Mobile Games Development – Report

Rehan Abdullah

P1320702

Game Name: JUMP

Game Implementation

Jethro Shell

Game Overview

Introduction

During the first assignment we were asked as a group to create a game and deliver the results via a presentation. We then had the chance to revise our strategies and take this concept forward to the next stage and code this idea into a reality. But due to a solid foundation required, this was a mammoth task and hence the reason to switch the content and start a new game with brand new designs.

For this assignment we are using Android Studio, an application that allows us to create games from scratch and then those games can be played on cross-platform devices such as: phones, tablets and even computers. The assignment is mostly based in Java programming with some XML. We are creating a single tap/ 2-D game that may have additional functionalities such as levels, menu screens, different activities from Splash, Main to Game. As part of this assignment, the task will be to describe the basics of the game, definition of the game and then the game story. We may also have to make different decisions in terms of design, how do we construct different aspects of the game features? Implementing different mechanics for the game will be a very good challenge. We may also reflect on what went well and how we implemented it and what would be changed if we were to re-do the game or do it differently.

The game is called JUMP, as all we do in the game is jump through a single tap gesture. The main aim of the game it to avoid the spanner that comes your way and you are in moving car. You can pick up points by avoiding the spanner. If the Spanner hits the end of the screen, you will lose some health from 10-25%, although this can be configured from anything lower too as we intend to make a game that can be played for 5 minutes or more. If that was not the case, the game would end within a few tabs due to its basic nature of single tap, jumping and avoiding obstacles through collision detection.

Definition of the game

The purpose of this game is to jump whenever possible and henceforth its name, jump. As you move through the game your task will be to jump and you are able to collect items such as cakes, coins that will push up your score. The game can be played for more than five minutes, and it is your choice to save the car from dying, as the car dies the game is over. You can now check your highscore from the highscore panel. You are able to turn on the sound and increase the game's brightness from the setting panel. As the game is 2D, we had to forward a basic understanding of how the gaming architecture is put together and create a simple game.

Game story

The game is about a car that wants to make it past the finish line but has been wrecked so badly through the last 10 years. Before the game actually starts we see the car in storyboard one as a brand-new car, and then the car has been driven by 20 rough riders and hence its current state. As the car moves through the city, it faces new challenges such as avoiding a huge spanner by having to jump over it.

As the car progresses through the game, its task is to avoid all weapons at any cost. As the game starts you see the car, the car's task is to jump and avoid all weapons at every cost. If the car is hit by a weapon it will lose all its health and have to restart the game. Once the car has gone through all the stages in the game, the car is still in a critical condition but everything that was collected during the game can be used to fix the car.

In the last scene we see that the car has been fixed and is the winner. The game story ends on a high note with the car's condition in top speed.

Design Decisions

To start of the whole project, we named the project/ game jump, and we can jump to the java folder that contains most of the project. For this assignment we used MVC to control different aspects of this game. All the activities (see Appendix, Figure 2) such as:

- GameAvtivity
- MainActivity
- SplashActivity

are all in the controller package, as the controller is “responsible for processing the user requests and building appropriate model and passes it to the view for rendering. (n/a, Spring - MVC Framework Tutorial, n/a)”

The next package is model that “represents a logical data structure in software productivity and a high-level class inter-linked with it. (Rouse, 2011)” Within the model package we were able to move in GameLoop class that “executes some instructions until we signal it to finish. (n/a, The Game Loop, 2010)” The other classes that were included in this package were highscore and settings. We had a highscore and settings class within the model package and we call this from the MainActivity class from the controller package.

The last package is called the view, that “is typically some form of user interface that lets humans interact with computer systems (Alexander, 2013).” In this package we are able to add classes such as: GameView, SurfaceView, and we are able to interrelate with cross-platform devices because of the classes that lie within this package.

We can then re-direct our attention to the res folder that has important components in it. The directory res/drawable contains all the images (see Appendix, Figure 1) that help the game to fully function because without the use of images we would not be able to see anything on the screen. The .xml

files in the directory, res/layout all hold the key to moving from viewing different screens

Discussion

We can now reflect on what has been achieved through a single tap 2D game that was created in Java, XML using Android Studio.

To create the game, we used Android Studio that supports Java Programming which is a cross platform language and instead of using many languages to create an app three times for three different devices such as: phones, tablets and computers, we used an Object Oriented language. As java is such a fantastic language, we may use functionality such as: inheritance, implements, encapsulation to link multiple classes together. The elements went very well due to the MVC layout, and each class was given its own package. We are now able to differentiate between what goes where and what happens in every package/ class.

The elements went well because everything was planned perfectly, a waterfall model (see Appendix) approach was used to create this game. In order to move on to the next step, one class had to be finished to start another class. The controller packages with the activities all had to be done before a loop and game view could be implemented. If a developer was to start by creating a loop and concurrent thread, there would be no way in which the game could be run through an emulator, as the .xml files are not visible. Henceforth it is mandatory to start from A and then finish with Z.

If I was to do anything differently with this game, I would consider more levels as a game with only one level may draw users away due to its repetitive nature. I would also consider adding sound to the game as this makes its users being able to listen as well as watch what is happening on the GameActivity.

Conclusion

In conclusion, we have used Android Studio to create a 2D single tap game from absolutely nothing. This is a fantastic achievement as we can turn code into a future profit by distributing our content on the Google play store. As this project is under the university's guidelines, it will stay as a project. The game itself was great to produce, we started off with an android studio project:

- Added blank activates
- Created different classes
- Added the MVC structure
- Assigned the classes to each package.

The game that was created looks good, but could have done with more implementation such as: levels, functionalities in the settings to work. Another attempt in the near future will put all these skills into practice for a better performance.

Bibliography

Alexander, A. (2013, August 3). *Model View Controller definitions and examples*. Retrieved April 21, 2016, from Alvin Alexander: http://alvinalexander.com/uml/model-view-controller-mvc-definitions-examples

n/a. (n/a). *Spring - MVC Framework Tutorial*. Retrieved April 20, 2016, from Tutorials Point: http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

n/a. (2010, August 29). *The Game Loop*. Retrieved April 19, 2016, from Against The Grain: http://obviam.net/index.php/the-android-game-loop/

Rouse, M. (2011, March). *model-view-controller (MVC)*. Retrieved April 25, 2016, from What Is: http://whatis.techtarget.com/definition/model-view-controller-MVC

Appendix

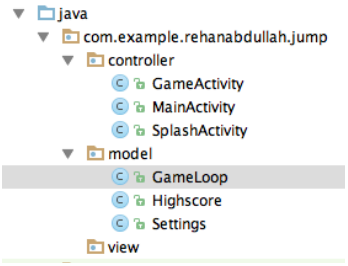


Figure 1(Project)

Figure 2 (Games Lab Diagram)

Figure 3 / http://s006.radikal.ru/i215/1010/ec/c29bf435b98b.jpg

Figure 4 (Car)
https://cms-assets.tutsplus.com/uploads/users/770/posts/24697/final_image/Car_art_complete.jpg

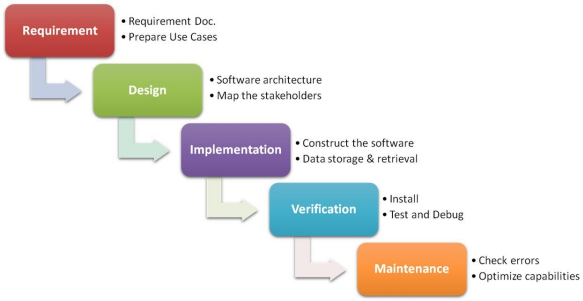


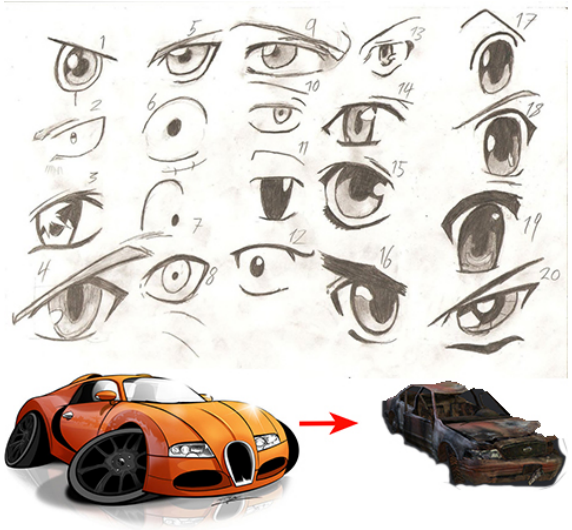
Figure 5 (Waterfall Method)
https://i.ytimg.com/vi/_ZKvvaZEFKE/maxresdefault.jpg

Mobile Games Development - Jump Game Storyboard

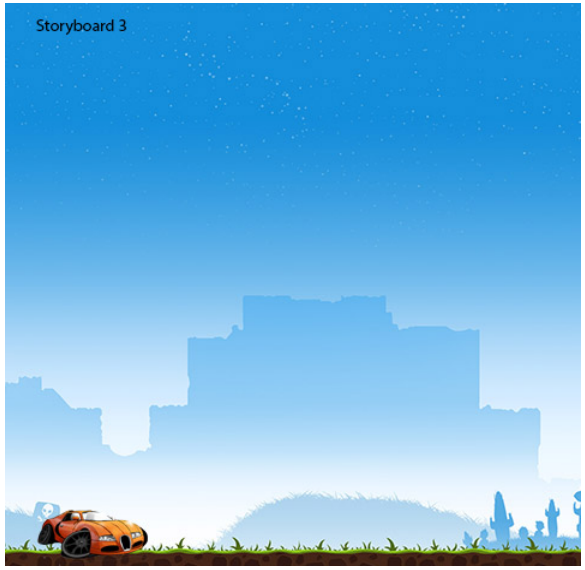
Storyboard 1

The Car Beforehand

Storyboard 1



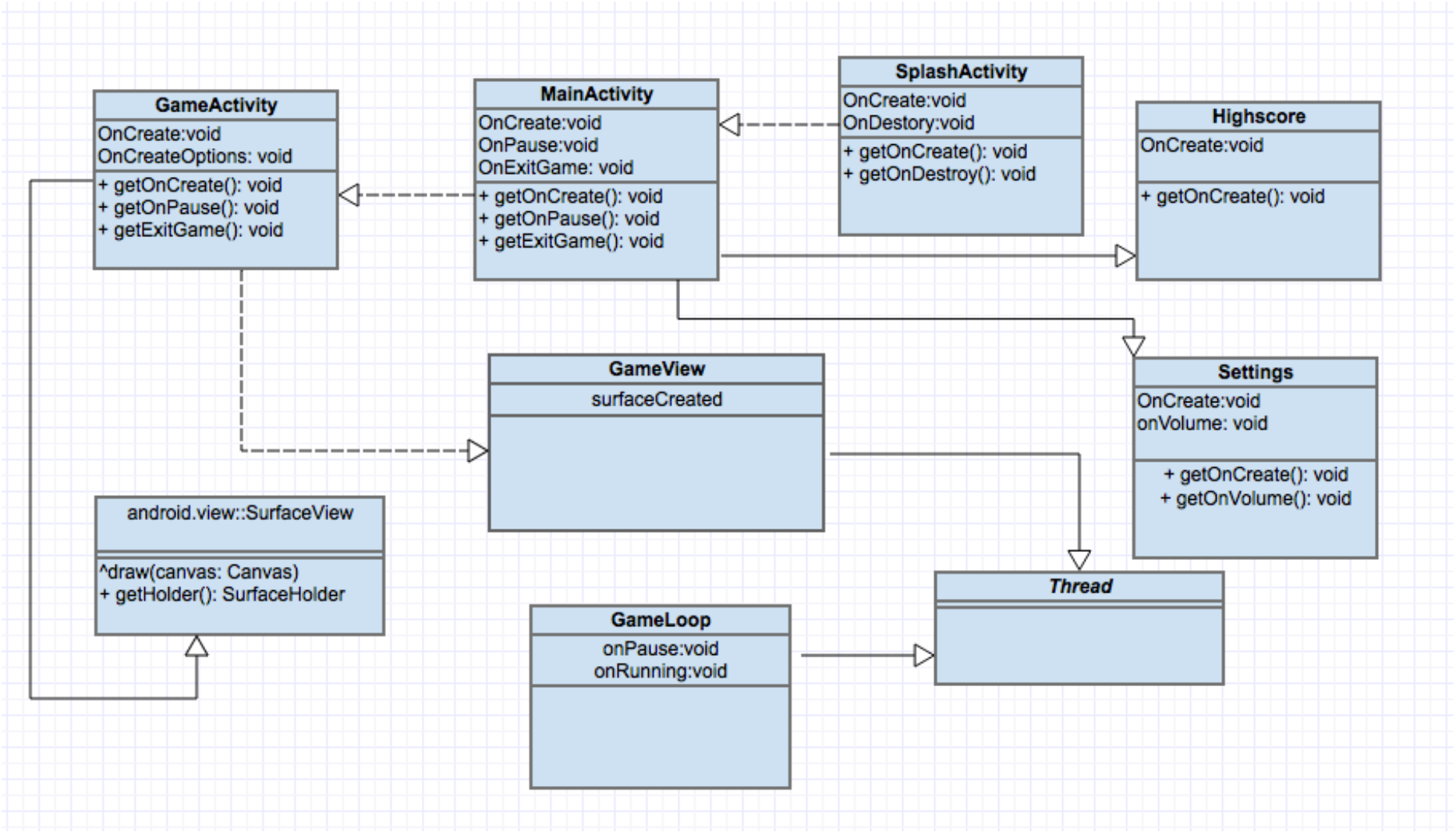
Storyboard 3



Storyboard 4

The aim is to jump with the single tap and avoid the weapons!

UML


GameActivity
OnCreate:void
OnCreateOptions: void
+ getOnCreate(): void
+ getOnPause(): void
+ getExitGame(): void
MainActivity
OnCreate:void
OnPause:void
OnExitGame: void
+ getOnCreate(): void
+ getOnPause(): void
+ getExitGame(): void
SplashActivity
OnCreate:void
OnDestory:void
+ getOnCreate(): void
+ getOnDestroy(): void
Highscore
OnCreate:void
+ getOnCreate(): void
GameView
surfaceCreated
Settings
OnCreate:void
onVolume: void
+ getOnCreate(): void
+ getOnVolume(): void
android.view::SurfaceView
^draw(canvas: Canvas)
+ getHolder(): SurfaceHolder
Thread
GameLoop
onPause:void
onRunning:void

Flow Diagram