Coursework Two

6048 Innovations in Software Development

Joshua Collins – 726681, Michael Grant - 725401

Liverpool John Moores University

Contents

[Figures 2](#_Toc503986825)

[1. Introduction 3](#_Toc503986826)

[2. Android Port Problems 3](#_Toc503986827)

[3. Data Storage Problems 6](#_Toc503986828)

[4. Animation Problems 6](#_Toc503986829)

[5. User Interface Problems 7](#_Toc503986830)

[References 10](#_Toc503986831)

# Figures

[Figure 1 C# switch and case Figure 2 Android switch and case port 5](#_Toc503980799)

[Figure 3 C# player counter Figure 4 Android player counter port 6](#_Toc503980800)

[Figure 5 C# player x rolls: 6](#_Toc503980801)

[Figure 6 Android port player x rolls: 6](#_Toc503980802)

[Figure 7 C# when a double is rolled 7](#_Toc503980803)

[Figure 8 Android solution: when a double is rolled 7](#_Toc503980804)

[Figure 9 Key functionality to allow animation in the RollerActivity 9](#_Toc503980805)

[Figure 10 If statement to identify what colour dice 1 should be 10](#_Toc503980806)

[Figure 11 Dice colour testing 11](#_Toc503980807)

[Figure 12 Play and stop music methods 11](#_Toc503980808)

[Figure 13 Toast notification for landscape/portrait 12](#_Toc503980809)

[Figure 14 Menu page in German Figure 15 Strings files for other languages 12](#_Toc503980810)

# 1. Introduction

This development report will address problems that have been encountered, solutions, and reasons behind the chosen solutions throughout this group coursework. In addition, this report will cite any useful sources to aid the solutions to some problems during the development of this mobile application. This coursework involves developing a mobile application on the topic of a dice rolling game. Using code provided in C#, this has been ported to execute likewise in Android Studio.

The aim of this mobile application is to develop an Android mobile application for a dice game that resembles the desktop dice game provided. However, the team have decided to include several additional features to make this dice app more attractive and fun to play. For example, this included using animation on the dice when being rolled, additional game modes, and some settings to improve the user experience such as music and changing colour of dice.

Upon completing of this mobile application, the team working on this aim to produce a fully functional dice mobile application. This mobile application will enable to the user to use a two-player dice roller, counting the number of rolls and unlimited maximum number of rolls. This page will notify the user which player has rolled and whether they have scored or not, also allowing the user to reset or exit at any point. Furthermore, this dice rolling game will have a similar page to the unlimited dice roller above, but rather than an unlimited dice roller, there is a maximum of 10 rolls in which the result of the game will be displayed. Likewise, in this game mode, the user will again be able to reset the score or exit at any point in the game. With both game modes, they will both use similar animations that will start when the user rolls the dice, and adjust accordingly if the user wishes to use the app landscape.

In addition, this dice application will include several features that may improve the user experience of the mobile application. For example, a splash screen will be included to indicate that the game is loading when selected from the mobile device. Also, there will be a settings menu that will allow the user to toggle music on or off in the game if they desire. Interestingly, the user will can change the colour of the dice for the two game modes described above.

Finally, the mobile application will have aim to deliver a professional look by using icons for the ic\_launcher as opposed to the default android logo for all mobile device screen sizes. Additionally, the game will be dice with a bold and simple text logo being displayed throughout the application.

# 2. Android Port Problems

The first challenge with developing this dice mobile application was porting the code from C# to Android. The team working on this do not have any previous experience with the C# programming language, but fortunately it is like java which the team is efficient with. Porting involves understanding the code and being able to translate this to write in another programming language. Some challenges faced with the port process were tackled easily and some were more difficult to understand and implement.

An important part of the porting was ensuring the Android code is almost identical to the C# code so anyone could see there is little or no difference between. For example, the case statement shown in figure 1 and figure 2 below have the function of taken the random generated dice number (1-6), and producing the output of the corresponding dice image to that number. This example was straight forward to follow as the team are efficient with java. The key difference and problem when porting this section was the translating “PictureBox1.Image = DiceBox1.Image;” to “img1.setImageResource(R.drawable.white1);. This basically says that for the case where the number generated is 1, set the first dice image to the dice with the face of 1.

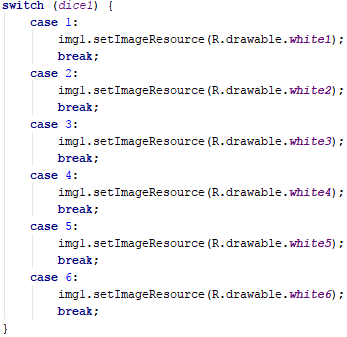
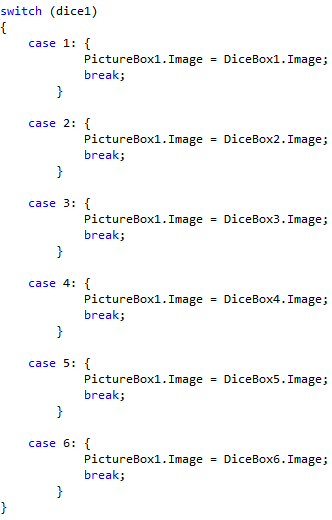


Figure C# switch and case Figure Android switch and case port

Another important section of the C# code is the if statement shown in figure 3 below. This statement can be transferred without any change in Android Studio. The function of this if is to declare which players turn it is to roll the dice allowing two players in the game. A key problem faced with this section of code is the where to place this within the RollerActivity. When working with this, the output of the player was affected whether this was placed inside onAnimationStart() and onAnimationEnd(), but fortunately works effectively before the onAnimationStart() method and within the button.setOnClickListener() method. An additional feature added into the port was to count the number of rolls each player has made. This was resolved by when it is player 1’s turn, player 2’s turn has finished therefore increase player 2’s roll count etc.

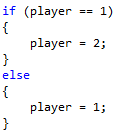
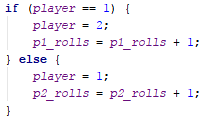
 

Figure C# player counter Figure Android player counter port

Additionally, another problem encountered in the port is shown below where text is updated after each roll to show which player has rolled that score. The C# snippet below is easily readable to show “Player (1 or 2) rolls:”. To translate this into Android, a textview must be created in the app and then when necessary this text view is updated accordingly. The Android port below shows that the same text will be displayed as in the C# code, but “Player” and “rolls” are being read from a strings file. This method is beneficial because it allows users of another language to read the application in another language if required, as Android will translate this.



Figure C# player x rolls:



Figure Android port player x rolls:

The most challenging section of C# code the team faced with porting is illustrated in figure 7 below. This code snippet serves the function of producing an output showing the winner if a dice is rolled, and allowing a sleep timer once 10 doubles have been rolled. The most difficult translation part of this snippet was porting “Label1.Visible = !(Label1.Visible);”. Therefore, the team produced an alternative solution to this snippet shown in figure 8. This statement calculates whether the game has reached 10 rolls, the maximum, and then compares the scores of players 1 and two. From this the result of the game is produced, otherwise continues game play.

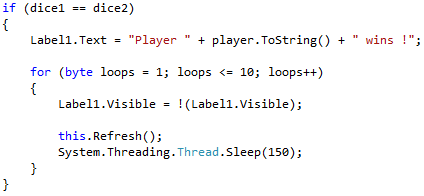


Figure C# when a double is rolled

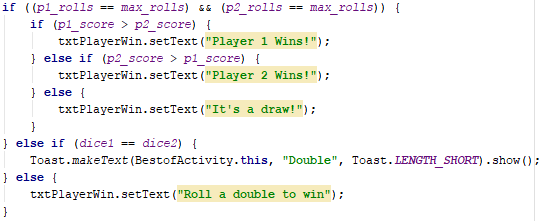


Figure Android solution: when a double is rolled

# 3. Data Storage Problems

To save the high score as a team we first looked to implement SQLite database to record and store the scores, but as of the time constraints and technical difficulty we decided to go for a quicker approach. We then looked about saving the high score to a text file but thought that it was unnecessary to save a text file when we could just use Shared Preferences and as we had some experience in using this method of saving before, so it was quick to implement.

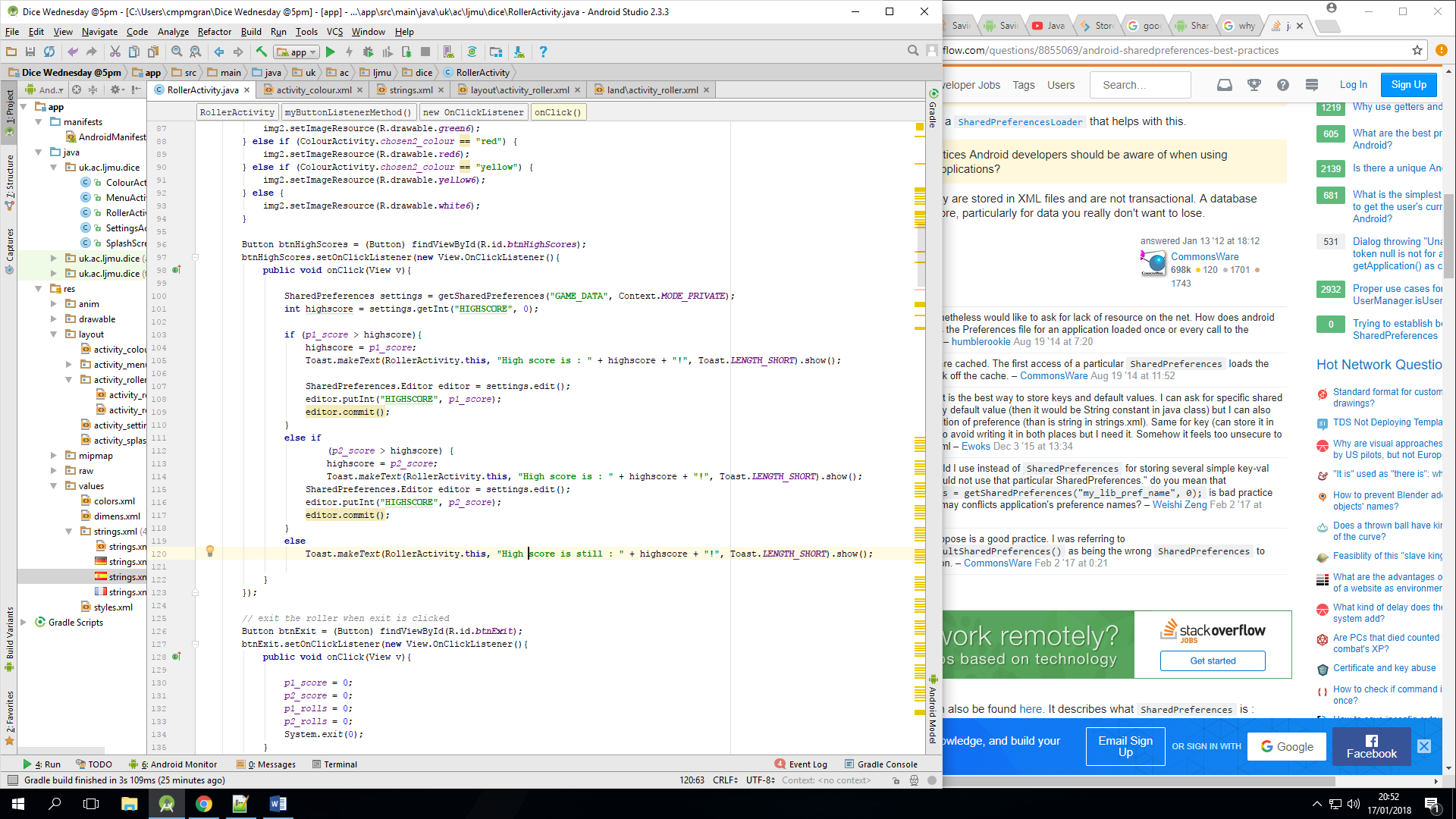


Figure 9 Shared Preferences

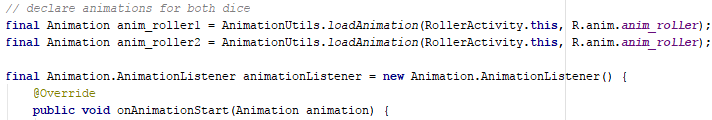
As you can see in the above Figure 9, the High score is updated in the If and else if statements when the player 1 score or player 2 score is greater than the current high score and the final else statement for when the high score is not beat in that game.

# 4. Animation Problems

Implementing animation into the mobile application was a difficult task. Using animation would allow the dice game to look more professional and give a more professional feel. The aims of implementing animation into this application was to produce a spinning dice image upon loading the splash screen. Also, allowing the dice to roll in the game modes to make it clear to the user that the dice are rolling.

The figure below shows the main code used to allow animations to work in the RollerActivity. This includes declaring the animation names for each of the dice in this activity. In addition, there are three methods that allow actions to be taken at different stages of the animation, start, end, and repeat. There were not any code to be executed in the onAnimationRepeat() method as this was not necessary in the situation. Within the onAnimationStart() method, this included setting various objects on the page to become invisible by using x.setAlpha(0);. Furthermore, the onAnimationEnd() method then set the various objects to re-appear using x.setAlpha(1);. Also, this method included the case statement described in 2. Where the dice images would be changed after the dice have rolled.

A problem faced with the animation was positioning code that calculated the player and number of rolls. After repeated testing this was found that calculating the number of rolls works most efficient whilst not included in the animation methods, but in the onClickListener instead. However, when calculating the players turn, this worked most effectively when put in the onAnimationEnd() method. Originally, the player and number of rolls were either inversed or not working at all.







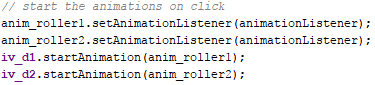


Figure 10 Key functionality to allow animation in the RollerActivity

# 5. User Interface Problems

The user interface is incredibly important when developing a mobile application and if performed well, can increase the popularity of the app and increased chance of it being a success. However, producing an attractive yet professional application can cause many different problems. Firstly, the team decided it would be interesting to try and implement a dice colour change set by the user. This involved taking the original dice and editing each nice number with five additional colours. The problem implementing this was understanding how to take the selected colours and implementing this to the game modes. The first problem was communicating between the ColourActivity and RollerActivity allowing the RollerActivity to fetch the selected\_colour variable. The solution for this was setting the selected\_colour variable global and as static. From this, the solution to load the selected dice colours in the game modes was to use an if statement in each case statement. This solution was long winded as each dice colour had to be scanned before finding which dice number that dice is associated with. The figure below shows the case statement as described in 2. With an if included to identify whether the dice number 1 should be whichever colour.

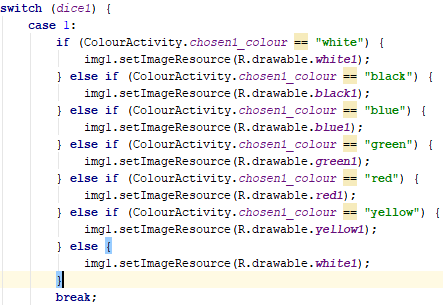


Figure If statement to identify what colour dice 1 should be

Upon testing this case statement, you can see that dice 1 has been set to blue and dice 2 has been set to red in the figure below.

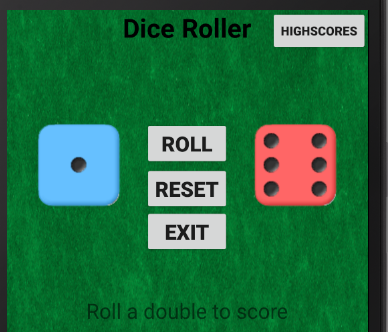


Figure Dice colour testing

Moreover, the team agreed on trying to implement background music to this dice game. The idea of this was to play music immediately when starting the app and using a small button to turn it off if necessary. However, this could be rather irritating to some users therefore starting the app with no music, but the user can toggle on or off in the settings page. Therefore, to implement this effectively, it would be suitable to use a toggle button to allow this. However, there were problems with the code not being to read whether the toggle button isChecked, therefore the music would either play or stop. To solution to this was to replace the toggle button with two simple buttons representing on and off.

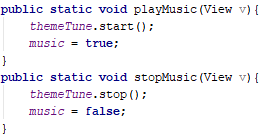


Figure Play and stop music methods

This dice application is compatible when the user wishes to use the device sideways. This was produced by adding a small amount of code and an additional xml file. When the phone is turned portrait or landscape, the code snippet shown in figure 13 shows that a toast message will pop up notifying the user of this. The problem with enabling landscape compatibility was re-positioning the items on that activity to display appropriately. For example, some buttons would automatically not appear on the screen causing the emulator to crash. Another example is when adding a reset button, this must be added into the landscape xml file too and be named the same to allow the full functionality from the desired java file, otherwise again the emulator would crash.

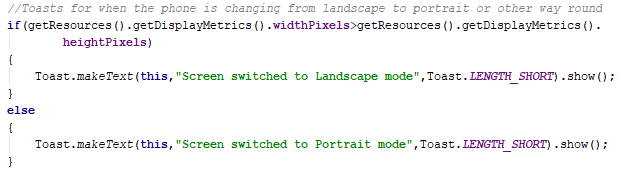


Figure Toast notification for landscape/portrait

Finally, implementing other languages was straight forward to implement by simply including additional strings xml files. The key problem encountered with this was retrieving accurate translations online, as some sources can be unreliable. Also, many languages have different spellings and pronunciations depending on the area of the country you are from. For example, when learning Welsh, there are some words that are spelt different and pronounced different depending on whether you are in North Wales or South Wales, although welsh is not implemented in this application, but this could be the case for French, German, and Spanish. The figures below show the menu page whilst the device is set to German, and the various strings files in this project.

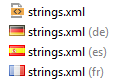
 

Figure Menu page in German Figure Strings files for other languages

# References

Bensound. (2018). *bensound-dance.mp3*. Retrieved January 2018, from ROYALTY FREE MUSIC by BENSOUND: https://www.bensound.com/royalty-free-music/electronica

FANDOM Games Community. (2017). *White Dice.png*. Retrieved January 2018, from Game of Dice Wikia: http://game-of-dice.wikia.com/wiki/File:White\_Dice.png

Outrageous Fortunes Fun Casino Hire. (2015). *poker-table-felt.jpg*. Retrieved January 2018, from Outrageous Fortunes Casino: http://www.outrageousfortunes.co.uk/wp-content/uploads/2015/05/poker-table-felt.jpg