Mobile Applications Development Coursework Report

Kieran Burns
40272382@live.napier.ac.uk
Edinburgh Napier University - Mobile Applications Development (SET08114)

1 Introduction

This report has the purpose of explaining and reflecting upon the creation of my recent mobile game named "Better Act Fast!". The project spanned the space of a few weeks however most of the actual development (programming and sprite/-graphical work) was done in around one week. Throughout the project my hope was to improve my existing skills as well as learn new skills within Android Studio as well as in Java in general.

When issued the challenge of creating a mobile application under the scope of creating anything of my choosing I initially had a hard time deciding what to work on. The idea of making a game definitely was the idea that appealed to me the most and after deciding that a game was what I wanted to work on I then had to come up with an idea of what sort of a game I was to make. A few different ideas came to mind in the initial ideas phase. Of the two main ideas I had, the first was a short narrative puzzle game where the player would solve basic puzzles to escape a dungeon they were trapped in while working out why they were trapped in the first place and the second a party game heavily inspired by video games such as "WarioWare" and old physical games such as "Bop It!" where the player would complete a series of timed mini-games to achieve as high a score as possible, which could then be compared to the score of friends and family to give a competitive and replayable experience.

After spending some time trying to figure out how both of my project ideas would be developed on a broad level I finally decided on working on the party style timed mini-games.

2 App Design

Once I had decided on what I intended to work on as my project I moved on to working on the design of the app. The design stage broke down into a few sections including figuring out specifics of the project from a gameplay perspective and what sort of hardware and software that would require, working out the general flow of the app (how it would function on a technical level for each of the mini-games and how they would join together) and designing basic interfaces and how they would be used and navigated.

2.1 Project Specifics

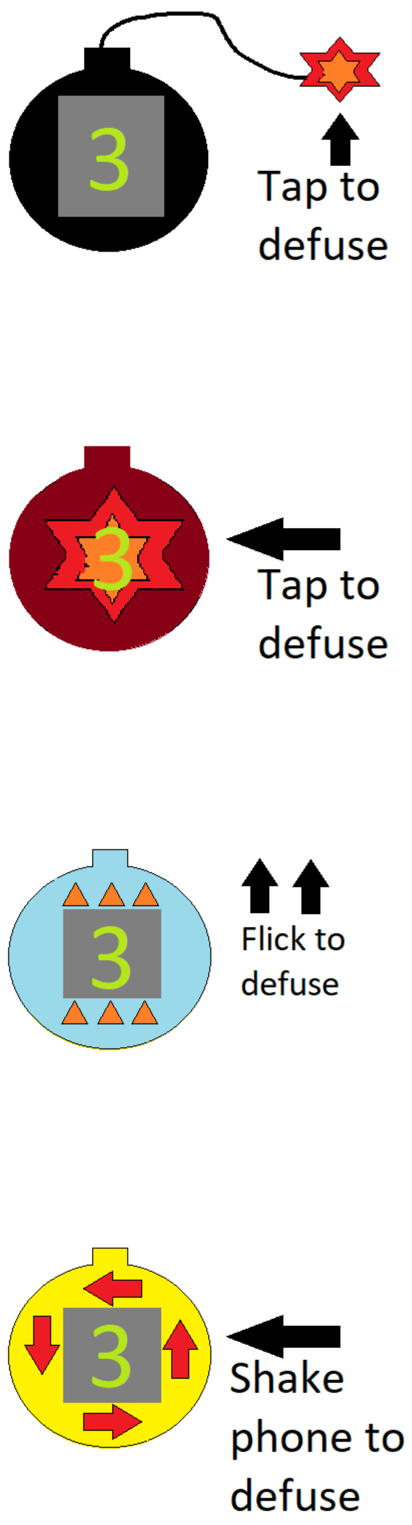
When it came down to planning the specifics of the project I came up with four general mini-game ideas using the hardware used within most modern Android devices. The first two ideas were tap based games, one requiring precise taps on certain areas of the screen and the other requiring faster taps in a much broader area of the screen, the third idea was one that used the devices gyroscope sensors to detect the device being shaken and the last of the ideas was to have the user drag and drop something from one area of the screen to another. I also had the idea of implementing a game which used microphone on a users device to take in the player shouting to complete an objective, however from a software perspective this would be much more complicated and would also limit the viability of the application being used in a more public setting such as on public transport or out on a busy street, hence that idea was sidelined for the initial development.

Originally I had planned a silly setting for each of the mini-games where respectively the user would be tasked to crush small spiders for the precision tap game, rapidly punch a punching bag with a giant boxing glove for the speed tap game, shake a bottle of chocolate milk for the shake game and feed a shark a cookie for the drag and drop game.

2.2 Consistent Visual Style

After beginning to draw up some basic concepts for each of the four mini-games I began to notice that the visual style was rather inconsistent which would make for a somewhat jarring experience for the user so I decided to redesign the visual aspects of the game to match a similar theme. I eventually conceived that the objectives of the game could be cartoon style bombs, akin to something that may be used for a gag in cartoons like "Tom and Jerry". Going off of this idea I planned the design for four bombs, a normal black bomb with a wick and fuse that must be tapped to defuse the bomb, a red bomb that must be tapped multiple times to break it to defuse it, a blue bomb that must be flicked away by the user to defuse and a yellow bomb that need the phone to be shaken to defuse. Each of the four bombs had the potential to have a countdown timer on them, allowing for the screen to both centre each bombs time limit and to reduce what would be needed within the user interface of the game screen. Initial drawings of each of the four planned bombs can be seen within *figure 1*. These designs would later be refined before moving on to the implementation phase to make them more even in shape and size as well as have edits to the colours and boxes to keep the designs consistent. The extra patterns were added as basic colourblind support so that each bomb would remain distinguishable.

figure 1.



2.3 General App Flow

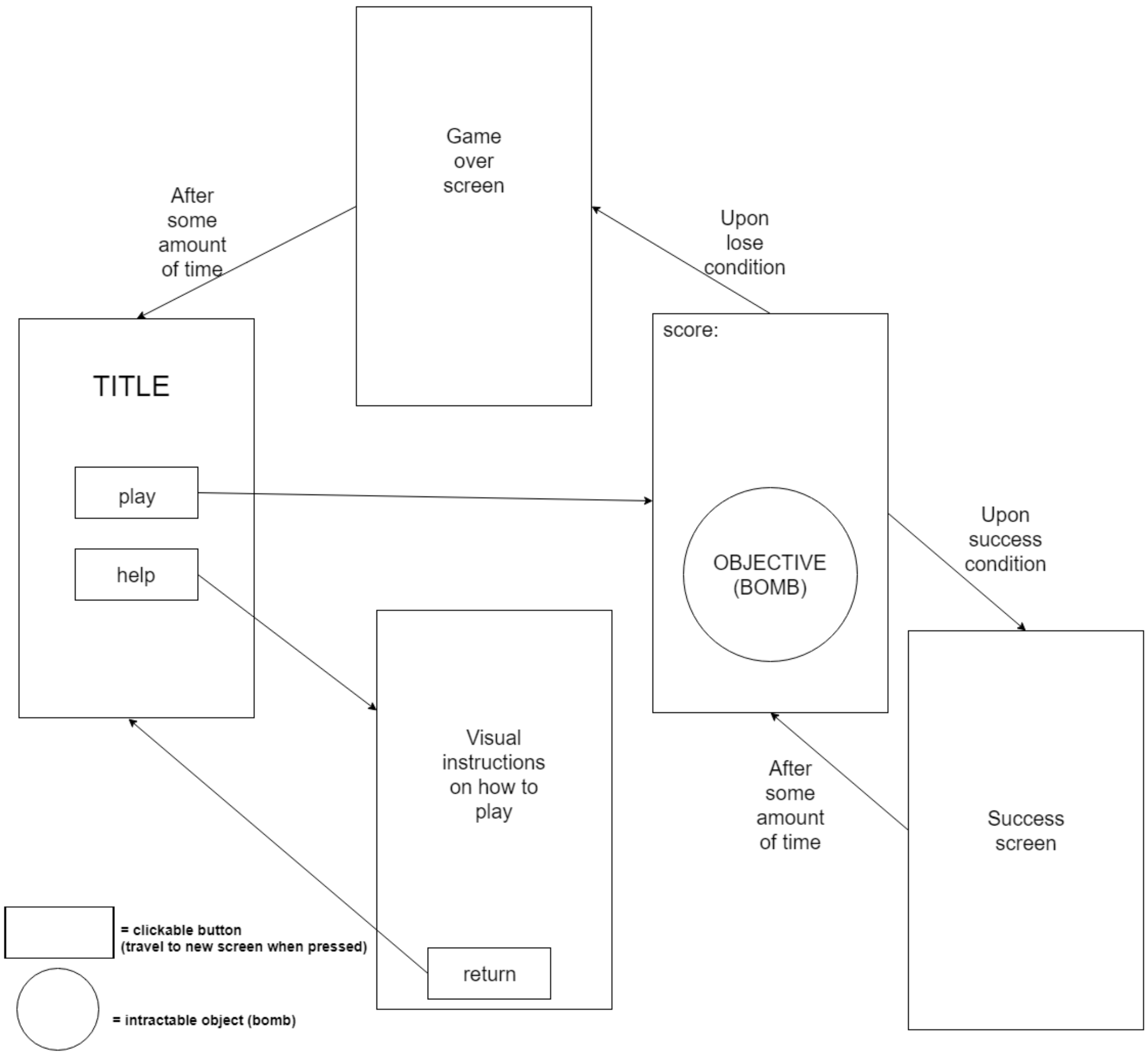
The next segment of the design that required planning was the functionality of the game. From a hardware perspective I had already figured out that each of the 4 mini-games would function as I had intended however the way they each connected was still to be decided. To begin work on this I decided to make each bomb initially count down from 4 to allow for a learning curve for players when they start playing the game so that new players can learn the controls and returning players could warm up when for a new high score attempt. After a bomb had its success condition cleared I planned on having some form of success message or screen appear before refreshing the bomb on the screen. I had also come to the decision to have a bomb be randomly selected out of the 4 options at the beginning of the game and after one was cleared to make the game interesting and allow players to hone their reaction speed. I also made the decision to have the starting timers on each bomb become lower on new bombs the longer the player managed to last for. If a player did not meet the success condition of the bomb that they were tasked to defuse by the time the timer on the bomb hit zero then the fail condition would be met and the game would end. The player would earn one point per bomb that the defuse until they meet the fail condition, their collective points would then be compared to a current high score and if the player earned a higher score the high score would then be set to the players score.

2.4 Interface Design

The interface design of the app was based around having three main activities/views being the home/menu screen, a help screen used to display a small guide on how to play the game and the actual game. The game screen would also have connections to two time-based screens showing that the player has met the success or fail condition of the game. The success screen would cover over the next bomb being generated for the game to continue, whilst the fail screen would overlap the game screen while returning to the main menu. A wireframe diagram for this design and flow can be seen within *figure 2*.

Each of the vertical rectangles depicted within *figure 2* represent a different display on the app, with the arrows between each being used to show how each interacts with each other, with each clickable button being shown as a horizontal rectangle. The circle used the screen that comes from pressing play is used to show where the bomb would be on the game screen.

figure 2.


Game over screen
After some amount of time
Upon lose condition
score:
TITLE
play
help
OBJECTIVE (BOMB)
Upon success condition
Visual instructions on how to play
After some amount of time
Success screen
= clickable button (travel to new screen when pressed)
return
= intractable object (bomb)

3 Implementation

With a design prepared I went on to begin development. This began with improved drawn images for each of the bombs as well as creating images for the backgrounds and buttons. Some of the designs were later tweaked and refined to make the game look as good as I could using my low-end drawing skills on my drawing tool of choice (Microsoft Paint).

3.1 Main Activity

After creating the required graphics the software end of the project began. Using Android Studio I began by creating a 'Main Activity' to act as my Home/Menu screen which would stay running in either the foreground or background of the app as long as the app was running. This activity handles the navigation to the Game Screen and the Help Screen as well as hold the global variables used within the program such as the loaded in high score. It contains methods for calling both the HelpButton activity and Play activity which is used to handle the Game Screen. When the Play activity is called the Main activity awaits a response which is set upon the destruction of (completion of) the Play activity which allows the Main activity to update its high score value based on the returned high score from the most recent run of the game.

3.2 HelpButton Activity

The help activity is a rather simple one, simply loading a different screen over the top of the home screen, containing a single button which closes the overlaid screen, returning the user back to the home screen.

3.3 Play Activity

The Play activity only contains onCreate and onDestroy override methods which are called at their respective times. The onCreate method calls the corresponding view class named GameWindow since the Play activity does not have a standard static view like the home and help screens. The GameWindow class is where most of the processing for the game is contained. The onDestroy method writes the updated high score (calculated after the GameWindow reaches a game over condition) to the file used to retain persistence (by the use of SharedPreferences), then returns a response telling Main Activity that the game was run and has ended.

3.4 GameWindow

As stated previously the GameWindow is where most the game and its required processing goes on. When first called it sets up the required variables and image paths for the game, checks to see if the required sensors for the yellow bomb are in place and if not disables the yellow bomb from being randomly selected by the game (to allow the game to be as accessible as possible to as many android devices as possible) then sets those sensors up to be used, collects information on the size of the devices screen to be used to dynamically size each of the on screen objects so the game can run the same on different devices, calls the creation of the first bomb from the separate Bombs Class, sets sizes of recurring images and the score text to what they are going to be for the game (based off of the collected screen size) so processing time is not used doing so later then calls the handler to start the game (first call of onDraw).

The images the game uses that are moved around the screen and placed in different positions (Spark and Fuse) are sized based off of the screen size in classes which are called when they are going to be needed.

The onDraw method runs the majority of the games conditions. When called it places the set background to the back of the screen, then places the current score over the top of it in the top left of the screen. The method then places the current bomb (initially set by the creation method) on the screen in the position and at the size it has been set to (also set in the creation method). If a game over is yet to be set and the success condition has yet to be met then the method will check what number of bomb the current bomb is (bomb number is a part of the Bombs class) and will act accordingly. If the bomb number is 0 or 1 then the program will act upon the bomb being a standard black bomb and will render the required spark on the fuse to be stubbed within the time limit and will then await the click on the spark to complete the success condition. If the bomb number is 2 then the program will act upon the bomb being a red bomb and will await the bomb being tapped 5 times to defuse it. If the bomb number is 3 then the program will act upon the bomb being a blue bomb and will render the target for the user to swipe the bomb to and will await them doing so. Lastly, if the bomb number is 4 then the program will act upon the bomb being a yellow bomb and will await the device being shaken at a sufficient velocity and over a sufficient distance to meet the success condition. If the success condition is met then the program will call for the success screen (displaying the word "Defused") to be shown for 8 ticks (800 milliseconds) then will generate a new bomb, lowing the users available time to defuse the new bomb if they have reached score increments of 5, 15, 30 and 50 successful defuses. If the bomb timer has reached 0 and the success condition has not been met then the game over screen will be displayed for 2 seconds before checking if the user has reached a new high score and setting the current score to become the high score and then calling finish on the Play class, returning the player to the main menu. After each check is made once, the game is progressed by 1 tick which is equal to a tenth of a second. Each number on the countdown is equal to 8 ticks rather than 1 second as after play testing the game using 1 second on the the count down timer it felt as if it was running too slow.

There are also override methods used within the class to take in touch and shake inputs from users during the corresponding bombs.

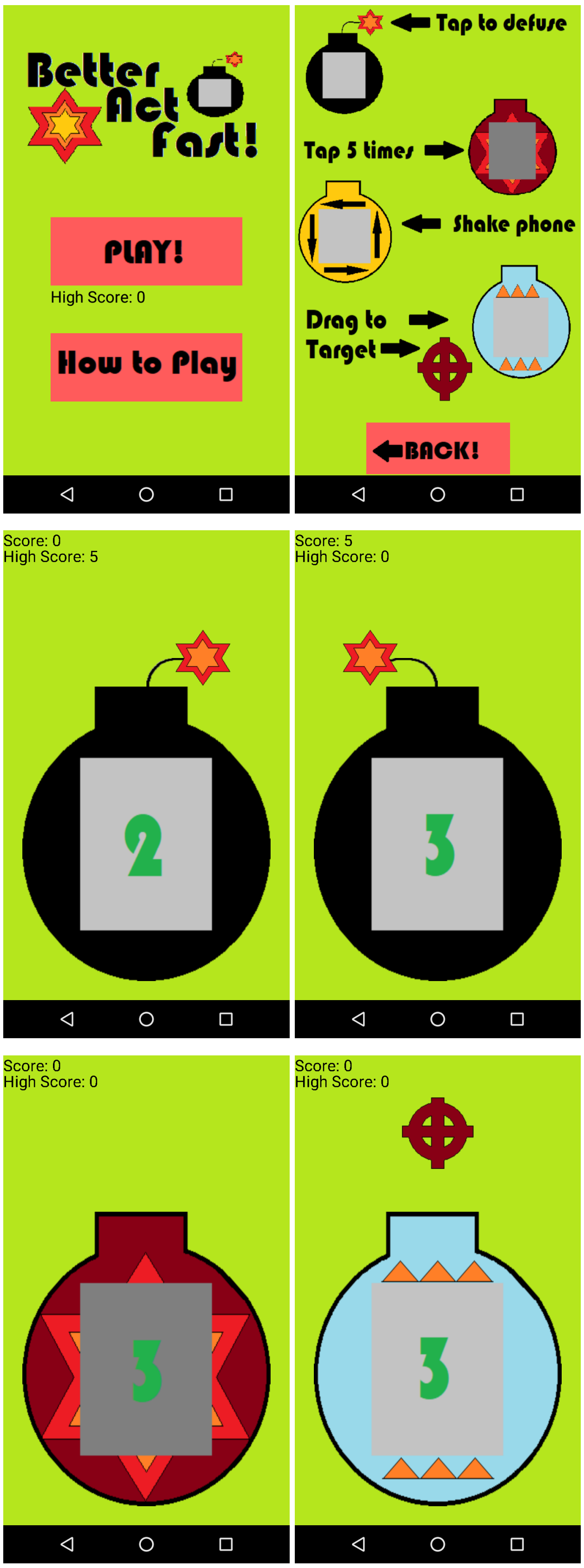
3.5 Bombs Class

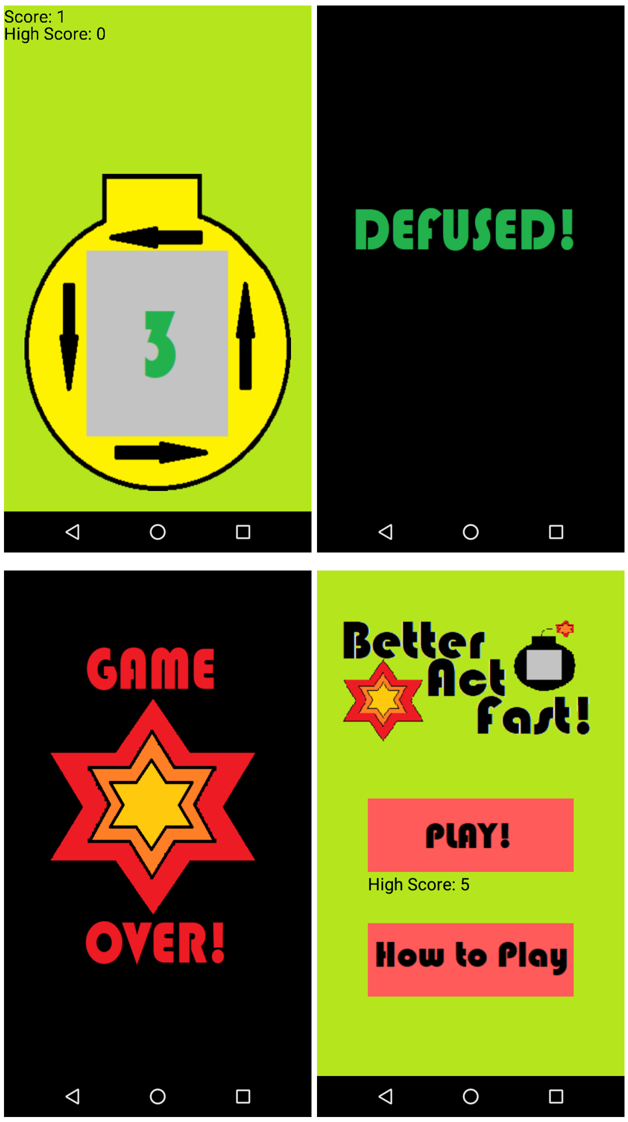
The Bombs class holds most of the information required when generating a new bomb for the game. When it is called (as "new Bombs") it generates a random number between zero and the allowed bombs by the GameWindow, this starts out as the numbers zero and four however if the yellow bomb is disabled then it will only go between zero and three. Once a bomb number has been selected the corresponding bomb has its frames (different images) for the countdown loaded into a bitmap array to be accessed by the GameWindow.

3.6 The finished application

Figure 3 shows (in order) the final versions of the home screen before the game is played,the help screen, the right and left versions of the black bomb running in the game, the red bomb running in the game, the blue bomb running in the game, the yellow bomb running in the game, the success screen, the fail screen and the home screen after the game is played.

figure 3.





4 Critical Evaluation

Compared to my original concept of sharks, chocolate milk and punching bags the finished product is almost unrecognizable however if broken down to a technical level each of the finished mini-games functions almost exactly the same as one of the planned mini-games so from that perspective the project was no failure. To some extent the finished product ended up in a much better place than the original idea as the visual style and design ended up being much more consistent and sensible.

Comparing the finished product to some of its inspirations can be a little difficult in the direction the game has gone however one can certainly assume some potential inspirations for the gameplay aspects. The "Bop It!" inspiration can still be seen from a general gameplay point of view in regards to the player having to repeat short reaction time tasks within a limited time frame, which also somewhat stands for the "WarioWare" comparison. The games name (despite some speculation possible about its origin) is actually a form of homage to another game I made as my first larger solo project ever named "Don't Go Splat!" which used an eye-catching emphasized naming convention of 'medium word - small word - medium word' and is also where the titles in-game style spans from. See *figure 4* for the "Don't Go Splat!" in-game title.

figure 4.

I had a small group of family and friends test out the finished product to see how different age groups could perform taking the best score of each different players first three tries and averaging them over the members of the age range. Teenagers and Younger adults (around the 18 to late 20's range) performed the best having scores averaging between the 40 and 50 points mark, younger children (below 12 years old) had score averaging around the 15 points mark (however seemed to do slightly better after having a longer time to learn the way the game works) and lastly older adults (between the 50 and 65 year-old age range) seemed to have the most difficulty with the speed increases of the game which left the average score of just over 10 points (similar to the younger children this score did somewhat improve after a longer time playing the game). The feedback I received from this group was generally positive however this is likely skewed by the relation they have to me. The two main complaints I received from the users were the teens and younger adults complaining about how there was not enough variation in the mini-games for longer runs (over around 30 points) and the older adults complaining about how the bombs 'exploded too fast so they didn't even get a chance to react some times'.

Improvements to remedy the complaints I received would be to implement more bombs to appeal to the younger audience and add different difficulties to give an easier option to those who find the timers a little too fast. Other improvements I personally would liked to have made alongside the more different bombs improvement especially would be to improve the graphics of the game to make it look a bit more appealing and eye-catching for players and to add sound effects and perhaps background music to the game, neither of which were done during development due to my lack of skills from an art perspective and my lack of knowledge in regards to writing background music and creating sound effects.

5 Personal Evaluation

From a learning standpoint this project was an excellent experience. I learned many of the important features of Android Studio and learned about a few different errors I had never personally experienced before and figured out how to fix them.

The first of the errors that were new to me was a somewhat serious memory leak which occurred when I was trying to figure out a way to save the high score and involved me calling open input and output streams (which I was originally trying to use for the persistent data of the high score) where I was trying to call the output stream from an external method and was generally having a tough time getting those streams to function. I fixed this error by using SharedPreferances to keep the persistent data in a more simple fashion and began using the memory write code within the area it was needed in directly rather than calling it from a different method. Another somewhat large issue I had was during the creation of the blue bombs mini-game where the bomb's hit-box originally overlapped that of the target as the bomb image had an extra third of size that was empty/invisible pixels that were above the bomb so the user could win the stage by simply tapping the target area as the press still technically went down on the the bomb and came up on the target. This was fixed by redrawing the sprite for the bomb and each of the other bombs also received and updated sprite at this point too. The last somewhat major issue I had was to do with the scale of the game screen. Originally I used a range of hard-coded numbers to set the sizes and locations of each object on the screen based off of how it looked on the virtual device I was using to test the game during creation. This of course ran into some problems when I began testing how the app functioned on larger and smaller devices where the game did not scale to the different shaped and sized screens. This was fixed by making each of the objects on screen dynamically scale based off of the users screen size.

I feel that overall I performed pretty well despite some of the difficulties I faced and skills I lacked and that the outcome of the project was a relatively fun and fully functional game which leads me to looking forward to learning new skills and developing even better games, apps and projects as a whole in the future.

6 References

- developer.android.com - Minor troubleshooting and further explanation of features of Android.