UNIVERSITY OF MUMBAI

A PROJECT REPORT ON

"FLAPPY BIRD CLONE "

(Android game application)

SUBMITTED BY

MR. DANIEL SIVAPRASAD

UNDER THE GUIDANCE OF

(PROF. PRAGATI MESTRY)

`

Late Shri. Vishnu Waman Thakur Charitable Trust's

VIVA INSTITUTE OF TECHNOLOGY

Shirgaon, Virar(East)

2021-22

Late Shri. Vishnu Waman Thakur Charitable Trust's

VIVA INSTITUTE OF TECHNOLOGY

Shirgaon, Virar(East)

CERTIFICATE

This is to certify that

MR. DANIEL SIVAPRASAD

Has satisfactorily completed the project entitled

FLAPPY BIRD CLONE

(Android game application)

Towards the partial fulfillment of the
MASTER OF COMPUTER APPLICATION (MCA)
As laid by University of Mumbai.

Principal	External Examiner	Internal Guide

DECLARATION

I "Mr. Daniel Sivaprasad" hereby declare that I myself have completed the project under the guidance of **Prof. Pragati Mestry**. I on my own have designed the system and have done all the programming required.

It may require some modifications in the future as per the user's requirements. From practical implementation point of view, flexibility in the changes have incorporated in the package.

I am sure that I can do any kind of modification suggested while practical implementation by modifying file design or the program code if necessary.

DANIEL SIVAPRASAD

(author)

PLAGIARISM REPORT

SmallSEOTools

PLAGIARISM SCAN REPORT

Words		Date	March 04, 2022
Characters	6513	Excluded URL	

2% Plagiarism	98% Unique	1 Plagiarized Sentences	40 Unique Sentences

Content Checked For Plagiarism

UNIVERSITY OF MUMBAI
A PROJECT REPORT ON
“FLAPPY BIRD CLONE “
(Android game application)
SUBMITTED BY
MR. DANIEL SIVAPRASAD
UNDER THE GUIDANCE OF
(PROF. PRAGATI MESTRY
Late Shri. Vishnu Waman Thakur Charitable Trust’s
VIVA INSTITUTE OF TECHNOLOGY Shirgaon, Virar(East)
2021-22
Late Shri. Vishnu Waman Thakur Charitable Trust’s
VIVA INSTITUTE OF TECHNOLOGY
Shirgaon, Virar(East)
CERTIFICATE
This is to certify that
MR. DANIEL SIVAPRASAD
Has satisfactorily completed the project entitled
FLAPPY BIRD CLONE
(Android game application)
Towards the partial fulfillment of the
MASTER OF COMPUTER APPLICATION (MCA)
As laid by University of Mumbai.
_____ _____ _____
Principal External Examiner Internal Guide
DECLARATION
I “Mr. Daniel Sivaprasad” hereby declare that I myself have completed the project under the guidance of Prof. Pragati Mestry. I on my own have designed the system and have done all the programming required.
It may require some modifications within the future as per the user’s requirements. From practical implementation point of view flexibility in the changes have incorporated in the package.
I am sure that I can do any kind of modification suggested while practical implementation by modifying file design or the program code if necessary.
DANIEL SIVAPRASAD
ACKNOWLEDGEMENT
First and foremost. I bow my head to the Almighty for being my light and his showers of blessing during this mini project. We extend our gratitude to Dr. ARUN KUMAR, Principal. Viva Institute of Technology, to provide all the resources to do this mini project work.
I record my sincere thanks and respect to our Head of Department, Prof. Chandani Patel, Department of Master of Computer Application, Viva Institute of Technology, with her good attention and outstanding guidance in all the work of

Page 1 of 3

ACKNOWLEDGEMENT

First and foremost. I bow my head to the Almighty for being my light and His showers of blessing during this mini project. I extend my gratitude to **Dr. ARUN KUMAR**, Principal. Viva Institute of Technology, to provide all the resources to do this mini project work. I record my sincere thanks and respect to our Head of Department, **Prof. Chandani Patel**, Department of Master of Computer Application, Viva Institute of Technology, with her good attention and outstanding guidance in all the work of my project.

I am indebted to my project coordinator and project guide **Prof. Pragati Mestry**, Department of Master of Computer Application, Viva Institute of Technology for her guidance and important suggestions for all the ways to complete my project.

I extend my deepest gratitude to all the other staff for their support and all assistance during the study and course work of the project. Finally, I truly thank my parents and friends for giving us valuable and suggestion for the improvement of my project.

ABSTRACT

- This project is a fully functional gaming application that runs a gaming object called flappy bird which should successfully manage the following objectives:
    - Moving Pipes
    - Score Counting
    - Collision detection

- Flappy Bird is a side scroll game based on 2D Graphics.

- The goal of this project is to control the bird, avoid it and pass as many obstacles as possible. This will run indefinitely until the bird hits an obstacle or the ground.

INDEX

Sr. No.	Contents	Page No.
1	**Introduction**	
	1.1 Introduction	10
	1.1.1 Problem definition	11
	1.1.2 Objectives of Project	11
	1.1.3 Scope of Project	11
	1.2 Technical Details	
	1.2.1Overview of Front End	12
	1.2.2Overview of Back End	12
2	**System Study and Planning**	13
	2.1 System Study	
	2.1.1 Existing System	14
	2.1.2 Disadvantages of Existing system	14
	2.1.3 Proposed System	14
	2.2 System Planning and Schedule	
	2.2.1 Activity Sheet	15
	2.2.2 Waterfall Model	16
	2.2.3 Gantt Chart	17
3	**System Design**	18
	3.1 Software Requirement Specification (SRS)	19
	3.1.1 Introduction of SRS	20
	3.1.1.1 Purpose	
	3.1.1.2 Definition	
	3.1.2 Overall Descriptions	21
	3.1.2.1 Software Requirements	
	3.1.2.2 Hardware Requirements	
	3.1.2.3 Design & Implementation	
	3.1.3 Functional Requirements	22
	3.1.4 Non-Functional Requirements	23
	3.2 Detailed life Cycle of the Project	
	3.2.1 Use Case Diagram	24
	3.2.2 Class Diagram	25

	3.2.3 Flowchart diagram 3.2.4 Screen Layout 3.2.5 Lines of Code	26 27 30
4	**Testing** (Any Model explanation in terms of your project) 4.1 Requirement Traceability Table	34 35
5	**Conclusion**	36
6	**Future Enhancements**	37
7	**References**	38

INTRODUCTION

1.1 INTRODUCTION

Flappy Bird was a side scroll game developed by Dong Nguyen. Although the game is based on 2D Graphics, is very popular on both Android and iOS platforms. The purpose of the game is simple, try flying a bird, called “Faby”. between the green pipes without hitting them. Due to its addictive nature, the game developer has suspended the game on both channels (Google Play Store and App Store) but an updated version of Amazon Fire TV launched on Amazon App Store. Time thunderstorms in most similar simulations and integrated versions have been released. In this project, I took the game concept and developed Flappy Bird Game using libGDX framework and Android Studio. Since LibGDX is a free game development and open-source software written in Java programming language with specific C and C ++ component-based code functions. It allows desktop and mobile game development using the same code base platform. I do not want to invest too much into the project so I developed Flappy Bird Game using Android Studio platform which is also a free IDE for and android app development.

1.1.1
PROBLEM DEFINITION

To create a fully functional gaming application that runs a 2d game called flappy bird which should successfully manage the following objectives:

- Moving Pipes
- Score Counting
- Collision detection

1.1.2
OBJECTIVE OF PROJECT

The goal of this game is to control the bird, avoid it and pass as many obstacles as possible. This will run indefinitely until the bird hits an obstacle or the ground.

1.1.3
SCOPE OF PROJECT

At the end of January 2014, it was the most downloaded free game in the App Store for iOS. During this period, its developer said that Flappy Bird was earning $50,000 a day from in-app advertisements as well as sales.

1.2 TECHNICAL DETAILS

1.2.1
SOFTWARE OVERVIEW

Flappy Bird is to be developed under the Android operating system using libGDX which is a cross-platform java game development framework based on OpenGL (ES) that works on Windows, Linux, macOS, Android, your browser and iOS.
ANDROID STUDIO: IDE (Integrated Development Environment) contains a code editor, a compiler or interpreter, and a debugger, accessed through a single graphical user interchange (GUI).

1.2.2
HARDWARE OVERVIEW

Flappy Bird is a mobile gaming application designed specifically for the Android platform and is functional on both mobile smart phones and tablets. Flappy Bird has been developed for Android Version 2.2 and all subsequent releases. This project was built on windows 64 bit processor with 8 GB of RAM.

SYSTEM STUDY AND PLANNING

2.1 SYSTEM STUDY.

2.1.1
EXISTING SYSTEM

Flappy Bird is an arcade style game where the player controls the Faby bird, moving continuously to the right. The player is tasked with navigating Faby with a pair of pipelines with equal spaces placed on a random high point. Faby goes down automatically and only up when the player touches the touch screen. Each successful pass with a pair of pipes rewards the player with one point. A collision with a pipe or a floor ends a game.

2.1.2
DISADVANTAGE OF EXISITING SYSTEM

We all know that, Flappy Bird was removed from both the App Store and Google Play by its creator on February 10, 2014. The main disadvantage of the existing system is that, it quickly becomes addictive or frustrating when you start trying to compete with others players around the world. And that was the main reason why the creator of the game removed it from the play store and apple store as he felt guilty about what he considered to be addictive and overuse.

2.1.3
PROPOSED SYSTEM

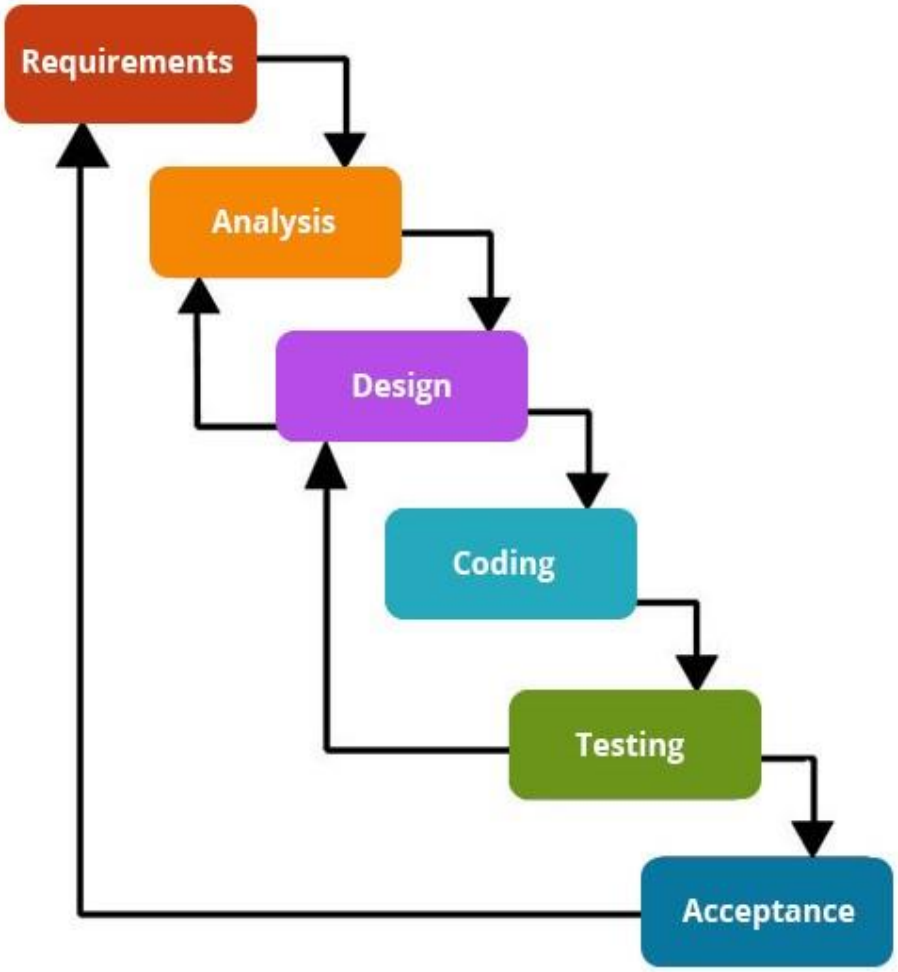
In the proposed game design, as soon as the game begins, obstacles will keep appearing from the right side of the screen and move leftwards which will make bird seem to be flying in the forward direction. The goal of this game would be to control the bird, dodging and passing it through as many obstacles as possible. Will try to add time limitations in the game for every day so that the users should not get too addicted to this game.

2.2 SYSTEM PLANNING AND SCHEDULING.

2.2.1
ACTIVITY SHEET

Sr. No.	Activity			Sign
		Planned Date	Actual Date	
1.	Preliminary Investigation	17/01/2022	18/01/2022	
2.	System Study and Analysis	18/01/2022	20/01/2022	
3.	System Coding and Report	31/01/2022	01/02/2022	
4.	System Testing	23/02/2022	26/02/2022	
5.	Project Submission	04/03/2022	04/03/2022	

2.2.2
WATERFALL MODEL



2.2.3
GANTT CHART

Period / Process	17/01/22 to 23/01/22 (Week 1)	24/01/22 to 30/01/22 (Week 2)	31/01/22 to 21/02/22 (Week 3 & Week 4)	22/02/22 to 04/03/22 (Week 5)
Preliminary Investigation				
System Analysis				
System Design				
System Coding				
Testing & Evaluation				
Project Submission				

- Planned Date

- Actual Date

SYSTEM DESIGN

3.1 SOFTWARE REQUIREMENT SPECIFICATION

Software (SRS) specification document captures a complete description of how the system is expected to operate. It is usually signed at the end of the needs engineering phase. The production of the requirements section of the software development process is Software Requirements Specifications (SRS) (also called the requirements document). This report lays the foundation for software engineering activities and builds on where all the requirements are requested and analyzed. SRS is an official report, acting as a representative software that enables customers to review whether (SRS) complies with their needs. Also, it includes system user requirements as well as detailed specification of system requirements.

The diagram below shows the different types of requirements captured during SRS.

3.1.1
INTRODUCTION OF SRS

3.1.1.1 Purpose

This Software Requirements Specification (SRS) document is intended to give a complete overview of Android game project Flappy Bird, including the game mechanics, user interface and story therein. The SRS document details all features upon which Flappy Bird have currently decided with reference to the manner and importance of their implementation.

3.1.1.2 Definition

Flappy Bird is an arcade-style game in which the player controls the bird Faby, which moves persistently to the right. The player is tasked with navigating Faby through pairs of pipes that have equally sized gaps placed at random heights.

3.1.2
OVERALL DESCRIPTIONS

3.1.2.1 Software Requirements

- Windows 64/32 bit.
- ANDROID STUDIO: IDE (Integrated Development Environment) contains a code editor, a compiler or interpreter, and a debugger, accessed through a single graphical user interchange (GUI).
- libGDX: A cross-platform game development framework used for developing this project.

3.1.2.2 Hardware Requirements (minimum)

- Pentium 3 processor.
- Processor speed of 300 MHZ and above.

3.1.2.3 Design and Implementation Constraints

Design
This section contains the requirements for compiling and creating a game software architecture. The phase will be reviewed several times during the implementation phase. My design process is to first create all static interfaces and buttons associated with each interface. Then I started to create pipe and background shifting, bird falling or jumping and score tracking,

Implementation
This section contains the repetitive process of creating a game. The whole structure should be used, and then the actual code will be written, as well as tests and complete documentation. The start-up phase is divided into three sprints.

3.1.3
FUNCTIONAL REQUIREMENTS

3.1.3.1 2D Animation
Animation is a complex subject in the game system. Animation is a fast-paced animation that creates the illusion of movement. Java games are expected to run on multiple operating systems with different hardware definitions.

3.1.3.2 Objective Selection
We build a flying bird object until any collision occurs and the bird flies to the wall posts starting from the top and bottom of the screen.

3.1.3.3 Moving Wall
The wall moves forward and will come randomly in size and distances. The bird flies through the wall.

3.1.3.4 Collision Detection
When a bird touches anything on the wall it creates a collision. Collision detection is one of the key functions of the game. If the bird touches any wall (pipes) the game will end.

3.1.3.5 Score Counting
Counting points is of interest to the user. By earning points the player knows his performance. If a bird crosses a pipe without colliding or does not fall to the ground its points rise.

3.1.4
NON-FUNCTIONAL REQUIREMENTS

3.1.4.1 Performance Requirements

Based on the capabilities of current phones and the Android system, performance should not be a problem. However, phones with weak computer hardware may cause inconvenience and slow performance. Game design will be done correctly to give a pleasant feeling to all Android phones, regardless of hardware. The performance of the game will be simple enough, but not trivial, and the graphics will not be too detailed so the system will not slow down.

3.1.4.2 Safety Requirements

Flappy Bird will not affect or harm any other applications installed on the player's phone. The game also will not cause any heat to the player's phone; therefore, the internal parts of the phone will not be damaged. Flappy Bird should not be played when a player's attention is divided between multiple tasks to prevent possible injury to the player.

3.1.4.3 Security Requirements

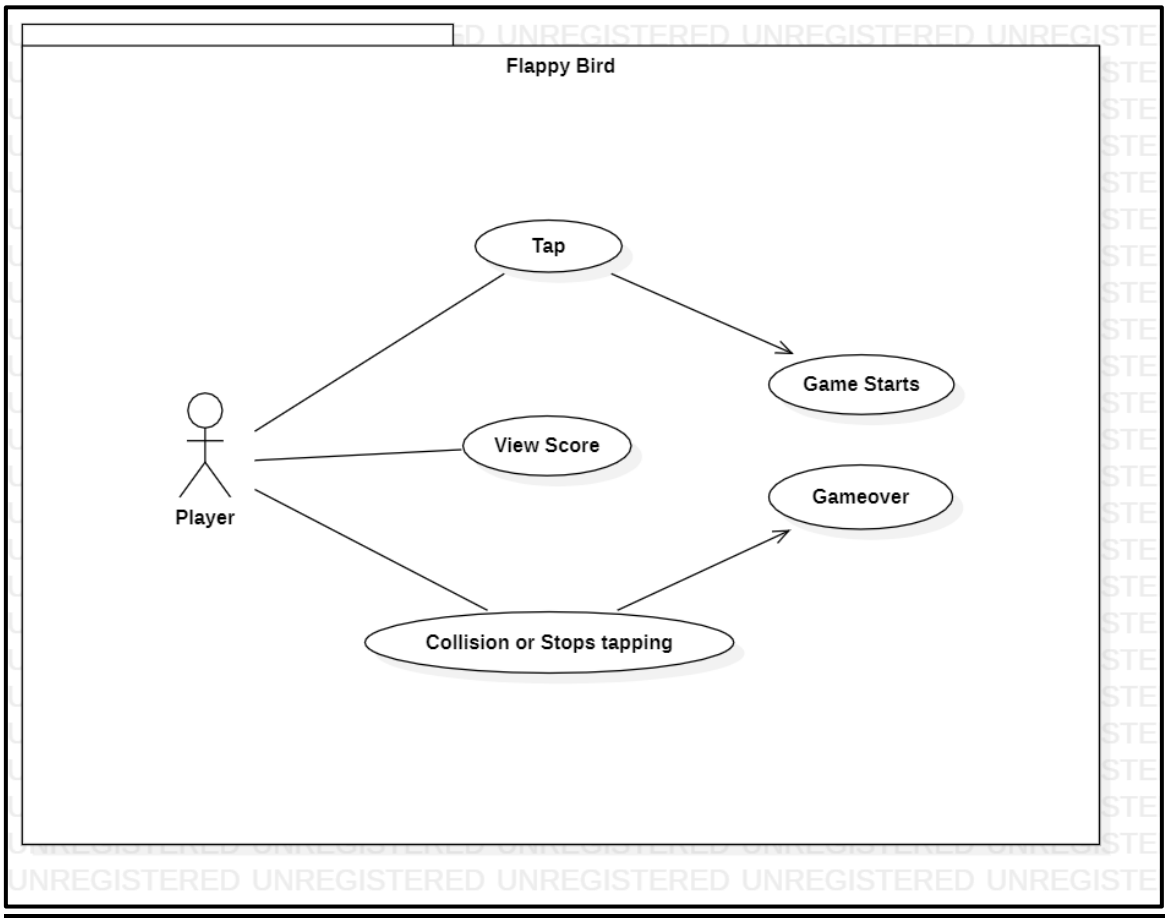
Flappy Bird will not ask for any personal information from the player and thus will not be able to compromise that information. No player authentication is required to play Flappy Bird. The player simply downloads the app to start playing Flappy Bird. That being said, anyone with access to the player's phone will have the ability to play Flappy Bird. If any unauthorized player gets the first player's phone, that unauthorized player will be able to play Flappy Bird. It is the Player 's job to make sure that no unauthorized player / person will have access to his / her phone.

3.1.4.4 Software Quality Attributes

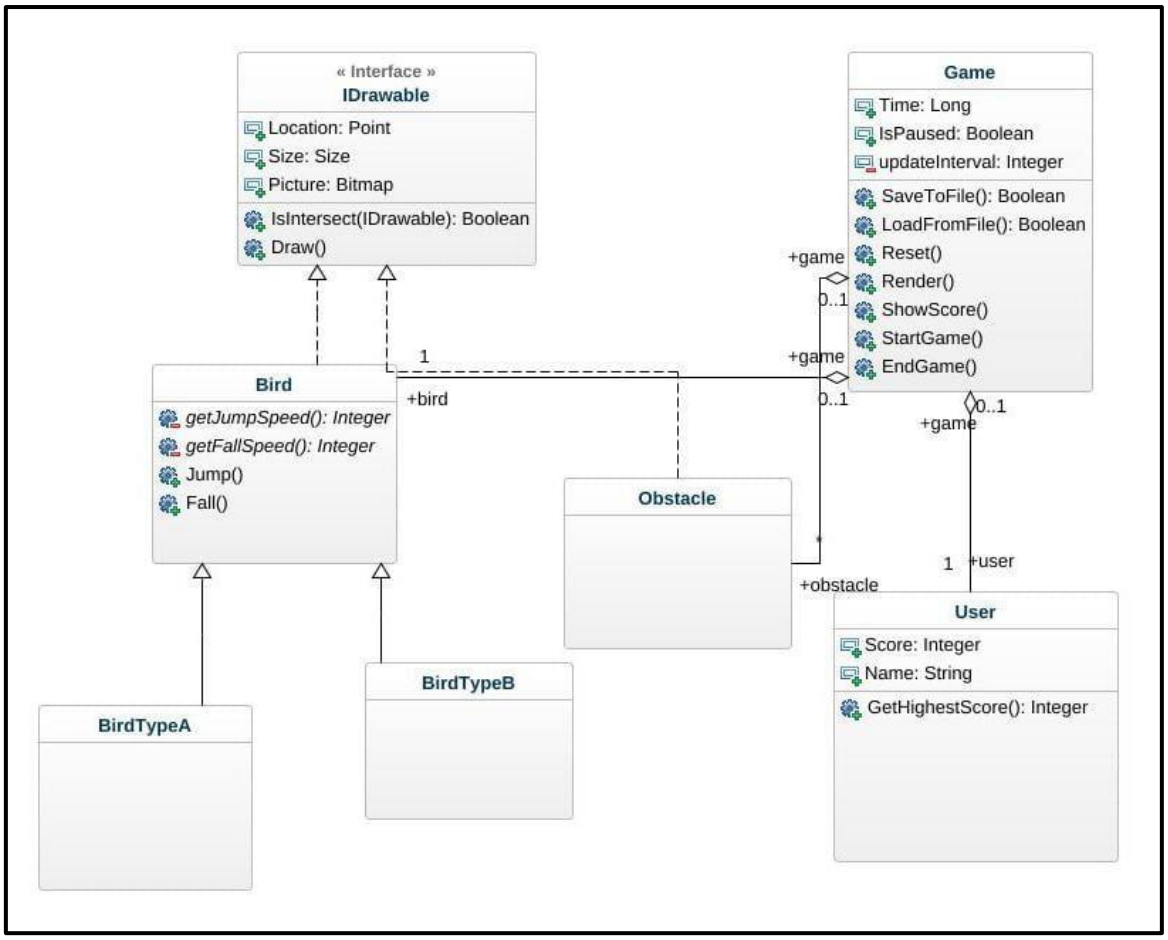
To ensure integrity and fairness, Flappy Bird will respond to player instructions in a timely manner. My game focuses on both ease of use and ease of learning while not relying on one or the other. Flappy Bird should be a game any player can download and play immediately without spending a lot of time gaining controls. The controls will be simple and easy to use and the player will not be bombed by using all the controls at once in the first levels.

3.2. LIFE CYCLE

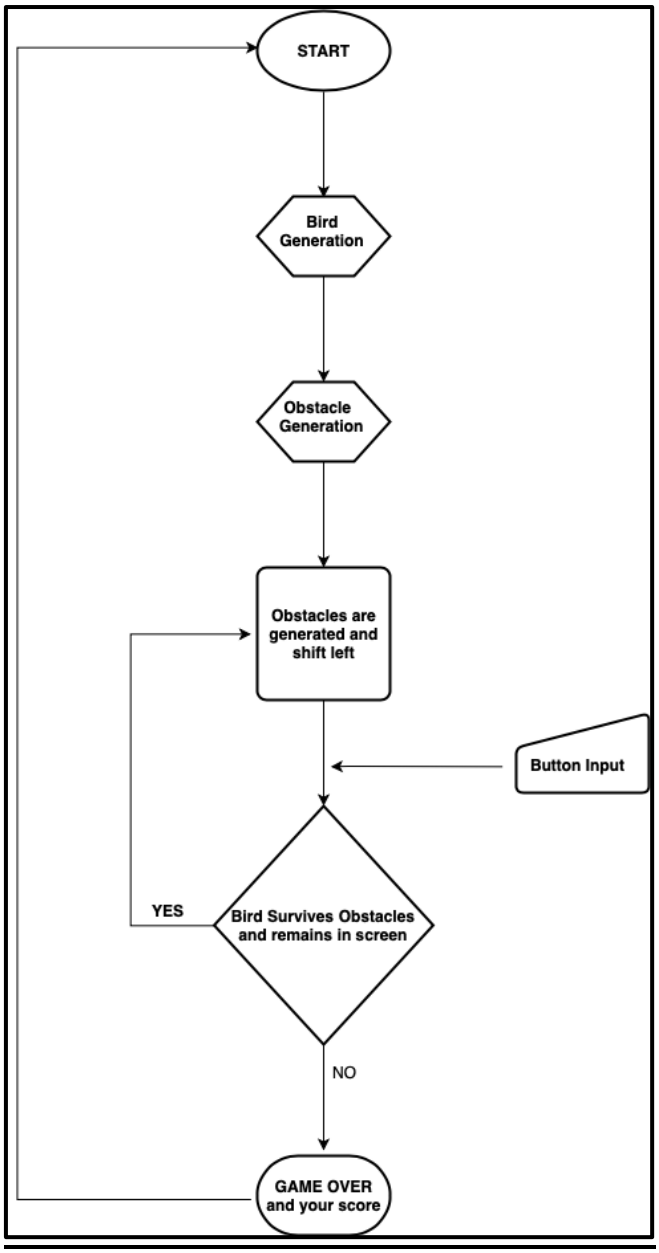
3.2.1
USE CASE DIAGRAM



3.2.2
CLASS DIAGRAM

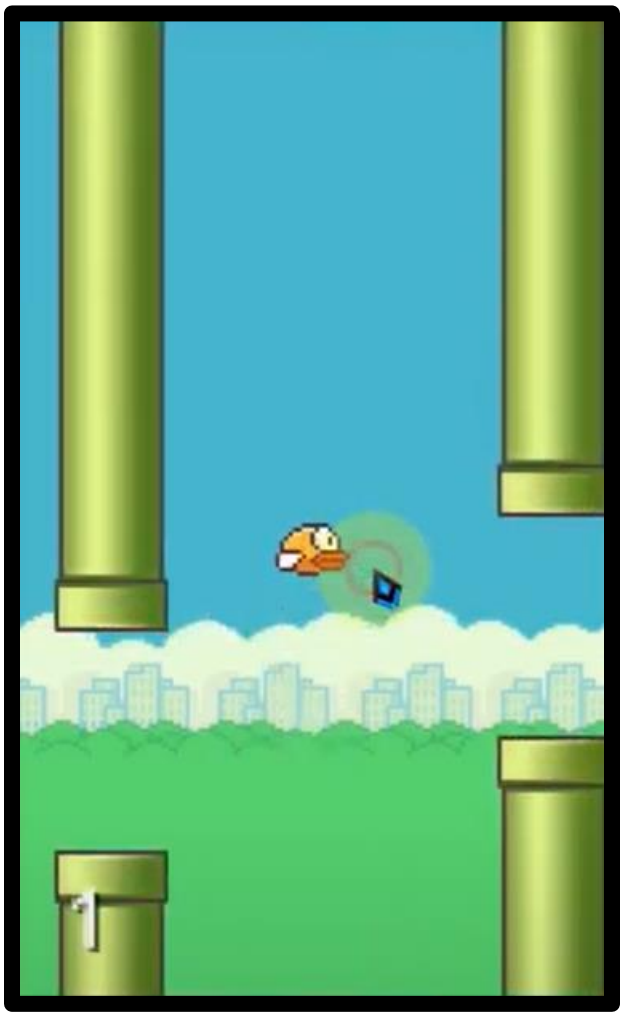


3.2.3
FLOWCHART DIAGRAM



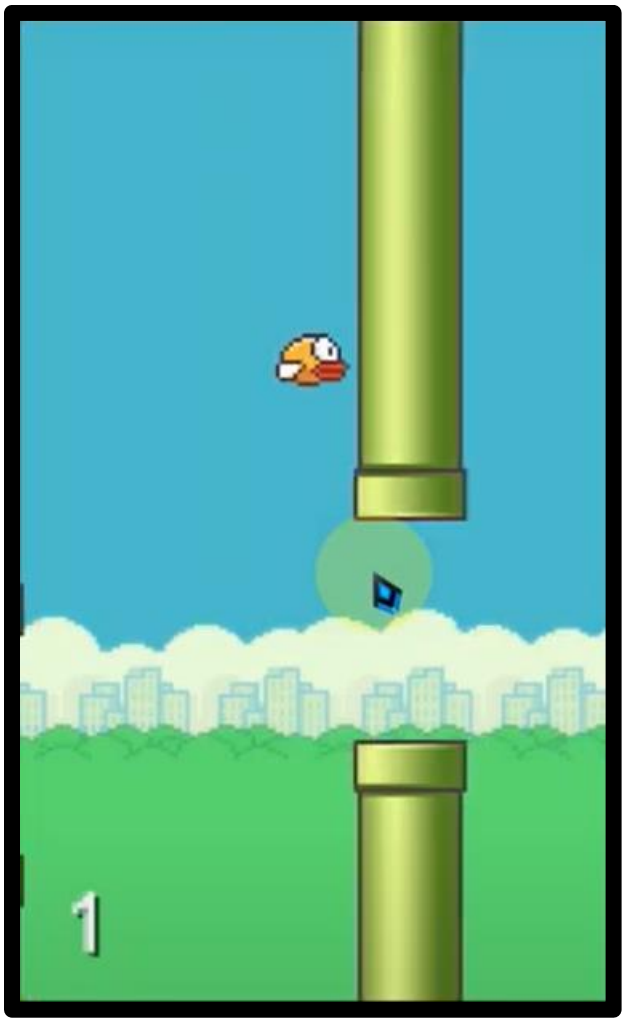
3.2.4
SCREEN LAYOUT

1. Flying bird



3.2.4
SCREEN LAYOUT

2. Before Collision



3.2.4
SCREEN LAYOUT

3. Game Over Screen

3.2.5
LINES OF CODE

```
package com.danielsp.flappybird;

import com.badlogic.gdx.ApplicationAdapter;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.graphics.g2d.BitmapFont;
import com.badlogic.gdx.graphics.g2d.SpriteBatch;
import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
import com.badlogic.gdx.math.Circle;
import com.badlogic.gdx.math.Intersector;
import com.badlogic.gdx.math.Rectangle;


import java.util.Random;

public class FlappyBird extends ApplicationAdapter {
    SpriteBatch batch;
    Texture background;
    //ShapeRenderer shapeRenderer;
    Texture gameover;

    Texture[] birds;
    int flapState = 0;
    float birdY = 0;
    float velocity = 0;
    Circle birdCircle;
    int score = 0;
    int scoringTube = 0;
    BitmapFont font;

    int gameState = 0;
    float gravity = 2;
    Texture topTube;
    Texture bottomTube;
    float gap = 400;
    float maxTubeOffset;
    Random randomGenerator;

    float tubeVelocity = 2;

    int numberOfTubes = 4;
    float[] tubeX = new float[numberOfTubes];
    float[] tubeOffset = new float[numberOfTubes];
    float distanceBetweenTheTube;
    Rectangle[] topTubeRectangles;
    Rectangle[] bottomTubeRectangles;
```

```
@Override
    public void create() {
        batch = new SpriteBatch();
        background = new Texture("bg.png");
        gameover = new Texture("gameover.png");
        //shapeRenderer = new ShapeRenderer();
        birdCircle = new Circle();

        birds = new Texture[2];
        birds[0] = new Texture("bird.png");
        birds[1] = new Texture("bird2.png");

        font = new BitmapFont();
        font.setColor(Color.WHITE);
        font.getData().setScale(10);

        topTube = new Texture("toptube.png");
        bottomTube = new Texture("bottomtube.png");

        maxTubeOffset = Gdx.graphics.getHeight() / 2 - gap / 2 - 100;
        randomGenerator = new Random();
        distanceBetweenTheTube = Gdx.graphics.getWidth() * 3 / 4;
        topTubeRectangles = new Rectangle[numberOfTubes];
        bottomTubeRectangles = new Rectangle[numberOfTubes];

        startGame();

    }

    public void startGame()
    {
        birdY = Gdx.graphics.getHeight() / 2 -
birds[flapState].getHeight() / 2;
        for (int i = 0; i < numberOfTubes; i++) {
            tubeOffset[i] = (randomGenerator.nextFloat() - 0.5f) *
(Gdx.graphics.getHeight() - gap - 200);
            tubeX[i] = Gdx.graphics.getWidth() / 2 - topTube.getWidth()
/ 2 + Gdx.graphics.getWidth() + i * distanceBetweenTheTube;

            topTubeRectangles[i] = new Rectangle();
            bottomTubeRectangles[i] = new Rectangle();
        }
    }

    @Override
    public void render() {

        batch.begin();
        batch.draw(background, 0, 0, Gdx.graphics.getWidth(),
Gdx.graphics.getHeight());

        if (gameState == 1) {

            if (tubeX[scoringTube] < Gdx.graphics.getWidth() / 2) {
                Gdx.app.log("Score", String.valueOf(score));
                score++;
```

```
                if (scoringTube < numberOfTubes - 1) {
                    scoringTube++;
                } else {
                    scoringTube = 0;
                }
            }

            if (Gdx.input.justTouched()) {
                velocity = -30;

            }

            for (int i = 0; i < numberOfTubes; i++) {

                if (tubeX[i] < -topTube.getWidth()) {
                    tubeX[i] += numberOfTubes * distanceBetweenTheTube;
                    tubeOffset[i] = (randomGenerator.nextFloat() -
0.5f) * (Gdx.graphics.getHeight() - gap - 200);
                } else {
                    tubeX[i] = tubeX[i] - tubeVelocity;


                }
                batch.draw(topTube, tubeX[i], Gdx.graphics.getHeight()
/ 2 + gap / 2 + tubeOffset[i]);
                batch.draw(bottomTube, tubeX[i],
Gdx.graphics.getHeight() / 2 - gap / 2 - bottomTube.getHeight() +
tubeOffset[i]);

                topTubeRectangles[i] = new Rectangle(tubeX[i],
Gdx.graphics.getHeight() / 2 + gap / 2 + tubeOffset[i],
topTube.getWidth(), topTube.getHeight());
                bottomTubeRectangles[i] = new Rectangle(tubeX[i],
Gdx.graphics.getHeight() / 2 - gap / 2 - bottomTube.getHeight() +
tubeOffset[i], bottomTube.getWidth(), bottomTube.getHeight());


            }


            if (birdY > 0) {
                velocity = velocity + gravity;
                birdY -= velocity;
            }else {
                gameState = 2; // 2 is the gameover state
            }
        } else if (gameState ==0){
            if (Gdx.input.justTouched())
            {
                gameState = 1;
            }
        }else if (gameState == 2)
        {
            batch.draw(gameover , Gdx.graphics.getWidth()/2 -
gameover.getWidth()/2 , Gdx.graphics.getHeight()/2 -
gameover.getHeight());
```

```
if (Gdx.input.justTouched())
            {
                gameState = 1;
                startGame();
                score = 0;
                scoringTube = 0;
                velocity = 0;

            }
        }

        if (flapState == 0) {
            flapState = 1;
        } else {
            flapState = 0;
        }


        batch.draw(birds[flapState], Gdx.graphics.getWidth() / 2 -
birds[flapState].getWidth() / 2, birdY);
        font.draw(batch , String.valueOf(score) , 100 , 200);


        batch.end();

        birdCircle.set(Gdx.graphics.getWidth() / 2, birdY +
birds[flapState].getHeight() / 2, birds[flapState].getWidth() / 2);

//        shapeRenderer.begin(ShapeRenderer.ShapeType.Filled);
//        shapeRenderer.setColor(Color.RED);
//        shapeRenderer.circle(birdCircle.x , birdCircle.y ,
birdCircle.radius);

        for (int i = 0; i < numberOfTubes; i++) {
            //shapeRenderer.rect(tubeX[i] , Gdx.graphics.getHeight() /
2 + gap / 2 + tubeOffset[i] , topTube.getWidth() ,
topTube.getHeight());
            //shapeRenderer.rect(tubeX[i] , Gdx.graphics.getHeight() /
2 - gap / 2 - bottomTube.getHeight() + tubeOffset[i] ,
bottomTube.getWidth() , bottomTube.getHeight());

            if (Intersector.overlaps(birdCircle, topTubeRectangles[i])
|| Intersector.overlaps(birdCircle, bottomTubeRectangles[i])) {
                gameState = 2;
            }
        }

        //shapeRenderer.end();

    }


}
```

TESTING

4.1
REQUIREMENT TRACEABILITY TABLE

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.

Req1 = 2D Animation

Req2 = Objectives Selection

Req3 = Moving wall

Req4 = Collision Detection

Req5 = Moving Background

Req6 = Score Counting

Requirement Traceability Table

Requirement Test Case	**Req1**	**Req2**	**Req3**	**Req4**	**Req5**	**Req6**
Test Case 1						
Test Case 2	✓					
Test Case 3	✓	✓				
Test Case 4	✓	✓				
Test Case 5	✓	✓	✓	✓		
Test Case 6	✓	✓	✓	✓	✓	✓

5. CONCLUSIONS

Games are a type of media that is often associated with negative health effects. However, when games are played with balance and consideration, they are an effective source of stress relief and a source of mental health development and community development. Mobile games themselves are a form of modern entertainment. They get involved and focus on a different level than normal board games and other forms of entertainment. The player actively contributes to the level of satisfaction he/she receives from this device and thus is more invested and willing to participate in the features of the mobile game. The amount of playing time is also an important factor in the results of games. While a lot of play time can have negative consequences, moderate play can be healthy, enjoyable, and educational.

6. FUTURE ENHANCEMENTS

- Bird skins and backgrounds will change according to the levels and preferences of the user.
- The status and history will be saved and will be displayed flexibly on a graph where the user can see his or her full performance.
- Pause and Resume

7. REFERENCES

- https://en.wikipedia.org/wiki/Flappy_Bird
- https://libgdx.com/
- Java Tutorial | Learn Java Programming - javatpoint
- https://flappybird.io/

* * * * *