

William Morris - Assignment 1

by William Morris

FILE	11680_WILLIAM_MORRIS_WILLIAM_MORRIS_-_ASSIGNMENT_1_164031_2081313299.DOCX		
TIME SUBMITTED	21-DEC-2018 10:31PM (UTC+0000)	WORD COUNT	5074
SUBMISSION ID	97926395	CHARACTER COUNT	31450

Application Development



DECEMBER 21ST

ASSIGNMENT 1 – DESIGN PATTERNS

Authored by: William Morris



Contents

Introduction	3
Part A	4
Introduction.....	4
Background.....	4
The Difference	5
Comparing Two Patterns.....	6
MVVM Pattern.....	8
Part B.....	11
Introduction.....	11
Exercise 1 – Dice Game	11
Exercise 2 – Music App.....	14
Exercise 3 – Text Based Story Game	16
References.....	19
Appendix	20
Exercise 1 – Dice Game	20
Exercise 2 – Music App.....	24
Exercise 3 – Text Based Story Game	28

Introduction

This report intends to demonstrate a competent understanding of the subjects covered in the Application Development module. It will be separated into two parts, Part A & B. Part A will evaluate the concepts, principles and technologies used in the development of interactive experiences as per the first module learning outcome. Part B will be used to demonstrate a successful selection and deployment of appropriate scripting and programming languages and techniques in the implementation of applications for real-world problem scenarios.

Part A

Introduction

In order to fulfill the learning objectives outlined above I will be investigating and discussing the use of software patterns and their relevance in the development of mobile applications. I will be directly comparing two design patterns and providing examples of their use that I have created myself in order to illustrate their differences.

Background

We must first define what a design pattern is before we discuss its purpose and use. Originally patterns in design were an architectural concept proposed by Christopher Wolfgang Alexander, an influential figure in architecture and design theory. His book "Notes on the Synthesis of Form" was considered "one of the most important contemporary books about the art of design, what it is, and how to go about it." [1] and was considered essential reading for anyone who was entering the field of research within computer science throughout the 1960's and 1970's.

It can be said that during this time there was very little consistency within software design and little structured planning. People were more concerned with creating things rather than how to create things *well*. It wasn't until 1987 that others began to try implement Alexander's theory of design into programming. In the book "Design Patterns: Elements of Reusable Object-Oriented Software" written by the "Gang of Four" it is said that "Designing object-oriented software is hard, and designing reusable object-oriented software is even harder." [2] This shows that design patterns were born from a necessity to improve design and implementation processes in software to improve efficiency and ease of creation.

In the book they also use an analogy that I think helps define a design pattern well. It is as follows, "Novelists and playwrights rarely design their plots from scratch. Instead, they follow patterns like "Tragically Flawed Hero" (Macbeth, Hamlet, etc.) or "The Romantic Novel" (countless romance novels). In the same way, object-oriented designers follow patterns like "represent states with objects" and "decorate objects so

you can easily add/remove features." Once you know the pattern, a lot of design decisions follow automatically." [2]

The Difference

It is important to make a distinction between the different types of patterns which have been developed and are used in application development.

"Simply put, a design pattern is a solution to a recurring problem." [3] This tells us that a design pattern is meant to be a reusable solution which can be altered and applied to tackle commonly occurring problems within application development. They provide more general solutions to general problems in a well-documented fashion. They are also useful to developers as they provide a common naming convention for different parts of an application, making it easier to understand what people are talking about.

Generally Architectural patterns are considered to be higher level patterns than design patterns. While a design pattern deals with the separation of concerns within an application, architectural patterns aren't really concerned with individual problems which an application is trying to solve. They are instead used to tackle more common problems within software engineering itself, like hardware performance limitations and storage restrictions.

Finally, UX design patterns are not concerned with any of the application's back end. UX stands for User Experience meaning these patterns are used when designing user interfaces for applications. When developing an application, it can be easy to focus on the back-end functionality and neglect the front-end UX. As Rex Hartson says in his book "The UX Book", "Computer systems are being worn by people and embedded within appliances, homes, offices, stereos and entertainment systems, vehicles, and roads" [4]. This means good quality UX design is more important than ever with the rapidly changing world of user interfaces in computing. As a result of this UX design patterns must be used to ensure a good outcome for the user. Correct use of UX design patterns also facilitate a more consistent look throughout an application which is generally considered beneficial to user experience.

Comparing Two Patterns

For this section I will be comparing two patterns, MVC and MVP.

MVC

MVC or Model – View – Controller is a design pattern which is concerned with the separation of concerns within the application. The Model contains the core data structure, the View is what is presented to the user and the Controller is used to handle user inputs and data being sent from the view to the model and vice versa. One of the most important rules to remember is that the model represents the data and only the data. It should never depend on either the view or the controller. It is also possible for the view and the controller to be the same object when using MVC.

One of the biggest advantages of MVC is the separation of dependencies from the model. By moving the view dependencies to the controller model classes can be reused with no modification. As an example, I have created a model class called Student which might be used for a school admin application and can be easily reused for another scenario.

```
namespace example.Models
{
    public class Student
    {
        public int StudentId { get; set; }
        public string FirstName { get; set; }
        public string Surname { get; set; }
        public string D.O.B { get; set; }
        public string Address1 { get; set; }
        public string Address2 { get; set; }
        public string Address3 { get; set; }
        public string Postcode { get; set; }
    }
}
```

This class could easily be copied over and by simply changing the name of the class and the first field it could then be used as a part of a company's payroll system.

```
namespace example.Models
{
    public class Employee
    {
        public int EmployeeId { get; set; }
        public string FirstName { get; set; }
        public string Surname { get; set; }
        public string D.O.B { get; set; }
        public string Address1 { get; set; }
        public string Address2 { get; set; }
        public string Address3 { get; set; }
        public string Postcode { get; set; }
    }
}
```

Another great advantage of using MVC is increased efficiency, for a couple of reasons. Firstly, the model returns data without any kind of formatting. This can help prevent code duplication as you can use the same data for multiple different interfaces. Secondly, MVC allows for more agile development as people can be working on separate sections of the application without affecting each other. For example, a C# developer could be working on the back end of a system while a web designer is building the views.

Finally, another advantage off using MVC is the ability to create SEO friendly URLs. Take the two examples I have created below. The first URL is not as SEO friendly as the second one due to the fact that we are using an ID as the route path for the URL.

http://localhost:45125/Home/News_Events/Featured_News/1

In the second URL we are using the name of the object in the model as the route path.

http://localhost:45125/Home/News_Events/Featured_News/william-morris-launches-tech-blog

This means that if a user searches for an article about the launch of my tech blog then the second URL will feature higher on the SEO rankings.

MVP

“Model-view-presenter (MVP) is a derivative of the MVC design pattern which focuses on improving presentation logic” [5]. It was originally conceived in the early 1990’s by a group called Taligent. Taligent was made up of Apple, IBM and HP. Similarly to MVC, MVP is a design pattern which facilitates the separation of concerns between elements within an application. MVP is “one of the most important patterns used for extracting business logic outside of User Interface (UI) elements and by that, enabling unit testing

the UI without the need for using specific UI based testing tools.” and “specifically geared towards a page event model such as ASP.NET.” [6]

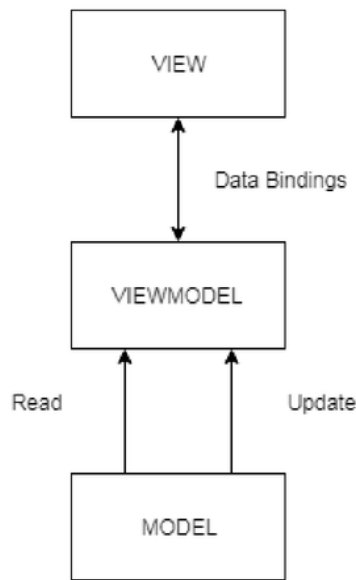
“MVC is introduced for small applications that are easy to manage and have simple navigation. MVP is introduced for test driven development and decoupling of presenter to the model by using passive view.” [7] Passive View is the name that is given to a view which is as dumb as possible. i.e. no behind code. All the middle work is done by the Presenter, similarly to the controller in MVC. Passive View uses the view to expose setter properties which are then set by the Presenter. This has the advantage of true separation of concerns but does come with more work.

MVVM Pattern

In this section of my report I am going to discuss a widely used software architectural pattern called MVVM or Model – View - ViewModel. I will be discussing the purpose of this pattern as well as the benefits and drawbacks of using it.

Purpose

The use of MVVM allows for a true separation of model and view by using the ViewModel to process data in order to keep the view and model simple. When using MVVM, the model should only be used to carry out core application logic while the view should only be used to display the GUI to the user. The ViewModel is what is used to communicate data from the View to the Model and vice versa. When using this pattern, a process called data binding is carried out in the ViewModel where properties from the UI are bound to interfaces which are then read/updated by the model. This is the biggest thing that separates MVVM design from other patterns which also support separation of elements such as MVC. I have included a basic diagram of typical data flow in MVVM architecture.



Benefits

Three of the primary benefits of using the MVVM pattern are maintainability, testability and extendability, which, as we know, are essential attributes when designing and developing any kind of software. If your code is not easily maintained or extended, then it will cost you and your business time and money. Requirements for projects are often changed or extended regularly, meaning these characteristics are crucial in developing and releasing a successful product or service. If your code cannot be easily tested, then you will not be able to test and find every bug or issue in your software leading to a poorer end user experience, and often a bad time for the QA/tester.

Because MVVM allows a truly clean separation of views and models it is much easier to maintain. This is because a developer is able to update code within classes without having to worry as much about affecting other parts of the program or needing to update code in other areas as well. The same can be said for testability. When implemented correctly, MVVM provides a separation of code which carries out core logic from internal and external dependencies, which exist in the ViewModel. This means that carrying out unit tests is much easier as you can more easily test only the specific logic which needs to be tested.

Further, MVVM pattern software is easy to extend for many of the same reasons discussed for maintainability and testability. Due to the nature of the separation of the code it also means that many elements within the Model can be reused for other projects/software as they should be more generic than code which does not separate the logic from the dependencies. For example, if a new project is started which requires a sign-up form to be filled out then it could be possible to retrieve code from a previous project, which carries out much of the same functionality, and then extend it to fit the new project's specific needs.

Drawbacks

While MVVM has seen it's fair share of use, just like any other design pattern it is entirely situational and not every problem can be effectively tackled using the same approach. With that said, there are some drawbacks. Many people consider MVVM architecture to be excessively complicated if you are only making a simple program. Sometimes it just isn't all that necessary to have such a complete separation of dependencies. Conversely, if you are working with a large amount of complicated data then creating the data binding in the ViewModel can become very complex as well as making debugging harder due to how complicated the data bindings can be.

Conclusion

In conclusion MVVM is a powerful design pattern that works well when you need to separate the graphical interface from the application logic. Some issues can arise when working with larger datasets as this can mean data binding becomes very complicated, very quickly and it is often considered overkill when developing very simple applications.

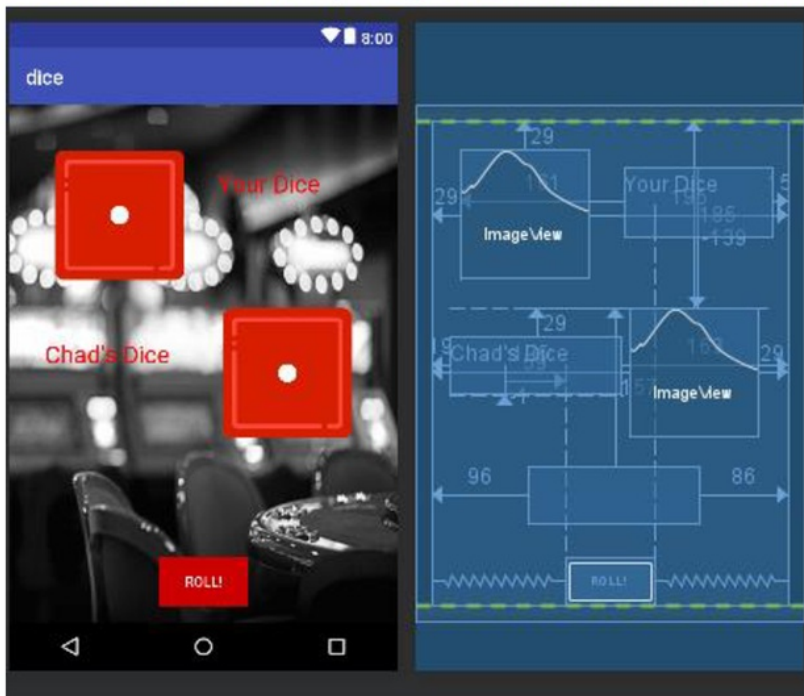
Part B

Introduction

In this section I will be providing evidence of the completion and testing of my applications created for Part B of the assignment. I will be providing my testing below as well as commented code for all applications in the appendix of the document. Testing carried out will be in the form of White Box Testing as I will be carrying out the tests myself. Accompanying screenshots for the testing will be included in the appendix.

Exercise 1 – Dice Game

This application is a simple 'Las Vegas Dice Game' in which the user is presented with a view which contains two dice, one assigned to the player and one assigned to the NPC, as well as a button for rolling the dice. When the player hits "ROLL" the two dice are rolled, and a point given to the player with the highest number. No points are awarded for a draw. Pictured below is a screenshot of the app being played as well as a screenshot of the activity design view from within Android Studio.





Testing

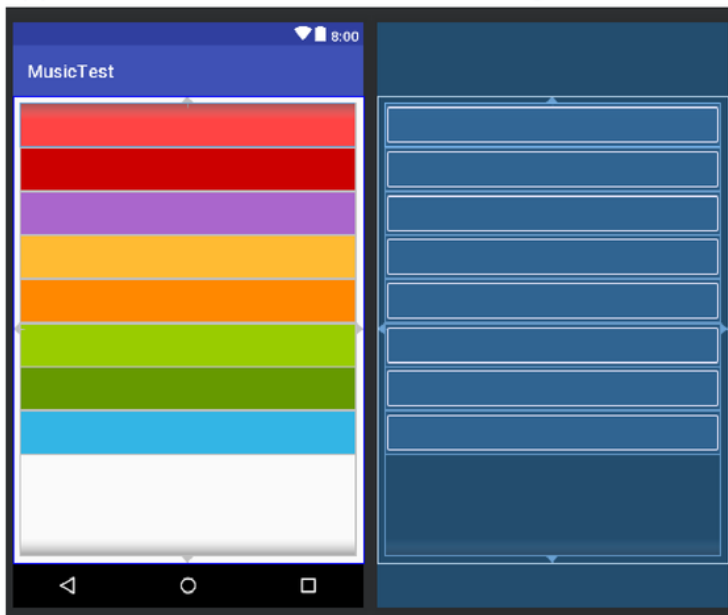
Test	Method	Expected Outcome	Outcome
Build Project	Test will be completed by building the project in Android Studio	Project should build with no errors	<i>Dice fig 1.</i> Project built successfully with no errors.
Launch App	App will be launched by tethering my phone to my PC and running the project	App should launch and user should see main activity	<i>Dice fig 2.</i> App launches successfully on each attempt
Roll Dice	Roll button will be clicked	When rolled the dice should display	<i>Dice fig 3.</i> Dice rolls successfully.

		two new values, one for each player	
Score Increments	Roll button will be clicked multiple times to make sure both scores increment, and a draw does not affect the scores.	When a round is won the winner should get +1 added to their score	<i>Dice fig 4.</i> Score is added to chad because he won. <i>Dice fig 5.</i> Score is added to player because they won. While the score successfully incremented once, each time the roll button is pressed the scores reset to 0. From investigating this I have learnt that Java passes arguments by values and not by reference. This means that when I pass the score values in they are only updated within the calculateScore() method. To solve this, I would need to change to code to return the two score values, possibly by separating the increment method

			into separate ones for each player.
Game is won	Game will be played until a winner is decided.	When a player reaches 3 points the app should display the winner	Dice <i>fig 6 & 7</i> . Due to the issues in the test above I was only able to test by hardcoding the score variables to 2 so that when someone scores, they instantly win.

Exercise 2 – Music App

This application is a launchpad emulator in the form of a xylophone app. Sounds are assigned to buttons which are then played on user input. Below is a screenshot of the application in action as well as the design view for the application.



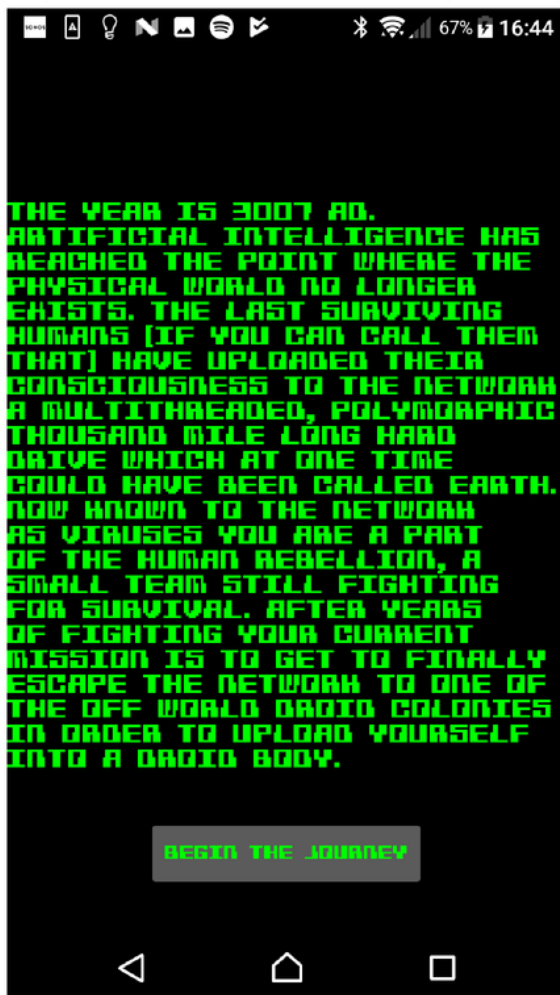


Test	Method	Expected Outcome	Outcome
Build Project	Test will be completed by building the project in Android Studio	Project should build with no errors	<i>Music fig 1.</i> Project built successfully with no errors.
Launch App	App will be launched by tethering my phone to my PC and running the project	App should launch and user should see main activity	<i>Music fig 2.</i> App launches successfully on each attempt
Play Notes	All notes will be clicked individually	All buttons should play corresponding notes	All notes play successfully

Play Notes Simultaneously	Multiple notes pressed at once	Up to 8 notes should be played together	Multi note playing operates correctly
------------------------------	-----------------------------------	---	--

Exercise 3 – Text Based Story Game

Finally, this application is a text-based story game in which the user is presented with an introductory story text. Once the user clicks the begin button, they are given further story and then two options. Depending on which choices they make they will be taken down a different route in the story. Below are some screenshots from the game being played as well as the main activity in design view.





Testing

Test	Method	Expected Outcome	Outcome
Build Project	Test will be completed by building the project in Android Studio	Project should build with no errors	<i>Text Game fig 1.</i> Project built successfully with no errors.
Launch App	App will be launched by tethering my phone to my PC and running the project	App should launch and user should see main activity	<i>Text Game fig 2.</i> App launches successfully on each attempt

Music Plays	App will be launched to main menu	Music should start playing and stop when app is closed	Music plays and stops successfully
Start Game	Begin the journey button will be clicked	Music should stop and player should be taken to the first story index in main activity	<i>Text Game fig 3.</i> Working as expected, story is initiated
Progress through story	Different choices will be made to progress the story	The next story message displayed should carry on from the decision made by the user	For the most part the story progress is working but some of the options jump to incorrect points in the story.

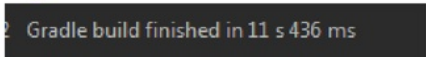
References

- [1] magazine, Industrial Design, 1964.
- [2] E. G. R. H. R. J. J. V. The Gang of Four, Design Patterns: Elements of Reusable Object-Oriented Software, United States: Addison-Wesley, 1994.
- [3] J. A. M. D. E. A. Andre DeHon ´, "Design Patterns for Reconfigurable Computing," *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004.
- [4] R. Hartson, The UX Book: Process and Guidelines for Ensuring a Quality User Experience, Google Books, 2012.
- [5] A. Osmani, Learning JavaScript Design Patterns, Sebastopol : O'Reilly, 2012.
- [6] P. E. Al, "Model View Presenter Design Pattern," *Journal of Computer Science and Control Systems*, vol. 3, no. 1, pp. 173-176, 2010.
- [7] M. R. J. Quresh, "A COMPARISON OF MODEL VIEW CONTROLLER AND MODEL," *Science International - Lahore*, vol. 25, no. 1, pp. 7-9, 7 9 2013.

Appendix

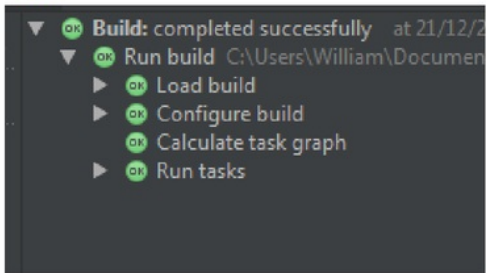
Exercise 1 – Dice Game

Fig 1



Gradle build finished in 11 s 436 ms

Fig 2



```
▼ ok Build: completed successfully at 21/12/2020 19:37:37
  ▼ ok Run build C:\Users\William\Documents\DiceGame\DiceGame
    ▶ ok Load build
    ▶ ok Configure build
    ▶ ok Calculate task graph
    ▶ ok Run tasks
```

Fig 3 & 4



Fig 5

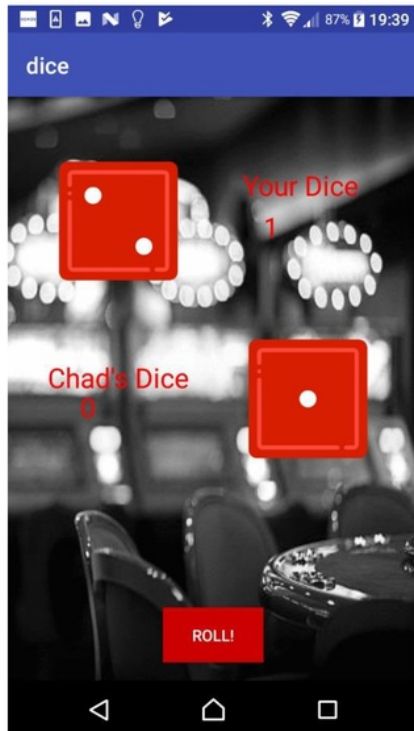


Fig 6

```
public static int intPlayerScore = 2;  
public static int intNPCScore = 2;
```

Fig 7



Fig 8



CODE

```
package com.example.william.dice;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import java.util.Random;

public class MainActivity extends AppCompatActivity { //the class used for main activity

    //declarations of public variables which will be used to calculate scores and the
    //winner, public so that they can be passed as arguments into my methods
    public static int intPlayerScore = 0; //scores declared as ints due to only needing to
    count each win
    public static int intNPCScore = 0;
    public static String strWinner = new String(); //string declared to hold name of winner

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```

        setContentView(R.layout.activity_main); //sets the view layout to activity main

        final ImageView playerDice = findViewById(R.id.playerDice); //assigning imageview
variables to the image views on the main activity so that the image can be updated with
each roll
        final ImageView npcDice = findViewById(R.id.npcDice);

        Button myButton = findViewById(R.id.btnRoll); //assigning a button variable to the
roll button

        final int[] playerDiceArray = { //this is an an int array used to hold the IDs of
each of the six sides of the dice for the player
            R.drawable.dice1,
            R.drawable.dice2,
            R.drawable.dice3,
            R.drawable.dice4,
            R.drawable.dice5,
            R.drawable.dice6
        };

        final int[] npcDiceArray = { //this is an an int array used to hold the IDs of each
of the six sides of the dice for the npc
            R.drawable.dice1,
            R.drawable.dice2,
            R.drawable.dice3,
            R.drawable.dice4,
            R.drawable.dice5,
            R.drawable.dice6
        };

        TextView strWinnerBox = (TextView) findViewById(R.id.strWinner); //finds the text
view used to display the winner.
        strWinnerBox.setText(strWinner); //then we set the text to the value string variable
which will be used to display the winner

        myButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { //this is the on click listener for the roll
button
                roll(playerDiceArray, npcDiceArray, npcDice, playerDice); //when the button
is clicked the roll method is called in order to begin the game, four arguments that are
needed are passed.
            }
        });
    }

    public void roll(int[] playerDiceArray, int[] npcDiceArray, ImageView npcDice,
ImageView playerDice) { //this is the roll method
        Random playerRandomNumberGenerator = new Random(); //when this method is called it
defines two new random number generators, one for the player and one for the npc which are
used to decide what number has been rolled
        Random npcRandomNumberGenerator = new Random();

        int intPlayerNumber = playerRandomNumberGenerator.nextInt(6); //this line sets the
playerNumber variable to the next random number generated. We are providing a max bound of
6 because our dice only have six sides
        int intNPCNumber = npcRandomNumberGenerator.nextInt(6); //this line sets the
npcNumber variable to the next random number generated. We are again providing a max bound
of 6 because our dice only have six sides

        playerDice.setImageResource(playerDiceArray[intPlayerNumber]); //this line then
takes the random numbers we have just set and uses it to find the corresponding image ID
from the array of dice images which is then used to set the image view to display the
correct image

```

```

        npcDice.setImageResource(npcDiceArray[intNPCNumber]); //this line then takes the
random numbers we have just set and uses it to find the corresponding image ID from the
array of dice images which is then used to set the image view to display the correct image

        calculateScore(intPlayerScore, intNPCScore, intPlayerNumber, intNPCNumber); //once
we reach the end of the method the we pass in the required arguments and call the method to
calculate who won the roll
    }

    public void calculateScore(int intPlayerScore, int intNPCScore, int intPlayerNumber,
int intNPCNumber){ //this is the calculateScore method which is called at the end of roll()

        if (intPlayerNumber > intNPCNumber) //here we have some very basic conditional
logic to decide who won the roll
        {
            intPlayerScore++; //if the player had a higher number then increment their score
        }
        else if (intPlayerNumber < intNPCNumber)
        {
            intNPCScore++; //if the npc had a higher number than the player then increment
their score
        }

        TextView playerScore = (TextView) findViewById(R.id.playerScore); //next we find the
text view used to hold the player score
        playerScore.setText(String.valueOf(intPlayerScore)); //then we set the text
to the value of the integer variable which holds the score

        TextView npcScore = (TextView) findViewById(R.id.npcScore); //then we do the same
for the npc score
        npcScore.setText(String.valueOf(intNPCScore));

        win(intNPCScore, intPlayerScore); //finally this method calls the win method which
is used to see if either player has won the game yet.
    }

    private void win(int intNPCScore, int intPlayerScore) { //this is the win method called
by calculateScore()
        if (intPlayerScore == 3) //more simple logic to decide if either player has won.
This game is best of five so once someone reaches three the game is over
        {
            strWinner = "You Win!!!!"; //set the winner string to display player as winner
        }
        else if (intNPCScore == 3)
        {
            strWinner = "Chad Wins!!!!"; //set winner string to display npc as winner
        }
    }
}

```

Exercise 2 – Music App

Fig 1

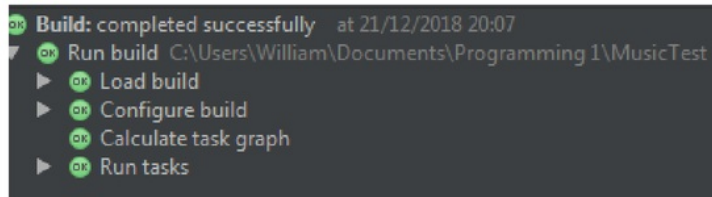


Fig 2



```
package com.example.william.musictest;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.media.AudioManager;
import android.media.SoundPool;
import android.view.View;

public class MainActivity extends AppCompatActivity {
    //instantiate the sound pool
    SoundPool mySoundPool;

    //this declares int to store all of the musical notes
    private int intSoundID1;
    private int intSoundID2;
    private int intSoundID3;
    private int intSoundID4;
    private int intSoundID5;
    private int intSoundID6;
    private int intSoundID7;
    private int intSoundID8;

    private final int NR_OF_SIMULTANEOUS_SOUNDS = 8; //this is needed to set the maximum
    number oif noises played at once. it is set to 8 so all notes can be played at once
    private final float LEFT VOLUME = 1.0f; //these two lines set the left and right volume
    for non mono audio. they are set the same so the sound plays equally on both sides
    private final float RIGHT VOLUME = 1.0f;
```

```

private final int NO_LOOP = 0; //this is set to 0 to allow sounds to be looped
private final int PRIORITY = 0;
private final float NORMAL_PLAY_RATE = 1.0f;

@Override
protected void onCreate(Bundle savedInstanceState) { //all code within this class is
executed when the app is started and the main activity is loaded
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main); //sets the content to display the main
activity

    mySoundPool = new SoundPool(NR_OF_SIMULTANEOUS_SOUNDS, AudioManager.STREAM_MUSIC,
0); //here we are assigning a value of data type sound pool to the empty variable using in
parameters to set the properties

    intSoundID1 = mySoundPool.load(getApplicationContext(), R.raw.note1, 1); //these 8
lines are used to assign the notes from the sound pool to the individual integer variables
which are played on button click
    intSoundID2 = mySoundPool.load(getApplicationContext(), R.raw.note2, 1);
    intSoundID3 = mySoundPool.load(getApplicationContext(), R.raw.note3, 1);
    intSoundID4 = mySoundPool.load(getApplicationContext(), R.raw.note4, 1);
    intSoundID5 = mySoundPool.load(getApplicationContext(), R.raw.note5, 1);
    intSoundID6 = mySoundPool.load(getApplicationContext(), R.raw.note6, 1);
    intSoundID7 = mySoundPool.load(getApplicationContext(), R.raw.note7, 1);
    intSoundID8 = mySoundPool.load(getApplicationContext(), R.raw.note8, 1);

}

//each of these classes is used to play the sounds when the on click listener is
triggered by pressing one of the buttons on the activity. In this example the on click is
declared in the xml view for the main activity
public void play1(View v)
{
    mySoundPool.play(intSoundID1, LEFT_VOLUME, RIGHT_VOLUME, PRIORITY, NO_LOOP,
NORMAL_PLAY_RATE); //this line calls the sound pool and starts the play method. The int
parameters that were declared earlier are passed in so that the sound is played in the way
we configured
}

public void play2(View v)
{
    mySoundPool.play(intSoundID2, LEFT_VOLUME, RIGHT_VOLUME, PRIORITY, NO_LOOP,
NORMAL_PLAY_RATE);
}

public void play3(View v)
{
    mySoundPool.play(intSoundID3, LEFT_VOLUME, RIGHT_VOLUME, PRIORITY, NO_LOOP,
NORMAL_PLAY_RATE);
}

public void play4(View v)
{
    mySoundPool.play(intSoundID4, LEFT_VOLUME, RIGHT_VOLUME, PRIORITY, NO_LOOP,
NORMAL_PLAY_RATE);
}

public void play5(View v)
{
    mySoundPool.play(intSoundID5, LEFT_VOLUME, RIGHT_VOLUME, PRIORITY, NO_LOOP,
NORMAL_PLAY_RATE);
}

public void play6(View v)

```

```
    {
        mySoundPool.play(intSoundID6, LEFT_VOLUME, RIGHT_VOLUME, PRIORITY, NO_LOOP,
NORMAL_PLAY_RATE);
    }

    public void play7(View v)
    {
        mySoundPool.play(intSoundID7, LEFT_VOLUME, RIGHT_VOLUME, PRIORITY, NO_LOOP,
NORMAL_PLAY_RATE);
    }

    public void play8(View v)
    {
        mySoundPool.play(intSoundID8, LEFT_VOLUME, RIGHT_VOLUME, PRIORITY, NO_LOOP,
NORMAL_PLAY_RATE);
    }
}
```

Exercise 3 – Text Based Story Game

Fig 1

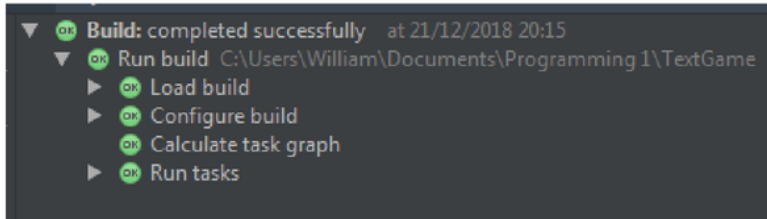


Fig 2

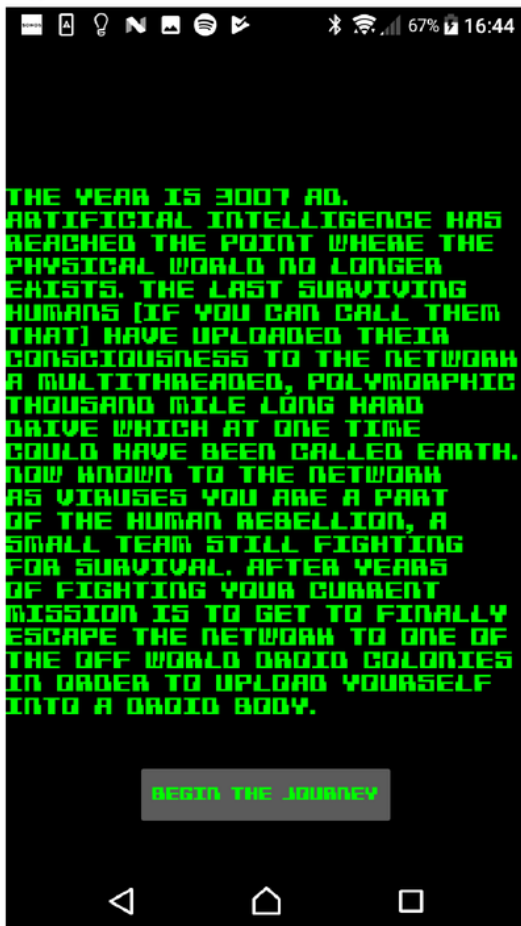


Fig 3



Fig 4

```
package com.example.william.textgame;

import android.content.DialogInterface;
import android.content.Intent;
import android.media.MediaPlayer;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainMenu extends AppCompatActivity {

    public MediaPlayer player;
```

```

    @Override//this class starts the music when the activity is created and an onclick
listener is generated for the begin button
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(com.example.william.textgame.R.layout.activity_main_menu);
        player = MediaPlayer.create(this, com.example.william.textgame.R.raw.appmusic);
        player.setLooping(true);
        player.start();
        Button startButton = (Button) findViewById(R.id.button3); //find the button
        startButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { // on click listener
                Intent intent = new Intent(view.getContext(), MainActivity.class); //create
a new intent to move to main activity
                startActivity(intent); //move ot main activity
            }
        });
    }

    //this class is needed to pause the background music.
    @Override
    protected void onPause() {
        super.onPause();
        player.pause();
    }

    //this class is needed to resume the music when the app is reopened.
    @Override
    protected void onStart() {
        super.onStart();
        player.start();
    }
}

```

```

package com.example.william.textgame;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    //declarations of variables for the controls on the activity
    private Button mTopButton;
    private Button mBottomButton;
    private TextView mTextView;

    //declarations used to hold the IDs of story position and button text
    private int mStoryIndex;
    private int mStory;
    private int mTopButtonIndex;
    private int mBottomButtonIndex;

    private clsStoryProgress[] mStoryBank = new clsStoryProgress[] { //uses the story
progress class to store the indexes of the story points and answers
        new clsStoryProgress(com.example.william.textgame.R.string.A1_Story, 1),
        new clsStoryProgress(com.example.william.textgame.R.string.A1_Ans1, 2),
        new clsStoryProgress(com.example.william.textgame.R.string.A1_Ans2, 3),
        new clsStoryProgress(com.example.william.textgame.R.string.A2_Story, 4),
        new clsStoryProgress(com.example.william.textgame.R.string.A2_Ans1, 5),
    }
}

```

```

        new clsStoryProgress(com.example.william.textgame.R.string.A2_Ans2, 6),
        new clsStoryProgress(com.example.william.textgame.R.string.A3_Story, 7),
        new clsStoryProgress(com.example.william.textgame.R.string.A3_Ans1, 8),
        new clsStoryProgress(com.example.william.textgame.R.string.A3_Ans2, 9),
        new clsStoryProgress(com.example.william.textgame.R.string.A4_End, 10),
        new clsStoryProgress(com.example.william.textgame.R.string.A5_End, 11),
        new clsStoryProgress(com.example.william.textgame.R.string.A6_End, 12)
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(com.example.william.textgame.R.layout.activity_main);

        if (savedInstanceState != null){
            mStoryIndex = savedInstanceState.getInt("StoryIndex");//if the user has played
before then re-initiate the app at the saved state
        }
        else//if they havent played before then start the game att the beginning by setting
the story index and button indexes
        {
            mStoryIndex = 0;
            mTopButtonIndex = 1;
            mBottomButtonIndex = 2;
        }

        mTopButton = (Button)
findViewById(com.example.william.textgame.R.id.button);//these three lines set the text in
the text view and the buttons for choices
        mBottomButton = (Button) findViewById(com.example.william.textgame.R.id.button2);
        mTextView = (TextView) findViewById(com.example.william.textgame.R.id.textView);

        mStory = mStoryBank[mStoryIndex].getStoryID();
        mTextView.setText(mStory);

        mTopButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {//on click listener for the top choice
                checkStoryPath(1);//calls the checkstorypath method and passes in the top
button's index
                UpdateStory();//updates the story
            }
        });

        mBottomButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {//on click listener for the second choice
                checkStoryPath(2);//calls the checkstorypath method and passes in the
bottom button's index
                UpdateStory();//updates the story
            }
        });
    }

    private void checkStoryPath(int userSelection) //this is the logic which is used to
determine the story path depending on which selections are made by the player
    {
        int ChosenAnswer = mStoryBank[mStoryIndex].isAnswer();

        if (userSelection == 1)
        {
            if (mStoryIndex == 0 || mStoryIndex == 3)

```

```

        {
            mStoryIndex = 5;
            mTopButtonIndex = 7;
            mBottomButtonIndex = 8;
        }
        else if (mStoryIndex == 6)
        {
            mStoryIndex = 10;
            End_Of_story();
        }
    }
    else
    {
        if (mStoryIndex == 0)
        {
            mStoryIndex = 2;
            mTopButtonIndex = 4;
            mBottomButtonIndex = 5;
        } else if (mStoryIndex == 3)
        {
            mStoryIndex = 8;
            End_Of_story();
        } else if (mStoryIndex == 6)
        {
            mStoryIndex = 9;
            End_Of_story();
        }
    }
}

@Override
public void onSaveInstanceState(Bundle outState){ //this method saves the progress of
the story when the user exits
    outState.putInt("Story Index", mStoryIndex);
    super.onSaveInstanceState(outState);
}

private void End_Of_story(){ //this method is called when the player reaches the end of
the story.
    mTopButton.setVisibility(View.GONE); //these two lines hide the buttons
    mBottomButton.setVisibility(View.GONE);

    AlertDialog.Builder alert = new AlertDialog.Builder(this); //builds the alert dialog
    alert.setTitle("TO BE CONTINUED"); //sets the alert dialog
    alert.setCancelable(false); //prevents the player from cancelling the dialogue
    alert.setPositiveButton("Close Application", new DialogInterface.OnClickListener()
    {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) { //sets the
positive button on the dialog to close the application.
            System.exit(0);
        }
    });
    alert.show(); //shows the game over alert
}

private void UpdateStory(){ //this class updates the story when the user makes a choice
{
    mStoryIndex = (mStoryIndex + 1) % mStoryBank.length;

    if (mStoryIndex == 0){
        End_Of_story();
    }
}

```

```

        mStory = mStoryBank[mStoryIndex].getStoryID();
        mTextView.setText(mStory);
        mStory = mStoryBank[mTopButtonIndex].getStoryID();
        mTopButton.setText(mStory);
        mStory = mStoryBank[mBottomButtonIndex].getStoryID();
        mBottomButton.setText(mStory);

    }
}

```

```

package com.example.william.textgame;

public class clsStoryProgress { //this class is used to get and return the story path
    depending on which indexes are passed in

    private int mStoryID;
    private int mAnswer;

    public clsStoryProgress(int StoryResourceID, int intAnswer){
        mStoryID = StoryResourceID;
        mAnswer = intAnswer;
    }

    public int getStoryID() {return mStoryID;}

    public void setmStoryID(int StoryID) {mStoryID = StoryID;}

    public int isAnswer() {return mAnswer;}

    public void setmAnswer(int answer) {mAnswer = answer;}

}

```