**Skill Development for Mobile Game and Application**

Training for Mobile Application Developer

Report on Single Project

**Project Title: KryptoSuperman**

**Submitted by**

**Name:** Nafeeul Alam Walee

**Registration No:** L1A-C-11-0-22

**Course Registered:** Android

**Batch Id**: L1A-C-11-0

**District:** Khulna

**Introduction:**

The game consists of the idea of survival. The concept is similar as the game called “FlappyBird”. The concept goes as, player (superman) is heavily injured and his greatest weakness is a material called “Kryptonite” is surrounded everywhere. As the time goes he must fly over these kryptonite spikes and avoid kryptonite pillars to survive as long as possible.

A score will continuously increase as the survival score of the player. If he touches any of the obstacles, he will die instantly and the final score will be shown.

**Software Requirement:**

-Android Studio

-Libgdx

-Photoshop (optional)

-Camtasia 9(Audio/Video editor) (optional)

**Project Management:**

As developing a game, many assets are required in order to successfully create the animations as it were visualized. As the assets were taken from several sites, even from a large part of a sprite batch and photoshopped as it required.

Not only assets but also some modules of android, core, desktop for libgdx were required to download manually for technical difficulties.

**Development:**

Libgdx generator will generate an empty game project by including the features required to create the game , main as the Desktop test mode feature to run the project on desktop and Core of java class, were main codes are written for both desktop and android.

After creating, developer manually needs to setup the desktop test mode feature.

The first step is the test and run the project to check if it’s successfully generated.

The main part of the project is states; Games can switch between several states continuously. For this, there’s needs to be a State Manager that handles the switches.

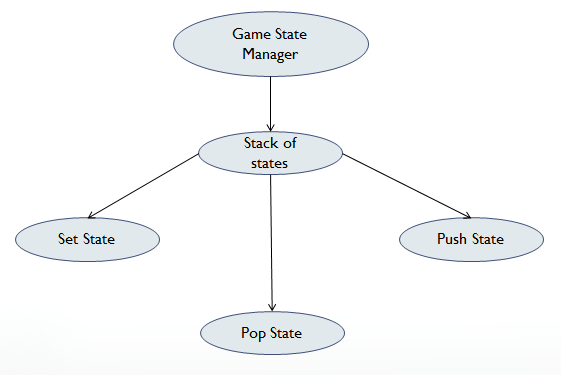


Figure: Game State Manager

The game state manager consists of an array of states. This means the manager implements a class of states that contains a stack of states. These states can be pushed and set to display on the screen, after when the job of done for a particular state it can be pop for disposal to avoid memory leaks.

The game has four states:

* Splash State
* Menu State
* Play State
* Game Over State

**Splash State:**

Similar to a splash screen, splash screen has been added into states instead of showing the splash screen before the game function starts is mainly because of simplicity.

When the game starts the first state is loads is the splash state, where the screens shows superman icon and a loading text with an animation of increment of dots. The function of the dots has implemented in a way where dot increases with respect to delta time. By an if-branch and a simple condition with respect to delta time, each dot are rendered in specific time and drawn on the screen. After a predefined time splash states will push Menu state and splash state will be popped and disposed.

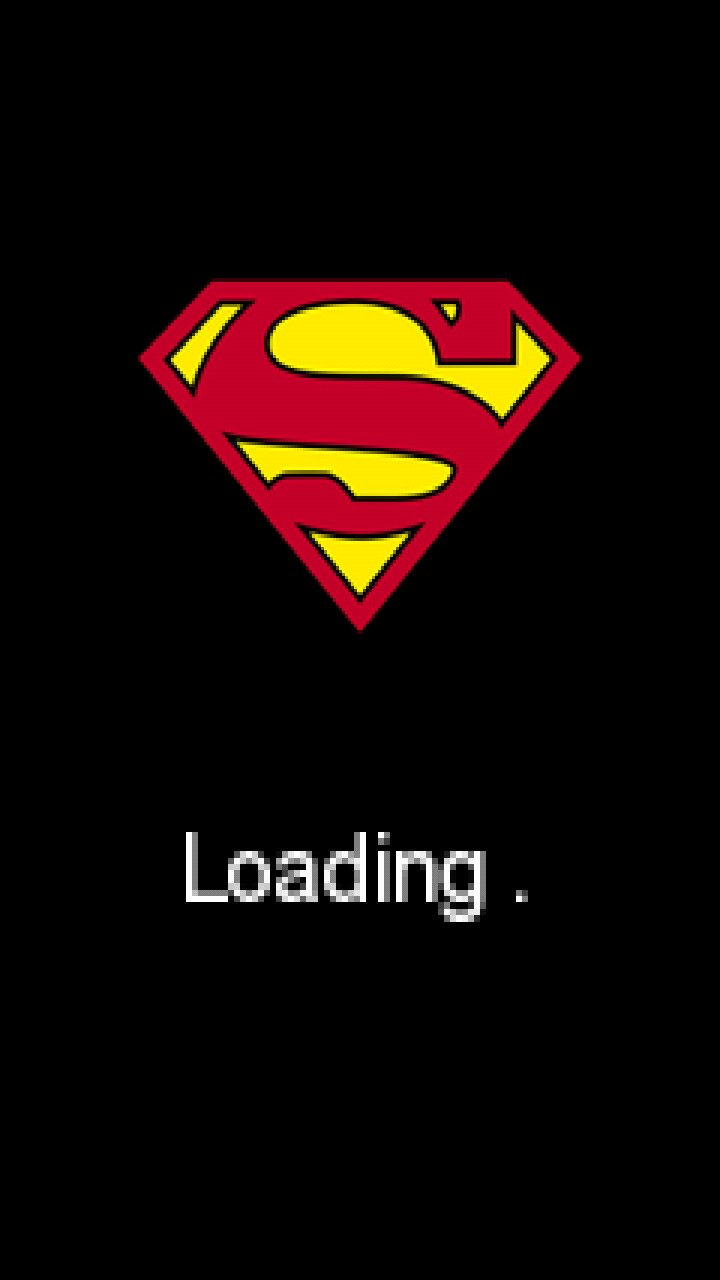
 

Figure: Splash Animation

**Menu State:**

Menu state is made by four textures- background, superman logo, superman’s sprite and “tap to start” text.

The main function of this state is to take player to play state. Menu state looks for handle Input function where the function pushes state to play state when the screen is touched. Similar to splash state pushing method, menu state will be disposed and play state will be pushed and set to screen for display.

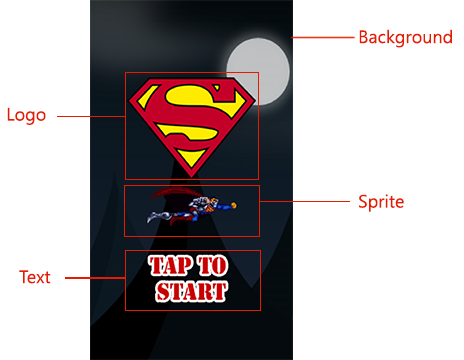


Figure:Main Menu

**Play State:**

This is the main state of the game. It consists of player flying animation, obstacles and obstacle control, score, camera control etc.

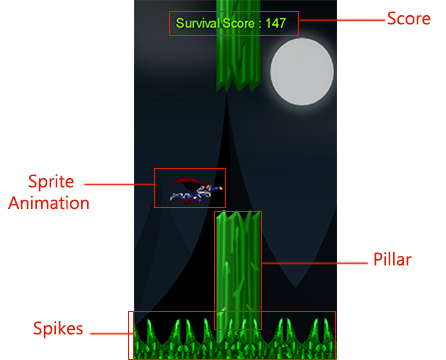


Figure: In-Game Screen

The main three parts here are-

* Player Animation
* Pillars and Spikes
* Survival Time

**Player Animation:**

The animation contains several sprites into one imagine. The sprites can be separated by measuring the width of the image and dividing them by the number of sprites it contains; here the image given below contains 6 superman sprites. Libgdx can calculate the width of the image and that width can be divided by 6 to get a singular sprite-



Figure: Before division



Figure: Division Measurement



Figure: After division

Now program can run loop between them to keep showing all the separated sprites simultaneously.

Secondly, player will fall down for a constant gravity pulling toward -y-axis while player also moving toward + x-axis with another constant velocity. A jump function will be activated toward +y-axis when screen is touched.

**Pillars and Spikes:**

The pillars are only two images, for TOP side and BOTTOM side, while spikes only shows up in BOTTOM side representing the ground. The spikes are bunch of images sticks together to make an image large enough to cover the whole width of the game screen. The spikes works as, when the camera reaches the end of the large image of spikes on the ground, it program regenerates the spike ground right next to when it ended. As a result, user will see an endless loop of spike ground.

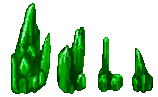


Figure: Spikes

While pillars follows a different pattern, maximum of 4 pillars can be seen on screen, when two pillars(bot and top) shows on screen, program generates two more ahead of the screen, when previous two pillars goes off of the screen, program just dispose of them. Thereby, there will always be four pillars shown on screen constantly.

Another technique has been used for user to take a rest by making the appearance of the pillar. By adding a delay after showing 4 pillars a moment of no pillar screen will appear.

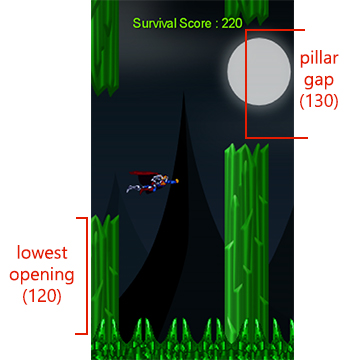


Figure: Pillars properties

Here, pillars are huge object having large height, user can only see the tip and some parts of the pillar, and other parts are out of the screen.

Lowest Opening is for the BOTTOM pillar set to value 120, where the pillar needs to show at least of value 120 no less than that and pillar gap considers the gap between TOP and BOTTOM pillars set to 130. No matter that lowest opening the programs sets, the pillar gap is always at 130 so the TOP side pillar will always show at 130 values above BOTTOM pillar.

**Survival Score:**

The survival score follows the delta time, is in not optimal to show score this way, but for the sake of simplicity the score increase whenever there’s a difference between (delta time) and (delta time +1). As a result there will always be difference and score will always increase.

Always, it is better to set up a stage and table to show score board. Here score board is a text that changes with respect to delta time. And the text needs to follow the camera in order to always keep on showing above the screen. As a result, when camera is coming the text will move too which can be noticeable. But by setting up a stage, score will be always still and do similar work as before.

**Advantage**: Simple and easy to implement.

**Disadvantage:** Scoreboard is not still.

**Advantage of stage and table:** Scoreboard stays still and can easily add new features in scoreboard.  
**Disadvantage of stage and table:** Takes time to implement.

**Game Over State:**

The main point of game over screen is to show the final survival score and continue back to game.

Here, a method waits for touch on screen to switch states to play state and dispose of game over state.

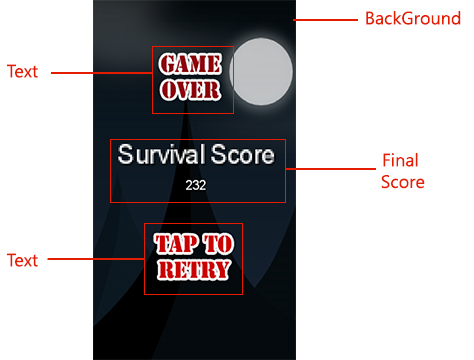


Figure: Final Score and Continue Screen

**Final Architecture of Game:**

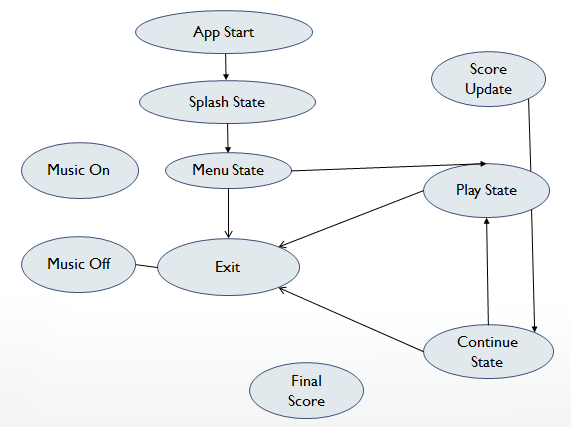


Figure: Flow Chart of the game