VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-590 018

A Mini -Project Work on

"Wing Master (Flappy Bird Game)"

A Dissertation work submitted in partial fulfillment of the requirement
for the award of the degree

Bachelor of Engineering

In

Information Science & Engineering

Submitted by

Chethan Holla BV **1AY18IS030**

Jeevan V **1AY18IS048**

Under the guidance of

Prof. Ramesh Sengodan

Assistant Professor

ACHARYA

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

ACHARYA INSTITUTE OF TECHNOLOGY

(AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI.APPROVED BY AICTE, NEW DELHI, ACCREDITEDBY NAAC, NEW DELHI)

Acharya Dr. Sarvepalli Radhakrishnan Road, Soldevanahalli, Bengaluru-560107

2020-21

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
ACHARYA INSTITUTE OF TECHNOLOGY
(AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI.APPROVED BY AICTE, NEW DELHI, ACCREDITEDBY NAAC, NEW DELHI)
Acharya Dr. Sarvepalli Radhakrishnan Road, Soldevanahalli, Bengaluru-560107

Certificate

This is to Certify that the Mini-Project work entitled **"Wing Master(Flappy Bird Game)"** is a bonafide work carried out by **Chethan Holla BV(1AY18IS030) and Jeevan V (1AY18IS048)** in partial fulfillment for the award of the degree of **Bachelor of Engineering** in **Information Science and Engineering** of the **Visvesvaraya Technological University**, Belagavi during the year 2020-21. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The Project has been approved as it satisfies the academic requirements in respect of Project work prescribed for the Bachelor of Engineering Degree.

Prof.Ramesh Sengodan Guide	**Prof. Marigowda C K** HOD

Name of the Examiners	**Signature with date**
1. _____	_____
2. _____	_____

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this mini-project would be incomplete without the mention of the people who made it possible through constant guidance and encouragement.

We would take this opportunity to express our heart-felt gratitude to **Sri. B.Premnath Reddy**, Chairman, Acharya Institutes and **Dr. Prakash M R**, Principal, Acharya Institute of Technology for providing the necessary infrastructure to complete this mini-project.

We wish to express our deepest gratitude and thanks to **Prof. Marigowda C K**, Head of the Department, Information Science and Engineering.

We wish to express sincere thanks to our guide **Prof. Ramesh Sengodan**, Assistant Professor, Department of Information Science and Engineering for helping us throughout and guiding us from time to time.

A warm thanks to all the faculty of Department of Information Science and Engineering, who have helped us with their views and encouraging ideas.

ABSTRACT

This Game is an Android application which is a clone of famous flappy bird game application developed using android studio and it is compatible with almost all old and new version of Android.

We try to bring the popular game Flappy Bird, which is originally on Android platform, to the FPGA board. The game is a side-scroller where the player controls a bird, attempting to fly between rows of green pipes without coming into contact them. If the player touches the pipes, it ends the game. The bird briefly flaps upward each time the player taps the screen; if the screen is not tapped, the bird falls due to gravity.

Flappy Bird is truly a marketer's dream. Simple design, a universally understood concept and doesn't require any previous knowledge. It is easy to see why such a simple and unassuming product became one of the biggest mobile application successes and resulted in emotional devastation when it was taken away. Flappy Bird is a reminder that simplicity in a product can be key and that you don't necessarily need all the bells and whistles; just a well put-together bike.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

Figures

Tables

CHAPTER 1

INTRODUCTION

1.1 Introduction to Mobile Application Development

Mobile application development is the process to making software for smartphones and digital assistants, most commonly for Android and iOS. The software can be preinstalled on the device, downloaded from a mobile app store or accessed through a mobile web browser. The programming and markup languages used for this kind of software development include Java, Swift, C# and HTML5.

Mobile app development is rapidly growing. From retail, telecommunications and e-commerce to insurance, healthcare and government, organizations across industries must meet user expectations for real-time, convenient ways to conduct transactions and access information.

Today, mobile devices—and the mobile applications that unlock their value—are the most popular way for people and businesses to connect to the internet. To stay relevant, responsive and successful, organizations need to develop the mobile applications that their customers, partners and employee's demand.

1.1.1 What is Mobile App?

A mobile application, also referred to as a mobile app or simply an app, is a computer program or software application designed to run on a mobile device such as a phone, tablet, or watch. Apps were originally intended for productivity assistance such as email, calendar, and contact databases, but the public demand for apps caused rapid expansion into other areas such as mobile games, factory automation, GPS and location-based services, order-tracking, and ticket purchases, so that there are now millions of apps available.

Apps are generally downloaded from application distribution platforms which are operated by the owner of the mobile operating system, such as the App Store (iOS) or Google Play Store. Some apps are free, and others have a price, with the profit being split between the application's creator and the distribution platform. Mobile applications often stand in contrast to desktop applications which are designed to run on desktop computers, and web applications which run in mobile web browsers rather than directly on the mobile device.

1.1.2 What is Mobile OS?

A mobile operating system (OS) is software that allows smartphones, tablet PCs (personal computers) and other devices to run applications and programs. A mobile OS typically starts up when a device powers on, presenting a screen with icons or tiles that present information and provide application access. Mobile operating systems also manage cellular and wireless network connectivity, as well as phone access.

1.1.3 Types of Mobile Apps

Native Apps: Such apps are developed for a single mobile operating system exclusively, therefore they are "native" for a particular platform or device. App built for systems like iOS, Android, Windows phone, Symbian, Blackberry can't be used on a platform other than their own.

Hybrid Apps: They are built using multi-platform web technologies (for example HTML5, CSS and JavaScript). So-called hybrid apps are mainly website applications disguised in a native wrapper. Apps possess usual pros and cons of both native and web mobile applications.

Web Apps: These are software applications that behave in a fashion similar to native applications. Web apps use a browser to run and are usually written in HTML5, JavaScript or CSS. These apps redirect a user to URL and offer "install" option by simply creating a bookmark to their page.

1.1.4 Different categories of Mobile Apps

Educational Apps: Educational and informative apps do just that–educate and inform. While the purpose of this type of app is fairly straightforward, there is a lot of diversity when it comes to educational apps, like news and language apps. If you're looking to break into this crowded space, you'll need to serve up news or other information in a fun and unique format for learners of all ages, interests, and levels.

Lifestyle Apps: This app category covers a lot of ground, literally. Where you're going, how you're getting there, what you're going to order off the menu–it all falls under lifestyle apps. Think of apps you use for convenience, like fitness, dating, food, and travel. Lifestyle and

leisure apps are increasingly popular, especially for tasks that require an extra step aside from the search itself (e.g., the scary action of actually picking up the phone to make a call).

Social Media Apps: Social media apps give users the opportunity to connect with people inside or outside their social circles. For the most part, social media apps are universal and have a very diverse user base. These apps are used to share live video, post images, facilitate conversations, and more. Social media apps have quickly become part of our everyday lives.

Productivity Apps: Also known as business apps, productivity apps typically organize and complete complex tasks for you, anything from sending an email to figuring out the tip on your dinner bill. Most productivity apps serve a single purpose and are built with a very intuitive interface and design to increase efficiency and improve user experience.

Entertainment Apps: This category of apps has one sole focus–keeping you busy. Entertainment apps are often used to fill your time, whether you are jet-setting across the country, lounging at home, or really anywhere in-between. Along with their websites, a lot of popular streaming services have mobile applications so users can access their library wherever they are. Entertainment apps can include video, text, or audio content.

Game Apps: This app category is pretty self-explanatory and represents the biggest portion of app downloads by far. With such a crowded category, it makes sense that there are so many types of game apps for different target audiences, such as arcade games, brain training puzzles, or just plain silly games, like launching tiny birds at pigs. Some mobile app games are played solo, others online, and occasionally in-person with a group of friends.

1.1.5 What is Mobile Application Development?

Mobile app development is the act or process by which a mobile app is developed for mobile devices, such as personal digital assistants, enterprise digital assistants or mobile phones. These applications can be pre-installed on phones during manufacturing platforms, or delivered as web applications using server-side or client-side processing (e.g., JavaScript) to provide an "application-like" experience within a Web browser.

To develop apps using the SDK, use the Java programming language for developing the app and Extensible Markup Language (XML) files for describing data resources. By writing the code in Java and creating a single app binary, you will have an app that can run on both phone and tablet.To help you develop your apps efficiently, Google offers a full Java Integrated Development Environment (IDE) called Android Studio, with advanced features

for developing, debugging, and packaging Android apps. Using Android Studio, you can develop apps on any available Android device, or create virtual devices that emulate any hardware configuration.

1.1.6 Mobile Application Development Challenges

While the Android platform provide 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 secure
- Remaining compatible with older platform versions
- Understanding the market and the user.

1.1.7 How to develop Mobile Apps

Step 1: Set a Goal

Step away from any form of technology and get out a pen and paper and define what it is you want to accomplish. The starting line in the app development word is a pen and paper, not complex coding and designing.

Step 2: Sketch your ideas

You need to use the pen and paper that has the answers to the questions about your apps purpose to develop a sketch of what it will look like. Here you move your clearly worded ideas into visual representations of your thoughts. Decide if you are going to give your app away and offer ads to generate money, or are you going to offer it as a paid download. You can also choose the option to offer in app purchases.

Step 3: Research

You have to dig deep and research the competition of your app idea. I know you think you have one of a kind idea, but the numbers are not in your favor—odds are someone has already tried it. You can look at this in two different ways. One you can become deflated and give up, or two, you can examine the competition and make your app better.

Step 4: Wireframe

In the technology world, a wireframe is a glorified story board. Here is where you take your sketch and your design idea, and you give your idea a little more clarity and functionality. This will become the foundation for your app's development, so it really is a crucial step.

Step 5: Start Defining the Back End of Your App

We left off with your wireframe, so at this point in your app development, you have a storyboard of how you want your app to function. Now it's time to use that storyboard to start examine functionality.

Step 6: Check Your Model

Here's where you need to call in the troops. Show your demo to friends, family, and anyone else who is willing to give you constructive criticism. Don't waste your time with people who will tell you, "Wow, that's neat." Seek out those cynics and critics. Brutal honesty is crucial at this phase.

Step 7: Get Building

With the foundation in place, you can start to put the puzzle together to building your app. First, your developer will set up your servers, databases, and APIs.

Step 8: Design the Look

Now it's time to employ the designers to create your UI, user interface. The user interface is a very important part of your app because people are attracted to how things look and how easy they are to navigate.

Step 9: Test Your App, again

A second round of testing is imperative. In this round, you will have both a functioning app as well as a user interface to test. All the screens of your app should properly work at this point, and your app should be visually appealing as well.

Step 10: Modify and Adjust

You've taken your prototype for a spin, and you've learned that there are still a few tweaks you need to make. Now that you've seen your app in it's fully functioning form, you need to call the troops back and ask they to do the same.

Step 11: Beta Testing

You've looked at your app through several different lenses, and you think you've managed to develop a smoothly functioning, aesthetically pleasing, problem solving app. Now, you need to examine how your app is going to function in a live environment.

Step 12: Release Your App

You've made it to the finish line. You've brought your idea to fruition, and the last step is to share it with the world. Hopefully, you've gone on to solve a major problem. If not, with any luck your app has some features that can simplify or bring enjoyment to someone's life. Regardless, you've accomplished something big. Now it's time to distribute it.

1.2 Introduction to Android Studio

Android is one of the most popular mobile device platforms. The Android platform allows developers to write managed code using Java to manage and control the Android device. Android Studio is a popular IDE developed by Google for developing applications that are targeted at the Android platform. Android Studio has replaced Eclipse as the IDE of choice for developing Android applications.

1.2.1 What is Android?

Android is a software package and Linux based operating system for mobile devices such as tablet computers and smartphones.It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.

1.2.2 Different versions of Android OS

Table 1.1: Different versions of Android OS

Name	Version Numbers	Release Date	API Level
No official Name	1.0	Sept, 2008	1
No official Name	1.1	Feb, 2009	2
Cupcake	1.5	Apr, 2009	3
Donut	1.6	Sept, 2009	4
Éclair	2.0-2.1	Oct, 2009	5-7
Froyo	2.2-2.2.3	May, 2010	8

Gingerbread	2.3-2.3.7	Dec, 2010	9-10
Honeycomb	3.0-3.2.6	Feb, 2011	11-13
Ice Cream Sandwich	4.0-4.0.4	Oct, 2011	14-15
Jelly Bean	4.1-4.3.1	July, 2012	16-18
KitKat	4.4-4.4.4	Oct, 2013	19-20
Lollipop	5.0-5.1.1	Nov, 2014	21-22
Marshmallow	6.0-6.0.1	Oct, 2015	23
Nougat	7.0-7.1.2	Aug, 2016	24-25
Oreo	8.0-8.1	Aug, 2017	26-27
Pie	9	Aug, 2018	28
Android 10	10	Sept, 2019	29
Android 11	11	Sept 2020	30

1.2.3 Android Development Architecture

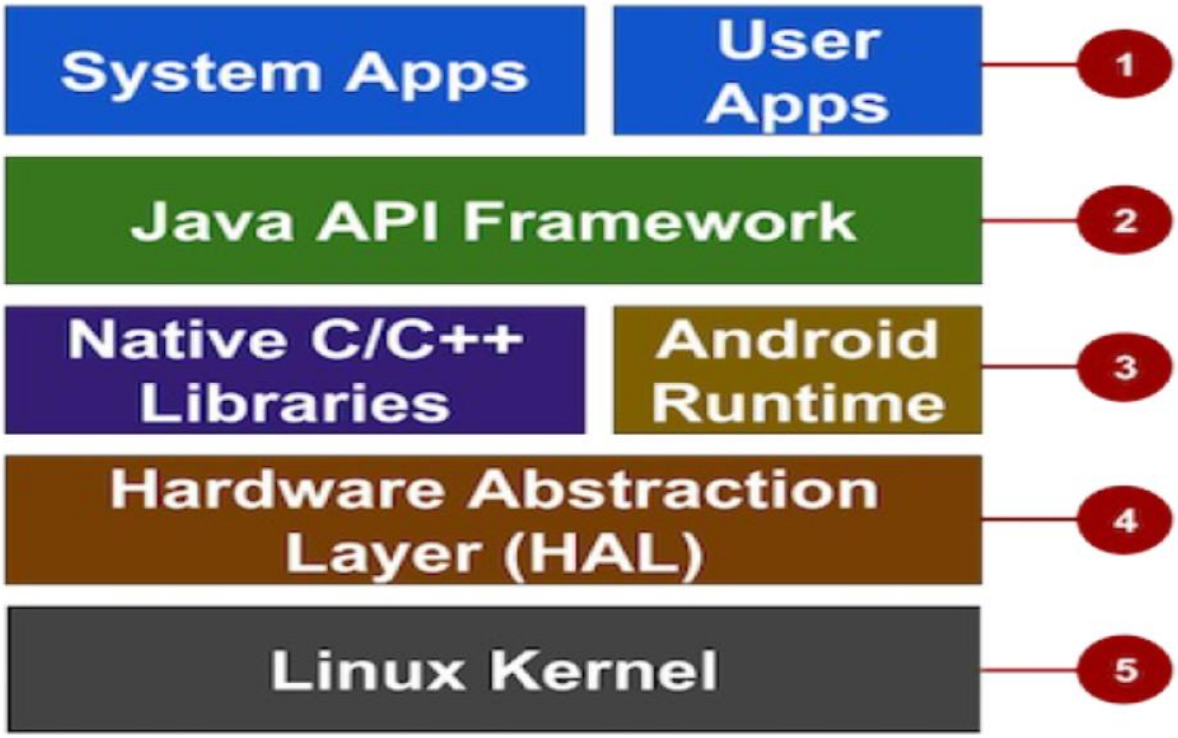


Fig 1.1: Android development architecture.

Apps: Your apps live at this level, along with core system apps for email, SMS messaging, calendars, Internet browsing, or contacts.

Java API Framework: All features of Android are available to developers through application programming interfaces (APIs) written in the Java language. You don't need to know the details of all of the APIs to learn how to develop Android apps, but you can learn more about the following APIs, which are useful for creating apps:

- **View System** used to build an app's UI, including lists, buttons, and menus.
- **Resource Manager** used to access to non-code resources such as localized strings, graphics, and layout files.
- **Notification Manager** used to display custom alerts in the status bar.
- **Activity Manager** that manages the lifecycle of apps.
- **Content Providers** that enable apps to access data from other apps.
- **All framework APIs** that Android system apps use.

Libraries and Android Runtime: Each app runs in its own process and with its own instance of the Android Runtime, which enables multiple virtual machines on low-memory devices. Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some Java 8 language features that the Java API framework uses. Many core Android system components and services are built from native code that require native libraries written in C and C++. These native libraries are available to apps through the Java API framework.

Hardware Abstraction Layer (HAL): This layer provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework.

Linux Kernel: The foundation of the Android platform is the Linux kernel. The above layers rely on the Linux kernel for underlying functionalities such as threading and low-level memory management.

1.2.4 Installing Android Studio

System Requirements:

- Microsoft® Windows® 7/8/10, Mac or Linux (64-bit).
- 4 GB RAM minimum, 8 GB RAM recommended.
- GB of available disk space minimum, 4 GB Recommended.
- 1280 x 800 minimum screen resolution.

Windows

To install Android Studio on Windows, proceed as follows:

- Download ***android-studio-ide-201.7199119-windows.exe*** file from the https://developer.android.com/studio
- Double-click .exe file to launch it.
- Follow the setup wizard in Android Studio and install any SDK packages that it recommends.

Mac

To install Android Studio on your Mac, proceed as follows:

- Launch the Android Studio DMG file.
- Drag and drop Android Studio into the Applications folder, then launch Android Studio.
- Select whether you want to import previous Android Studio settings, then click OK.
- The Android Studio Setup Wizard guides you through the rest of the setup, which includes downloading Android SDK components that are required for development.

Linux

To install Android Studio on Linux, proceed as follows:

- Unpack the .zip file you downloaded to an appropriate location for your applications, such as within /usr/local/ for your user profile, or /opt/ for shared users.
- If you're using a 64-bit version of Linux, make sure you first install the required libraries for 64-bit machines.
- To launch Android Studio, open a terminal, navigate to the android-studio/bin/ directory, and execute studio.sh.
- Select whether you want to import previous Android Studio settings or not, then click OK.
- The Android Studio Setup Wizard guides you through the rest of the setup, which includes downloading Android SDK components that are required for development.

1.2.5 Android Studio Window Panes

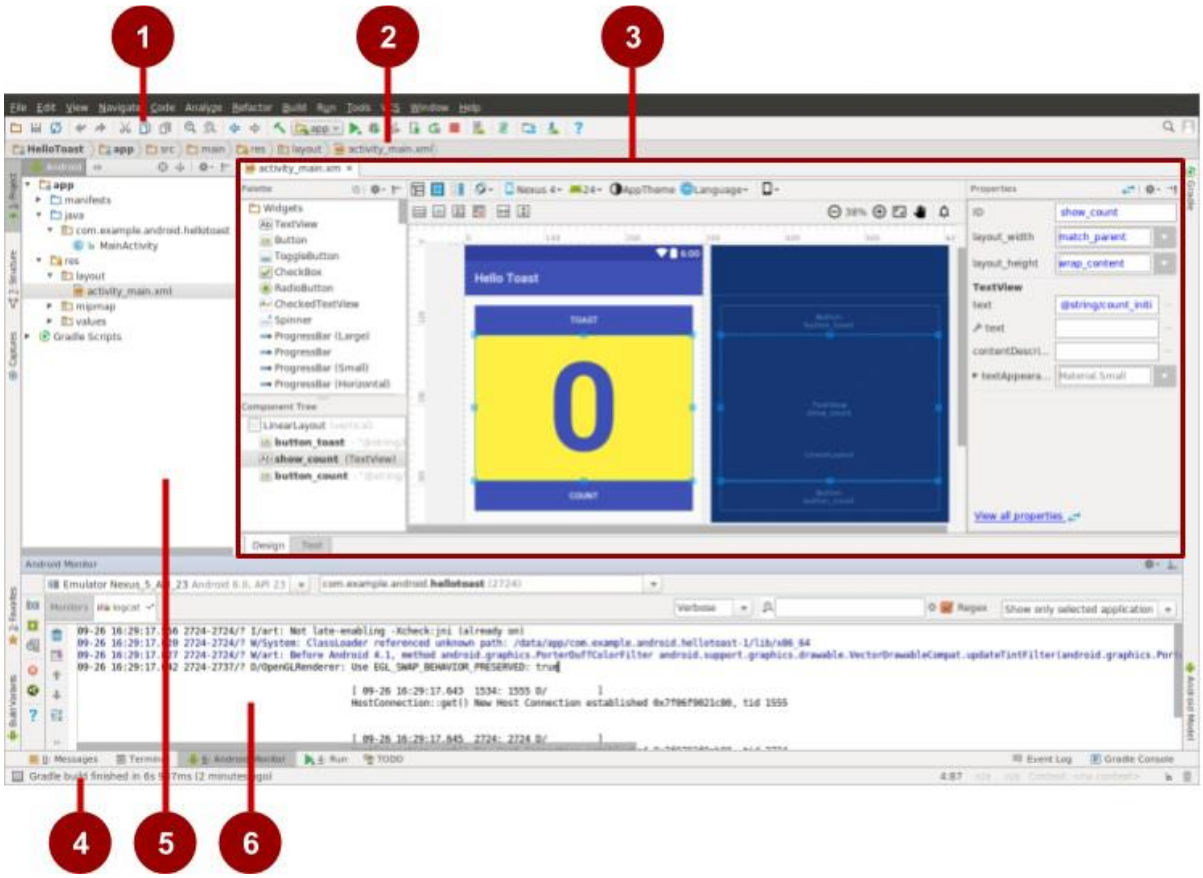


Fig 1.2: Android studio window panes.

- **The Toolbar.** The toolbar carries out a wide range of actions, including running the Android app and launching Android tools.
- **The Navigation Bar**. The navigation bar allows navigation through the project and open files for editing. It provides a more compact view of the project structure.
- **The Editor Pane**. This pane shows the contents of a selected file in the project. For example, after selecting a layout (as shown in the figure), this pane shows the layout editor with tools to edit the layout. After selecting a Java code file, this pane shows the code with tools for editing the code.
- **The Status Bar**. The status bar displays the status of the project and Android Studio itself, as well as any warnings or messages. You can watch the build progress in the status bar.
- **The Project Pane**. The project pane shows the project files and project hierarchy.

- **The Monitor Pane**. The monitor pane offers access to the TODO list for managing tasks, the Android Monitor for monitoring app execution (shown in the figure), the logcat for viewing log messages, and the Terminal application for performing Terminal activities.

1.2.6 Viewing the Android Manifest

Before the Android system can start an app component, the system must know that the component exists by reading the app's AndroidManifest.xml file. The app must declare all its components in this file, which must be at the root of the app project directory. To view this file, expand the manifests folder in the Project: Android view, and double-click the file (AndroidManifest.xml). Its contents appear in the editing pane as shown in the figure below.

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Hello World"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Fig 1.3: Android manifest file.

1.2.7 Viewing and editing Java code

Components are written in Java and listed within module folders in the java folder in the Project: Android view. Each module name begins with the domain name (such as com.example.android) and includes the app name.

The following example shows an activity component:

- Click the module folder to expand it and show the MainActivity file for the activity written in Java (the MainActivity class).
- Double-click MainActivity to see the source file in the editing pane, as shown in the figure below.

```
package com.example.android.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Fig 1.4: MainActivity.java file.

1.2.8 Viewing and editing layouts

Layout resources are written in XML and listed within the layout folder in the res folder in the Project: Android view. Click res > layout and then double-click activity_main.xml to see the layout file in the editing pane.Android Studio shows the Design view of the layout, as shown in the figure below. This view provides a Palette pane of user interface elements, and a grid showing the screen layout.

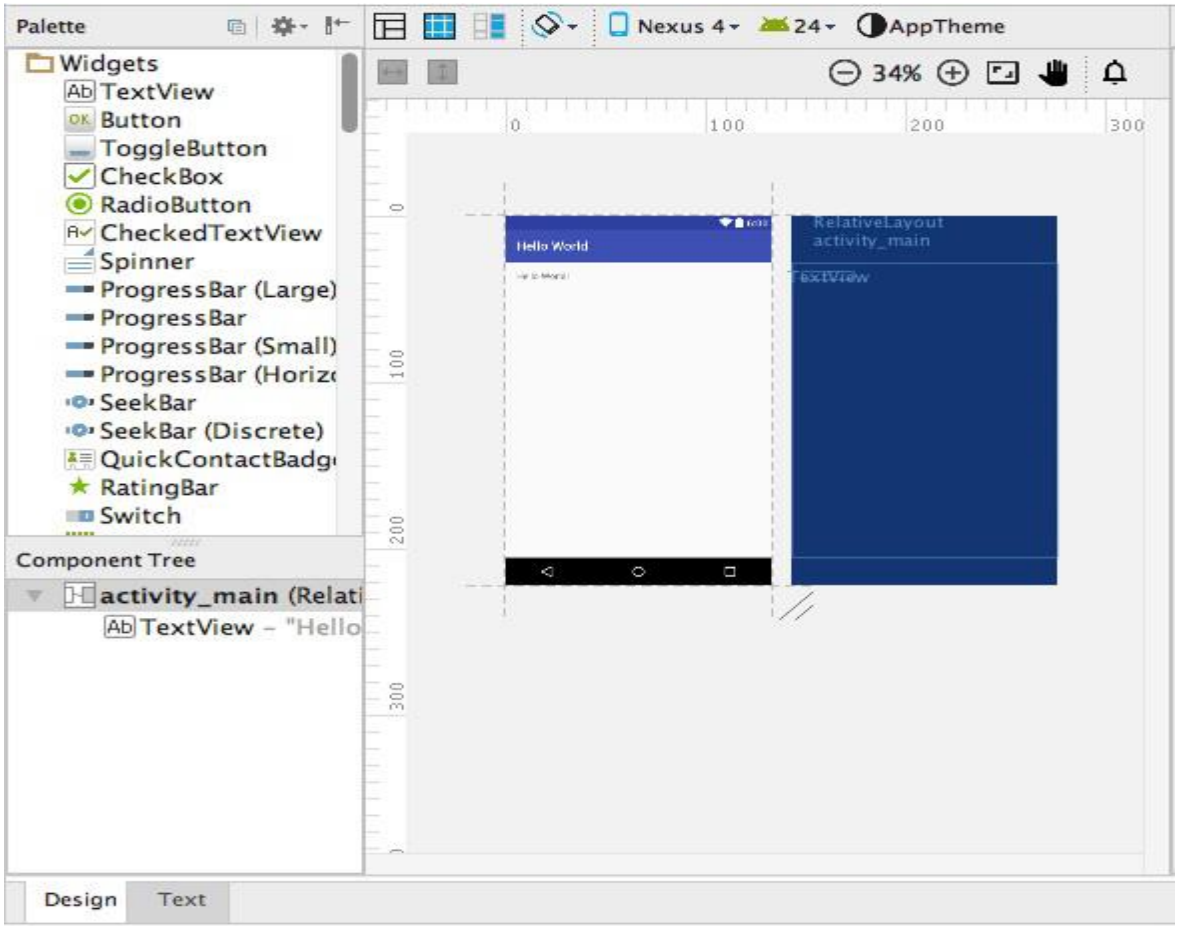


Fig 1.5: Design view of layout.

1.3 Introduction to Project

Our software project is Flappy Bird. It's a game application. The game was designed and built by Dong Nguyen, a developer who lives in Vietnam. Flappy bird is a side - scroller game where the player controls a bird, attempting to fly between columns of green pipes. The bird will be flying until it collisions with a pipe or it fall on ground. It's a simple game of infinite level type. It's a challenging game for all.

The game was released in May 2013 but received a sudden rise in popularity in early 2014 and became a sleeper hit. Flappy Bird received poor reviews from some critics, who criticized its high level of difficulty and alleged plagiarism in graphics and game mechanics, while other reviewers found it addictive.

1.3.1 Problem Statement

Design Physic based 2D Platform game, the game provides Enjoyment and learning for uses regarding fundamental concept of physic like Velocity, Friction, Bounciness and Gravity.

1.3.2 Motivation and Objectives

It's flappy bird with more randomness. the goal is to get past the obstacles without getting detracted by everything always changing. If you touch the ground or an obstacle you lose. In this game, the objective is to pass through all the obstacles and not hit the ground or the obstacles.

1.3.3 Project Applications

. This project was developed for Entertainment purpose.

. This may help ease anxiety and Depression.

. This helps to improve Dexterity.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Hardware Requirements

MINIMUM RAM: 128MB

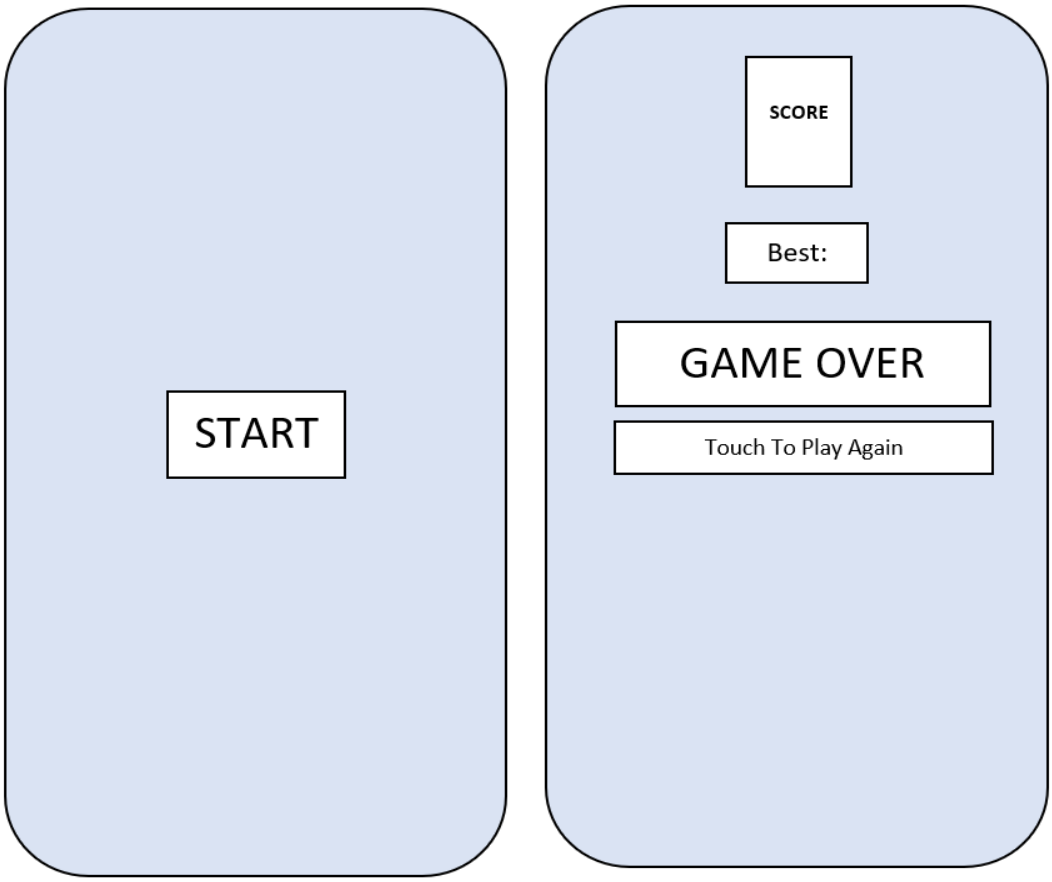
2.2 Software Requirements

ANDROID VERSION: ANDROID 4.0 or above.

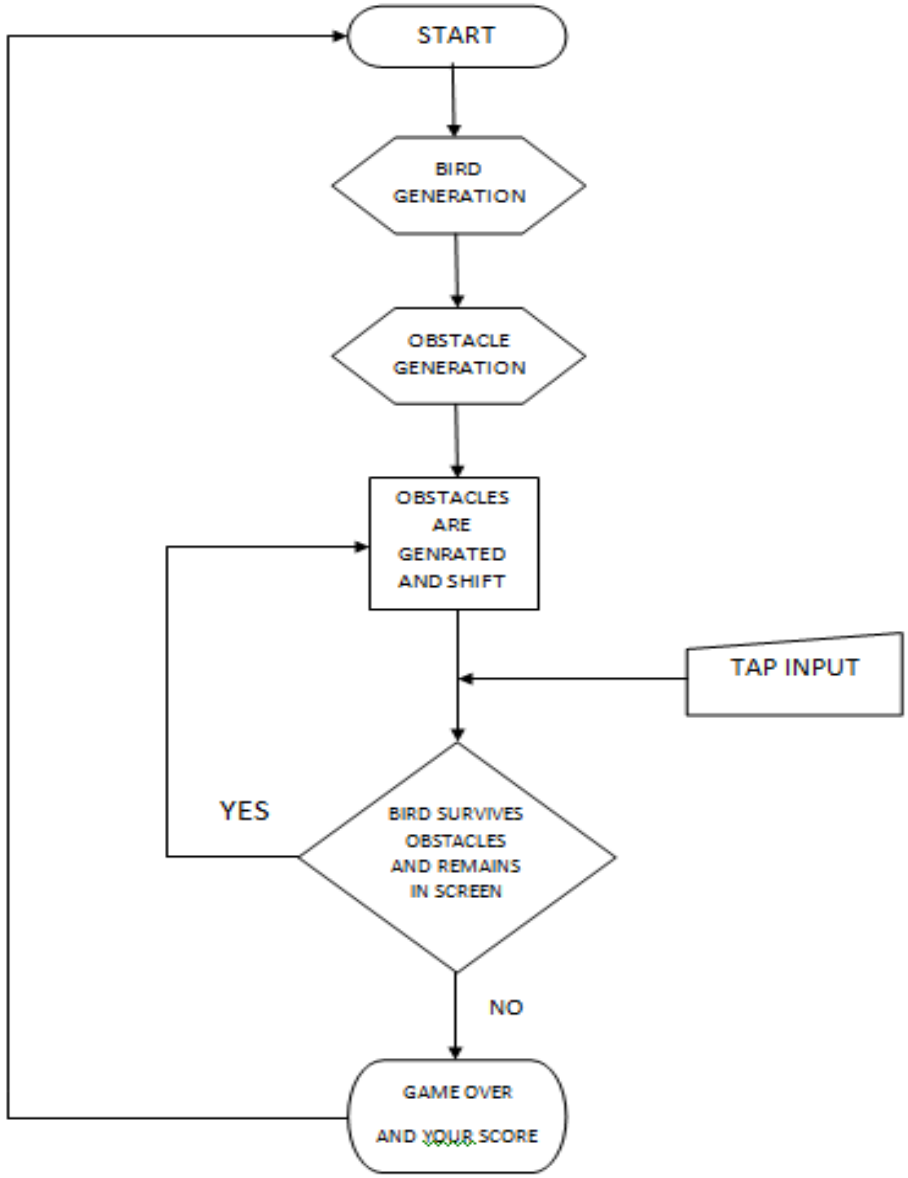
CHAPTER 3

DESIGN

3.1 User Interface Design



3.2 ER Diagram /Data Flow Diagram



CHAPTER 4

IMPLEMENTATION

4.1 Source Code

BaseObject.java

```
package com.example.wingmaster;

import android.graphics.Bitmap;
import android.graphics.Rect;

public class BaseObject {
    protected float x,y;
    protected int width,height;
    protected Rectrect;
    protected Bitmap bm;

    public BaseObject() {
    }

    public BaseObject(float x, float y, int width, int height) {
this.x = x;
this.y = y;
this.width = width;
this.height = height;
    }

    public float getX() {
        return x;
    }

    public void setX(float x) {
this.x = x;
```

```
    }

    public float getY() {
        return y;
    }

    public void setY(float y) {
this.y = y;
    }

    public int getWidth() {
        return width;
    }

    public void setWidth(int width) {
this.width = width;
    }

    public int getHeight() {
        return height;
    }

    public void setHeight(int height) {
this.height = height;
    }

    public Bitmap getBm() {
        return bm;
    }

    public void setBm(Bitmap bm) {
        this.bm = bm;
    }
```

```
    public RectgetRect() {
        return new Rect((int)this.x, (int)this.y, (int)this.x+this.width, (int)this.y+this.height);
    }

    public void setRect(Rectrect) {
this.rect = rect;
    }
}
```

Bird.java

```
package com.example.wingmaster;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;

import java.util.ArrayList;

public class Bird extends BaseObject {
    private ArrayList<Bitmap>arrBms = new ArrayList<>();
    private int count, vFlap, idCurrentBitmap;
    public float drop;
    public Bird(){
this.count = 0;
this.vFlap = 8;
this.idCurrentBitmap = 0;
this.drop = 0;
    }
    public void draw(Canvas canvas){
drop();
canvas.drawBitmap(this.getBm(), this.x, this.y, null);
    }
```

```
    private void drop() {
this.drop+=0.5;
this.y+=this.drop;
    }

    public ArrayList<Bitmap>getArrBms() {
        return arrBms;
    }

    public void setArrtms(ArrayList<Bitmap>arrBms) {
this.arrBms = arrBms;
for(int i=0; i<arrBms.size(); i++){
this.arrBms.set(i, Bitmap.createScaledBitmap(this.arrBms.get(i), this.width, this.height,
true));
        }
    }
    @Override
    public Bitmap getBm(){
        count++;
if(this.count == this.vFlap){
for(int i = 0; i<arrBms.size(); i++){
            if (i == arrBms.size()-1){
this.idCurrentBitmap = 0;
                break;
}else if (this.idCurrentBitmap == i){
idCurrentBitmap = i+1;
                break;
            }
        }
        count = 0;
    }
    if (this.drop<0){
        Matrix matrix = new Matrix();
```

```
matrix.postRotate(-25);
        return Bitmap.createBitmap(arrBms.get(idCurrentBitmap), 0, 0,
arrBms.get(idCurrentBitmap).getWidth(), arrBms.get(idCurrentBitmap).getHeight());
}else if (drop>=0){
        Matrix matrix = new Matrix();
        if (drop<70){
matrix.postRotate(-25+(drop*2));
}else{
matrix.postRotate(45);
        }
        return Bitmap.createBitmap(arrBms.get(idCurrentBitmap), 0, 0,
arrBms.get(idCurrentBitmap).getWidth(), arrBms.get(idCurrentBitmap).getHeight());
      }
      return this.arrBms.get(idCurrentBitmap);
   }

   public float getDrop() {
      return drop;
   }

   public void setDrop(float drop) {
this.drop = drop;
   }
}
```

Constants.java

```
package com.example.wingmaster;

public class Constants {
   public static int SCREEN_WIDTH;
   public static int SCREEN_HEIGHT;
}
```

GameView.java

```
package com.example.wingmaster;

import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.os.Handler;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;

import androidx.annotation.Nullable;

import java.util.ArrayList;

public class GameView extends View {
    private Bird bird;
    private Handler handler;
    private Runnable r;
    private ArrayList<Pipe>arrPipes;
    private int sumpipe, distance;
    private int score, bestscore = 0;
    private boolean start;
    private Context context;
    public GameView(Context context, @Nullable AttributeSetattrs) {
super(context, attrs);
this.context = context;
SharedPreferencessp = context.getSharedPreferences("gamesetting",
Context.MODE_PRIVATE);
        if (sp!=null){
```

```
bestscore = sp.getInt("bestscore", 0);
        }
        score = 0;
        start = false;
initBird();
initPipe();
        handler = new Handler();
        r = new Runnable() {
            @Override
            public void run() {
invalidate();
            }
        };
    }

    private void initPipe() {
sumpipe = 5;
        distance = 300*Constants.SCREEN_HEIGHT/1070;
arrPipes = new ArrayList<>();
        for (int i = 0; i<sumpipe; i++){
            if (i<sumpipe/2){
this.arrPipes.add(new
Pipe(Constants.SCREEN_WIDTH+i*((Constants.SCREEN_WIDTH+200*Constants.SCRE
EN_WIDTH/1080)/(sumpipe/2)),
                    0, 200*Constants.SCREEN_WIDTH/1080, Constants.SCREEN_HEIGHT/2));
                this.arrPipes.get(this.arrPipes.size()-
1).setBm(BitmapFactory.decodeResource(this.getResources(), R.drawable.pipe2));
this.arrPipes.get(this.arrPipes.size()-1).randomY();
}else{
this.arrPipes.add(new Pipe(this.arrPipes.get(i-sumpipe/2).getX(), this.arrPipes.get(i-
sumpipe/2).getY()
                    +this.arrPipes.get(i-sumpipe/2).getHeight() + this.distance,
200*Constants.SCREEN_WIDTH/1080, Constants.SCREEN_HEIGHT/2));
```

```
            this.arrPipes.get(this.arrPipes.size()-
1).setBm(BitmapFactory.decodeResource(this.getResources(), R.drawable.pipe1));
          }
       }
    }

    private void initBird() {
       bird = new Bird();
bird.setWidth(100*Constants.SCREEN_WIDTH/1000);
bird.setHeight(100*Constants.SCREEN_HEIGHT/1000);
bird.setX(100*Constants.SCREEN_WIDTH/1000);
bird.setY(Constants.SCREEN_HEIGHT/2-bird.getHeight()/2);
ArrayList<Bitmap>arrBms = new ArrayList<>();
arrBms.add(BitmapFactory.decodeResource(this.getResources(), R.drawable.bird1));
arrBms.add(BitmapFactory.decodeResource(this.getResources(), R.drawable.bird2));
bird.setArrtms(arrBms);
    }

    public void draw(Canvas canvas){
super.draw(canvas);
       if (start){
bird.draw(canvas);
for(int i = 0; i<sumpipe; i++){
            if                    (bird.getRect().intersect(arrPipes.get(i).getRect())||bird.getY()-
bird.getHeight()<0||bird.getY()>Constants.SCREEN_HEIGHT){
Pipe.speed = 0;
               MainActivity.txt_score_over.setText(MainActivity.txt_score.getText());
MainActivity.txt_best_score.setText("Best: " + bestscore);
MainActivity.txt_score.setVisibility(INVISIBLE);
               MainActivity.r1_game_over.setVisibility(VISIBLE);
            }
            if
(this.bird.getX()+this.bird.getWidth()>arrPipes.get(i).getX()+arrPipes.get(i).getWidth()/2
```

```
&&this.bird.getX()+this.bird.getWidth()<=arrPipes.get(i).getX()+arrPipes.get(i).getWidth()/2
+Pipe.speed
&&i<sumpipe/2){
                score++;
                if (score>bestscore){
bestscore = score;
SharedPreferencessp                =                context.getSharedPreferences("gamesetting",
Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sp.edit();
editor.putInt("bestscore", bestscore);
editor.apply();
                }
MainActivity.txt_score.setText(""+score);
            }
            if(this.arrPipes.get(i).getX() < -arrPipes.get(i).getWidth()){
this.arrPipes.get(i).setX(Constants.SCREEN_WIDTH);
if(i<sumpipe/2){
arrPipes.get(i).randomY();
}else{
arrPipes.get(i).setY(this.arrPipes.get(i-sumpipe/2).getY()
                        +this.arrPipes.get(i-sumpipe/2).getHeight() + this.distance);
                }
            }
this.arrPipes.get(i).draw(canvas);
        }
}else{
        if (bird.getY()>Constants.SCREEN_HEIGHT/2){
bird.setDrop(-15*Constants.SCREEN_HEIGHT/1920);
        }
bird.draw(canvas);
      }
handler.postDelayed(r, 10);
   }
```

```
    @Override
    public booleanonTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN){
bird.setDrop(-15);
        }
        return true;
    }

    public booleanisStart() {
        return start;
    }

    public void setStart(boolean start) {
this.start = start;
    }

    public void reset() {
MainActivity.txt_score.setText("0");
        score = 0;
initPipe();
initBird();
    }
}
```

MainActivity.java

```
package com.example.wingmaster;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.View;
```

```
import android.view.WindowManager;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.RelativeLayout;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    public static TextViewtxt_score, txt_best_score, txt_score_over;
    public static RelativeLayout r1_game_over;
    public static Button btn_start;
    private GameViewgv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,Window
Manager.LayoutParams.FLAG_FULLSCREEN);
DisplayMetrics ds = new DisplayMetrics();
this.getWindowManager().getDefaultDisplay().getMetrics(ds);
Constants.SCREEN_WIDTH = ds.widthPixels;
Constants.SCREEN_HEIGHT = ds.heightPixels;
setContentView(R.layout.activity_main);
txt_score = findViewById(R.id.txt_score);
txt_best_score = findViewById(R.id.txt_best_score);
txt_score_over = findViewById(R.id.txt_score_over);
        r1_game_over = findViewById(R.id.r1_game_over);
btn_start = findViewById(R.id.btn_start);
gv = findViewById(R.id.gv);
btn_start.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
gv.setStart(true);
txt_score.setVisibility(View.VISIBLE);
btn_start.setVisibility(View.INVISIBLE);
            }
```

```
        });
        r1_game_over.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
btn_start.setVisibility(View.VISIBLE);
                r1_game_over.setVisibility(View.INVISIBLE);
gv.setStart(false);
gv.reset();
            }
        });
    }
}
```

Pipe.java

```
package com.example.wingmaster;

import android.graphics.Bitmap;
import android.graphics.Canvas;

import java.util.Random;

public class Pipe extends BaseObject {
    public static int speed;
    public Pipe(float x, float y, int width, int height){
super(x, y, width, height);
        speed = 10*Constants.SCREEN_WIDTH/1080;
    }
    public void draw(Canvas canvas){
this.x-=speed;
canvas.drawBitmap(this.bm, this.x, this.y, null);
    }
    public void randomY(){
        Random r = new Random();
```

```
this.y = r.nextInt((0+this.height/4)+1)-this.height/4;
    }

    @Override
    public void setBm(Bitmap bm) {
        this.bm = Bitmap.createScaledBitmap(bm, width, height, true);
    }
}
```

CHAPTER 5

RESULTS

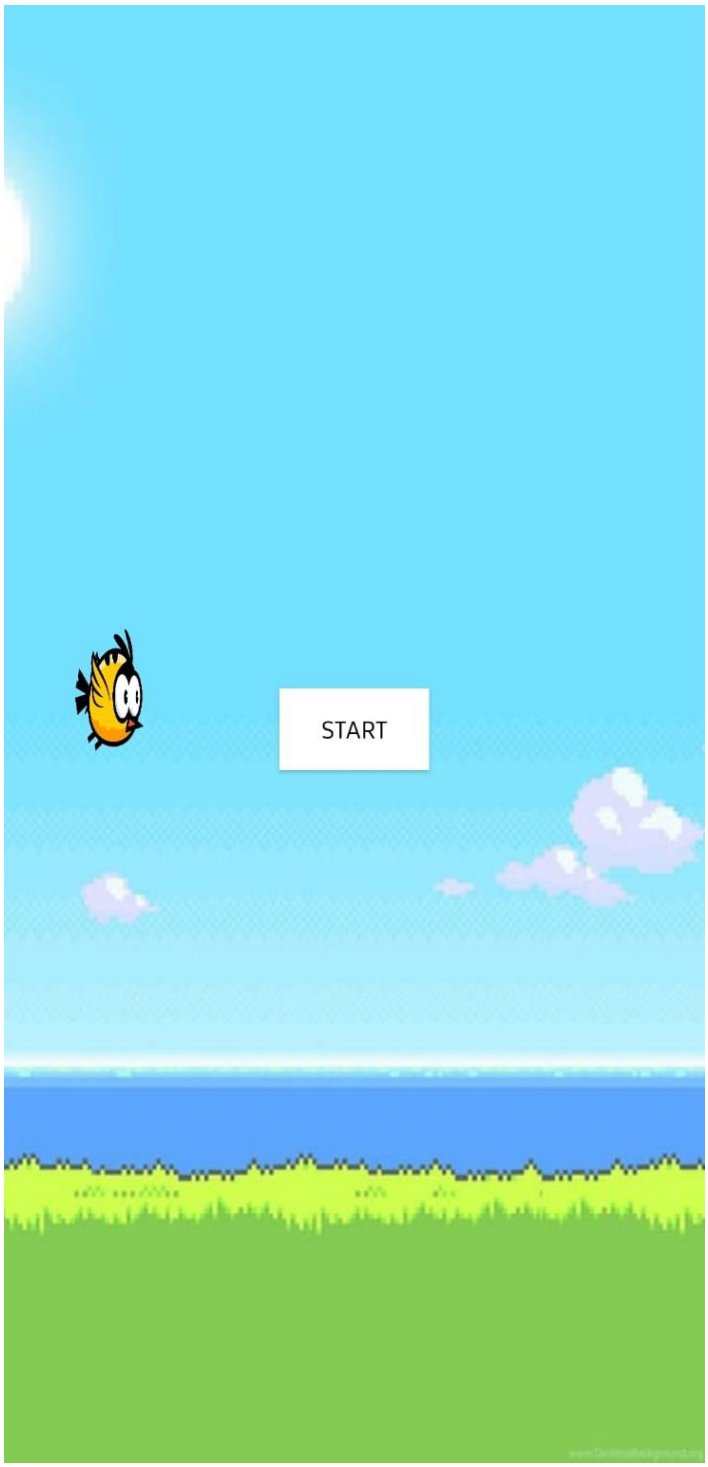


Fig 5.1: Start Screen

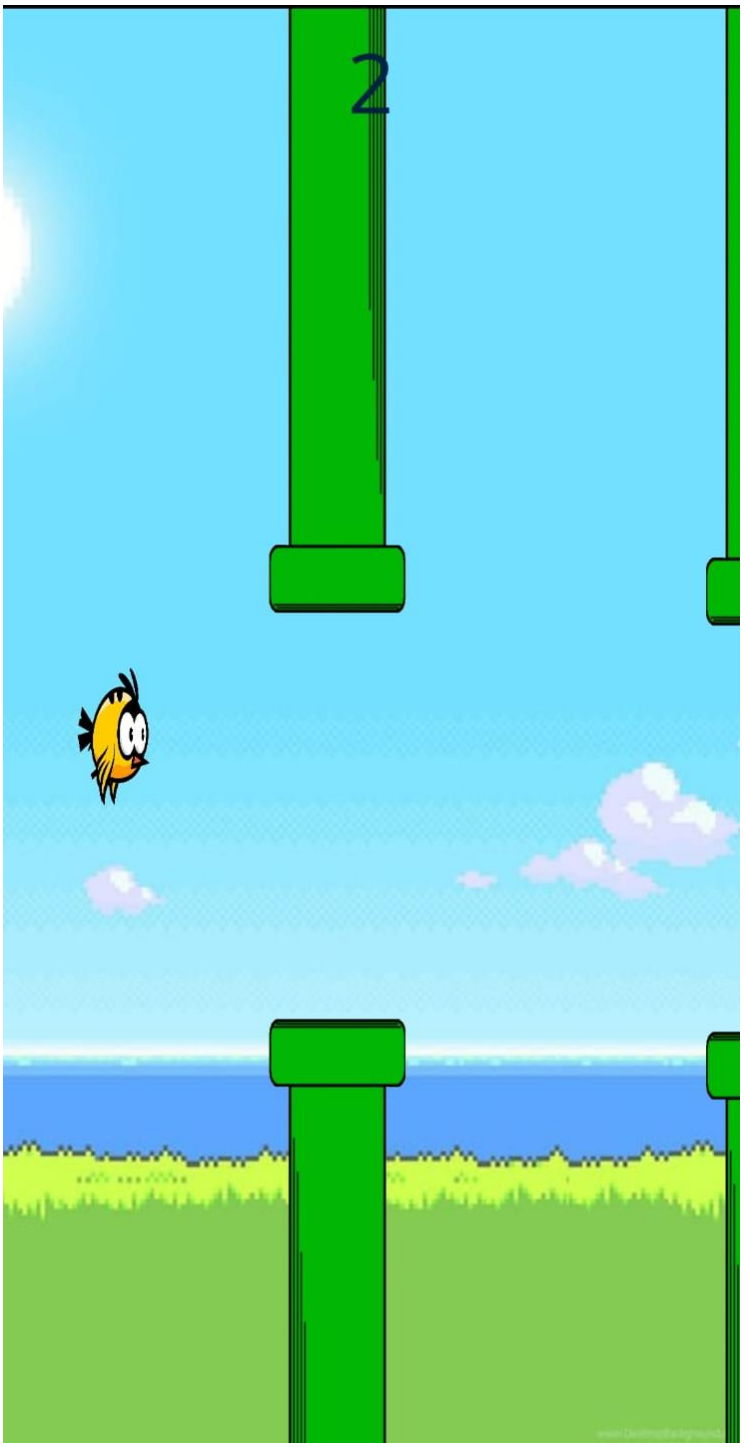


Fig 5.2: Game Screen

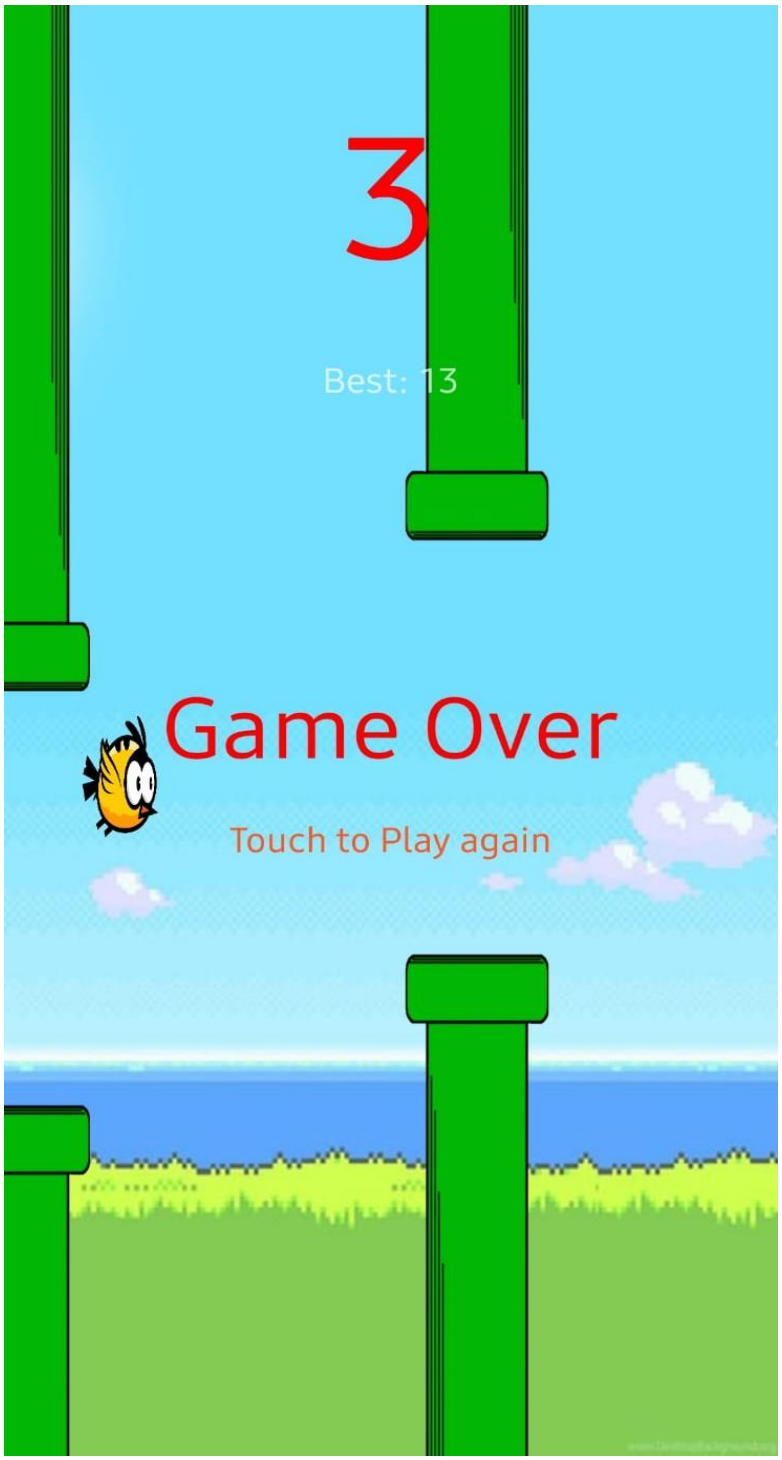


Fig 5.3:Game Over Screen

CONCLUSION & FUTURE ENHANCEMENT

Conclusion

It's a beautifully manipulative game that sells advertising against your base-level tendencies to keep trying at something that seems within grasp, but rather is designed to mask its clear and utter propensity to grind you into failure. It's a work of a genius.

From a mechanics perspective, you simply tap to keep your bird afloat and alter your frequency of taps to maneuver the bird through different size obstacles that, if touched, end your game immediately.

Future Enhancement

- We can add levels and increase the difficulty of the game.
- We can add different bird characters and make it more interesting.
- We can add sounds and make the game more fun.
- We can make it online and play with our friends making a scoreboard.

BIBLIOGRAPHY

Book References

[1] Google Developer Training, "Android Developer Fundamentals Course – Concept Reference", Google Developer Training Team, 2017.

[2] Erik Hellman, "Android Programming – Pushing the Limits", 1st Edition, Wiley India Pvt Ltd, 2014. ISBN-13: 978-8126547197.

[3] Dawn Griffiths and David Griffiths, "Head First Android Development", 1st Edition, O'Reilly

[4] SPD Publishers, 2015. ISBN-13: 978-9352131341.

[5] Bill Phillips, Chris Stewart and Kristin Marsicano, "Android Programming: The Big Nerd Ranch Guide", 3rd Edition, Big Nerd Ranch Guides, 2017. ISBN-13: 978-0134706054.

Web References

[1] Android Developers https://developer.android.com/

[2] Android Tutorial https://www.tutorialspoint.com/android/index.htm

[3] Android Tutorial https://www.w3schools.in/category/android-tutorial/

[4] Java Tutorial https://www.javatpoint.com/java-tutorial

[5] Java Tutorial https://www.w3schools.com/java/