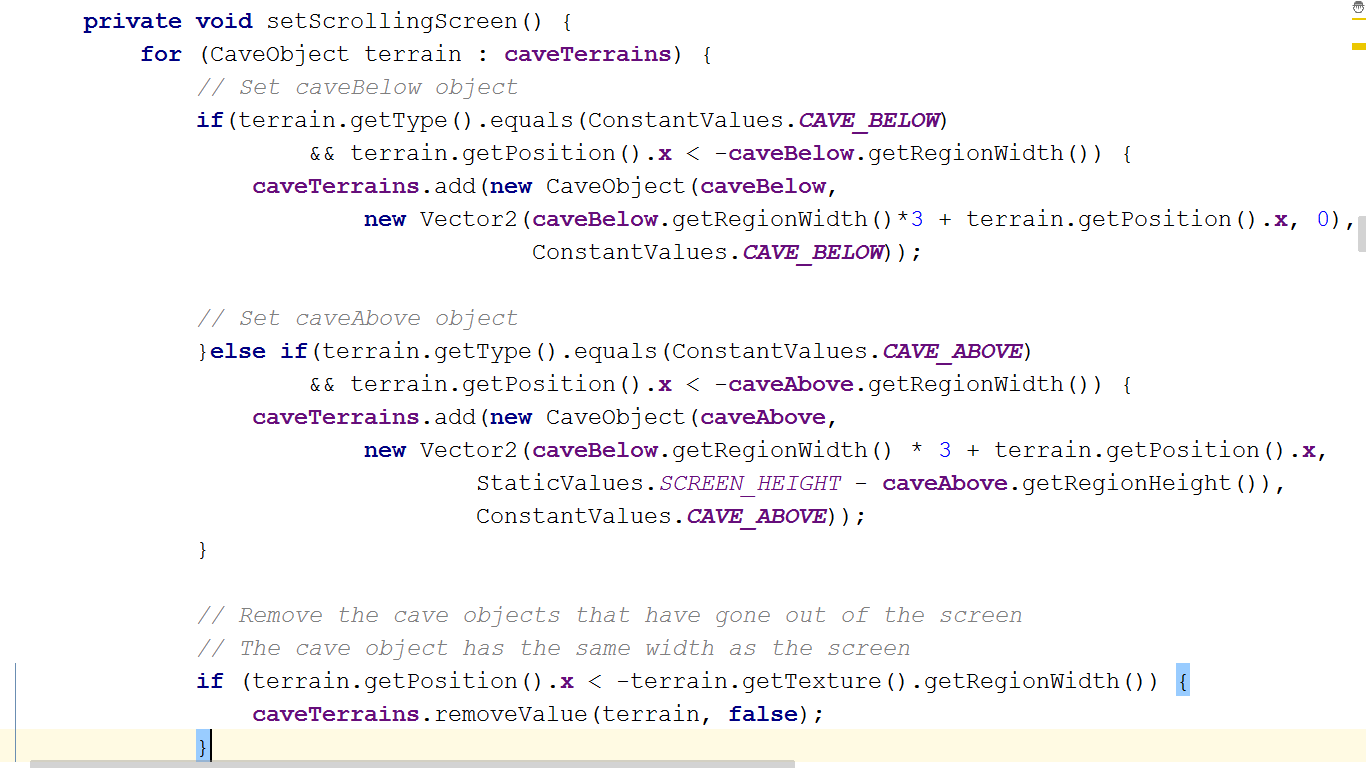
Then, add 3 CaveObject objects that define the lower cave terrain in the same way

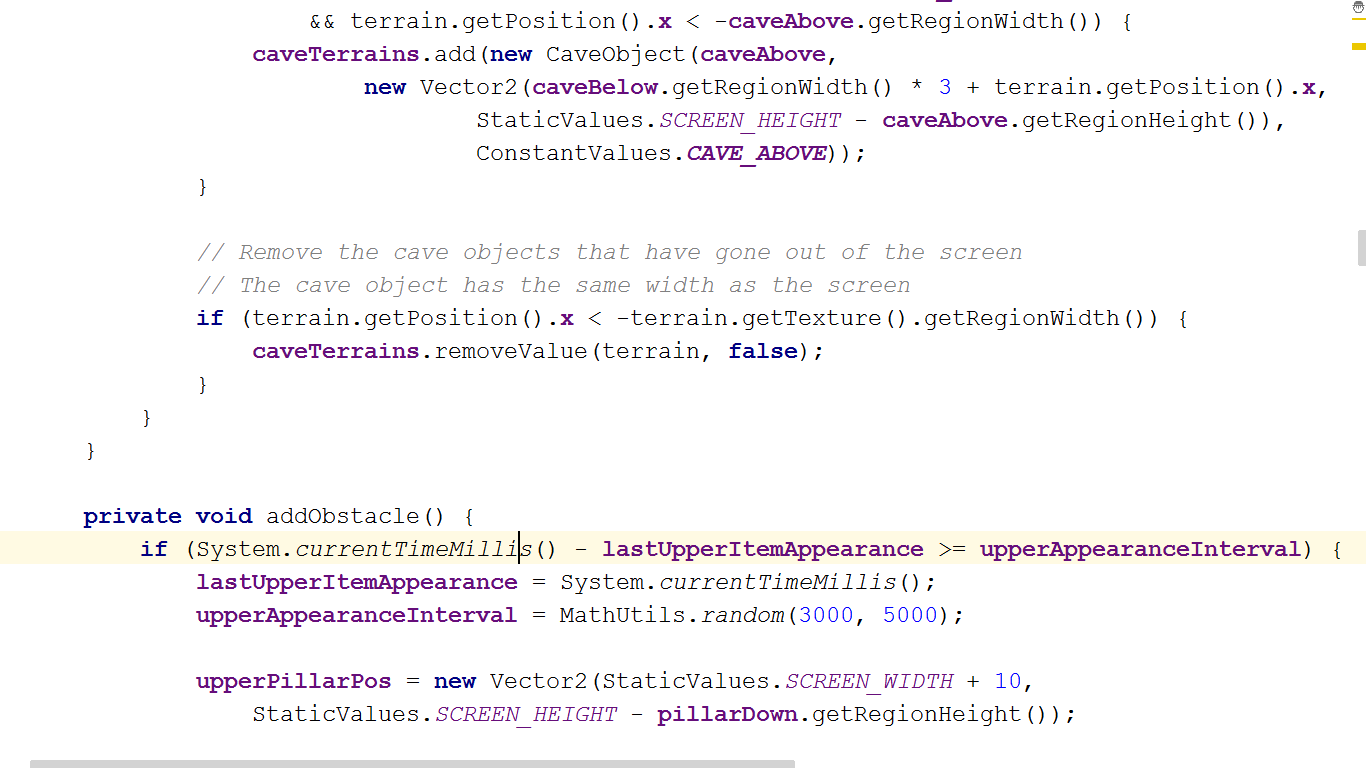




### Create setScrollingEffect() that moves all the cave terrain images to the left.

As the image goes out of the screen (the screen size is the same as the image size), it will be destroyed. Then, a new image is added to the caveTerrains Array.





### Create updateObjects() to update the positions of the terrains as time passes.







### Create renderObjects (SpriteBatch sprite) to render the image and updated positions of the terrains.





## Step 4:

Create a class called MainScreen.class that implements the interface Screen and defines game screen, which means the camera, as well as the view port for the game scene. Then, create a GameController object that creates, renders and updates the cave terrains on the screen

### Define the variable that holds the camera.

The OrthoCamera is a class that extends already-defined OrthographicCamera class in libgdx. It specifies the viewport of the camera and how it will be resized due to the different resolutions of different devices.

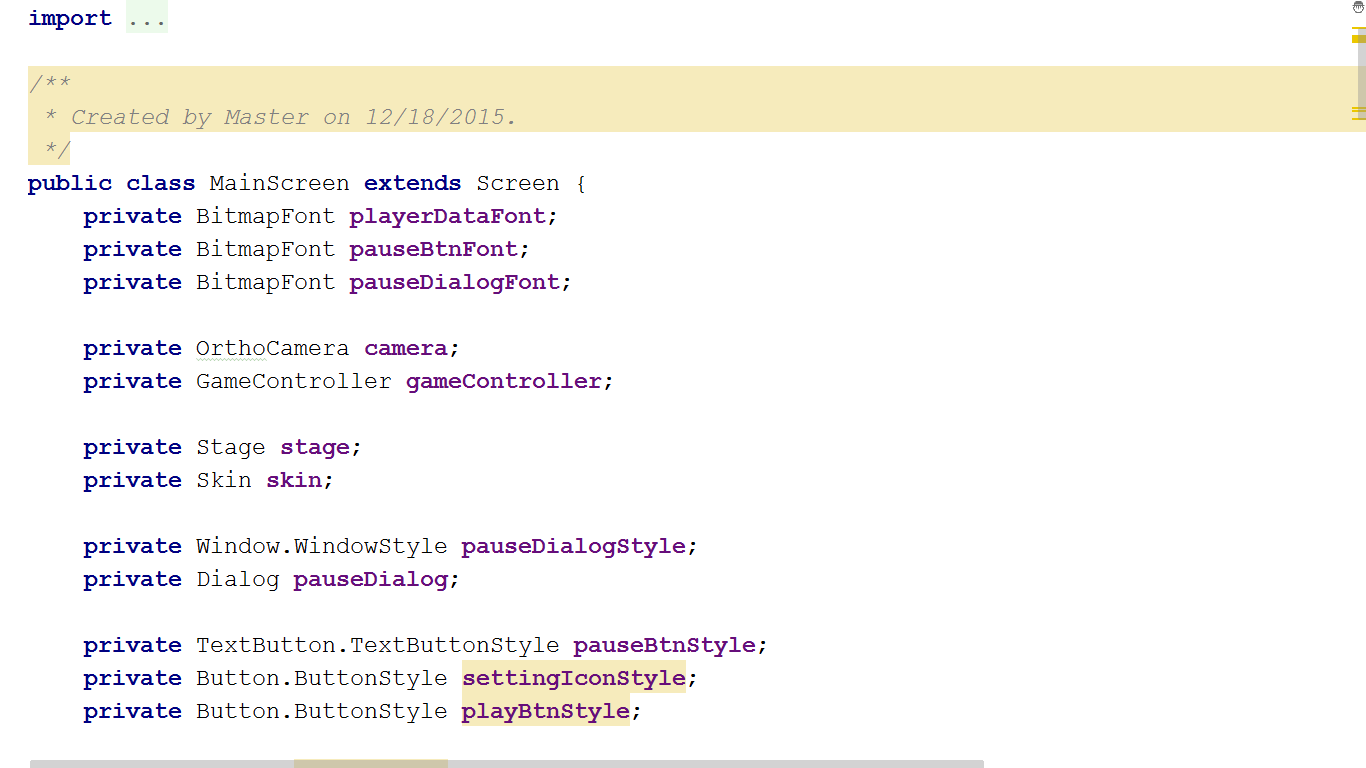


### Instantiate and resize the camera in the create() method



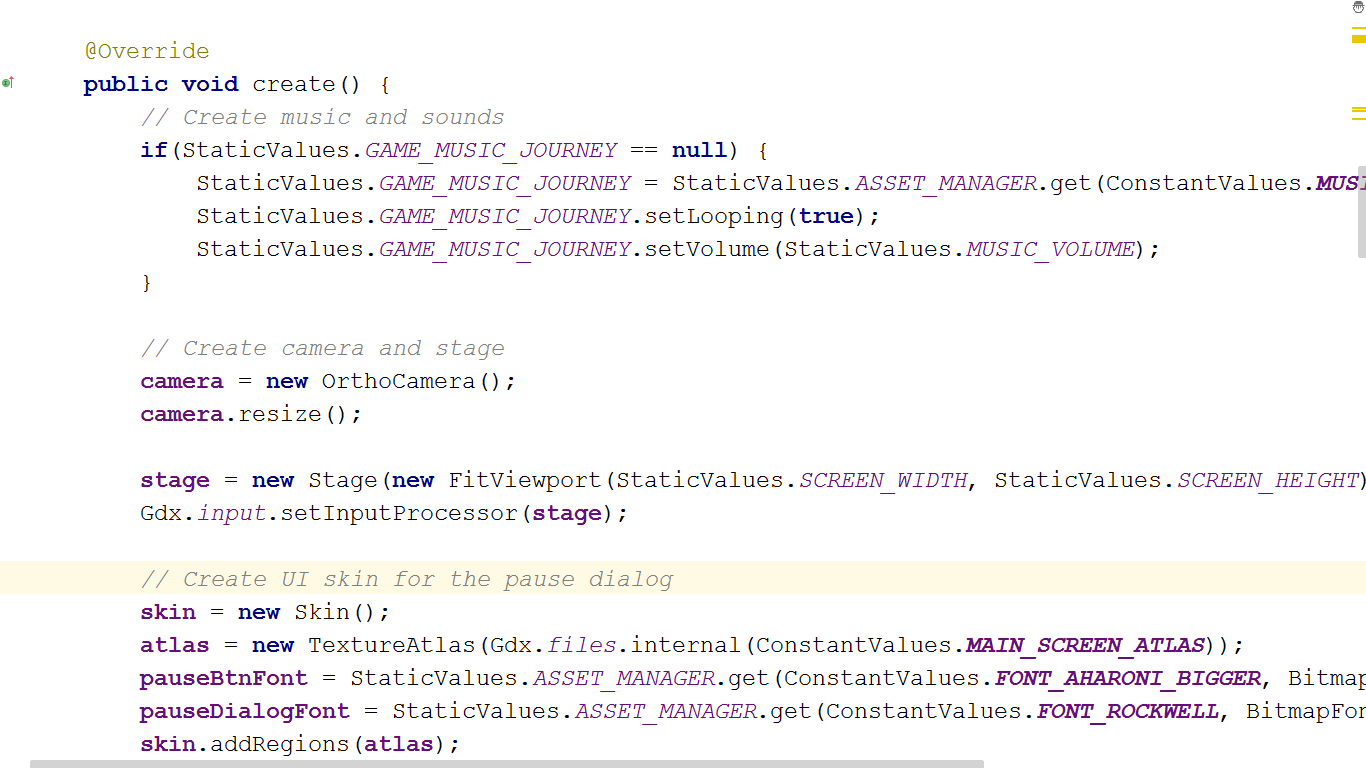
### Create a variable to hold a GameController object and initialize it in the create() method

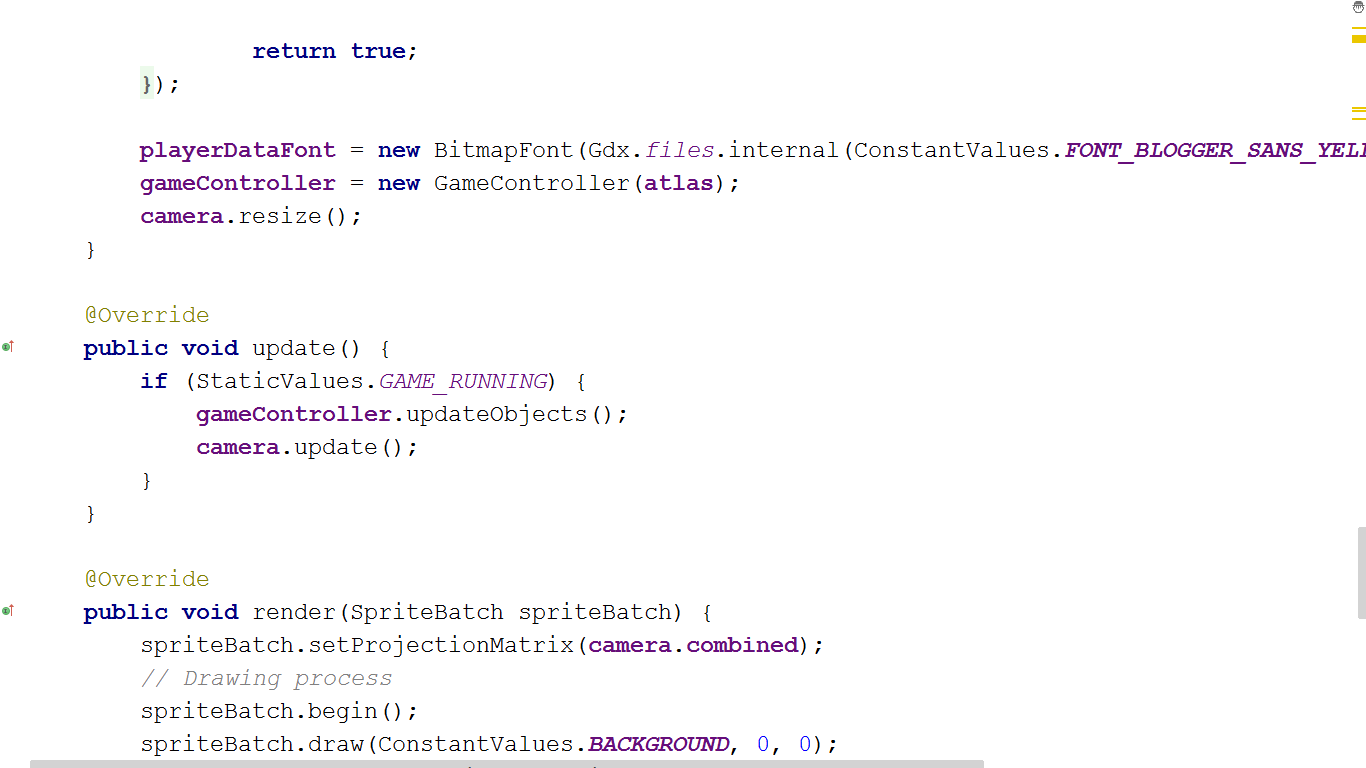
* Declare variables to hold GameController object, as well as a TextureAtlas for later passing to the GameController object’s contructor.

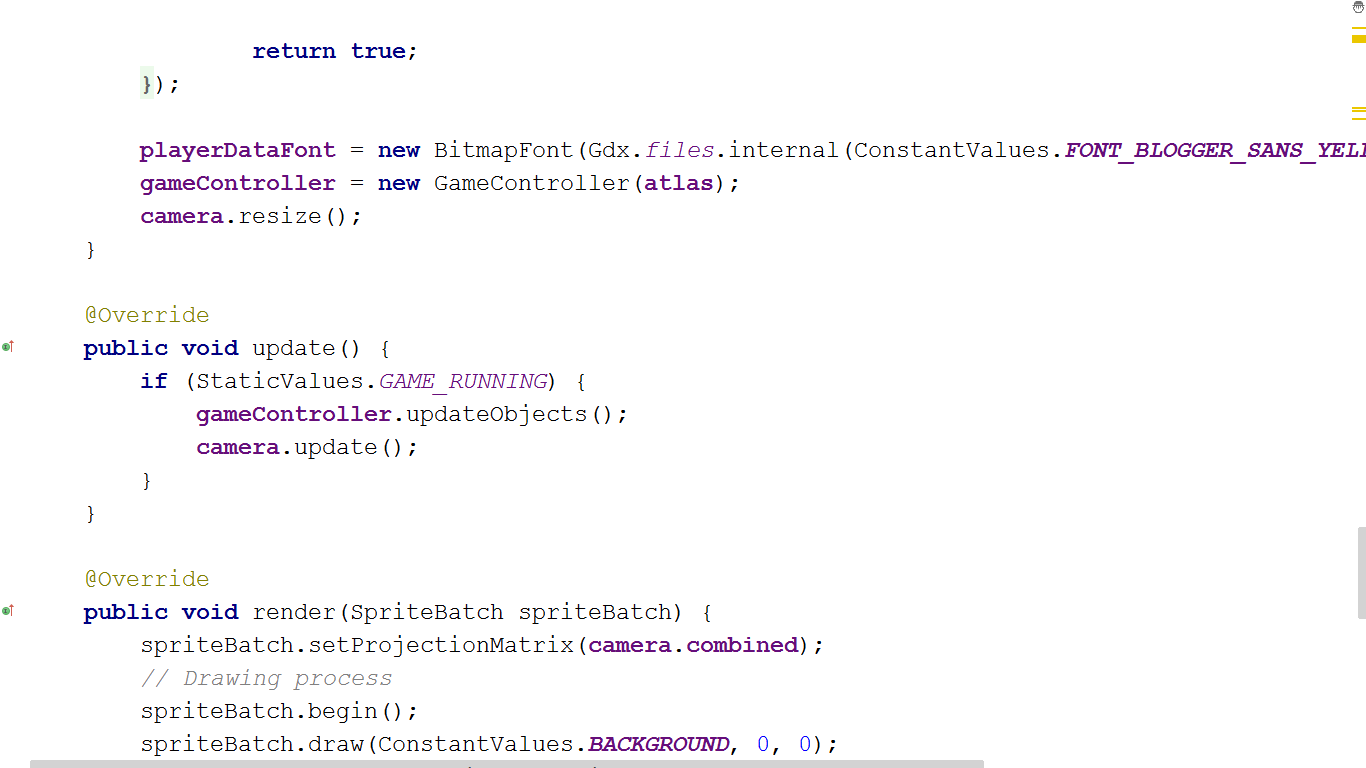




* Instantiate the GameController object







### Call the updateObjects() and renderObjects(SpriteBatch sprite) of the GameController object the update() and render() method of the MainScreen

## Step 5:

Create a GameExecution class that extends ApplicationAdpater. This is where the game actually initiates and runs.

### Declare a variable to hold the MainScreen object and instantiate it in the create() method.

### Create a SpriteBatch object called sprite to draw to scene.

### Call the update() and render(SpriteBatch sprite) method of the MainScreen object in the render() method of the GameExecution. Pass the SpriteBatch sprite to the MainScreen’s render method.

# Additional Part – How To Play The Game

## Play new game

The first screen shown to the user when the application starts is the Menu screen, which is illustrated in Figure 1. To play a new game, just simply click NEW GAME button.



Figure 3: The game menu - the first screen to be displayed when the game starts

Then, you will be navigated to the Main screen. To start the game, tap the big play button in the middle of the screen. In the beginning, there are 5 lives and 200 fuel points given to the user.



Figure 4: The opening of the Main screen

## Navigate the helicopter

To play the game, the user needs to control the helicopter avoid the obstacles mentioned above. The x-coordination of the copter is fixed and the object is always pulled down by the gravity. Therefore, the user has to tap the screen to move it up. Not taping the screen will make the helicopter go down.

## Pause the game

To temporarily stop the game, click the  button on the top-right of the main screen. Then, a dialog, shown in the picture below, will appear and the game will be paused. To resume the game from this main screen, just simply click Resume button in this dialog.



Figure 5: The main screen with pause dialog

## Go to the menu from the main screen

To temporarily exit the game and move to menu, click the  button on the top-right of the main screen. Then, click “GO TO MENU” button to save the current game and view the menu. Below is the picture of what the game menu looks like.



Figure 6: The game menu - the first screen to be displayed when the game starts

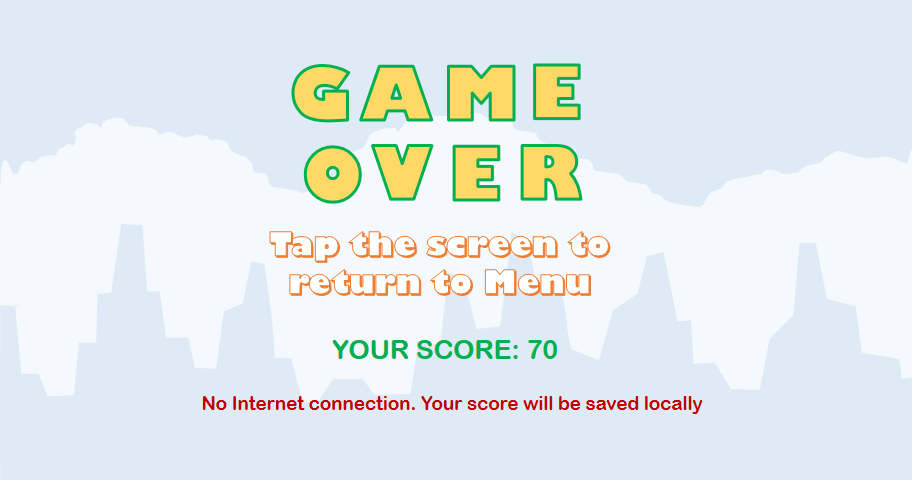
## Resume the game from the menu

To continue the game, click “RESUME GAME” button in the menu. You will be returned to the game state where you left. Click to big play button in the middle of the screen to start.

## Return to menu from Game Over Screen

After the game is over, the user is directed to the Game Over screen, where he/she could view his/her score again. To return to the Menu, just tap the screen.

\*\* Note: At the end of this screen, there will be a red notification about the Internet connection just in the case the device has the Internet problem. Otherwise, this red line never shows on the screen



## View scores

To view the scores, just click the SCORES button in the Menu. A dialog with the 10 highest scores will be shown to the user. To exit this dialog, click the OK button at the end of it. However, it seems like the cloud backend, Parse.com, we use to store the score has some problem. It returns a

## Adjust the settings

To enter the setting dialog, click the SETTINGS button in the menu.

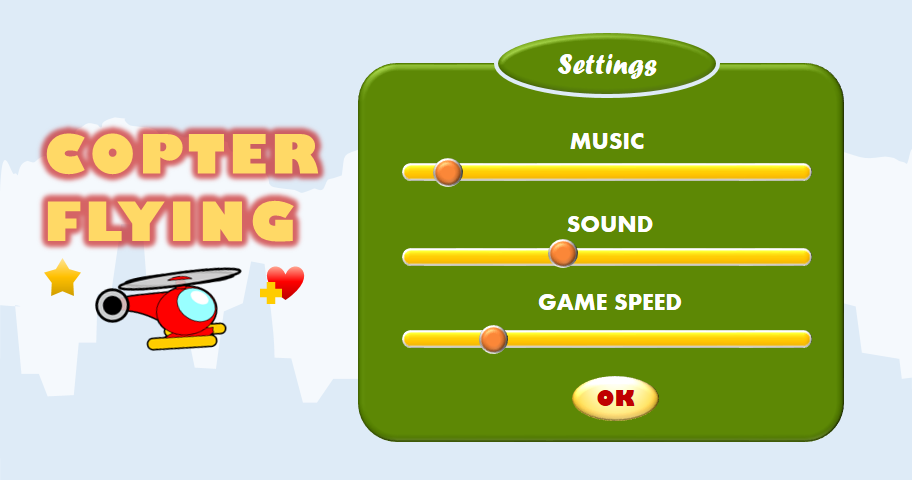


Figure 7: The setting dialog in the game

In this dialog, you can adjust the music volume, sound effect volume and the game speed, also known as the difficulty level of the game. Click OK to exit the dialog and return to the Menu.

## Exit the game

Click EXIT button in the Menu to quit the game.

# References

Libgdx Game Development Essentials – online pdf version: <https://vk.com/doc19838261_370886923?hash=6c97a4dd38e49824e3&dl=be3439cd0022728cfe>

Libgdx wiki on Github: <https://github.com/libgdx/libgdx/wiki>