Final Year Project Report

Full Unit – Final Report

Android Game Development

Patryk Pawlicki

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Jasper Lyons

Department of Computer Science

Royal Holloway, University of London

March 28, 2018

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 15247

Student Name: Patryk Pawlicki

Date of Submission: 28/03/2018

Signature: P.Pawlicki

Table of Contents

Abstract

The aim of my final year project is to develop a simple game for Android platform using the Android Software Development Kit which will cope with the variations of hardware configurations and be compatible with different devices and factors such as battery saving will be taken into consideration along with other constraints present when developing phone applications. Mobile applications as well as games are a very big market which is constantly growing with a massive potential for monetization, android as well as IOS are the leading systems for mobile games. It's relatively easy to publish apps on the Android Market making Android App development accessible to anyone interested wishing to dedicate their time. There are a lot of resources available on the web with a big community of android developers. I have always been interested in game development, especially for Android systems. My motivation behind choosing this particular project was mainly due to the fact I would like to get involved with Android Game Development to potentially pursue my future career in app development after graduating from university.

I am an Android phone user myself, who has experience with apps and games available on the Android Market, therefore I understand the industry from the user side and have a bit of insight on what users want and expect from applications they download. Seeing as the market is very densely saturated with app developers, there are a lot of free applications available for download, therefore if an app or game does not satisfy the user they will simply uninstall and install another one. It's very easy to download and install apps and users rarely get attached to using a single application. I would like to try and get into the realm of Android app development myself and contribute to the community of developers and android game users, hopefully this project will be my gateway.

Chapter 1: **A description of Android, its history and its internals**

1.1 Brief history of Android

1.1.1 Early days of Android as a Company

Android started out in October 2003 in Palo Alto in California. The company has been co funded by Andy Rubin along with Nick Sears and Chris White which played the key role in the company. Android has started out as a company which primary focus was on developing operating systems for digital cameras. However the company shifted their focus on mobile phones as there was a bigger potential in that market and was aiming to rival with the market leaders at the time; Symbian as well as the Microsoft’s Windows Mobile.

Interestingly Andy Rubin spoke about his belief in developing smarter mobile devices around 2 months before Android as a company hit the market. In an interview with BusinessWeek he spoke about seeing a great potential in developing devices aware of user’s location and preferences and how this information could be useful to model consumer products. Rubin said: *“If people are smart, that information starts getting aggregated into consumer products”[1].*

Initially Android struggled to attract investors and was on the edge of bankruptcy, it has all changed when Google purchased the company in 2005, taking on Rubin as well as the other key employees to become part of Google.

Rubin came from a background where he already had a good portfolio of experience working on operating systems and interfaces for hand-held mobile devices. He has previously worked for Apple Inc between 1992 and 1995, he worked on the development o Magic Cap operating system, MSN TV as well as Danger Inc, which he was a cofounder of, at the time of Android’s creation.

1.1.2 Android bought by Google

Around the time when Google purchased Android in 2005, Google has been purchasing many start-up companies, a lot of which never really made it to be part of Google’s arsenal of services. Initially the little known Android didn’t receive much sympathy and belief from the tech industry, even the chief executive of the company Eric Schmidt joked *“One day Larry and Sergey bough Android, and I didn’t even notice”[2],* which goes to show how little significance Android had to the tech giant Google when they initially took it on as part of their portfolio. Little did the industry know, Android would grow to become the most popular operating system in the world, currently one in two computers sold run on the Android OS.

1.2 Android OS and its internals

1.2.1 What is Android?

Android is an operating system designed for mobile devices such as smart phones, tablets, televisions as well as watches, with smart phones and tablets being the primary focus due to being the biggest market for Android.

Android is open source software with its Kernel based on one of Linux long term support branches, The source code is made available to the developers so that is can be further customised to accommodate different devices and device drivers, any modifications to the Android source code are however under the Android licence. Due to software being open source, the community has a chance of contributing to developing patches, updates and features quicker than the official developers, availability of the source code drives community driven projects and aids development.

1.2.2 Open source

The fact that android is open source also means that it’s more accessible to handset manufacturers which are looking for a platform to operate on. Devices can be produced for all kinds of price segments to target a broader audience and the operating system can be modified in order to accommodate the hardware specification of a given device.

The open nature of the OS also means that individual manufacturers as well as hobbyists have the ability to modify the operating system as well as create their own versions. Modified versions of android are referred to as “Mods” or “ROMs”. Some handheld device manufacturers created their own versions, an example of that is HTC.

One of the most known ROMs developed for Android is called “CyanogenMod”.[3]

1.2.3 CyanogenMod

CyanogenMod has been the most successful ROM to date and at its peak it has been used on around 50 million devices. The main advantage of using the ROM was the fact that it allowed more freedom over the conventional Android (which is already a fairly liberal OS, compared to competitors such as iOS for instance). It allowed its users to fully customise their devices, including themes, colour and fonts; it also made it possible to uninstall preinstalled applications and bloatware which came on the devices from the manufacturer. This is something that is not possible on conventional Android versions; it went as far as allowing the users to fully uninstall the Google Play Store as well as granted the user Root access.

Cyanogen even reached the point where the company tried to commercialise on their success and released a CyangenOS. The operating system came pre installed on some devices such as the OnePlus One; however that didn’t come with root access.

Unfortunately the CyanogenMod has shut down its services in late 2016 due to conflict within the company which resulted in the CEO Steve Kondik stepping down from his position. The ROM has been since revived under the new name “LineageOS” which has been built on top the CyanogenMod version 13 and 14.1.

1.2.4 Open Handset Alliance (OHA)[4]

OHA is a business alliance between 84 companies which aims to develop open standards for mobile devices. It has been established on 5th November 2007 by Google, Android is the primary operating system within the alliance.

The alliance includes companies from various sectors of the industry, including mobile handset makers, service providers, chip makers as well as application developers.

Some of the more notable companies which are part of the alliance are: Intel, Sony, Samsung Electronics, LG Electronics, HTC, Toshiba, Vodafone, T-Mobile, Texas Instruments, Motorola, Nvidia and many more.

1.3 Android Version Control

Android versions are named after desserts, starting from the release of version 1.5 Cupcake in April 2009. The previous API versions had no official codename. The code names for OS releases are in alphabetical order.

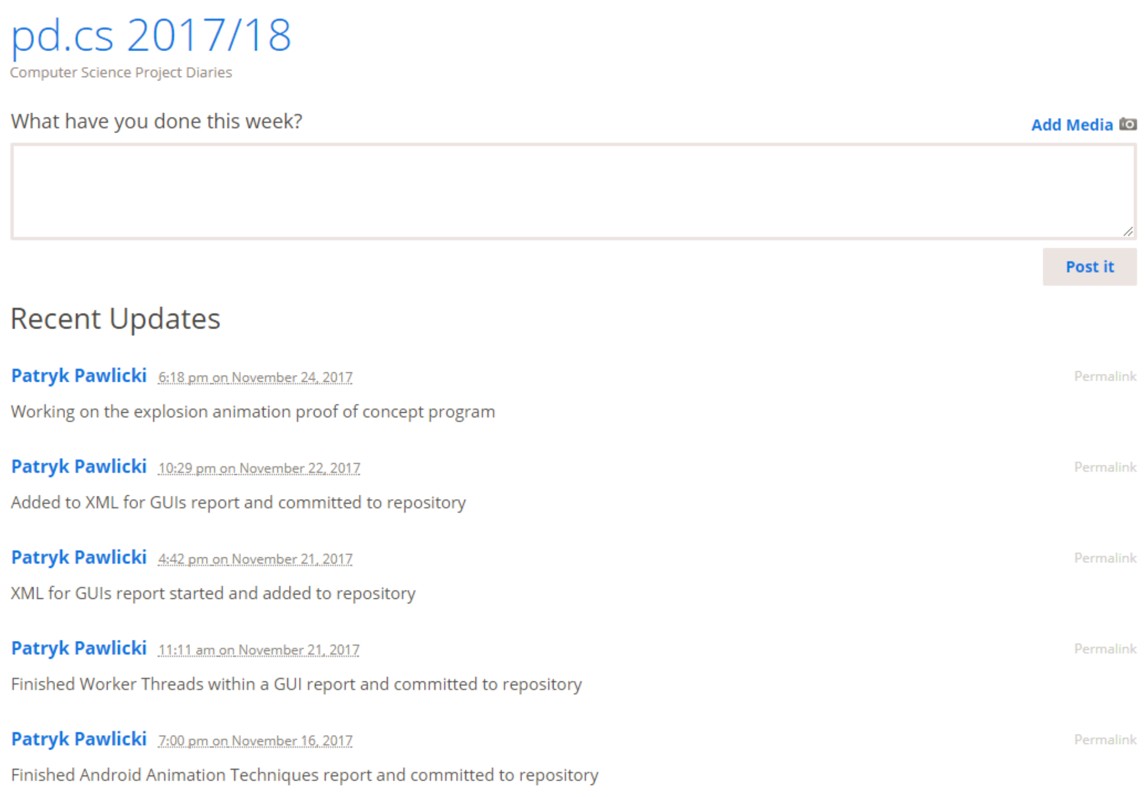
Figure 1. Different android versions[5]

Below we can see a table of all android versions with their corresponding release date, version and API level. I will also cover some of the most notable features and changes implemented at different version levels.

Version name:	Version number:	Release date:	API level:
	1.0	September 2008	1
	1.1	February 2009	2
Cupcake	1.5	April 2009	3
Donut	1.6	September 2009	4
Eclair	2.0 - 2.1	October 2009	5-7
Froyo	2.2 – 2.2.3	May 2010	8
Gingerbread	2.3 – 2.3.7	December 2010	9-10
Honeycomb	3.0 – 3.2.6	February 2011	11-13
Ice cream sandwich	4.0 – 4.0.4	October 2011	14-15
Jelly Bean	4.1 – 4.3.1	July 2012	16-18
Kitkat	4.4 – 4.4.4	October 2013	19-20
Lollipop	5.0 – 5.1.1	November 2014	21-22
Marshmallow	6.0 – 6.0.1	October 2015	23
Nougat	7.0 – 7.1.2	August 2016	24-25
Oreo	8.0 – Present	August 2017	26-27

Android is a backwards compatible OS, therefore the newer OS versions will work fine with software designed in older APIs, the reverse however is not the case. Therefore choosing the target API level is crucial when developing for Android, I cover this topic further in Chapter 4, Section 4.3.2.

Chapter 2: **Project Diary**



pd.cs 2017/18

Computer Science Project Diaries

What have you done this week?

Recent Updates

Patryk Pawlicki 6:18 pm on November 24, 2017 Permalink

Working on the explosion animation proof of concept program

Patryk Pawlicki 10:29 pm on November 22, 2017 Permalink

Added to XML for GUIs report and committed to repository

Patryk Pawlicki 4:42 pm on November 21, 2017 Permalink

XML for GUIs report started and added to repository

Patryk Pawlicki 11:11 am on November 21, 2017 Permalink

Finished Worker Threads within a GUI report and committed to repository

Patryk Pawlicki 7:00 pm on November 16, 2017 Permalink

Finished Android Animation Techniques report and committed to repository

Figure 2. Computer Science Project Diaries[6]

In order to log my progress on the project, I have been posting updates on the blog. These aid the development and clearly illustrate the progress at any given point of the project. The blog helps me to stay organised and make sure that all of the tasks I am currently working on, planning on working on, or the tasks I have finished are accordingly logged.

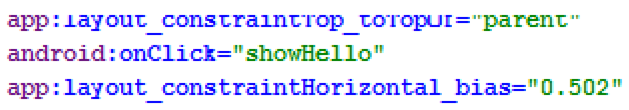
Figure 1 shows a screenshot of my diary and some of the updates I have made to log my progress. I will be using the diary throughout my project to stay organised and up to date with the previously written plan of action for the project.

Chapter 3: **Proof of concept programs**

3.1 Hello World

My first proof of concept program is the classic “Hello World” program. The idea behind the program is very simple, it includes an empty interface with a button, which when pressed displays the “Hello World” message in the middle of the UI.

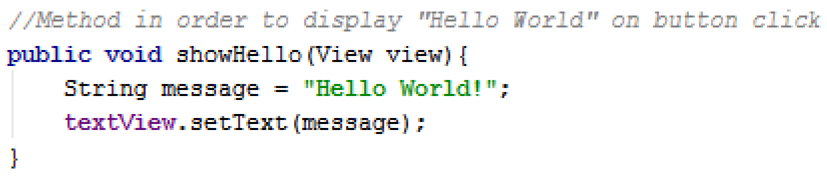
This functionality is achieved through the use of an onClick() method called “showHello” in order to alter the text of a blank text field. We can see that on Figure 2 below.



```
app:layout_constraintTop_toTopOf="parent"
android:onClick="showHello"
app:layout_constraintHorizontal_bias="0.502"
```

Figure 3. showHello method linked in the button XML to be triggered on button click

In order to make the method work, I had to write relevant code which will change the text of the empty TextView object which has also been added to the View. We can see that in Figure 3.



```
//Method in order to display "Hello World" on button click
public void showHello(View view){
    String message = "Hello World!";
    textView.setText(message);
}
```

Figure 4. showHello method defined in the java file

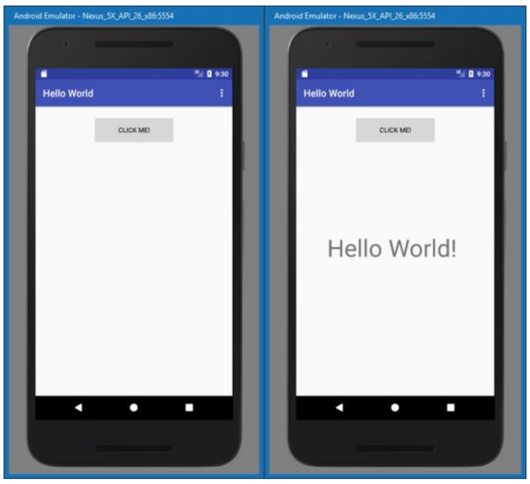

Android Emulator - Nexus_5X_API_26_x86:5554
Hello World
CLICK ME!
Android Emulator - Nexus_5X_API_26_x86:5554
Hello World
CLICK ME!
Hello World!

Figure 5. Hello World application before button click (left) and after the button click (right)

The Hello World program is a very basic structure however it has helped me to understand how to use the GUI builder in order to add objects as well as change their state through the use of onClick methods. It was a necessary learning step to learn the basics about manipulating objects programmatically as well as creating basic user interfaces.

3.2 Explosion Animation

The explosion animation proof of concept program has been my introduction to implementing animation into the user interface. The animation technique I have decided to use for this purpose is by using drawable objects.

The first thing I had to do is to source my images for the explosion animation and split them up into individual frames so that when played back quickly in order, motion will be formed and the animation will be displayed smoothly.

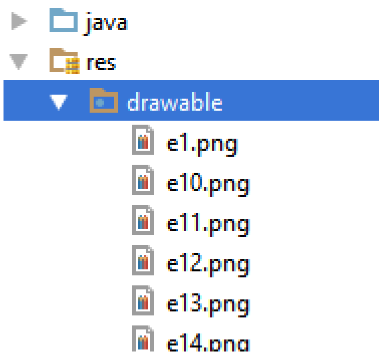
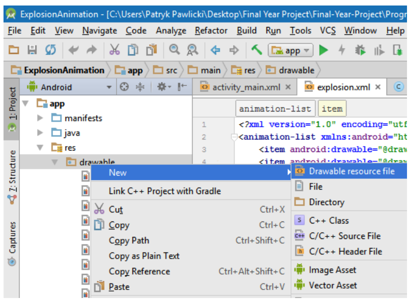


Figure 6. Source folder with all the drawable frames of my animation

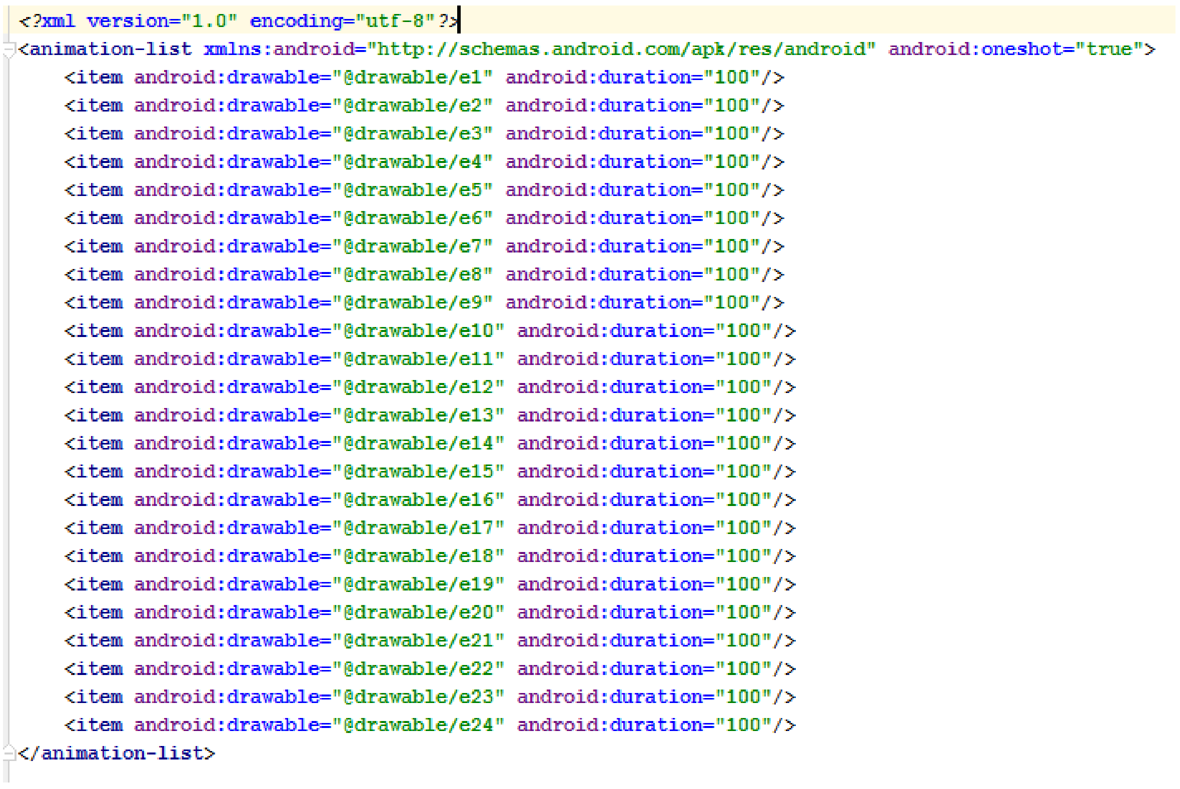
The first step is copying my images into the project resources under the drawable folder as displayed in Figure 5.



```
<?xml version="1.0" encoding="utf
<animation-list xmlns:android="ht
    <item android:drawable="@draw
```

Figure 7. Creating a new drawable resource file

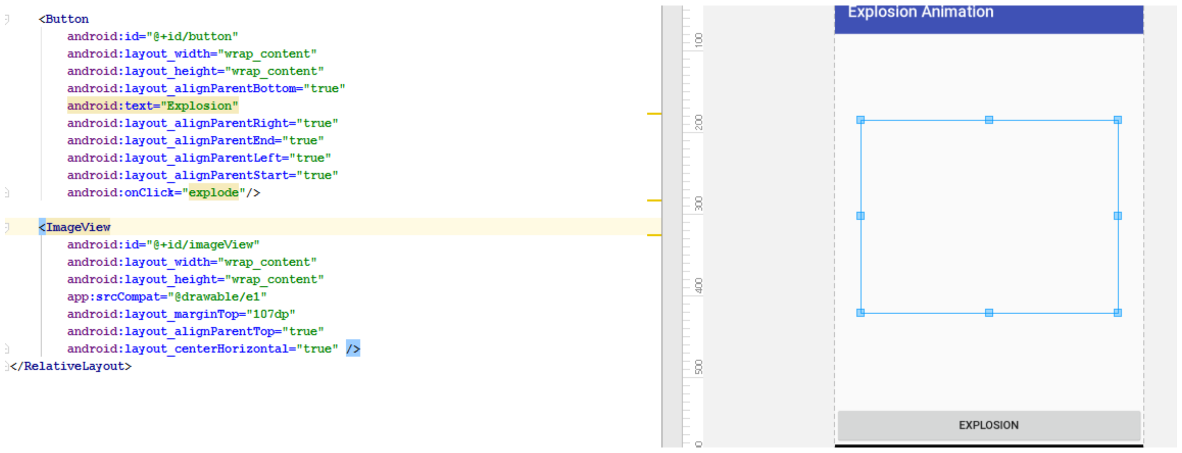
The next step is creating a drawable resource file in order to define our drawable animation and all of its frames in order so that they can all be animated. We can see the process of me defining a new resource file in Figure 6.



```xml
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android" android:oneshot="true">
    <item android:drawable="@drawable/e1" android:duration="100"/>
    <item android:drawable="@drawable/e2" android:duration="100"/>
    <item android:drawable="@drawable/e3" android:duration="100"/>
    <item android:drawable="@drawable/e4" android:duration="100"/>
    <item android:drawable="@drawable/e5" android:duration="100"/>
    <item android:drawable="@drawable/e6" android:duration="100"/>
    <item android:drawable="@drawable/e7" android:duration="100"/>
    <item android:drawable="@drawable/e8" android:duration="100"/>
    <item android:drawable="@drawable/e9" android:duration="100"/>
    <item android:drawable="@drawable/e10" android:duration="100"/>
    <item android:drawable="@drawable/e11" android:duration="100"/>
    <item android:drawable="@drawable/e12" android:duration="100"/>
    <item android:drawable="@drawable/e13" android:duration="100"/>
    <item android:drawable="@drawable/e14" android:duration="100"/>
    <item android:drawable="@drawable/e15" android:duration="100"/>
    <item android:drawable="@drawable/e16" android:duration="100"/>
    <item android:drawable="@drawable/e17" android:duration="100"/>
    <item android:drawable="@drawable/e18" android:duration="100"/>
    <item android:drawable="@drawable/e19" android:duration="100"/>
    <item android:drawable="@drawable/e20" android:duration="100"/>
    <item android:drawable="@drawable/e21" android:duration="100"/>
    <item android:drawable="@drawable/e22" android:duration="100"/>
    <item android:drawable="@drawable/e23" android:duration="100"/>
    <item android:drawable="@drawable/e24" android:duration="100"/>
</animation-list>
```

Figure 8. Drawable resource file defining all of the frames as drawable items in order to compose an animation-list

The next step is to add all of the drawable animation frame images as items to the and set the duration so that the animation list can be composed. The second line also includes a definition "onceshot" which is set to true, this basically means the animation will display once and stop, instead of looping constantly.



```xml
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Explosion"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:onClick="explode"/>

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/e1"
        android:layout_marginTop="107dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

Figure 9. GUI builder displaying my animation added as an ImageView as well as a button to trigger the "explode" animation

After we have our animation list composed we can add it to the UI y adding a ImageView object onto the canvas and referring to the first image of the animation in the XML properties.

Finally we are ready to edit our java file corresponding to this Activity. We have to define all of the new objects that have been added to the View, such as the button and the ImageView. After that we link our ImageView animation with the XML file which defines all of the frames which compose the animation list which I have described earlier and define the animation.

We can see that in Figure 9.

```
Button explosion_btn = (Button)findViewById(R.id.button);
ImageView animation = (ImageView)findViewById(R.id.imageView);
animation.setImageResource(R.drawable.explosion);
final AnimationDrawable explosionAnimation = (AnimationDrawable)animation.getDrawable();
```

Figure 10. Definition of all of the objects and animation

The last step is to add a button listener which will trigger the animation when the button is pressed.

```
explosion_btn.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        explosionAnimation.stop();
        explosionAnimation.start();
    }
});
```

Figure 11. Explosion button listener that triggers the animation

Figure 10 shows the structure of the listener method. An interesting thing worth mentioning is that the stop() method gets called before the start() method is called every time the button is click. This is in order so that if the animation is already running and the button is pressed, the application fill restart the animation. I have found that this is a necessary in order to make the animation run without any problems.

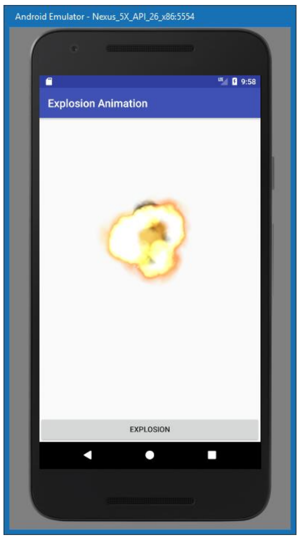


Figure 12. Explosion animation at work

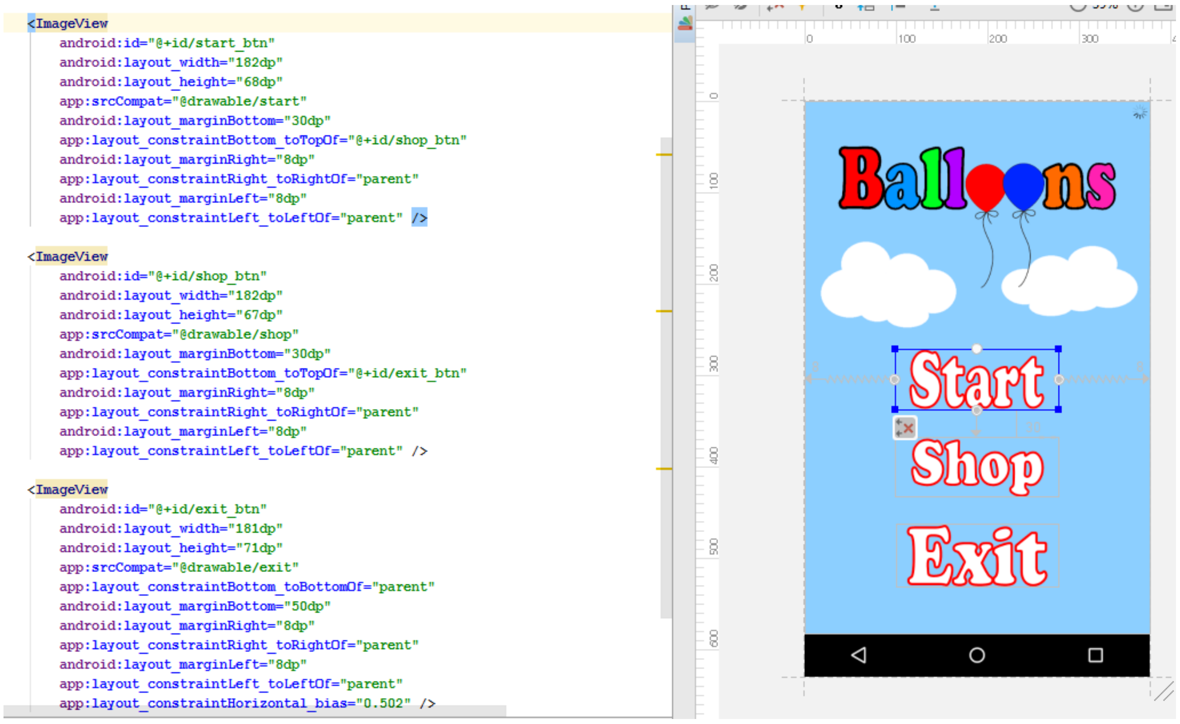
This proof of concept program has taught me the fundamentals of creating animation in the android framework. I have learnt how to use individual frames to compose an animation, how to control the playback speed and how to use button listeners in order to trigger events. I have previously used an onClick method in the XML code, this has been my first attempt at creating a button listener.

3.3 XML based User Interface

For my final proof of concept program I had to learn how to create multiple activities so that I can have more than user interface. I have also spent a significant amount of time using a graphical editing software in order to create the design for all of the objects required for a UI.

I wanted my UI to be aesthetically pleasing therefore I couldn’t use standard buttons which are provided by the Android SDK. I decided to design my own and use them to serve the purpose of a button.

I had to also learn how to navigate between activities when adequate buttons are pressed as well as how to close an application on button click.



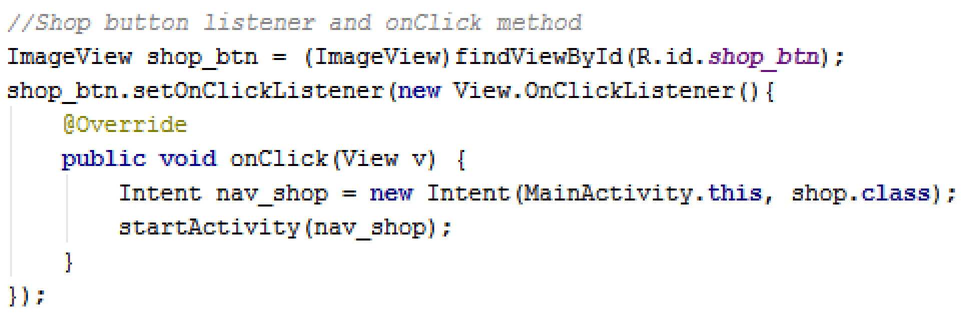
```
<ImageView
    android:id="@+id/start_btn"
    android:layout_width="182dp"
    android:layout_height="68dp"
    app:srcCompat="@drawable/start"
    android:layout_marginBottom="30dp"
    app:layout_constraintBottom_toTopOf="@+id/shop_btn"
    android:layout_marginRight="8dp"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginLeft="8dp"
    app:layout_constraintLeft_toLeftOf="parent" />

<ImageView
    android:id="@+id/shop_btn"
    android:layout_width="182dp"
    android:layout_height="67dp"
    app:srcCompat="@drawable/shop"
    android:layout_marginBottom="30dp"
    app:layout_constraintBottom_toTopOf="@+id/exit_btn"
    android:layout_marginRight="8dp"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginLeft="8dp"
    app:layout_constraintLeft_toLeftOf="parent" />

<ImageView
    android:id="@+id/exit_btn"
    android:layout_width="181dp"
    android:layout_height="71dp"
    app:srcCompat="@drawable/exit"
    app:layout_constraintBottom_toBottomOf="parent"
    android:layout_marginBottom="50dp"
    android:layout_marginRight="8dp"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginLeft="8dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintHorizontal_bias="0.502" />
```

Figure 13. Screenshot of the GUI builder and the corresponding XML file

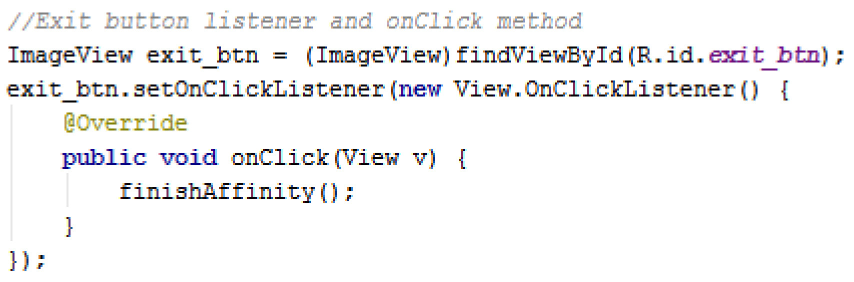
Figure 12 shows how all of my custom made buttons are added to the UI as ImageView items, in order to make them functional I had to come up with button listeners for each one. I will now show an example of how I made my buttons navigate between different views as well as how the exit button works. The process of navigating between the user interfaces is the same for all of the buttons, therefore I will illustrate one.



```
//Shop button listener and onClick method
ImageView shop_btn = (ImageView)findViewById(R.id.shop_btn);
shop_btn.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        Intent nav_shop = new Intent(MainActivity.this, shop.class);
        startActivity(nav_shop);
    }
});
```

Figure 14. Button listener for the "Shop" button

Figure 13 shows how I have used the "Intent" object in order to link my interfaces together. The intent basically defines the starting Activity (MainActivity.this) as well as the Activity we are navigating to (shop.class). This is all encapsulated in an inClick() event as standard.

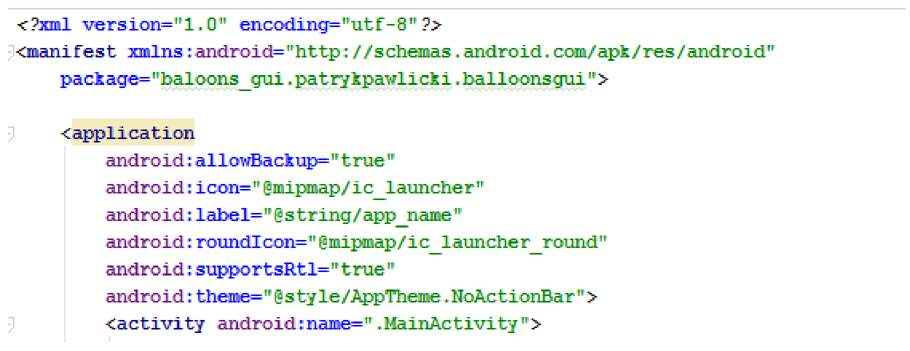


```
//Exit button listener and onClick method
ImageView exit_btn = (ImageView)findViewById(R.id.exit_btn);
exit_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finishAffinity();
    }
});
```

Figure 15. Button listener for the "Exit" button

Figure 14 illustrates how the exit button works in order to close the application. This functionality is very easily achieved as it involves one method, however I am mentioning it as it is something now which I had to research in order to implement it with the use of finishAffinity() method.

Finally I wanted to cover how I managed to make the application display in full screen so that no name bar is included, this required some additional changes in the Android Manifest XML file. In order to make the views display in full screen, I had to change the theme of each one to "@style/AppTheme.NoActionBar". This can be seen in Figure 15.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="baloons_gui.patrykpawlicki.balloonsgui">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme.NoActionBar">
        <activity android:name=".MainActivity">
```

Figure 16. AndroidManifest XML file defining the Activities within the application

Chapter 4: **Design of a GUI based mobile game**

4.1 Initial Game Concept

The concept I have came up for my GUI based mobile game is a game of balloons. The aim of the game is to pop helium filled balloons which are flying up from the bottom of the screen to the top. The game will progressively get harder, generating an increasing amount of balloons as well as increasing the speed of their movement. The game will end when the user has failed to catch a predefined number of balloons (based on difficulty).

The game will include special effects such as the ability to catch balloons which carry an additional balloon allowance, increasing the number of balloons a user can miss before losing a game, instant destruction of all balloons which are currently on the screen or coin rewards.

Coins are obtained through playing the game; the score achieved at the end of the game will be converted into coins which can be spent in the shop. The shop will allow the users to purchase upgrades which increase their balloon allowance or increase the multiplier which awards coins based on their score. After every round ends, the user will see a screen which will inform him on their score and display some statistics such as: time in game, balloons popped and coins earned.

A potential extension to my game would be implementing a High Score table which would store player’s scores in order to enable competition. Such high score table could contain different categories in which the players could compete on. These could include: most points scored in a round, most balloons popped (overall), longest survival time, etc.

4.2 Initial Game Design

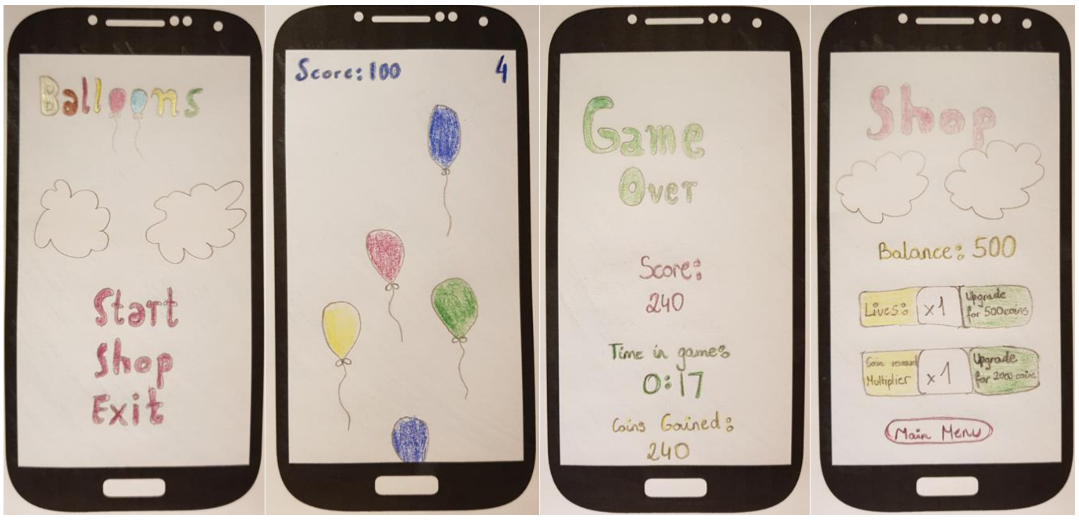


Figure 17. Initial sketches of my game design
(from left to right: Main Menu, Gameplay, Shop UI, Game over UI)

Figure 16 illustrates some of my initial sketches I have done in order to come up with a design. I have then developed my design further and created it using graphical editing software called paint.net [7].

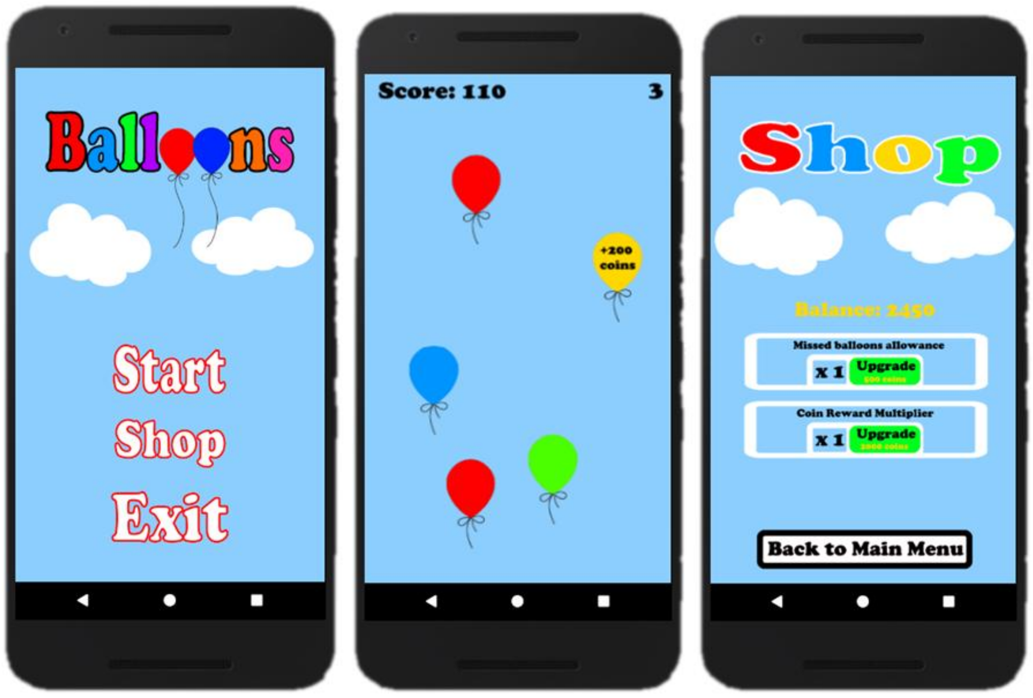


Figure 18. Game Designs created with the use a graphical editing software

Figure 17 illustrated the designs I have came up with using graphical editing software. The design is likely to change during the development process. I have aimed to come up with a game design that will be colourful, readable and easy to navigate.

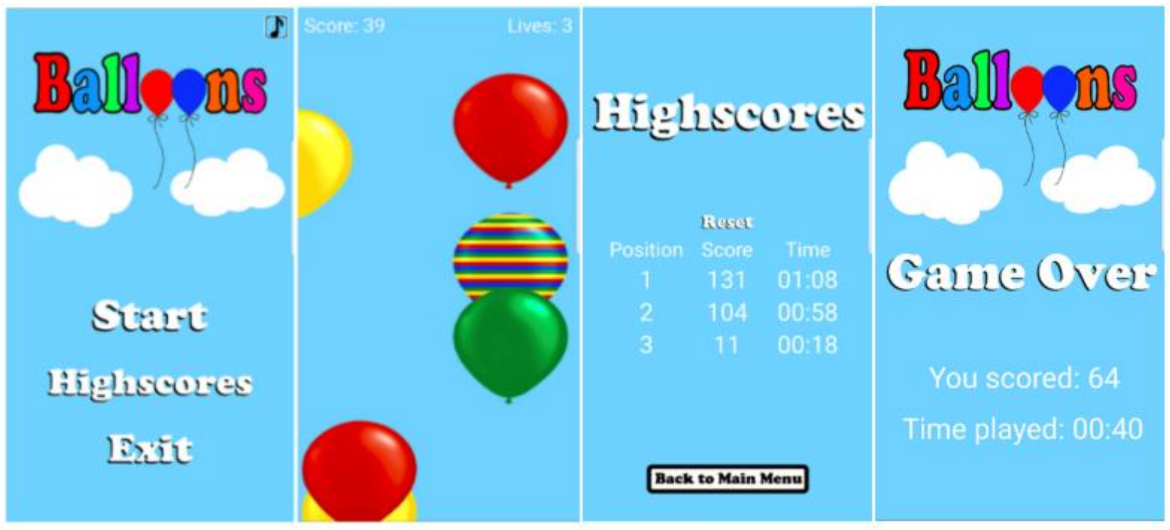
4.3 Final Game Concept

My final game concept turned out slightly different to what I originally planned for my game. It is based on the same idea of a Balloons game, the aim is still the same, pop as many balloons as possible to achieve the highest score without missing any which results in a lost game.

I have decided to get rid of the idea of the shop and coins which allows user to purchase upgrades which will give them an advantage over other players, essentially playing the game more in order to obtain coins would put users in a major advantage and would eliminate the aspect of competition as well as high score being a result of skill and reflexes. I wanted to develop the game to be fun and competitive therefore upgrades granting advantages over other players had to go, instead I decided to implement the Highscores which I have originally thought of as a potential extension.

4.4 Final Game Design

The final game design remained mostly the same, with some minor changes due to the changed concept and some UI improvements, such as buttons for better aesthetics.



Position	Score	Time
1	131	01:08
2	104	00:58
3	11	00:18

You scored: 64
Time played: 00:40

Figure 19. Final Game Design
(from left to right: Main Menu, Gameplay, Highscores, Game over UI)

Figure 18 shows what the application looks in its final form, the pictures are screenshots taken from the application running on a phone. As we can see the main design remained with some graphical improvements.

4.5 Framework

4.5.1 Android Studio

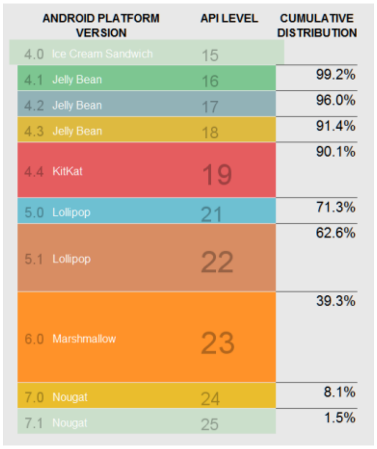
Android Studio is the official IDE (Integrated Development Environment) made for Android application development. *"Android Studio is built on IntelliJ and is capable of advanced code completion, refactoring, and code analysis."***Error! Reference source not found.** IntelliJ[9] is one of the most popular IDE with growing popularity which provides a range of useful tools for software development.

The software also includes a range of other useful tools such a built in Emulator which allows us for rapid testing of our application as we develop, it allows us to test on different devices, screen sizes and preferences. This greatly aids development and software optimisation on order to make it run well on all sorts of android devices.

Development of the graphical UI in Android Studio is carried out through the use of a built in GUI builder which allows the user to add objects onto the View and easily manipulate them in order to achieve the desired structure. The XML code for the View is automatically generated. I cover this topic in more detail in Section 9.2.

Android Studio is the most popular choice amongst Android app developers therefore it was an obvious choice as it comes with a very broad support from the community, the SDK is very well documented and due to the large following, it's very easy to find answers to common problems on various forums.

4.5.2 Version Control



ANDROID PLATFORM VERSION		API LEVEL	CUMULATIVE DISTRIBUTION
4.0	Ice Cream Sandwich	15	
4.1	Jelly Bean	16	99.2%
4.2	Jelly Bean	17	96.0%
4.3	Jelly Bean	18	91.4%
4.4	KitKat	19	90.1%
5.0	Lollipop	21	71.3%
5.1	Lollipop	22	62.6%
6.0	Marshmallow	23	39.3%
7.0	Nougat	24	8.1%
7.1	Nougat	25	1.5%

Figure 20. Supported devices based on API level [10]

Figure 18 illustrates a table with all of the latest Android platform versions and their support amongst Android devices currently used on the market. Choosing the right API Level was an important decision I had to make prior to starting my app development.

When choosing the API, the goal is to achieve the right balance. I aimed to maximize the amount of devices my app can run on, without having to choose a version which is too outdated, meaning it might lack some of the new features and improvements implemented in the newer iterations.

I have decided to choose Android 4.4 Kitkat[11], due to it supporting over 90% of devices currently on the market whilst still offering some of the fresher features of the Android framework, most notably, the immersive full-screen mode **[23]** which has been added in this API version.

Chapter 5: **Android Life Cycle**

5.1 Activities in Android

In Android, the users navigate through different activity instances as the applications are being used, or navigated in and out of. All the transitions between applications or within a single application correspond to a change in an activity state of an application. This is done through the Activity class via a series of callback methods which are used for pausing, stopping, starting or resuming an activity.

It's very important to consider what happens when an application is interrupted by things such as phone calls, alarms or even user input which will result in the application being paused or stopped. It's important to define what happens in those instances so that the progress of our game is not lost and the user can switch between the applications fairly seamlessly without consuming system resources without using them.

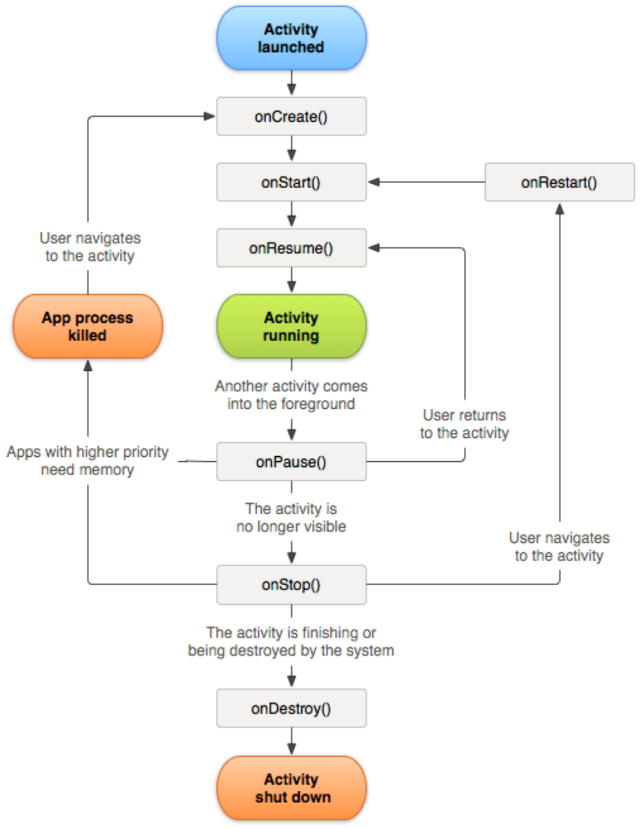

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 21. Activity Life Cycle [13]

5.2 Callbacks

There are 6 main callbacks:

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroy()

5.2.1 OnCreate()

In this state the system will create and configure the backbone of the program required for the lifecycle of its operation, also things like layout for the main activity or the first UI would be configured here.
However onCreate is not a state that the application will stay on for long and after everything is set up, the activity will always move onto the onStart() state.

5.2.2 OnStart()

The started state is another quick state in which the activity doesn't stay for long, its main purpose is to make the app interactive and make the activity available to the user. If the activity comes to the foreground it is then followed by onResume(), if it doesn't it will be followed by the onStop() callback.

5.2.3 OnResume()

When applications are being actively used by the user and being interacted with, the activity will spend the most time in the Resumed state, until it gets disrupted by user input or other applications so that the focus is taken off it.

The resume state is at the top of the activity stack and has the least likelihood of being killed to free up memory. It has priority over other running applications and is currently in the spotlight.

5.2.4 OnPause()

onPause() state is called when the focus is taken off the application, it does exactly what it says, it pauses the application without killing it so that the system can come back to it later without losing any progress, the application is still running but is paused and in the background. This is an opposite state from resume. When in this state, there are two possibilities, either onResume() will be called and the application will carry on or onStop() will be called in order to stop it.

5.2.5 OnStop()

This callback is called when the activity has either finished running or when the UI is no longer visible to the user. After the call, the activity releases the resources it doesn't need to free up memory, all the necessary CPU intensive operations can be performed on this callback, such as saving data to the database, etc. This callback is often followed by onRestart() or onDestroy() callback.

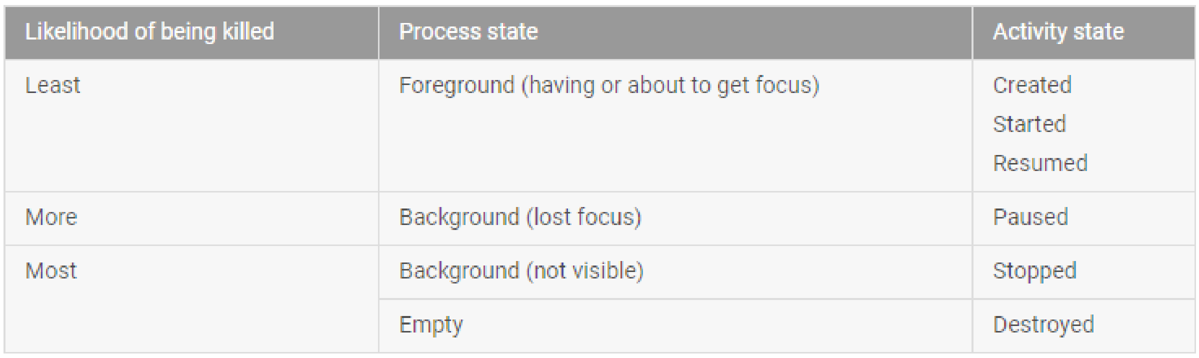
5.2.6 OnDestroy()

This is the final call of an activity, after it is invoked the activity gets destroyed and all the remaining resources gets released.

5.3 Activity State and ejection from memory

The system will kill processes in order to free up memory when needed to do so, the decision on what processes should be killed is dependent on the state of the activity within that process. This ensures that the system will not kill the application actively being used by the user but will rather prioritize processes which correspond to applications currently not in use.

Below we can see a table which talks about different activity states and the likelihood of the system killing them.



Likelihood of being killed	Process state	Activity state
Least	Foreground (having or about to get focus)	Created Started Resumed
More	Background (lost focus)	Paused
Most	Background (not visible)	Stopped
	Empty	Destroyed

Figure 22. Relationship between process lifecycle and activity state [23][17]

Chapter 6: **Worker Threads within a GUI**

6.1 UI Thread

By default, the User Interface in the android framework is controlled by one thread, often referred to as the UI thread or the "main" thread of execution. It is responsible for everything the user sees on the screen while interacting with the application. This is not ideal as operations which take longer to compute will block the UI making our app unresponsive for the user and force them to wait until the operation finishes or even worst, trigger the Application Not Responding notification, which allows the user to close our application. We can avoid this with the user of worker threads.

6.2 What are worker threads

Worker threads have been introduced in order to take the load off the UI thread which is responsible for controlling the UI. Having all of our operations on the UI thread can have detrimental effects to the performance of our application. All the time intensive tasks such as querying the database, carrying out complex calculation or performing network functionality would mean that the GUI that the user is interacting with, would appear to be hanging as it becomes unresponsive to input until the operation finishes.

With game development it is essential to carry out all of the work and calculations in a worker thread, whilst the UI thread should focus on the graphical aspect of our application such as drawing animations and responding to user input.

Worker threads are basically threads which work in the background of our application, they are referred to as "worker" as they carry out work which the user can't see and they are not allowed to interact with the UI directly, only objects which are part of the UI have access to the thread, an example of this would be the View objects.

In order to allow the background threads to communicate with the UI thread, the Android framework provides us with a few useful tools in order to allow communication between the worker threads as well as the UI thread. The most common one is to use a Handler. Handlers are responsible for thread management and allow messages to be exchanged between the worker thread as well as the UI thread.

6.3 AsyncTask

A common alternative to using Handlers is the use of AsyncTask subclass which provides us with a set of useful methods which make it even easier for background threads to communicate with the UI thread and exchange messages.
The process of passing on a message from the worker thread onto the UI thread is as simple as using a few methods which I will briefly cover below:

6.3.1 doInBackground():

This is where all of the work in our worker thread is carried out, all of the complex calculations, database queries and other time intensive operations would go here. Whenever we would want to communicate with the UI thread and update it with the progress from our operation, we call the **publishProgress()** method from within this method which invokes the **onProgressUpdate()**, this is where we specify the reaction to our method call, this could be as simple as updating the progress bar.

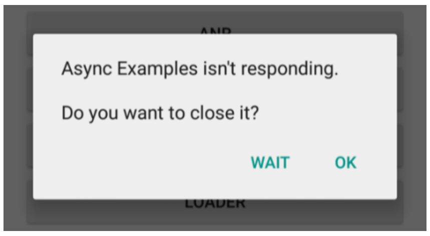
6.3.2 onPostExectute():

This method would be called after all of our time intensive background work has been finished, inside of this method we would specify the action carried out after that has taken place. It could be some meaningful change in UI or as simple as displaying a little message.

For example, if we used the above described structure to represent a user downloading an application of the web, all the intensive processing would go iniside of the doInBackground() method, which would allow the user to still interact with the UI. Once in a while the publishProgress() method would be called to update the UI on the progress of that activity, could be as simple as updating the progress bar. Finally the onPostExecute() the UI could display a little window notifying the user that the file has finished downloading.

6.4 Application Not Responding (ANR)

Android has a system guard implemented against unresponsive applications which hang for extensive amounts of time, making the user wait. In the event of an application being insufficiently responsive for a period of time by displaying a dialog that says the app has stopped working, Figure 21 illustrates such dialog box. [24]


Async Examples isn't responding.

Do you want to close it?

Figure 23. Application Not Responding Dialog Box [25]

Generally the ANR dialog box will appear after the application has been unresponsive to user input for about 5 seconds; it will then throw up the dialog box asking the user whether they want to close the application as it is not responding.

In order to avoid the ANR dialog boxes from appearing on our application it is vital to make sure that the operations do not block the UI thread which would result in the GUI being unresponsive.

The use of worker threads allows us to minimize the risk of our application being sluggish or unresponsive and having to deal with the dreaded ANR dialog box. On top of using worker threads alongside our UI thread it is important to make sure that the workload that is carried out directly in our UI thread takes as little processing time as possible.

Chapter 7: **Design Patterns**

7.1 What are design patterns

Design patterns aim to create reusable solutions to common software development problems. They can *“even improve the documentation and maintenance of existing systems by furnishing an explicit specification of class and object interactions and their underlying intent.”* [17]

Design patterns add clarity to the code and make it much easier to maintain, edit and test. Design patterns are solutions which are widely recognised by the industry therefore the intent of our implementation is much easier to understand as it can be explicitly specified in the documentation.

There are 3 main types of design patterns: Creational, Structural and Behavioural. I will cover some of the design patterns that can be used in the Android framework and can potentially aid the development of my game.

7.2 Types of design patterns

7.2.1 Creational

Builder:

The builder pattern is a commonly used in order to create multiple complex objects out of basic objects collectively. It makes it much easier to create them in mass amounts, the implementation makes sure our code is much cleaner and reduces the repetition. It allows us to use the same objects for various representations, or representations of other complex objects, which might consist of different basic objects.

Dependency Injection:

Dependency injection is a pattern where the dependency is not given to an object on creation; it is rather “injected” into it externally. In simple terms, this basically means that we are passing instance variables to our objects externally after they are created rather than on creation. There are multiple ways we can inject dependencies into our objects, two most basic methods to “inject” dependencies are through the constructors or setters. [23]
Dependency injection is especially useful when testing, it allows us to isolate classes in order so that a stub object can be passed and appropriate functionality can be easily tested.

Singleton:

This design pattern is used where only a single instance of a class is needed to be globally accessed. The class, with the way it’s written, implements a static method which returns an instance of the class, which can be initialized only once in the entire program, the constructor cannot be invoked. This is useful for modelling real life objects which can only exist as a single instance, singleton allows the instance to be accesses from anywhere within the program, it has a global access. [24]

7.2.2 Structural

Adapter:

The purpose of the adapter design pattern is to create a bridge between two classes which are not compatible and make it possible for them to work. The way that this is achieved is by converting our current class into an interface which we want the user to see. It’s very useful when working with classes that are not compatible but provide essential functionality for us, we can “adapt” them through a use of specifically written methods so that they convert their interface into one more suitable to achieve our goal.

Facade:

The goal behind the facade design pattern is to hide the underlying complexity of the system from the user. It’s about building an interface which will contain what the user needs without the unnecessary complexity behind it. We can think of it just like the literal real world use of facade, it’s a cover up, a fancy clean interface to hide all the nasty inner workings of a program.

7.2.3 Behavioural

Command:

“Command decouples the object that invokes the operation from the one that knows how to perform it. “[25] The invoker received a command hidden in an object and then passes it to the appropriate object which handles a given request. This pattern is useful when wanting to issue a command without knowing the receiver; the pattern handles our request and forwards the command to an appropriate object so that it can be dealt with.

Observer:

The observer pattern is used in order to be able to monitor changes in other objects. Whenever a change is detected, an observable object will notify all of its observers and the changes could be acted upon through the use of various methods. The observer pattern aims to decouple objects so that they are easy to change and maintain.

Chapter 8: **Animation Techniques**

8.1 Animation Techniques used in the Android Framework

In the Android framework there are several animation techniques which can be used for both 2D as well as 3D graphics. The choice on which technique to use greatly depends on what we are trying to achieve since all of them have their pros and cons.

There are three main systems used in Android to create animation:

- Property Animation
- View Animation
- Drawable Animation

Along with the animation techniques there are frameworks which allow us to create 2D and 3D graphics in order to be used for our animations. The most common method is using Canvas and Drawables, which is powerful enough for most animation needs; there is also an alternative of using the OPEN GL ES framework which allows us to create more sophisticated and powerful graphics.

8.1.1 Property Animation

Property Animation focuses on the object properties in order to animate it. That can involve changing its X and Y coordinates in order to move it around the screen, altering its size properties to scale or rotate 2D or even 3D object by choosing a pivot point. We can also change alpha transparency of objects, which is useful to fade objects in and out of the view.

All of this is performed based on the duration of the animation; we can choose what kind of movement we want, over a specific timeframe. Longer duration or smaller change in properties will result in a slower animation, where as shorter animation duration or bigger change in properties over a timeframe will result in a quicker animation.

The animation interpolation can be linear meaning the movement will be the same within the duration of an animation (for example; change X coordinate from 0 to 50 over 50ms, 10ms being in increase of X coordinate by 10) resulting in a smooth consistent animation, or non-linear meaning the animation will change speed as the change in properties will not be constant within a time frame. We can even make the animation stop and resume after a given timeframe. All of this is possible due to the use of the “TimeInterpolators” provided by android or the use of Key frames which we can create and set up ourselves. Keyframes are a tool commonly used in filmmaking and animation, Keyframe is also a class used by Value Animator.

Property animation can be declared in both XML as well as programmatically. The main advantage of doing it in XML is the fact that we can reuse the code for more than one object or more than one activity, synchronising the animation sequence is also easier using XML.

8.1.2 View Animation

View animation, also referred to as “Tween Animation” is an older animation system implemented in the Android API. Just like property animation it allows us to animate view objects by moving them around, scaling, rotating and even changing the alpha transparency settings. The main disadvantage of using View Animation is the fact that it only allows us to work with view objects, whereas property animation allows us to work with any object.

Just like property animation we have the option to animate objects using XML or programmatically, with XML being the preferred option due to the ability to reuse code for more than one object; it is also a much cleaner, more readable implementation.

The speed of an animation is defined by assigning an Interpolar, the same technique which I have covered in the Property Animation.

The main problem with view animation compared to property animation is the fact that if we wanted to animate a button to move across the screen, visually it would do exactly what we ask it to do, but the physical button which we can click that provides functionality would stay in the original button location. Therefore if we wanted to animate a button using a view animation, it would no longer work when clicked and in order to move the functionality of the button to make it clickable again we would have to code that separately.

View animation is a very basic technique and it doesn’t require much code to write, it’s a good option when working with basic animation in views, where it’s flaws or inability to animate objects outside of view are not a problem.

8.1.3 Drawable Animation

Drawable animation is a technique where all of the images are loaded in and then played in sequence in order to create movement. It is done via the *AnimationDrawable* class.

Drawable animation is a very common technique for animation; it allows the use of spritesheets. Spritesheet is a series of images which are all in a slightly different positions, so when played back quickly in series, they appear to be moving, it’s just like having individual frames of a movie which are played back quickly to create movement. Spritesheets have been in most classic games and are still used today in modern games, for example a very successful game “Cut the rope” is also based on drawable animation using spritesheets.

Drawable Animation offers a lot of flexibility, the smoothness of animation can be chosen by deciding how many images we would like to use for our movement, the more we use, the smoother the animation and it will allow us to play it back slower without being able to notice the transition, however it will mean we will have to implement more individual images. It’s a bit like frame rate, the more we have the smoother it is, however it will also require more computational power in order to be played back.

8.2 2D and 3D Graphics

The android framework provides a variety of drawing APIs for 2D graphics which allow a user to generate and animate custom graphics for their application needs. There are two main ways of how graphics is done on the android frameworks, drawing graphics using the View object in the layout section or drawing graphics onto the Canvas object. Android framework also has a Drawable library which provides us with a range of tools for drawing shapes and graphics.

8.2.1 Canvas and Drawable

Canvas and Drawables are the most sensible option for creating custom 2D graphics for an android game, it’s perfect for rapidly changing animation unlike drawing straight on top of View. Graphics can be either drawn using the tools provided by the Drawable library or imported from our project resources, therefore techniques such as the use of sprites can be utilised.

The supported file types are PNG, JPG or GIF, however PNG is the proffered file type for the drawable library. When images are imported from project resource files, the images are automatically optimized using a lossless compression in order to reduce their size.

Drawable class has a useful subclass called ShapeDrawable which is a handy tool to create two dimensional graphics of primitive shapes. We can define and customise the size, position as well as colour properties and get our shape drawn onto the View.
An alternative to this is using vector drawables which are defined by a set of points, lines or curves which are also an option in the Drawable library.

8.2.2 OpenGL ES

OpenGL ES is a library in the Android framework which allows creating sophisticated 2D and 3D graphics which can be used for animation. The objects are formed through the use of matrices with a set of coordinates in order to define a shape. OpenGL has a very powerful toolset however it is very complex and requires a lot of coding in order to come up with 3D graphics of high quality, for my game I will most likely be focusing on the use of Canvas as well as drawables.

Chapter 9: **XML for GUIs**

9.1 GUI design in Android

In the android framework there are two ways of defining the layout for Graphical User Interfaces, we can either programmatically manipulate the properties of objects or use XML in order to format out GUI according to our preferences. In this section, the focus will be on XML and how GUI builder is used in order to generate it.

9.2 Using XML to set the layout:

The main advantage of choosing to use XML over formatting the GUI programmatically is that we still have the ability to modify the layout and state of objects at run time, yet we separate the main code from the UI layout.

Declaring the UI in XML also means that the code can be reused or with slight changes multiple views can be created in order to suit different devices or screen resolutions in order to achieve the best possible optimization. XML driven UI's are also much easier to edit and format as they allow the user to visualize the structure and make necessary changes using a graphical interface which generates the appropriate cod. It also provides the ability to replicate properties of a single object so that other objects so that consistency can be easily achieved amongst them when designing the UI.

The XML code is generated through the use of the inbuilt GUI builder which allows the user to drag and drop objects onto the View in order to create a GUI. Objects can easily be positioned and moved around whilst setting up various preferences is only click away and can be set up very easily. The XML code is automatically generated for the user so that no coding is required in order to set up the GUI. Therefore it's very easy to create, modify and reuse the View specification which has been constructed using the UI builder to generate XML code.

9.3 View Group Layouts

The android framework comes with a series of layouts and views which are prewritten, they come as a subclass of ViewGroup, and some of the main ones include:

Linear Layout:
This layout organises the children elements in vertical rows

Relative Layout:
This layout organises the children elements in relative position to other elements or in relation to the container that it resides in

Absolute Layout:
This layout organises the children in exact positions on the view, the positions are defined by X and Y coordinates of the object

List View:
This view groups the children and allows them to be viewed in the scrollable list

Grid View:
This view groups the children and allows them to be displayed in a scrollable grid

9.4 ID

Each object can be uniquely identified with by an ID which can be assigned to it. IDs are strings, but when the program gets compiled they get changed into integers. The ID can be used in order to refer to individual objects in order to manipulate them programmatically after we create an instance of them. We refer to object through the use of findViewById() method.

In order to declare a new ID we have to use a "+" sign, which basically means it will be added to the resources directory, "R.java" by default. After that they can just be referenced without the "+" sign.

Declaring and adding a new object in XML:

```
android:id="@+id/my_button"
```

Creating an instance:

```
Button myButton = (Button) findViewById(R.id.my_button);
```

9.5 Attributes

Attributes are used to define properties of different objects; they are different for various objects. For instance, we can create a button object, in order to define its properties we use attributes such as "*text*", "*layout_width*" or "*layout_height*".

"*id*" is an attribute that is available to all kinds of objects.

Before we refer to an attribute we use the "*android:*" tag to refer to the library. An example of an object reference with attributes:

```
<Button
     android:id="@+id/button_id"
     android:layout_height="wrap_content"
     android:layout_width="wrap_content"
     android:text="@string/self_destruct"
/>
```

Chapter 10: **Special considerations for mobile applications**

10.1 Devices

Android is an operating system which supports a plethora of different devices which differentiate in many different ways. These devices have to be taken into consideration when developing mobile applications for this platform to make sure that the application is compatible to the devices it has been designed to run on. The devices will come equipped with a variation of hardware configurations which will have an impact on the processing capabilities as well as usability, the devices will come equipped with different screen sizes which have to be accommodated for so that the applications can adapt to the various devices it will be ran on.

The android developer site states that as a developer *"you do need to consider whether your app is compatible with each potential device configuration. Because Android runs on a wide range of device configurations, some features are not available on all devices. For example, some devices may not include a compass sensor. If your app's core functionality requires the use of a compass sensor, then your app is compatible only with devices that include a compass sensor." [21]*

10.1.1 Features / Hardware

Different devices will come equipped with various hardware configurations which will impact the available functionality to a given device. For example, following the example of a compass sensor mentioned earlier, if a device does not contain a compass sensor, that app functionality won't be available. Android implements feature IDs which allow the developer to specify which hardware or software features may not be available on certain devices. The way that is done is through declaring the <uses-feature> element in the application's manifest file.

Below we can see an example of that which I have taken from the android developers' website which specifies that a given application utilizes the compass sensor.

```
<manifest ... >
    <uses-feature android:name="android.hardware.sensor.compass"
                  android:required="true" />
    ...
</manifest>
```

If a device does not require some of the functionality that the app requires, the Google Play Store will not allow the user to download it. An alternative to this, if the app's functionality doesn't entirely depend on a given feature is just to disable it within the app which will limit the functionality of it, but will still allow the users to download the app. This behaviour can be achieved by calling a function which disables the feature on systems which don't have the given functionality. The code which achieves this behaviour can be seen below.

```
PackageManager pm = getPackageManager();
if (!pm.hasSystemFeature(PackageManager.FEATURE_SENSOR_COMPASS)) {
    // This device does not have a compass, turn off the compass feature
    disableCompassFeature();
}
```

10.1.2 Platform versions

Devices might also differentiate in OS versions that they run on, the version number much like hardware has impact on what features and functionality is available and whether the app will be compatible with a given device or not.

Android is a system which is backwards compatible, meaning devices which run newer versions of the operating system will be compatible with applications designed for older versions of the OS. However if an application utilizes features which have been released in an API level higher than the one which is available to the device, the application might not be compatible, due to some functionality not being available.

This means that app developers need to take into consideration the API level they use in order to make their applications compatible with the widest range of devices as possible. The aim of the developer is to maximise the potential audience of the application whilst having access to the new functionality implemented in higher APIs.

The minimum version that the application can be ran on is specified in the build.gradle file, it will change depending on the functionality implemented. It is not good practice to use the newest available API version even though it will provide us with the most functionality because it will drastically reduce the number of devices which can run it, it a lot of the cases the developer would not even utilise all the newest features, therefore a lower API would be adequate.

10.1.3 Screen configuration

Devices that run on android come in different shapes and sizes, which directly translate onto the range of screen configurations used. When developing applications, screen configurations are an important factor to consider, applications will display differently on screen with different resolutions or densities.

In order to make sure that applications are best suited for different screen sizes Android allows multiple XML layouts which will be used depending on the detected screen configuration of a given device. It does by default try to be compatible with all screens, however naturally discrepancies will occur across devices.

Sometimes personal phone configurations can impact how the application displays on the same devices which I have found during the initial tests for my application. My game has been tested on two identical phones (Samsung S8), it displayed as intended on my device however my tester’s phone resolution appeared to be lower when running the application, it turned out to be the battery saving mode that he had enabled. After further investigation it turned out that default settings for battery saving don’t affect it, it starts being an issue when the settings get customised to drop the resolution to 1480x720 which is the lowest setting.

This brings us to the next important factor that has to be taken into consideration when developing mobile applications, battery life.

10.1.4 Battery consumption

Battery life is key when speaking on handheld devices; special consideration has to be taken ensuring that the applications are optimised to conserve battery life. App optimisation in order to reduce battery consumption can be achieved in a variety of different ways.

According to Android developers should aim to make their apps “Lazy First”. “*Making your app Lazy First means looking for ways to reduce and optimize operations that are particularly battery-intensive.”* [22]

The idea of making apps “Lazy First” is dissected onto 3 categories which aim to question the application design in order to optimise it to reduce battery consumption.

*"**Reduce:** Are there redundant operations your app can cut out? For example, can it cache downloaded data instead of repeatedly waking up the radio to re-download the data?*

***Defer:** Does an app need to perform an action right away? For example, can it wait until the device is charging before it backs data up to the cloud?*

***Coalesce:** Can work be batched, instead of putting the device into an active state many times?"[22]*

These 3 questions can help developers to consider battery consumption when designing the architecture of their application. However there are not the only ways to achieve less power hungry applications. Optimising our game through the use of good programming conventions in order to reduce processing power required to run them is also an option.

Additionally there are tools designed to help developers by highlighting the areas of the applications which can be further optimised. An example of a tool supplied by Google specifically for Android is "Library Historian".[23]

10.2 Accessibility and Ease of use

Accessibility is an area that has to be considered when working with applications to make sure that they can be used by everyone, that includes people with disabilities. Due to the fact that Android is a system used on many handheld and portable devices it is very likely that it will be used by people with some sort of disability.

"Common disabilities that affect a person's use of an Android device include blindness or low vision, color blindness, deafness or impaired hearing, and restricted motor skills. When you develop apps with accessibility in mind, you make the user experience better, particularly for users with these disabilities." [24] There is a range of techniques which can be used in order to develop applications in a way so that they can assist and make them easier to use especially, however not limited to people with disabilities.

It's all about having a good design to ensure ease of use. A well designed application should be very intuitive to use as well as clear with its functionality. I will cover some good practices some of which are recommended by Android that can be implemented in order to increase the accessibility of an application for people with disabilities as well as making the applications user friendly to all users in general, whether disabled or not.

10.2.1 Content layout / Grouping:

One of the common practices in order to make the applications more users friendly and accessible is making sure the content is laid out correctly. Content of an application should be grouped in such as manner that it's easy to browse and understand. The content should be arranged in a logical way where relevant sections are grouped together.

10.2.2 Navigation:

Navigation is a big aspect of app design; usually users are required to navigate between views or pages as it is unlikely that all of the app's features will fit on the screen all once. A good practice when working on navigation is to make sure it is easy to follow, that means making it consistent across the application and implementing it in a way which is fairly self explanatory and intuitive for the user. Inconsistencies in methods that users navigate their way around the app such as the use of buttons or gestures should be avoided, if a swipe gesture is used to navigate between windows, it should be across all interfaces rather than having a button on one page and a gesture on the other.

10.2.1 Adequate use of colours and contrast:

Just like anything, in order for our use of colours to be effective and clear it has to be implemented correctly. In general a colour palette has to be chosen carefully in order to be readable and appealing. There are tools out there which help developers and designers to select colour combinations that go well together, there is a whole study dedicated to colour theory and psychological reasons for the use of certain colour and colour combinations.

It is good to aim to achieve "Colour harmony" when choosing colour palettes for a design. *"The color harmony is about the arrangement of the colors in design in the most attractive and effective way for users' perception." [25]*

There are various colour schemes which can be used effectively, some of the popular ones include:

- Monochromatic – based on the use of 1 colour and various shades of that colour, results in a very plain, simple yet effectively and aesthetically pleasing colour combination.

- Analogous – is a scheme which uses a group of colours which are next to each other on the colour wheel, usually consists of 3 colours; the most dominant colour which is picked to be either a primary or secondary colour, as well as tertiary colours.

- Complementary – is a scheme which uses colour which are opposite of each other on the colour wheel meaning they are the total opposites. It results in a good contrast and a clear, well readable implementation.

Another important factor which determines the design of an application to achieve aesthetics as well as aim accessibility is an effective use of contrast and choice of the right contrast ratio in order to make the design clear and easily readable.

W3C (World Wide Web Consortium)[26] defines an acceptable contrast radio in order to determine whether colours are different enough from each other in order to be clearly distinguishable. This is especially important for people with impaired vision or colour blindness.

The ratio defined for small text (smaller than 18, 14 for bold text) is 4.5 to 1.

The ratio defined for medium/large text (larger than 18, 14 for bolt text) is 3.0 to 1.

10.2.2 Use of cues:

In order to increase accessibility of the user interface for people with impaired vision it is a good practice to use more than 1 cue for elements such as buttons. Using colour as a single cue to differentiate elements is not sufficient; a good practice is to use a minimum of 2 cues, an additional cue to colour is the use of shape and symbols as recommended by Android.

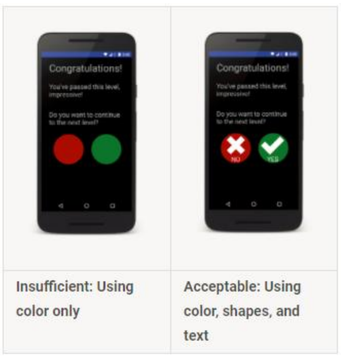

Congratulations!
Congratulations!
NO
YES
Insufficient: Using color only
Acceptable: Using color, shapes, and text

Figure 24. Use of multiple cues to differentiate IU elements [27]

10.2.3 Labelling UI Elements:

Labelling UI Elements is a technique used in order to accommodate disabled people which use screen readers. Since screen readers read out the text that is visible on the screen, we can extend the assortment of cues which are read out by the reader by labelling the elements that would normally not be read out. We can do that by adding a tag in our XML document which contains a description of the object or its functionality this will be picked up by the reader additionally aiding the user.

There is a range of tags that we can use in order to label our elements.

Below we can see a tag used to write a description for a given object which will aid disabled users:

`“android:contentDescription”`

An alternative to using a description would be to use a tag `“android.hint”` or for dynamic elements described programmatically the method `“setHint()”`.

It is advised by the android developer blog that objects such as TextView or other elements which contain text are not labelled as the text within elements is usually automatically picked up by screen readers.

Chapter 11: **Final Program**

11.1 Introduction

In this section of the report I will cover the technical achievements as well as the development process of my project. I will talk a little about the structure of the application, its internals as well as the method used in its creation.

Additionally I will talk about how I used version control in order to keep track of all my documentation as well as deliverables and a little about how I tested my application's functionality before releasing it to the public.

11.2 Method, Development & Technical Achievements

11.2.1 Main Menu

The development of my game started out by creating an activity called "MainActivity" which contains the main menu; once the application is started the content view is set at the MainActivity.

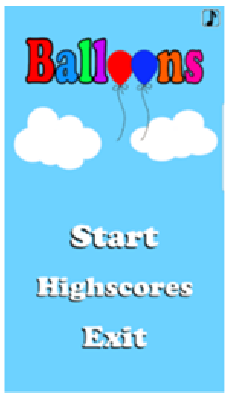


Figure 25. MainActivity layout (The main menu of my game)

The main menu contains all of the buttons which link my views together, the user can navigate to the game screen, the highscores or exit the game. Additionally the main menu features a mute button which mutes the sounds during gameplay.

The activity has been created using an XML layout, the buttons are graphics which I have created using a graphical editing software and implemented into the layout through the use of ImageView elements.

The navigation functionality has been achieved through the use of button listeners:

```
ImageView shop_btn = (ImageView)findViewById(R.id.shop_btn);
      shop_btn.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                Intent  nav_highscores  =  new  Intent(MainActivity.this,
highscores.class);
                startActivity(nav_highscores);
            }
        });
```

11.2.2 Game Loop

The second step to getting the game working was to create a game loop; the game loop handles all the processing as well as draw and update the screen. This is achieved thanks to the two classes, MainThread as well as GamePanel. The MainThread class allows the graphical updates to occur by locking the surfaceHolder which is used by the gameCanvas, the surface has to be locked in order for the updates to the UI to occur, the synchronisation is essential for successful operation.

MainThread creates a new Thread of execution as well as contains the main body of the game loop where the GamePanel draw() and update() method are called, the GamePanel is then responsible for updating and drawing of various game elements through calls to adequate update() and draw() methods which come from within the GamePanel draw() and update() methods.

This behaviour can be seen in the two code snippets below, first one being from MainThread and the second one being a snippet from the GamePanel class.

```
            try {
                gameCanvas = this.surfaceHolder.lockCanvas();
                isSurfaceLocked = true;
                synchronized (surfaceHolder) {
                    this.gamePanel.update();
                    gamePanel.draw(gameCanvas);
                }
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                if (gameCanvas != null) {
                    try {
                        surfaceHolder.unlockCanvasAndPost(gameCanvas);
                        isSurfaceLocked = false;
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
```

```
@Override
    public void draw(Canvas gameCanvas) {
        super.draw(gameCanvas);
        gameCanvas.drawColor(Color.rgb(140, 207, 255));

        if (!MainThread.gameOver){
            balloonManager.draw(gameCanvas);
            drawGamePaint(gameCanvas);
        } else {
            long  timeInGameSeconds  =  (balloonManager.getGameEndTime()  -
balloonManager.getStartTime()) / 1000;
            String                     timeInGame                      =
DateUtils.formatElapsedTime(timeInGameSeconds);
            drawGameOverPaint(gameCanvas, thread.score, timeInGame);
            updateHighscores(thread.score, timeInGame);
        }
    }

    public void update() {
        if (!MainThread.gameOver)
            balloonManager.update();
    }
```

11.2.3 Balloon Object and Balloon Manager

Balloons object class contains all the relevant methods to the balloon object, which includes the draw method, the methods responsible for handling touch events, randomising the position of the balloon object, randomising its colour and speed.

The balloon object class implements a BalloonController Interface which defines the methods which are overridden in the Balloon class.

The balloon object utilises a class called Circle which I have implemented, the Circle object is the basis of the balloon, it defines the coordinates as well as the radius of the balloon which effectively controls its size when the radius is passed as a generateBalloon() method parameter. The x and y coordinates set the centre point of the Circle via the Point object. The use of the circle class also allows me to handle the collisions with the balloon object based on the coordinates and the radius of the circle. We can see the relevant code for the collision detection method below:

```
    public boolean handleTouchEvent(float x, float y) {
        boolean collision = false;
            if(x  >=  getX()  -  getCircle().radius  &&  y  >=  getY()  -
getCircle().radius &&
                    x  <=  getX()  +  getCircle().radius  &&  y  <=  getY()  +
getCircle().radius){
                collision = true;
            }
            return collision;
    }
```

The method returns a Boolean which is then passed onto the Balloon manager to deal with the consequences of the collision (if true) in the respective touch event Handler. See below:

```
public void handleTouchEvent(float x, float y) {
        for (Balloon b : balloons) {
            if (b.handleTouchEvent(x, y) == true) {
                if (b.isPopped() != true) {
                    if (b.getType() == BalloonType.BLACK) {
                        if (!MainThread.muted)

soundController.snd.play(soundController.beep_sound,
                                    1, 1, 1, 0, 1);
                        b.setTypeStandard();
                    } else if (b.getType() == BalloonType.RAINBOW) {
                        b.setPopped(true);
                        MainThread.score = MainThread.score + 20;
                        if (MainThread.lives < 3) {MainThread.lives++;}
                        if (!MainThread.muted)

soundController.snd.play(soundController.rainbow_sound,
                                    1, 1, 1, 0, 1);
                    } else {
                        MainThread.score++;
                        b.setPopped(true);
                        if (!MainThread.muted)

soundController.snd.play(soundController.pop_sound,
                                    1, 1, 1, 0, 1);
                    }
                }
            }
        }
    }
```

All of the balloon objects are generated and stored in the BalloonManager object, which utilises a CopyOnWriteArrayList called balloons. The reason for not using a standard array list is the fact that modifications result in a ConcurrentModificationError otherwise as the array is being iterated over while new balloon objects are being added. The behaviour of my update() method from the BalloonManager can be seen below:

```
    public void update() {
        if (!MainThread.gameOver) {
            int elapsedTime = ((int) (System.currentTimeMillis() - 
startTime) / 1000);
            if (elapsedTime > spawnTime + 1) {
                generateBalloon(context, randX(), randY(), 
MainThread.SCREEN_WIDTH/5, elapsedTime);
                spawnTime++;
            }
            for (Balloon b : balloons) {
                b.updatePosition(b.getSpeed());
                if (b.getY() < 0 - b.getCircle().radius) {
                    if (b.isPopped() != true)
                        MainThread.lives--;
                    b.setPopped(true);
                }
                if (MainThread.lives <= 0) {
                    MainThread.gameOver = true;
                    balloons.clear();
                    balloons = null;
                    gameEndTime = System.currentTimeMillis();
                }
            }
        }
    }
```

The update() method handles the generating of new balloon objects and their addition to the ArrayList. Additionally it iterates through the array and modifies the coordinates of the balloon and handles the

The behaviour for the generateBalloon() method in the BalloonManager is based on the elapsed time, the further the player is in the game the more balloons will be spawned the way that this is achieved is through checking the elapsed time which then decides how many “spawn rounds” will be executed. The spawning takes place every second, the code can be seen below:

```
public void generateBalloon(Context context, int x, int y, int radius, 
int elapsedTime) {
        if (elapsedTime <= 30) { spawnRounds = 1; }
        else if (elapsedTime > 30 && elapsedTime <= 60) { spawnRounds = 
2; }
        else if (elapsedTime > 60) { spawnRounds = 3; }

        if ((elapsedTime % 10 == 0)) {
            balloons.add(new Balloon(context, BalloonType.BLACK, x, y, 
radius, elapsedTime));
        } else if ((elapsedTime % 35 ==0)){
            balloons.add(new Balloon(context, BalloonType.RAINBOW, x, y, 
radius, elapsedTime));
        }
        for (int i = 0; i < spawnRounds; i++) {
            balloons.add(new Balloon(context, BalloonType.STANDARD, x, y, 
radius, elapsedTime));
        }
    }
```

Additionally there are special balloons which will spawn every 10 seconds (Type.BLACK) as well as balloons which spawn every 35 seconds (Type.RAINBOW). The black balloon requires two clicks to be popped, 1 to change its colour and Type to STANDARD and then another one to pop. The RAINBOW balloon replenishes 1 life if under 3 as well as rewarding the player 20 points as opposed to 1 like with a standard balloon.

Balloon Types are defined by the BalloonType enum class:

```
public enum BalloonType {
    STANDARD,
    BLACK,
    RAINBOW
}
```

The balloon colours are assigned appropriately in the Balloon constructor through the use of the setBalloonColour() method which check its type and assigns a corresponding Bitmap.

Additionally to the previously mentioned method of controlling the amount of balloons spawned with increasing elapsed time, the speed of the balloon objects increases.

```
this.speed = 20 + elapsedTime/4 + randSpeed(elapsedTime);
```

It's achieved through a combination of the "base speed" = 20, the additional speed based on elapsed time/4 as well as the random speed returned by the randSpeed which also utilises the elapsed time. This is to create variety between the balloon speeds so they are not all moving at the constant pace, the random speed placed on top of the base speed is capped at 60 seconds to prevent the balloons from moving too fast.

```
    public static int randSpeed(int elapsedTime){
        if(elapsedTime >= 60) {elapsedTime = 60;}
        Random rand = new Random();
        int randomSpeed = rand.nextInt(elapsedTime);
        return randomSpeed;
    }
```

11.2.4 Sound

In order to handle the sound effects which are played when the balloon objects are popped by the player I have created a SoundController class which contains all the logic for setting up a SoundPool, the appropriate sounds are then imported from the project resources.

```
public SoundController(Context context) {
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP){
            AudioAttributes aa = new AudioAttributes.Builder()

.setContentType(AudioAttributes.CONTENT_TYPE_SONIFICATION)
                    .setUsage(AudioAttributes.USAGE_GAME)
                    .build();

            snd = new android.media.SoundPool.Builder()
                    .setMaxStreams(10)
                    .setAudioAttributes(aa)
                    .build();
            pop_sound = snd.load(context, R.raw.pop_sound, 1);
            beep_sound = snd.load(context, R.raw.beep_sound, 1);
            rainbow_sound = snd.load(context, R.raw.rainbow_sound, 1);
        } else {
            snd         =         new         android.media.SoundPool(10,
AudioManager.STREAM_MUSIC, 1);
            pop_sound = snd.load(context, R.raw.pop_sound, 1);
            beep_sound = snd.load(context, R.raw.beep_sound, 1);
            rainbow_sound = snd.load(context, R.raw.rainbow_sound, 1);
        }
    }
```

The sound effects are then called in the touch event handler in the BalloonManager class like so:

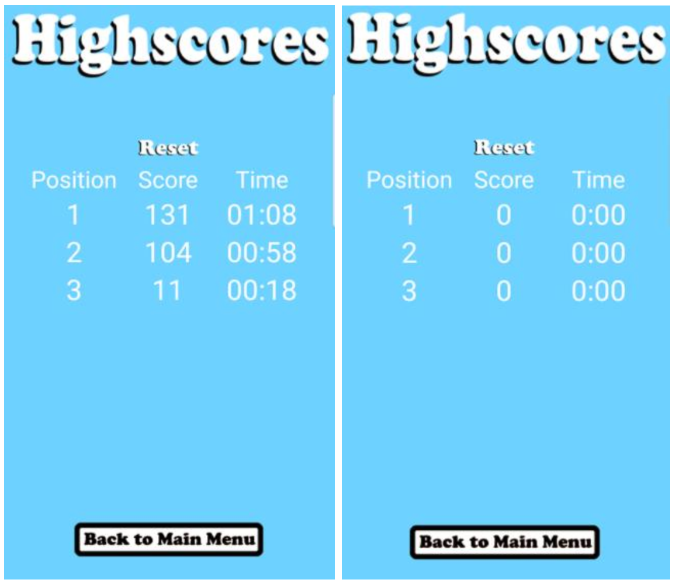
```
soundController.snd.play(soundController.pop_sound,
                                    1, 1, 1, 0, 1);
```

11.2.5 Highscores

I have implemented the Highscores through the use of SharedPreferences, this allows the application to retain the scored when the application is closed. The highscore table gets updated in the highscores onCreate() method as well as when the game over screen.

The gamePanel updates the highscores through the use of the updateHighscores() method when the game over screen is displayed, the method handles all the logic for comparing the current score to the ones saves in the high scores to determine whether the score needs to be updated and updates the table accordingly.

I have also implemented a clear button which clears the highscores table and wipes the shared preferences values to the default.



Position	Score	Time
1	131	01:08
2	104	00:58
3	11	00:18

Position	Score	Time
1	0	0:00
2	0	0:00
3	0	0:00

Figure 26. Highscores user interface

11.2.6 Splash Screen

Additionally I have implemented a splash screen which displays when the game is first launched:

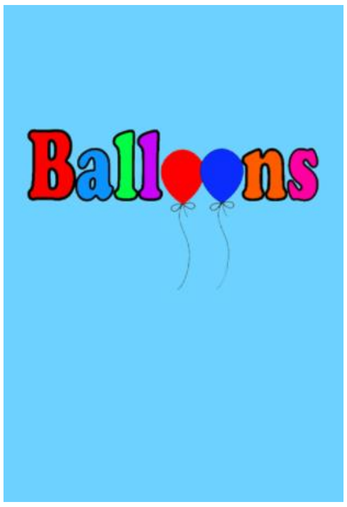


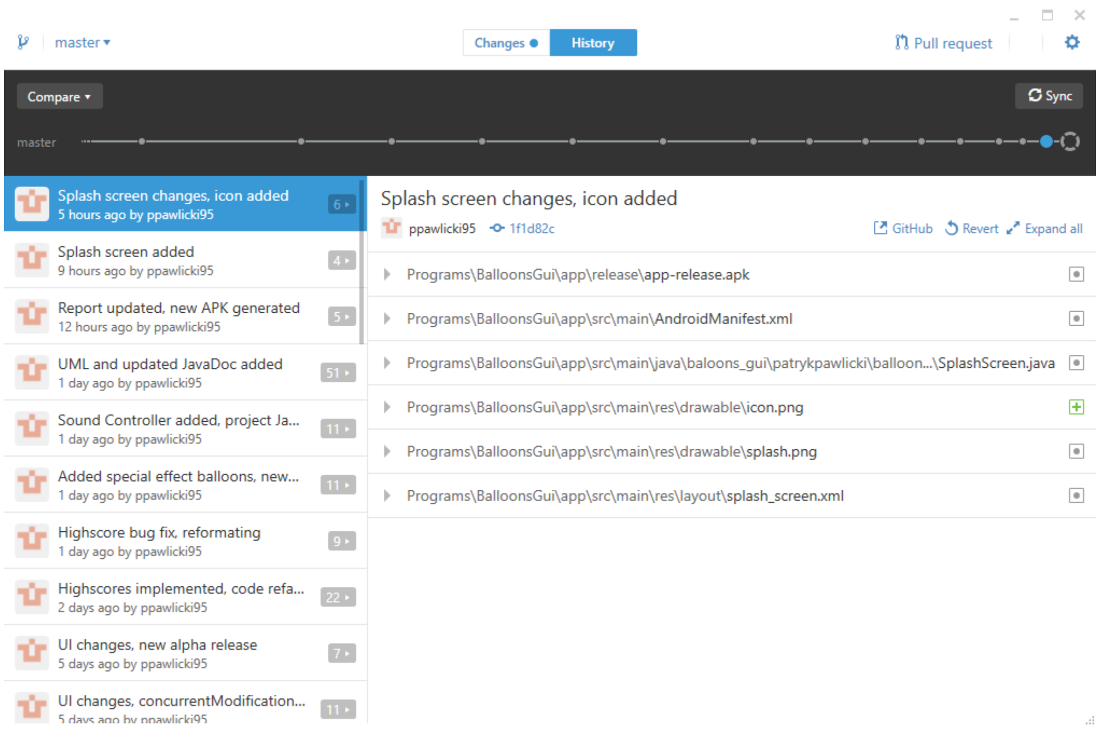
Figure 27. Splash Screen

11.3 Version Control

During my project development I have used version control in order to have all my resources backed up as well as have the ability to revert to any previous version if I need to. This saved me a lot of trouble when things didn't go my way and certain aspects of the game would break.

I have used GitHub as it works well with Android Studio as well as it allowed me to put my entire project structure under version control, that includes not only my project but also all the relevant reports and other deliverables.

My repository can be found here: https://github.com/ppawlicki95/Final-Year-Project



Splash screen changes, icon added — 5 hours ago by ppawlicki95 — 6
Splash screen added — 9 hours ago by ppawlicki95 — 4
Report updated, new APK generated — 12 hours ago by ppawlicki95 — 5
UML and updated JavaDoc added — 1 day ago by ppawlicki95 — 51
Sound Controller added, project Ja... — 1 day ago by ppawlicki95 — 11
Added special effect balloons, new... — 1 day ago by ppawlicki95 — 11
Highscore bug fix, reformating — 1 day ago by ppawlicki95 — 9
Highscores implemented, code refa... — 2 days ago by ppawlicki95 — 22
UI changes, new alpha release — 5 days ago by ppawlicki95 — 7
UI changes, concurrentModification... — 5 days ago by ppawlicki95 — 11

Splash screen changes, icon added
ppawlicki95 1f1d82c GitHub Revert Expand all

Programs\BalloonsGui\app\release\app-release.apk
Programs\BalloonsGui\app\src\main\AndroidManifest.xml
Programs\BalloonsGui\app\src\main\java\baloons_gui\patrykpawlicki\balloon...\SplashScreen.java
Programs\BalloonsGui\app\src\main\res\drawable\icon.png
Programs\BalloonsGui\app\src\main\res\drawable\splash.png
Programs\BalloonsGui\app\src\main\res\layout\splash_screen.xml

Figure 28. View of my GitHub repository

Additionally the GitHub is very useful for keeping track of my progress to assess my performance and see how much code I added or deleted over time.

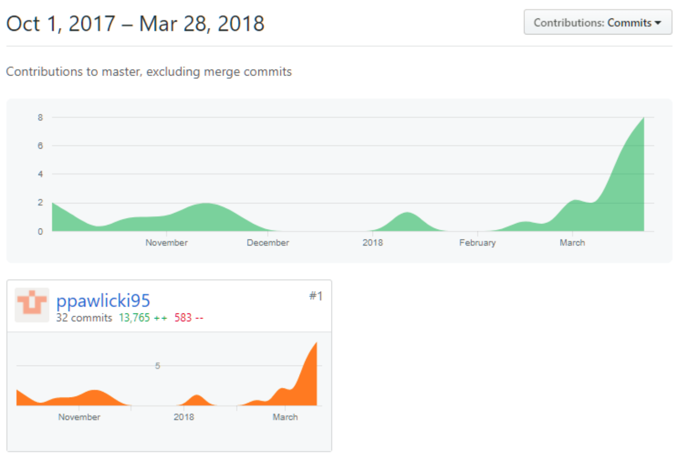


Figure 29. Contributions to my GitHub repository over time

11.4 Testing

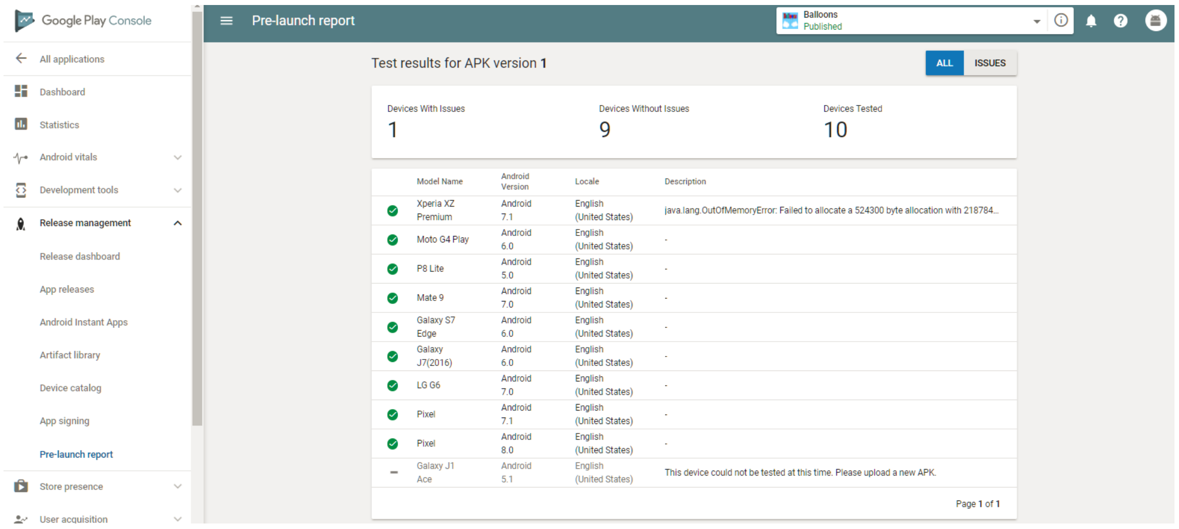
I have carried out extensive testing for my project; this was easily achieved through the use of Android Developer Console which supports Alpha as well as Beta testing. This allowed me to share my application with my testers so that they can test it on their own devices and send me feedback.

Testing done by different users on various different devices which operate on different hardware as well as utilise different screen types allowed me to optimise my application in order to make it compatible. It is much more effective than emulating an environment on my machine.

First series of test has been carried out in the alpha environment; I went through several iterations of the software as I was adding functionality as well as fixing bugs and issues which arose.

On top of individual user feedback, the Development Tools on the Google Play console carry out tests automatically in several different environment and report and application crashes.

Below we can see the test results from my first alpha release:



Test results for APK version **1**

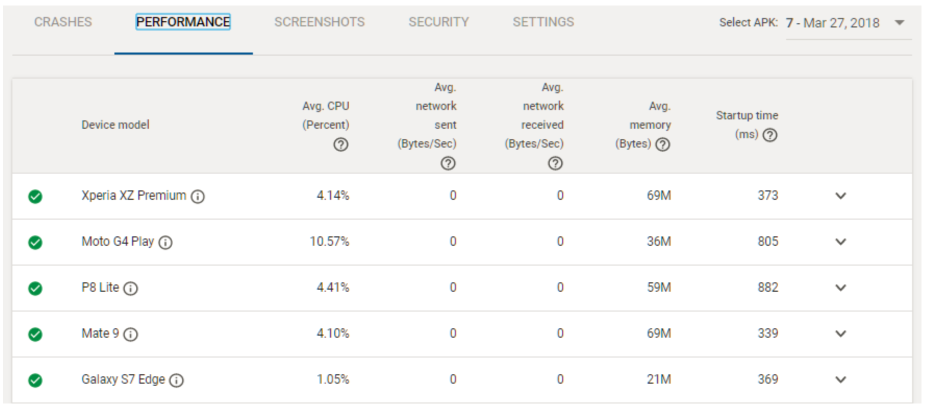
Devices With Issues	Devices Without Issues	Devices Tested
1	9	10

	Model Name	Android Version	Locale	Description
✓	Xperia XZ Premium	Android 7.1	English (United States)	java.lang.OutOfMemoryError: Failed to allocate a 524300 byte allocation with 218784...
✓	Moto G4 Play	Android 6.0	English (United States)	-
✓	P8 Lite	Android 5.0	English (United States)	-
✓	Mate 9	Android 7.0	English (United States)	-
✓	Galaxy S7 Edge	Android 6.0	English (United States)	-
✓	Galaxy J7(2016)	Android 6.0	English (United States)	-
✓	LG G6	Android 7.0	English (United States)	-
✓	Pixel	Android 7.1	English (United States)	-
✓	Pixel	Android 8.0	English (United States)	-
–	Galaxy J1 Ace	Android 5.1	English (United States)	This device could not be tested at this time. Please upload a new APK.

Page **1** of **1**

Figure 30. Tests results of my game APK version 1 on the Google Play Console

Additionally I have been provided with statistics from the various tests, such as CPU usage, any security vulnerabilities found in the software, etc. I have found the CPU usage especially useful as it can be used in order to try to optimise the application to be less process intensive.



Select APK: 7 - Mar 27, 2018

	Device model	Avg. CPU (Percent)	Avg. network sent (Bytes/Sec)	Avg. network received (Bytes/Sec)	Avg. memory (Bytes)	Startup time (ms)
✓	Xperia XZ Premium	4.14%	0	0	69M	373
✓	Moto G4 Play	10.57%	0	0	36M	805
✓	P8 Lite	4.41%	0	0	59M	882
✓	Mate 9	4.10%	0	0	69M	339
✓	Galaxy S7 Edge	1.05%	0	0	21M	369

Figure 31. Performance report on various devices from the Google Play Console Pre-launch testing environment

Chapter 12: **Professional Issues**

12.1 Introduction

In this section of my report I will discuss one of the professional issues that are prominent in the gaming industry, the topic of Plagiarism. My main concern will be the use of code and other game resources which are part of intellectual property and how citation should be used when elements are "borrowed" from another project. I will also cover some of the ethical issues with the use of someone else's intellectual property without adequately citing their work or using it without their consent if the license doesn't allow use and copying of their work.

12.2 History of plagiarism in the game industry

Plagiarism in video games goes as far as the 70s/80s where the first real game studios have started and video games became an increasingly more popular and profitable business. Back then it was a common practice for games to get "cloned", the game mechanics or even the entire game would be cloned and released under a different name. Many of the popular games at the time that most gamers will recognise such as "Pong", "Arkanoid" or "Centepede" were either cloned or themselves were a clone of a different game. Pong for instance, a game released by Atari[28] was a clone of a game released for the Magnavox Odyssey console**Error! Reference source not found.** which featured a Table Tennis game. Atari has been later sued for "Pong" and lost the case for plagiarism. A lot of cases however never reached court and the Pong example is a one of few which actually resulted in the dispute being resolved in the favour of the original creators.

The game industry back then was pretty much unregulated and game studios would steal each other's ideas in order to produce their own games. The main reason for this was the lack of copyright law in place which would adequately cover computer software.

The turning point happened in the mid 80s after the video game industry crash in 1983.[30]**Error! Reference source not found.** A major influence also came from the fact that the copyright law has also been updated in 1980 to protect computer software, the bill was entitled "Computer Software Copyright Act of 1980".[32]

12.3 Game industry now

The game industry is booming, especially the mobile game industry due to the raise of independent and hobbyist game developers for platform such as Android. Before game development was not as popular within independent game developers and was not that much of a concern, however the market is changing and the mobile game industry significantly lowered the barrier of entry for anyone to start developing their own games. It is not so much a concern if games are made for personal pleasure and non commercial use however the mobile game platforms make it very easy to monetise the game released by allowing ads or making them paid to download.

When intellectual property such as ideas, designs or code is stolen or used outside of its license such as commercial use or without the appropriate citation, this becomes an offence which violates the copyright placed upon the relevant piece of work.

All of the code, ideas or resources; such as graphics, sound effects or music has to be adequately referenced giving credit to the original creator, this is of course if the author allowed for the intellectual property to be used by others.

The copyright violations are not a thing of the past, the golden days of "Atari" or "Nintendo" where cloning has been a massive issue. Cloning is still very much alive nowadays especially in the mobile game industry. Very popular and successful games such as "Angry Birds"[33] or "Flappy Bird" (no longer available)[34] have a fair share of clones which base on the identical idea and mechanics, but are dressed up in different graphics.

12.4 Why is plagiarism bad

Plagiarism is bad not only from the legal standpoint as it could result in lawsuits, stealing someone else's intellectual property such as ideas or hard work developing code or graphics is also unethical. We have to take into consideration that it takes a lot of time to develop games, coming up with ideas is also not easy, we could argue that copying other games' ideas, design or mechanics is a creativity killer for the gaming industry, as it restrains the industry and creators from releasing original, never before seen ideas.

Creators who get their work and ideas copied may get discouraged, having spent hours of hard work and get it stolen by someone else and not get any recognition for must be devastating. Rami Ismail form "Vlambeer Games"[35] spoke how he felt when he found out that one of the games he worked on developing called "Radical Fishing" has been stolen*: "Seeing someone copy and release something you worked so hard on to create, something you put so many hours and thoughts and so much research into, something you tweaked for weeks ... it's painful. It stifles your ability be creative because your mind wanders to 'those guys' taking the credit for your hard work. It's an odd feeling."*[36]

12.5 Conclusion

I had to be aware of this during the development of my project too, all of the resources I have used are appropriately referenced and I had made sure that if I did use a resource from another persons' code, I have given credit for their achievements. I had to make sure the resources used were licensed in a way where they are free to use for my application.

Not only would I be breaching the law regarding copyright as well as the rules of the college by using material which is not open source or free to use or without appropriate citation, I would also be doing something unethical by stealing someone else's hard work and creativity.

Chapter 13: **Evaluation**

13.1 Introduction

In this section of the report I will critically evaluate my project as well as myself as a developer. I will talk about areas which have been a success, areas which could be improved as well as cover some of the problems I have encountered while working on my project.

13.2 Project evaluation

Overall I am satisfied with how the project turned out. I have managed to implement everything I have planned to implement; the game runs smoothly and is compatible with all devices above API level 19 that I have had the chance to get it tested on.

There is room for improvement, as well as potential extension which I will cover in this section.

13.2.1 What went well

The choice of API has been a good decision since I managed to make my application available to 90% Android devices whilst not ever felt the need that I missed out on any functionality due to API level being too low.

Graphically it turned out to be well designed as it displays as intended, even the aspects of the game which don't rely on XML layouts, this is due to the fact that I have implemented all the object references, paint as well as graphics relative to the screen size throughout.

The gameplay is engaging, the variable difficulty over time along with the implementation of Highscores make the game fun to play in a competitive manner.

The project is also well documented with comments where necessary as well as JavaDoc throughout. I have also aimed to implement my code in a consistent and easily readable manner, sticking to naming conventions across variables, methods as well as resources.

13.2.2 Areas that could be improved

The main area where I think the game could have been improved is the way the balloons spawn on the screen, in order to achieve randomisation I have used a Random() method however I don't think the spawn coordinates are well randomised. Balloons tend to spawn close to each other a lot and their locations seem a little repetitive, this results in balloons overlapping each other a lot, they spawn with various speeds which decreases the problem however they still tend to overlap on screen a lot more than I would like.

The earlier mentioned pseudo randomisation problem leads me onto the next area which could be improved or changed, when two balloons are stacked on top of each other, a single click will pop both of them. Not really a big problem however the poor randomisation amplifies the problem.

Additionally multi-touch could be a possible extension for popping multiple balloons at once however I have not found it being a massive improvement due to players usually targeting balloons one by one, even when clicking fairly fast with two thumbs. However I do recognise there would be instances where it could help.

One of the things which I think would be “game changing” would be the implementation of decentralised high scores using Google Services which would allow players to compete on a global scale rather than between friends in the real world by comparing high scores.

Additionally the game could be extended to include more features, for instance the idea where I implement “special effect” balloons could be extended in order to implement more variety and possibly some balloons with a negative effect such as taking away lives, score, speeding up balloons or increasing their spawn rate temporarily. There is tons of room for extension to add variety and make the game even more engaging and interesting to play. I feel like over time the game could lose the player’s interest as there is not enough that can surprise and challenge them.

One change in terms of my game design I thought of in order to improve performance, especially critical on less powerful devices would be to change the way balloon objects are used. Currently the balloon objects are added and stored in an arrayList which grows over time meaning the performance is gradually deteriorating as the memory and processing power usage increases.

A possible solution to this problem would be to only use a set amount of objects which cycle through, instead of generating new ones, the existing ones could be sent back to the bottom of the screen after they get popped or leave the screen. This would drastically improve performance and stop it from deteriorating along with elapsed time.

13.3 Self evaluation

This has been my first independent project of such magnitude, not only in terms of software engineering but in terms of commitment, organisation and self motivation. It has been the most involving, demanding and at times frustrating project I have done during my education on this course.

The project has been a great learning experience which has greatly broadened my knowledge in the area of Android game development. I have carried out extensive research in order to put together the application and it has given me the opportunity to get creative and put my ideas forward in order to create my first mobile game.

I had absolutely no experience with Android game development at the start of the project and the whole idea seemed greatly overwhelming at first. Over time I gained confidence and became more efficient at developing the application and implementing new functionality as well as resolving framework related problems.

Looking back at my journey from the start of the development to now having developed a complete, working piece of software I wish I had this knowledge from the start because I would be able to come up with a much better put together piece of work and avoid wasting a lot of time dealing with issues.

I have effectively utilised the tools, such as version control software and the project blog. I have made frequent and fairly consistent updates as well as developed my project incrementally with a quite good distribution of delivery and GitHub commits. However I believe that there is still room for improvement in terms of time management.

13.4 Problems encountered

During my project I have experienced a range of problems and issues; this was mainly due to the framework not willing to cooperate and not my code being incorrect. It has however made me lose a lot of valuable development time trying to pinpoint the source of the problem in my implementation where that wasn’t always the case.

For example one of the major setbacks I had was when I implemented my game loop as well as a way of controlling and modifying the view along with some objects I tried to draw on screen in order to test it. I have found that my objects don’t display as they should, the program didn’t throw any exceptions and no errors were present. I have spent countless hours trying to pinpoint the problem with my code with no avail; I rewrote my code from scratch for the corresponding functionality with no positive effect. Eventually I decided to reinstall Android Studio, also didn’t fix the problem. Eventually I decided to accept my framework along with emulator and that solved the issue.

Emulator in itself has been a challenge to work with at times due to it crashing, not willing to run my application or simply being incredibly slow and laggy. Some of the problems I encountered have been a result of the gradle files missing the appropriate dependencies and implements, however that was manageable.

Overall the Android Framework is quite a beast to deal with especially for a novice, as I gained experience and invested more hours into the development I became more efficient at resolving problems and navigating my way around the framework.

13.5 Conclusion

To conclude this section; I believe that developing my application has been an overall positive experience, I have gained a huge amount of knowledge which will no doubt aid me in future projects, not only android but of any nature. I might consider developing for Android in the future even if it will be purely as a hobby; it’s a really rewarding and fun environment to play with when the developer learns their way around it.

Bibliography

[1] Andy Rubin Interview – Business Week:

https://web.archive.org/web/20110205190729/http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm

[2] Eric Schmidt Quote about Android:

https://www.nytimes.com/2015/05/28/technology/personaltech/a-murky-road-ahead-for-android-despite-market-dominance.html

[3] CyanogenMod: https://github.com/CyanogenMod

[4] Open Handset Alliance: https://www.openhandsetalliance.com/

[5] Android versions image: https://i.ytimg.com/vi/pF_H4jbW41I/maxresdefault.jpg

[6] Project blog: https://pd.cs.rhul.ac.uk/2017-18/

[7] Paint.net software: https://www.getpaint.net/index.html

[8] Android Studio: https://developer.android.com/studio/index.html

[9] IntelliJ: https://www.jetbrains.com/idea/

[10] Sourced from Android Studio – Android Platform/API Version Distribution

[11] Android 4.4 API: https://developer.android.com/about/versions/android-4.4.html

[12] Immersive full-screen mode:
https://developer.android.com/reference/android/view/View.html#SYSTEM_UI_FLAG_IMMERSIVE

[13] Figure 1 – Activity-lifecycle concepts, A simplified illustration of the activity lifecycle:

https://developer.android.com/guide/components/images/activity_lifecycle.png

[14] Figure 2 – Activity state and ejection from memory, Table 1:
https://developer.android.com/guide/components/activities/activity-lifecycle.html

[15] Keeping your app responsive:

https://developer.android.com/training/articles/perf-anr.html

[16] Figure 3 – Keeping your app responsive, Figure 1 – An ANR dialog displayed to the user: https://developer.android.com/training/articles/perf-anr.html

[17] Gamme. E, Helm. R, Johnson. R, Vlissides. J (1994) Page 11
Design Patterns – Elements of Reusable Object-Oriented Software

[18] James Shore, Dependency Injection Demystified:
http://www.jamesshore.com/Blog/Dependency-Injection-Demystified.html

[19] D.A. Cohen, CS2800 Software Engineering, Week 7:
GUI Design and Design Patterns

[20] Source Making – Command Design Pattern;
https://sourcemaking.com/design_patterns/command

[21] Compatibility:

https://developer.android.com/guide/practices/compatibility.html#Versions

[22] Lazy First: https://developer.android.com/topic/performance/power/index.html

[23] Battery Historian Github:

https://github.com/google/battery-historian

[24] Android Accessibility

https://developer.android.com/guide/topics/ui/accessibility/apps.html

[25] Colour theory:

https://uxplanet.org/color-theory-brief-guide-for-designers-76e11c57eaa]

[26] W3C – Acceptable contrast ratio:

https://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-contrast.html

[27] Figure 24 - Use of multiple cues to differentiate IU elements:

https://developer.android.com/guide/topics/ui/accessibility/apps.html

[28] Atari: https://www.atari.com/

[29] Magnavox Odyssey console: https://en.wikipedia.org/wiki/Magnavox_Odyssey

[30] https://www.plagiarismtoday.com/2012/01/30/plagiarism-in-video-games/

[31] Videos on the gaming industry crash 1983:

https://www.youtube.com/watch?v=kv7DJrLAZus

[32] Computer Software Copyright Act of 1980

https://www.govtrack.us/congress/bills/96/hr6934

[33] Angry Birds:

https://play.google.com/store/apps/details?id=com.rovio.angrybirds&hl=en_GB

[34] Flappy Bird: https://en.wikipedia.org/wiki/Flappy_Bird

[35] Vlambeer Games: http://www.vlambeer.com/

[36] Rami Ismail Quote from The Guardian – Clone Wars

https://www.theguardian.com/technology/gamesblog/2011/dec/21/clone-wars-games-industry-plagiarism