VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belagavi-590018

Mini Project Report

on

"Aircraft War Game"

submitted in partial fulfillment of the requirement for the award of degree

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE & ENGINEERING

for the academic year

2021-2022

submitted by

Mr. Sandesh PY **1AR19CS046**
Mr. Bhuvan Jain K **1AR19CS011**

Under the guidance of
Mr. Ramesh Babu N
Associate Professor,
Dept. of CSE

AIEMS

BENGALURU

2021-22
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.V.V. Sangha's

AMRUTA INSTITUTE OF ENGINEERING & MANAGEMENT SCIENCES

Bidadi, Bengaluru-562109, Karnataka

BVV Sangha, Bagalkot
AMRUTA INSTITUTE OF ENGINEERING & MANAGEMENT SCIENCES
Approved by AICTE, New Delhi
Recognized by Government of Karnataka & Affiliated to VTU, Belagavi
AIEMS BENGALURU

Department of Computer Science & Engineering

AIEMS

BENGALURU

CERTIFICATE

This is to certify that the mini project report entitled **"Aircraft War Game**" is a bonafide work carried out by

Mr. Sandesh PY **[1AR19CS046]**

Mr. Bhuvan Jain K **[1AR19CS011]**

in partial fulfilment of award of **Bachelor of Engineering** in **Computer Science & Engineering** of the Visvesvaraya Technological University, Belagavi, during the academic year 2021-2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated. The mini project has been approved as it satisfies the academic requirements in respect to MAD Lab(18CSMP68) associated with the degree mentioned.

…… ……..……

Signature of Guide

Mr. Ramesh Babu N
Associate Professor,
Dept. of CSE
AIEMS

Signature of HOD

Dr. M S Patel
Professor & Head,
Dept. of CSE
AIEMS

External

Examiner Name	**Signature**
1. _____	_____
2. _____	_____

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned my effort with success.

We are grateful to our institution, **B.V.V Sangha's Amruta Institute of Engineering & Management Sciences (AIEMS),** with its ideals and inspirations for having provided us with the facilities, which has made this, project a success.

We earnestly thank **Dr. Santosh M Muranal, Principal, AIEMS,** for facilitating academic excellence in the college and providing us with the congenial environment to work in, that helped us in completing this project.

We wish to extend our profound thanks to **Dr. M S Patel, Professor & Head, Department of Computer Science & Engineering, AIEMS,** for giving us the consent to carry out this project.

We would like to express our sincere thanks to our guide **Mr. Ramesh Babu N, Associate Professor, Department of Computer Science & Engineering, AIEMS,** for his immense help during the project and also for his valuable suggestions on the project report preparations, which helped us in the successful completion of the project.

We would like to thank all the faculties of **Computer Science & Engineering Department**, for their valuable advice and support.

We would like to express our sincere thanks to our parents and friends for their support.

Mr. Sandesh PY **[1AR19CS046]**

Mr. Bhuvan Jain K **[1AR19CS011]**

DECLARATION

We, **Mr. Sandesh PY [1AR19CS046]** and **Mr. Bhuvan Jain K [1AR19CS011]** students of VI semester B.E, Department of Computer Science & Engineering, AMRUTA INSTITUTE OF ENGINEERING AND MANAGEMENT SCIENCES, Bengaluru, declare that mini project work entitled "**Aircraft War Game**" has been carried out by us and submitted in partial fulfillment of the course requirements for the award of degree in Bachelor of Engineering in Computer Science & Engineering of Visvesvaraya Technological University, Belgavi during the academic year 2021-2022. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

Place:

Date:

Mr. Sandesh PY **[1AR19CS046]**

Mr. Bhuvan Jain K **[1AR19CS011]**

ABSTRACT

A mobile game is a video game that is typically played on a mobile phone. The term also refers to all games that are played on any portable device, including from mobile phone (feature phone or smartphone). Shooter video games are a subgenre of action video games where the focus is almost entirely on the defeat of the character's enemies using the weapons given to the player. Shooter games test the player's spatial awareness, reflexes, and speed in both isolated single player or multiplayer environments. Vertically scrolling video game are a subgenre of shooters wherein the player may move, up, down, left or right around the screen, typically firing straight forward.

By making Simple Space, I have learned how the video-games can be designed and developed by a single person. This process includes the usage of different tools for graphic design such as android studio and java development tool, as well as the Unity engine for combining them into a seamlessly working game.

Using Android studio, I have designed a variety of objects for this game which include different space-ships made of simple figures, a big and complicated Boss ship and many miscellaneous objects used in level design and user interface, they were used to create sound effects that give the player an audio feedback when its ship gets damaged or an enemy ships explode. With respect to Unity, I have used it to design and develop a single working level with a careful crafted game logic, different particle effects and a complete menu system. For this purpose, I have utilized many different game creation tools that Unity offers in combination with the graphical and sound.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Introduction to Android

Android is an operating system and programming platform developed by Google for mobile phones and other mobile devices, such as tablets. It can run on many different devices from many different manufacturers. Android includes a software development kit (SDK) that helps you write original code and assemble software modules to create apps for Android users. Android also provides a marketplace to distribute apps. Altogether, Android represents an ecosystem for mobile apps.

Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

Improving the user interface, both in terms of functionality and performance. On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 Jelly Bean. Jelly Bean is an incremental update, with the primary aim of the source code for Android is available under free and open-source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

Google sponsored the project at initial stages and in the year 2005, it acquired the whole company. In September 2008, the first Android-powered device launched in the market. Android dominates the mobile OS industry because of the long list of features it provides.

It's user-friendly, has huge community support, provides a greater extent of customization, and a large number of companies build Android-compatible smart phones. As a result, the market observes a sharp increase in the demand for developing Android mobile applications, and with that companies need smart developers with the right skill set. At first, the purpose of Android was thought of as a mobile operating system However, with the advancement of code libraries and its popularity among developers of the divergent domain, Android becomes an absolute set of software for all devices like tablets, wearables, set-top boxes, smart TVs, notebooks, etc.

1.1.1 Why Android?

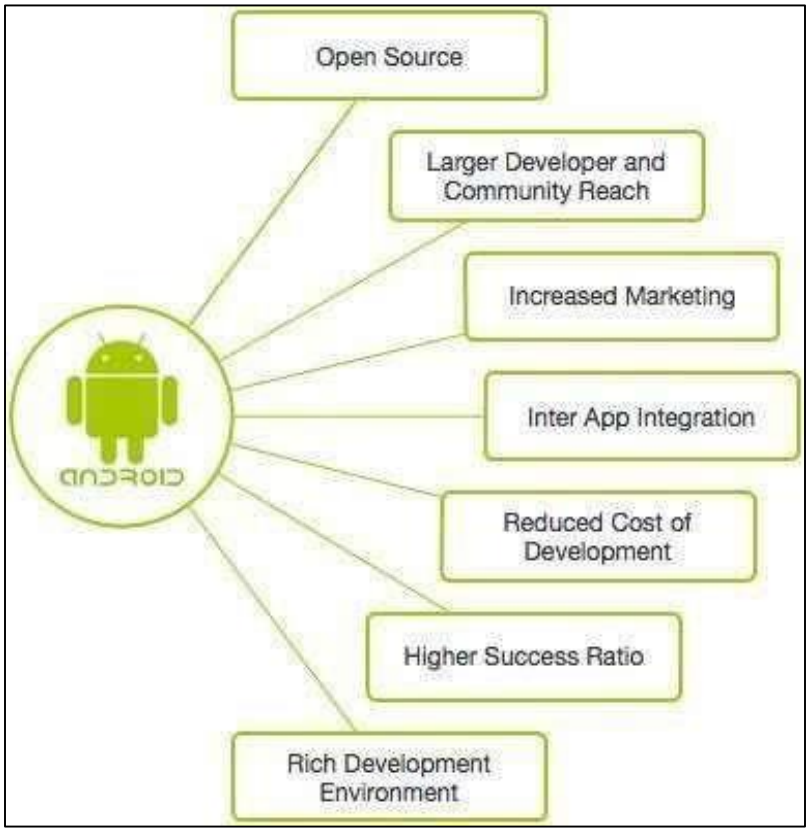


Figure 1.1: Advantages of Android

CG (Computer graphics) started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computer themselves.

Android Applications:

Android applications are usually developed in the Java language using the Android Software Development Kit.

Once developed, Android applications can be packaged easily and sold out either through a store such as Google Play, Slide ME, Opera Mobile Store, Mobbing, F- droid and the Amazon App store.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

This tutorial has been written with an aim to teach you how to develop and package Android application. We will start from environment setup for Android application programming and then drill down to look into various aspects of Android applications

Android Features:

Android is a powerful open-source operating system which provides a lot of great features

those are:

- It’s open-source and we can customize the OS based on our requirements
- It supports connectivity for GSM, CDMA, WIFI, NFC, Bluetooth, etc. For telephony or data transfer.
- It has a wide range of media supports like AVI, MPEG4, etc. to play or record a variety of audio/video and having a different image format like JPEG, PNG, GIF, MP3, etc. to perform playback or recording using camera and microphone.
- It supports a multi-tasking
- It will give a chance to reuse the application components and the replacement of native applications.
- It has support for 2D/3D Graphics
- It has an integrated open-source Web Kit layout-based web browser to support HTML5, CSS3

1.1.2 The challenges of Android app development

While the Android platform provides rich functionality for app development, there are still a number of challenges you need to address, such as:

- Building for a multi-screen world
- Getting performance right
- Keeping your code and your users more secure
- Making sure your app is compatible with older platform versions

1.1.3 Applications

Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc. And third-party applications downloaded from the play store, like chat applications, games etc. will be installed on this layer only. It runs within the android runtime with the help of the classes and services provided by the application framework**.**

1.2 Problem Statement / Aim

To analyze problem(bugs) that user may get by using Aircraft game application. To develop a mobile application of shooting game that can help people in improve their skills. Although shooting games are violent, they can be very efficient in teaching hand-eye coordination and spatial skills. The main objective of this project was to design and develop a 2D shoot 'me up video-game and learn the maximum amount of things possible while doing it. For this purpose, the different graphic elements of the video-game were designed, allowing me to learn how to efficiently use the Android studio software in order to achieve the desired results. In this manner, each design has been done faster and better than the previous one. Similarly, I have learned how to design simple albeit necessary sound effects for the game by combining audio software.

1.3 Scope

Gaming is a highly competitive sector where professionals are needed who have a good balance of creativity, fun and technology. The most important skills required in this field are creativity and passion. Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy shooter games test the player's spatial awareness, reflexes, and speed in both isolated single player or networked multiplayer environments. This Report describes all the requirements for the project. The purpose of this research is to provide a virtual image for the combination of both structured and unstructured information of my project "Space Shooter". This is a single-player strategy game on the Windows platform. The player will progress through levels which require precise manipulation of the environment, though the game Encourages creativity and daring via branching pathways. The episodic structure of the game facilitates the pace of the story. I demonstrate the action flow between inputs, script, display (output). Should be working mainly with story, levels, object, animation, graphics, scripts, game engine facilities.

1.4 Project Description

Top-down run-and-gun games are characterized by an on-screen overhead in a camera angle that shows players and the areas around them from above. Side-scrolling run-and-gun games combine elements of both shoot 'me up and platform games while the player characters move and jump around shooting with various guns and other long-range weapons. Shooting gallery games (also known as "target shooting" games) are a sub-genre of shooters where the player aims at moving targets on a stationary screen. They are distinguished from rail shooters, which move the player through levels on a fixed path, and first-person shooters, which allow player-guided navigation through a three-dimensional space. Shooter video games or shooters are a subgenre of action video games where the focus is almost entirely on the defeat of the character's enemies using the weapons given to the player. Usually these weapons are firearms or some other long-range weapons and can be used in combination with other tools such as grenades for indirect offense, armor for additional defense, or accessories such as telescopic sights to modify the behavior of the weapons. A common

resource found in many shooter games is ammunition, armor or health, or upgrades which augment the player character's weapons.

Table 1.1: Java SE Development Kit version

Java SE Development Kit 18.0.1.1	
Java 18	Java 17
Windows	x64 Compressed Archive
	x64 Installer
	x64 MSI Installer

Table1.2: List of Android Versions

slno	Androids	APIs	Versions	Launched MM/YY
1	Android 2.3	API level 9	Android 2.3, Revision 1	Dec-10
2	Android 2.3.3	API level 10	Android 2.3.3, Revision 1	Feb-11
			Android 2.3.3, Revision 2	Jul-11
3	Android 3.0	API level 11	Android 3.0, Revision 1	Jul-11
			Android 3.0, Revision 2	Jul-11
4	Android 3.1	API level 12	Android 3.1, Revision 1	May-11
			Android 3.1, Revision 2	May-11
			Android 3.1, Revision 3	May-11
5	Android 3.2	API level 13	Android 3.2, Revision 1	Jul-11
6	Android 4.0	API level 14	Android 4.0, Revision 1	Oct-11
			Android 4.0, Revision 2	Dec-11
7	Android 4.0.3	API level 15	Android4.0.3, Revision 1	Dec-11
			Android4.0.3, Revision 2	Jan-12
			Android4.0.3, Revision 3	Mar-12
8	Android 4.1	API level 16	Android 4.1, Revision 1	Jun-12
			Android 4.1, Revision 2	Jul-12
			Android 4.1, Revision 3	Oct-12
9	Android 4.2	API level 17	Android 4.2,Revision 1	Feb-13
			Android 4.2,Revision 2	Feb-13
10	Android 4.3	API level 18	Android 4.3,Revision 1	Jul-13
			Android 4.3,Revision 2	Aug-13
11	Android 4.4	API level 19	Android 4.4,Revision 1	Oct-13
			Android 4.4,Revision 2	Dec-13
12	Android 4.4W	API level 20	Android 4.4W,Revision 1	Jun-14
			Android 4.4W,Revision 2	Oct-14
13	Android 5.0	API level 21	Android 5.0,Revision 1	Oct-14
			Android 5.0,Revision 2	Dec-14
14	Android 5.1	API level 22	Android 5.1,Revision 1	Mar-15
15	Android 6.0	API level 23	Android 6.0,Revision 1	Aug-15
			Android 6.0,Revision 2	Nov-15
16	Android 7.0	API level 24	Android 7.0,Revision 1	Aug-16
17	Android 7.1	API level 25	Android 7.1,Revision 1	Oct-16
			Android 7.1,Revision 2	Nov-16
			Android 7.1,Revision 3	Dec-16
18	Android 8.0	API level 26	Android 8.0,Revision 2	Aug-17
19	Android 8.1	API level 27	Android 8.1,Revision 1	Dec-17
20	Android 9	API level 28	Android 9,Revision 1	Aug-18
21	Android 10	API level 29	Android 10,Revision 5	Jul-20
22	Android 11	API level 30	Android 11	Nov-20
23	Android 12	API levels 31,32	Android 12	Nov-21

CHAPTER 2

HARDWARE AND SOFTWARE REQUIREMENTS

In the development of any software applications, we require same particular system configuration of software and hardware components. This configuration helps in achieving the proper execution.

The various requirements that are essential for this project are specified over here. These requirements have to be fulfilled for successful of the project. The purpose scope along with hardware and software requirements helps for proper execution.

2.1 Functional Requirements

In software engineering and system engineering, a functional requirement defines a functions of a system., In the project it should have provided the switch to turn on and turn off the game music and also the start button which takes into a next user interface. It should allow the players in an offline mode without internet and also to take an in charge of ranking management of the game to maintain the score board database, the game user interface should be containing of ships of player and the enemy ships with a flying animation and also with firing of bullets from both ships.

2.2 Non-Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behavior's. They are contrasted with functional requirements that define specific behavior or functions.

- The user must experience a strong story.
- The game must be fun.
- The game must not crash.
- The game must be accessible for all user segments.

2.3 Hardware Requirements

The projects work with any IBM PC compatible with Intel or ADM processor.

- Processor: Intel Pentium 41.50 Giga HZ
- Memory: 512MB to 1GB
- Operating System: windows 9/10/11
- RAM: 16 GB

2.4 Software Requirements

A software requirements description of a software system to developed. This document enlists enough and necessary requirements that are required for the project development.

- Android studio (2021.2.1 Windows 64-bit)
- Application Programming Interface (API level 30,31,32)
- XML
- Java development tool kit (JDK 18.0.1.1

CHAPTER 3

DESIGN

3.1 SYSTEM DESIGN

This chapter presents detailed explanations about how the graphic and audio elements of this game were designed. It also includes a brief description of different programs and tools used for this purpose, in addition to step by step actions taken when designing graphic and audio elements as well as their uses inside the game

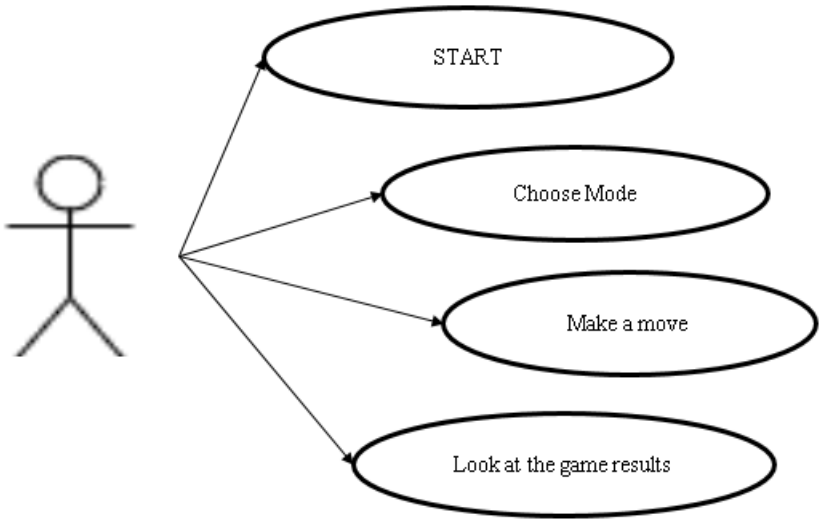


Figure 3.1: Use case diagram of player

Overview:

The project is all based on the galaxy space shooter which is available in the play store, by taking it as a reference we developed the project Aircraft war game. The starting user interface will be containing of button named “START”, It also contain a switch to turn on and turn off the game music. Then by clicking the start button it will move to the next user interface

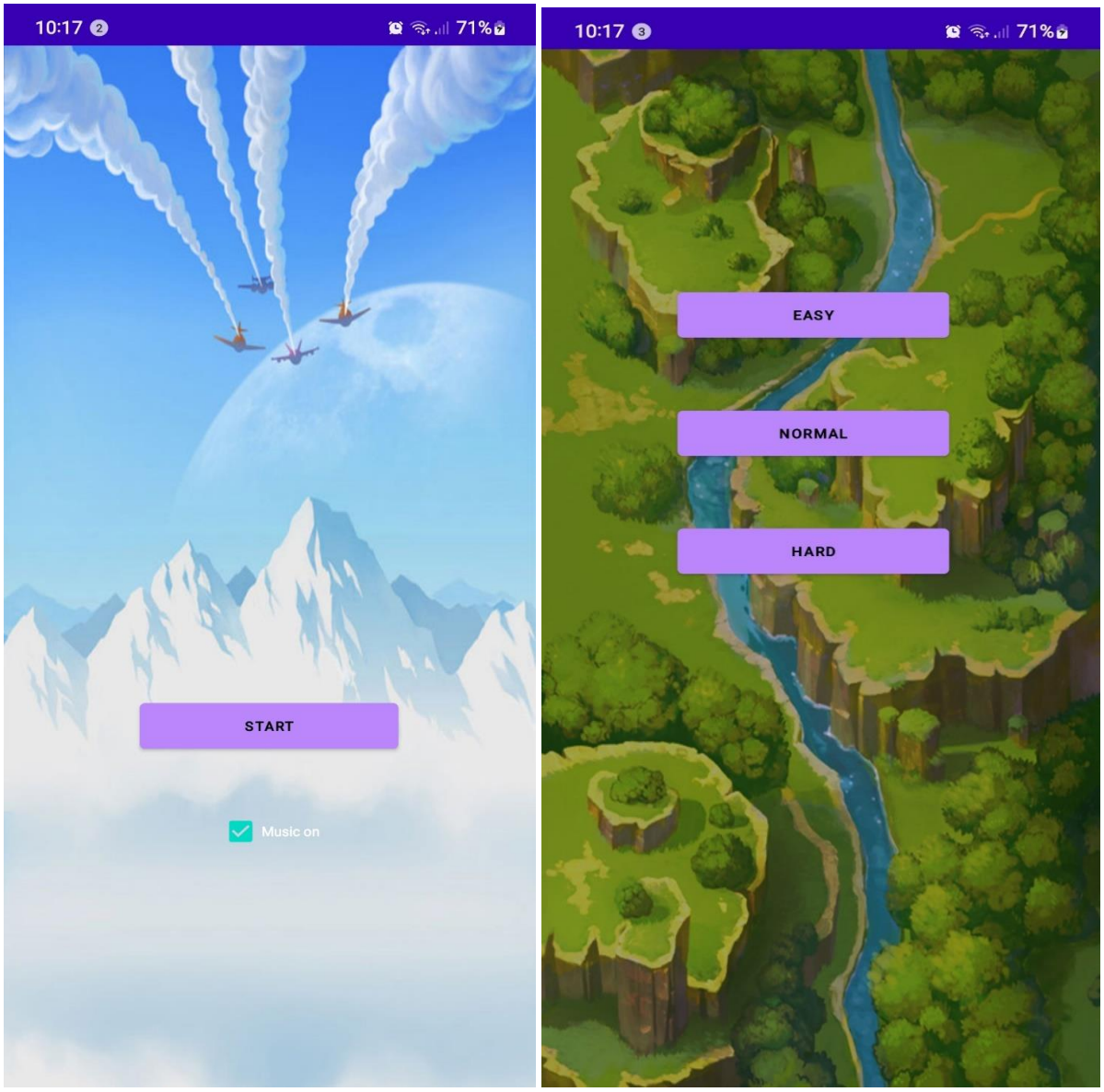
which contains the three buttons named as Easy, Normal, Hard. This are the buttons which defines the different modes of the game, by clicking any of these button the player will enter the game.

In the game player/user should protect the aircraft(ship) during war by obstacles of the enemy ship, in this game there will be three levels of difficulties they are Easy, Normal, Hard. The number of obstacles and attacking bullets will be less in easy mode. The number of obstacles and attacking bullets will be more. The number of obstacles and attacking bullets will be more than normal mode.

The games are designed in such a way that the player can play the game till the space ship blast by enemy attacks which means every mode has an infinity loop. The player ship will be get damaged by enemy bullets, if a player ship collides with any obstacles then the game will end.

When the game ends the process and the progress of the game is passed over to the next activity called ranking. In which it will separate the different modes score and it will display the player's score in the form of rank.

3.2 User Interface



(a) (b)

Figure 3.2: User Interfaces

User Interfaces is designed using the xml files in android studio in the figure(a) the user interface consists of a START button and also switch to operate the game music. In figure(b) the user interface consists of three buttons in which each button has operation of its own.

3.3 Process Flow

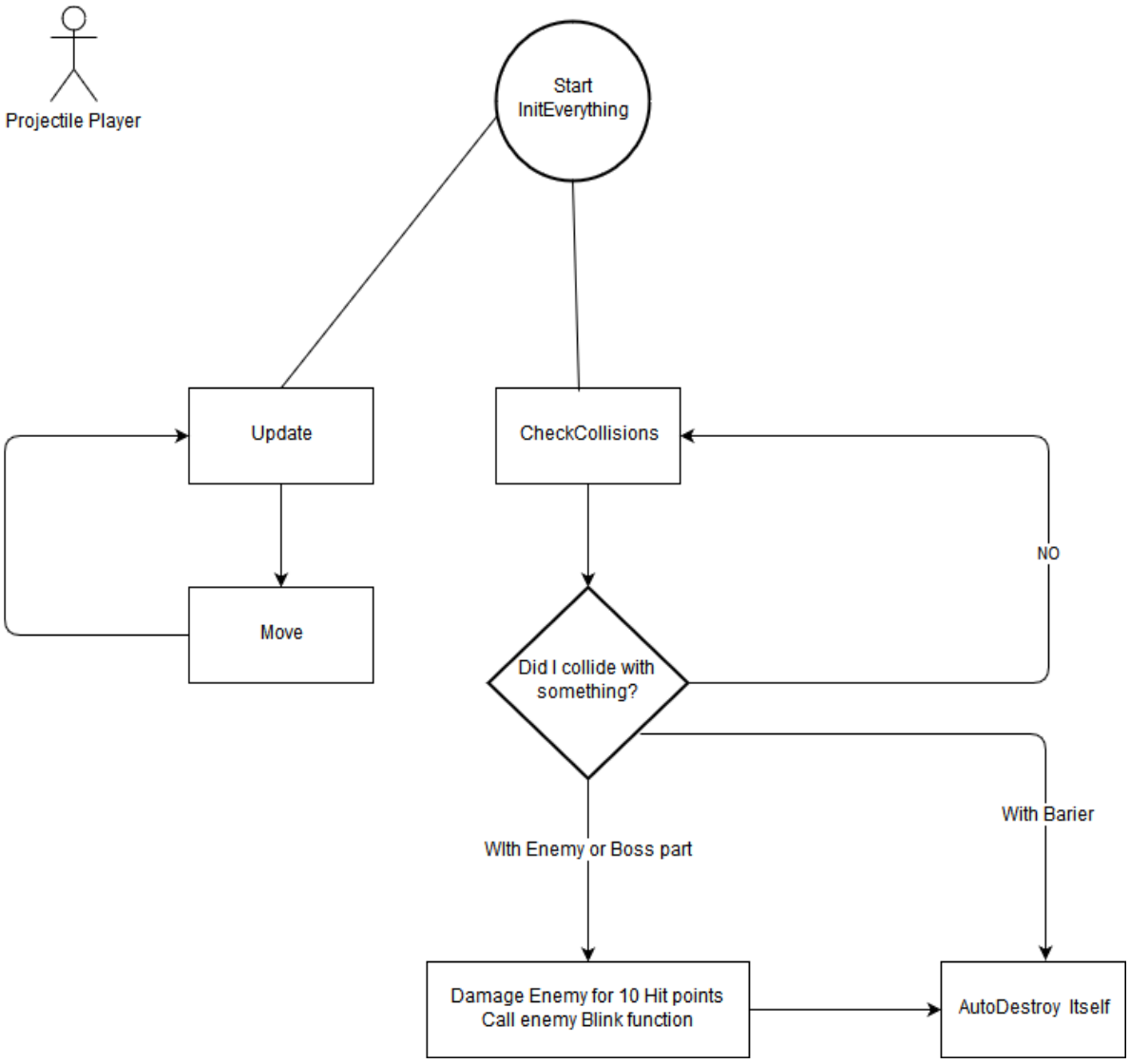


Figure 3.3: Flowchart diagram of the Player Projectile

In this flowchart it represents the flow of the player role in the game, the player has only one chance in the game, the player ship will get damage by the hit of the enemy ship bullets and also when the player ship got collide with any obstacles the ship will be destroyed and the game will end.

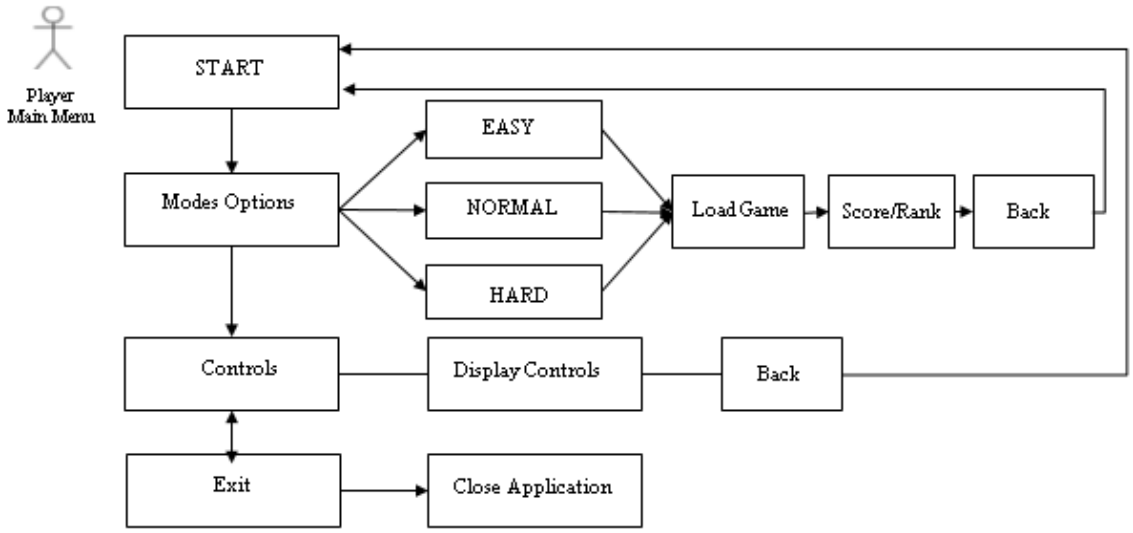


Figure 3.4: Flowchart diagram of Player Main Menu

In this flowchart the process of main menu in which is describes the selection of modes and operation done by the player/user in the application.

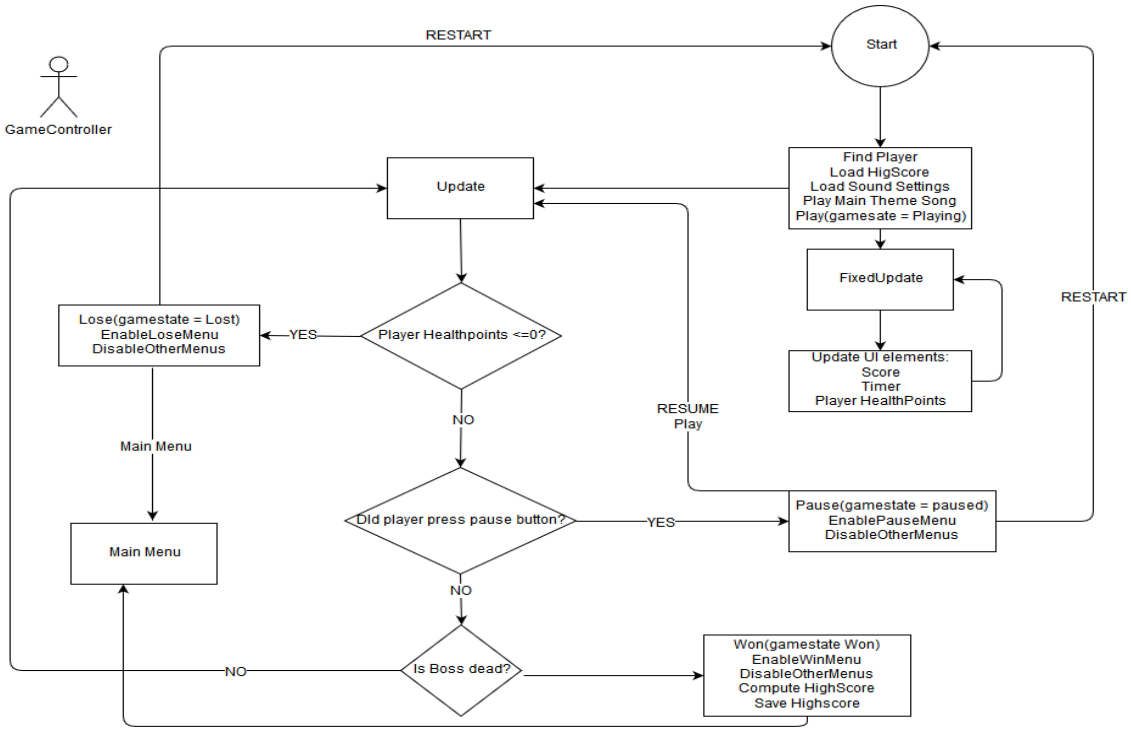


Figure 3.5: Flowchart diagram of the Game Controller

This flowchart shows the overall operations done to run the application to play the game.

3.4 Storage Database

A database is generally used for storing related, structured data, with well-defined data formats, in an efficient manner for insert, update and/or retrieval (depending on application). On the other hand, a file system is a more unstructured data store for storing arbitrary, probably unrelated data.

A database diagram is the very foundation of a database design and development effort. It represents the basic structure of a database; how information is stored, categorized and managed within it. In the project the database is used to store all the rankings of recently played games and to display the high score by comparing it.

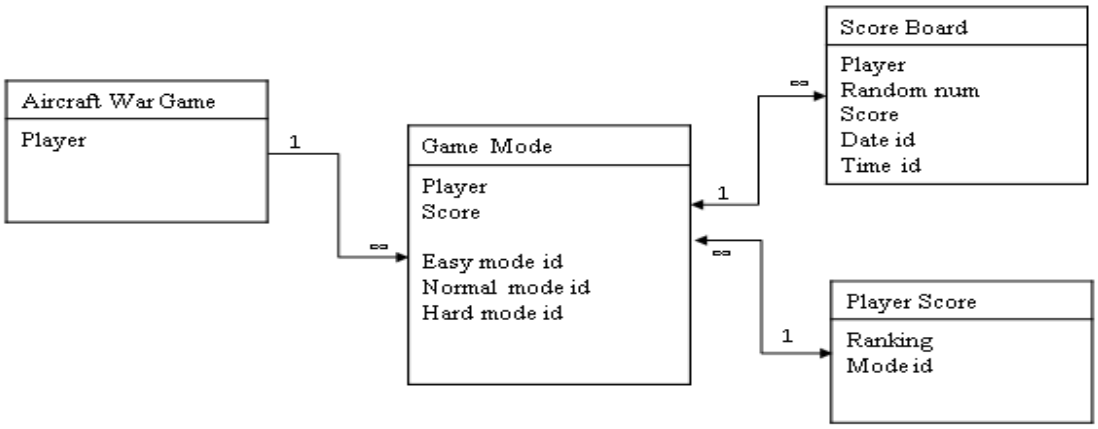


Figure 3.6: Database Design

The database will store the game results and it will display the ranking/score of the game played by the player. It will also differentiate the different modes scorecard and shows separately in the ranking system.

CHAPTER 4

IMPLEMENATION

4.1 Code Snippets

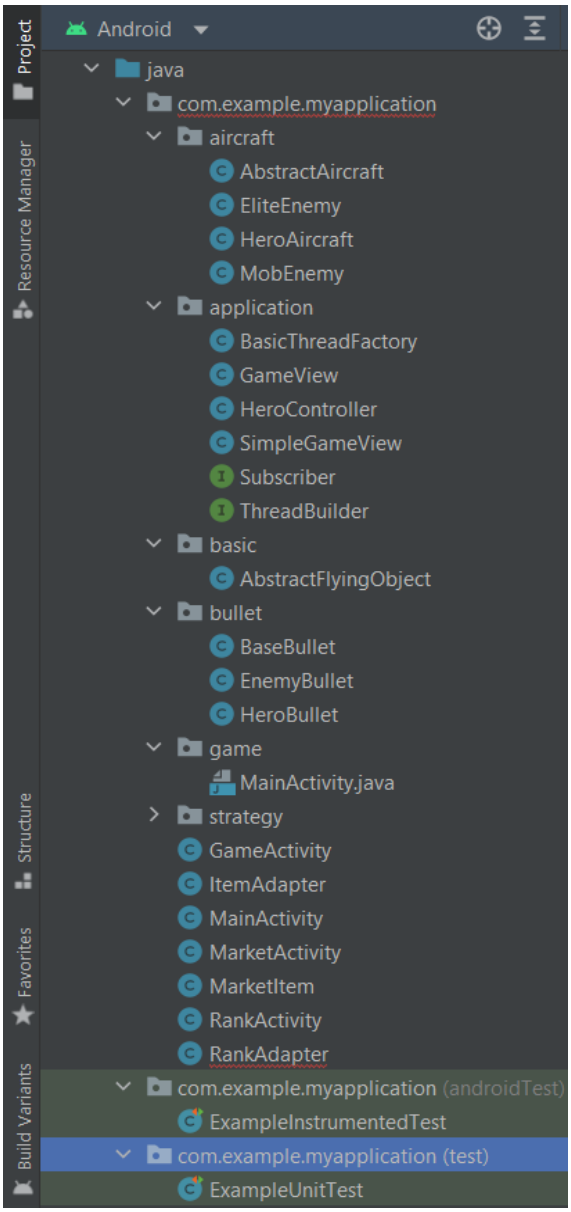


Figure 4.1: Java codes

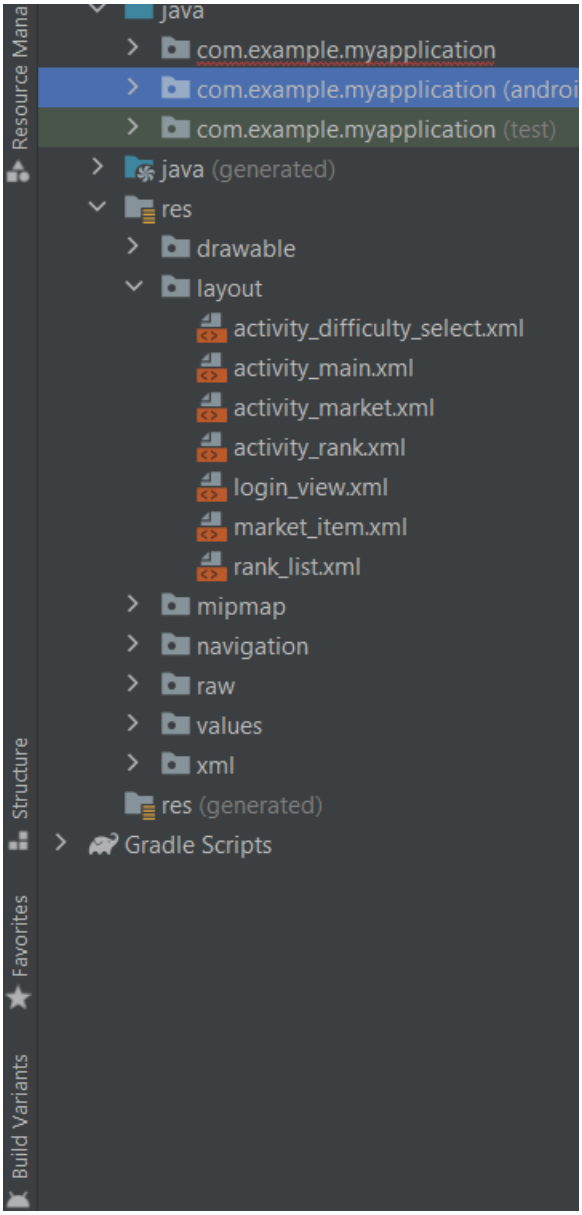


Figure 4.2:XML codes

The fig4.1 are the java codes used to develop the Aircraft War game. And in fig4.2 are the XML codes used to create user interface in the project.

Source code:

- AbstractMainActivity.java:

It is the java code used to make flying and control the flying speed of objects like player ship and enemy ship in the game.

- Gameactivity.java:

It is the java code used to perform an operation on selection of modes in the game activity.

- RankingActivity.java:

It is the java code used to set and maintain the ranking system of player's gameplay and display the rank.

- AbstractMainActivity.java

```
import android.graphics.Bitmap;

import com.example.myapplication.MainActivity;
import com.example.myapplication.aircraft.AbstractAircraft;
import com.example.myapplication.application.ImageManager;
public class AbstractFlyingObject {

    protected int locationX;
    protected int locationY;
    protected int speedX;
    protected int speedY;
    protected Bitmap image = null;
    protected int width = -1;
    protected int height = -1;
    protected boolean isValid = true;
```

```
    public AbstractFlyingObject() {
   }

 public AbstractFlyingObject(int locationX, int locationY, int speedX, int speedY) {
      this.locationX = locationX;
      this.locationY = locationY;
      this.speedX = speedX;
      this.speedY = speedY;
   }
 public void forward() {
      locationX += speedX;
      locationY += speedY;
      if (locationX <= 0 || locationX >= MainActivity.WINDOW_WIDTH) {
      speedX = -speedX;

       }
   }

public boolean crash(AbstractFlyingObject flyingObject) {
       int factor = this instanceof AbstractAircraft ? 2 : 1;
       int fFactor = flyingObject instanceof AbstractAircraft ? 2 : 1;

       int x = flyingObject.getLocationX();
       int y = flyingObject.getLocationY();
       int fWidth = flyingObject.getWidth();
       int fHeight = flyingObject.getHeight();
                                              return x + (fWidth+this.getWidth())/2 > locationX
                    && x - (fWidth+this.getWidth())/2 < locationX
                    && y + ( fHeight/fFactor+this.getHeight()/factor )/2 > locationY
                    && y - ( fHeight/fFactor+this.getHeight()/factor )/2 < locationY;
                   }
public int getLocationX() {
return locationX;
}
```

```
public int getLocationY() {
return locationY;
}
public void setLocation(double locationX, double locationY){
        this.locationX = (int) locationX;
        this.locationY = (int) locationY;
        }
public int getSpeedY() {
return speedY;
}
public  int getSpeedX(){
return speedX;
}
public Bitmap getImage() {
if (image == null){
image = ImageManager.get(this);
}
return image;
}
public int getWidth() {
if (width == -1){
width = ImageManager.get(this).getWidth();
}
return width;
}
public int getHeight() {
if (height == -1){
height = ImageManager.get(this).getHeight();
}
return height;
```

```
}
public boolean notValid() {
return !this.isValid;
}
public void vanish() {
isValid = false;
}
```

- Gameactivity.java

```
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.View;
import android.widget.Button;
import com.example.myapplication.application.DifficultGameView;
import com.example.myapplication.application.NormalGameView;
import com.example.myapplication.application.SimpleGameView;
import java.io.BufferedReader;
public class GameActivity extends AppCompatActivity implementsView.OnClickListener{
   public static int WINDOW_WIDTH;
   public static int WINDOW_HEIGHT;
   public static String whichMode;
   public static GameView gameView = null;
```

```
    public static final Object lock = new Object();
@Override
        protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_difficulty_select);
        DisplayMetrics dm = getResources().getDisplayMetrics();
        WINDOW_WIDTH = dm.widthPixels;
        WINDOW_HEIGHT = dm.heightPixels;
        Button easy = findViewById(R.id.easy);
        Button normal = findViewById(R.id.normal);
        Button hard = findViewById(R.id.hard);
        easy.setOnClickListener(this);
        normal.setOnClickListener(this);
        hard.setOnClickListener(this);
        getSupportActionBar().hide();
       }
@Override
    public void onClick(View view){
    Intent intent;
    switch (view.getId()){
    case R.id.easy:
                gameView = new SimpleGameView(this);
                setContentView(gameView);
```

```
            whichMode = "simple";
break;
      case R.id.normal:
            gameView = new NormalGameView(this);
            setContentView(gameView);
            whichMode = "normal";
            break;
      case R.id.hard:
            gameView = new DifficultGameView(this);
            setContentView(gameView);
            whichMode = "hard";
 break;
            default:
            gameView = new GameView(this);
            setContentView(gameView);
            whichMode = "error";
            break;
           }
     }
}
```

- RankingActivity.java

```
package com.example.myapplication;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Build;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import com.example.myapplication.dao.Record;
import com.example.myapplication.dao.RecordDaoImple;
import java.util.ArrayList;
import java.util.List;
public class RankActivity extends AppCompatActivity {
    public static int WINDOW_WIDTH;
    public static int WINDOW_HEIGHT;
    private static int score = 0;
    private static RecordDaoImple recordDaoImple;
    ListView listView;
    @RequiresApi(api = Build.VERSION_CODES.O)

@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_rank);

    listView = findViewById(R.id.rankList);
    TextView title = findViewById(R.id.rank_title);
    if(GameActivity.whichMode == "simple"){
    recordDaoImple = new RecordDaoImple("easy.txt",this);
    title.setText("Easy Rank");
    }else if(GameActivity.whichMode == "normal"){
            recordDaoImple = new RecordDaoImple("normal.txt",this);
                title.setText("Normal Rank");
                }else{
                recordDaoImple = new RecordDaoImple("hard.txt",this);
                title.setText("Hard Rank");
   }
DisplayMetrics dm = getResources().getDisplayMetrics();
WINDOW_WIDTH = dm.widthPixels;
WINDOW_HEIGHT = dm.heightPixels;
      Record record = new Record("Player",GameActivity.gameView.getScore());
      recordDaoImple.addRecord(record);
      List<Record> tableTata = recordDaoImple.printRecord();
```

```
            RankAdapter rankAdapter = new
RankAdapter(RankActivity.this,R.layout.rank_list,tableTata);

            listView.setAdapter(rankAdapter);

    getSupportActionBar().hide();

   }

}
```

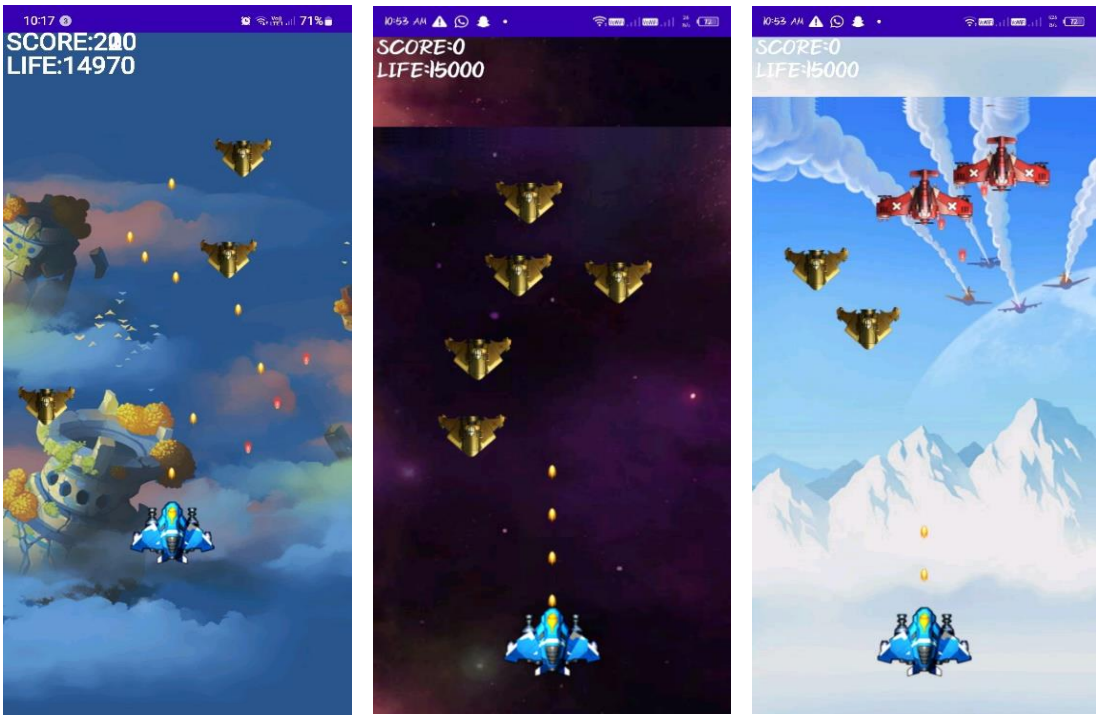
CHAPTER 5

RESULT AND DISCUSSION

5.1 Screen Shots

Figure 5.1: User interface of mode selection

This is the output of "GameActivityMain.xml". The user interface is designed using xml file and in this interface we have three buttons used to select different modes of the game.



(a) Easy mode (b) Normal mode (c) Hard mode

Figure 5.2: Different types of modes/Levels

After selecting the mode by the player, The game will be started, This is the view of each different modes (a)Easy (b)Normal (c)Hard.

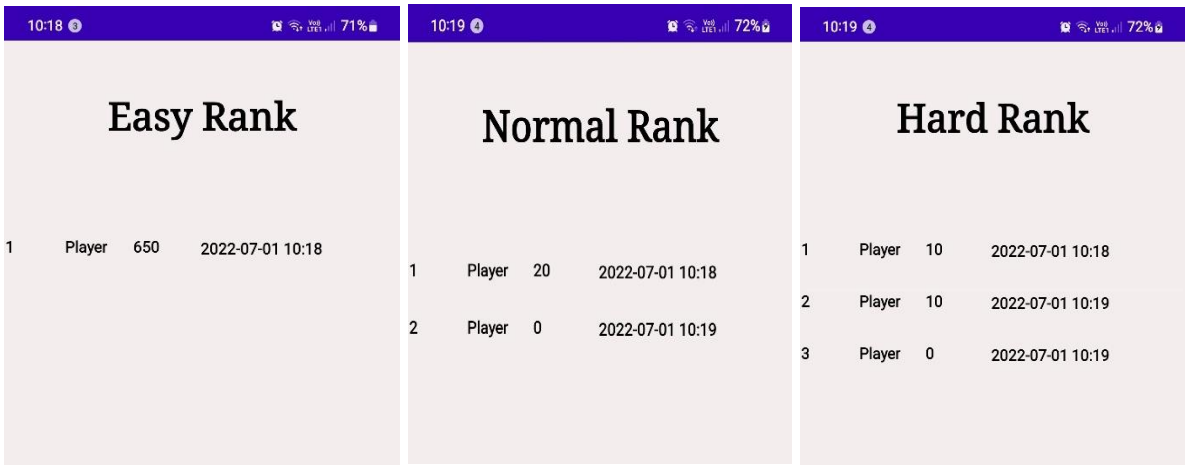


Figure 5.3: Ranking system of the game

This is the output of the “RankingActivityMain.xml”. The user interface is used to show the player game scoreboard of every different modes and also the date and time that when he was played.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

➢ Conclusion:

Working with game completely a new experience for me. Normally I am working with different languages, DBMS, mark up languages etc. It is very sensible work and it demands much time because the game engines try to connect game environment with the real world. The Exists game engines demands vast knowledge about its properties, sections and sub-sections. In this project, I have done all the necessary to achieve the main targets by designing and developing my own video-game. However, I already have a few ideas to extend this work and improve the video-game.

➢ Future enhancements:

Create a complete projectile system with many types of projectiles and ways to shoot them, using different approaches. By creating more levels with different game-play options, each of them with new enemies and bosses .and also to create a power-up system that allows players to get new powers, such as super-speed. To make a different kind of reward logic, like including a multiplier to calculate the final score and also to make different types of ships that can be used as the Player. Create a module for Unity to accept graphics in SVG format.

BIBLIOGRAPHY

- https://www.youtube.com/watch?v=1FRrUHL8l-g
- https://youtube.com/playlist?list=PLsOU6EOcj51e7YesVnTrEtvJDD016p9oS
- Main article: Run-and-gun shooter
- 1980's *Crazy Climber* (Nichibutsu, arcade) has the player scaling a vertically scrolling skyscraper.
- Data East's arcade game *Flash Boy* (1981) for the DECO Cassette System was released in two versions: a side-scrolling version and a vertical scrolling version.