PROJECT REPORT FOR BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

FLAPPY BIRD

AYUSH BHATTARAI [5-2-1113-10-2019]

DEWANAND GIRI [5-2-1113-126-2019]

OM KC [5-2-1113-23-2019]

UDAY SHRESTHA [5-2-1113-46-2019]

SAMRIDDHI COLLEGE

DEPARTMENT OF SCIENCE AND TECHNOLOGY

APRIL 25, 2021

PROJECT REPORT FOR BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

FLAPPY BIRD

SUPERVISED BY Er. MOHAN BHANDARI
FACULTY, SAMRIDDHI COLLEGE

A REPORT SUBMITTED
FOR
SECOND SEMESTER OPP PROJECT

AYUSH BHATTARAI [5-2-1113-10-2019]
DEWANAND GIRI [5-2-1113-126-2019]
OM KC [5-2-1113-23-2019]
UDAY SHRESTHA [5-2-1113-46-2019]

SAMRIDDHI COLLEGE
DEPARTMENT OF SCIENCE AND TECHNOLOGY

FEBRUARY, 2021

DECLARATION

I hereby declare that this project entitled **FLAPPY BIRD** is based on my original research work. Related works on this project by other researchers have been duly acknowledged. I owe all the liabilities relating to the accuracy and authenticity of the data and any other information included hereunder.

Ayush Bhattarai

Dewanand Giri

Om KC

Uday Shrestha

Date: 2021-04-25

RECOMMENDATION

This is to certify that this thesis entitled **FLAPPY BIRD** prepared and submitted by **Ayush Bhattarai, Dewanand Giri, Om KC, Uday Shrestha** for second semester OOP Project of Bachelor of Computer Science and Information Technology awarded by Tribhuvan University, has been completed under my supervision.

ER. MOHAN BHANDARI

Faculty, Samriddhi College

Date: February 2021

ACKNOWLEDGEMENT

First of all, the party that we would like to thank is Samriddhi College, for providing this excellent opportunity to participate in this project. Also, we would also like to thank the faculty for utilizing this minor project model to consolidate our knowledge on Object-Oriented Programming and many concepts that are associated with it. We would also like to extended our thanks to our faculty members of Object-Oriented Programming Mr. Mohan Bhandari and Mr. Jeeban Dhungel for sharing their invaluable knowledge and providing guidance on this particular topic. Furthermore, we would also like to thank our principal Mr. Sandeep Shrestha, along with Arpan Maskey sir and Lok Nath Regmi sir for providing us with the environment and facility so that we could complete the project on time without any time constraints and any pressures. Lastly, we would also like to thank our seniors and friends who have contributed their time to our project by helping us proof read, debug and guide us along the way. This endeavor wouldn’t be successful without the involvement of these parties and although not named we would also like to thank anyone remotely associated with this project and we would like them to know that their helps and advice have been invaluable in this endeavor.

ABSTRACT

Flappy bird is a simple game that has game that has the titular bird flaps its way through endless obstacles as it scores point for every obstacle avoided. The only way to end the game is to hit the obstacle and that makes the game finish. Thus, it is a very simple game that anybody with a PC can pick up and play

Our project, flappy bird, not only aims to consolidate our knowledge and showcase our skills in C++ programming language along with OpenGL graphics API, it also aims to provide some entertainment to the people that will try our game.

Flappy bird in its initial form was an android game developed by Dong Nguyen under his game development company dotGears. It was loved in its initial form thus we recreated this game in C++ as our semester project. This showcases the possibilities that C++ and Object-Oriented Programming Languages have in game development

This is, unappologetically, a simple game that pleases players and proves the metal of its developers without much grandeur to it.

Keyword: *Flappy bird, Bird, Obstacle, Endless Obstacle and Bird game*

TABLE OF CONTENT

LIST OF ABBREVIATIONS.....vi

1. INTRODUCTION.....1

    1.1 INTRODUCTION.....1

    1.2 PROBLEM STATEMENT.....1

    1.3 OBJECTVES.....2

    1.4 SCOPE.....2

    1.5 OVERVIEW OF REPORT.....2

2. BACKGROUND STUDY AND LITERATURE REVIEW.....3

    2.1 BACKGROUND STUDY.....3

    2.2 LITERATURE REVIEW.....4

3. SYSTEM ANALYSIS AND DESIGN.....6

    3.1 SYSTEM ANALYSIS.....6

        3.1.1 REQUIREMENT ANALYSIS.....6

        3.1.2 FEASIBILITY ANALYSIS.....8

    3.2 SYSTEM DESIGN.....8

        3.2.1 ALGORITHM.....8

        3.2.2 FLOWCHART.....9

4. RESULT AND DISCUSSION.....10

    4.1 RESULT.....10

    4.2 TEST CASES.....12

5. LIMITATION.....16

6. FUTURE WORK.....17

7. RECOMMENDATION AND CONCLUSION.....18

    7.1 RECOMMENDATION.....18

    7.2 CONCLUSION.....18

REFERENCES.....19

LIST OF FIGURES

FIGURE 3.1.1.1: USE CASE DIAGRAM FOR THE PROJECT.................................6
FIGURE 3.2.2.1: FLOWCHART OF THE PROJECT...9
FIGURE 4.1.1: SCREENSHOT OF THE MENU SCREEN.......................................10
FIGURE 4.1.2: SCREENSHOT OF THE GAME SCREEN.......................................11

LIST OF TABLES

TABLE 4.2.1: TEST CASE 1 12
TABLE 4.2.2: TEST CASE 2 12
TABLE 4.2.3: TEST CASE 3 13
TABLE 4.2.4: TEST CASE 4 13
TABLE 4.2.5: TEST CASE 5 14
TABLE 4.2.6: TEST CASE 6 14
TABLE 4.2.7: TEST CASE 7 15
TABLE 4.2.8: TEST CASE 8 15

LIST OF ABBREVIATIONS

2D	: Two Dimensional
3D	: Three Dimensional
API	: Application Programming Interface
DT	: Digital Trends
EXEC	: Executable
FPS	: Frames Per Second
GL	: Graphics Library
GUI	: Graphical User Interface
OOPL	: Object Oriented Programming Language

1. INTRODUCTION

1.1 INTRODUCTION

Flappy bird has been enjoyed all around the world. It is a fun game that use concept of endless obstacles to make people try to beat their own high scores this makes the game fun even when it is repetitive.

This game has very clear and simple objective to avoid obstacle, thus, is very simple to take up and start playing without hassle. This game releases hormones like adrenaline and dopamine because of the constant frustration of failing at the game and euphoria of going just a few obstacles farther. It provides the same kind of physical and emotional reactions that an artist gets when they are creating art.[1] This game is very great break from monotony of work and also in those times when you just don't feel like doing anything and just want to sleep, playing this game for some time will release dopamine and adrenaline which will wake you up pretty fast.[1]

This version of the game is ported using the C++ language thus it can be played on many devices due to the flexibility of the C++ language.

1.2 PROBLEM STATEMENT

The current reality of the world is that everyone is doing their very best to make life better for themselves and their loved ones. This means running in the rat races of the world, constantly trying to do better, to be better, to earn more and such. But in this hectic lifestyle and among the rat races, people are forgetting to take regular breaks of those that do take breaks are not able to think anything other than all the things they have yet to accomplish. This is giving people mild depressions and pessimism towards life and society and also impacting harshly on their mental health.

Thus, we are trying to make an easily accessible game that can take people's mind off their problems and give themselves a break, a true break with excitement that this game provides The release of the high excitement hormone adrenaline and satisfaction hormone dopamine is an added bonus that will make them feel good for a while.

1.3 OBJECTIVES

The objectives of our project are the following:

i. To make flappy bird more easily accessible using a highly versatile language.

ii. To provide people with a source of good clean fun that will alleviate their stress.

iii. To utilize our knowledge in Object-Oriented Programming to make something useful.

1.4 SCOPE

Our project is applicable in any place that has access to a compute regardless of its specifications and its hardware capabilities. This is made possible by the versatility of C++ language and OpenGL framework.

1.5 OVERVIEW OF REPORT

This report altogether contains four chapters. The first chapter “Introduction” contains overall information of the game “Flappy Bird”. The chapter is further sub-headed into ten topics. Contents of this chapter are background, problem statements, objectives, scopes and overview of our project. Similarly, chapter two contains literature review of the report. In addition, chapter three is all about the methods by which the project is being completed. Under the third chapter “Methodology”, systemflow diagram of flappy bird along with its working mechanism is given.

2. BACKGROUND STUDY AND LITERATURE REVIEW

2.1 BACKGROUND STUDY

Our team decided to port flappy bird in C++ because first off, it is a well liked game and thus our program is familiar to the users as there is a high chance that user has atleast seen the game. Having familiarity with the game is important because it will be easier to convince people to give a chance to our program instead of trying to make them give a chance to something that they have no idea what it is and no idea whether they will like it.

Secondly, we are also starting out with very beginner level knowledge of OOPL. Thus, we wanted our project to be moderately challenging without hopeless levels of difficulty that would put us off OOPL.

Thirdly, we wanted our project to incorporate technology that was new to us but learning it could prove to be very useful to us in the future. And we came upon OpenGL, it is a GL that is widely used in many places and in many avenues. Learning a GL like OpenGL is an in-demand skill as graphics are required in most places that utilize software in any capacity. Plus, the graphics.h graphics library that is inbuilt with the C++ compilers is quiet old and almost deprecated and instead of that using glfw.h and windows.h libraries give much modern looking GUI that makes the project look professional.[2]

Thus, because of these reasons, we decided that our project would be porting the popular mobile game flappy bird to PC using C++ and OpenGL.

2.2 LITERATURE REVIEW

For any game to prove itself irresistible to play, it should have a perfect amalgam of hassle and reward. Psychologist Mihaly Csikszentmihalyi terms this feeling “flow” and swaying below this feeling leads to sense of ennui and above it leads to anxiety. Our game should provide exactly that, the sense of joy on crossing the high score after battling yourself to get the bird across the pipes.[1]

Codes that have been written by someone else is often difficult to understand and horrible to maintain and extend. Countless program units, like classes, or functions, are very large, some of them with thousands of lines of code. Too many dependencies between these units lead to unwanted side effects if something changed. While our team might not be a team of strong software architects we’ll try to implement what they do so that our program would not look like a “big ball of mud”. Understanding basics of software entropies is a very good start and yes, we’ve already implemented that in our project.[4]

Mastering how to work with vectors in game development is one of the core building blocks to becoming a professional games programmer and gives you the ability to implement most of the mechanics found in modern games. Positioning of objects simply requires vectors and it will be used quite commonly in out project.[5]

The Broken Window Theory was developed in U.S.-American crime research. The theory states that a single destroyed window at an abandoned building can be the trigger for the dilapidation of an entire neighborhood. The broken window sends a fatal signal to the environment: “Look, nobody cares about this building!” This attracts further decay, vandalism, and other antisocial behavior.[4]

The Broken Window Theory has been used as the foundation for several reforms in criminal policy, especially for the development of Zero-Tolerance strategies. In game development, this theory can be taken up and applied to the quality of code Hacks and bad implementations, which are not compliant with the software design, are called “broken windows”. If these bad implementations are not repaired, more hacks to deal with them may appear in their neighborhood. And thus, the dilapidation of code is set into motion. And to prevent that, we can nullify any broken windows that might appear during the development of our project.[4]

One of the important aspect of good optimal gameplay is clear goal. And yes in our game, the rulw is very simple. Which means that the ones playing would not need to spend large chunks of their precious time to know what’s going around in the game. [6]

Aaron Isaksen and Andy Neelan with their simulations have showed us how changing attributes like bird flying speed, gravity, pipe width, and pipe spacing (among lots of others) could dramatically change the play of the game. Adjusting these attributes like gravity and jump velocity, and simulating repeatedly, the researchers plotted out a graph of game duration outcomes that ranged from impossible to easy. In the middle, a sweet spot of a very playable game. Although we might not get the same level of sweet spot, we can come close to it.

3. SYSTEM ANALYSIS AND DESIGN

3.1 SYSTEM ANALYSIS

3.1.1 REQUIREMENT ANALYSIS

i. FUNCTIONAL REQUIREMENT

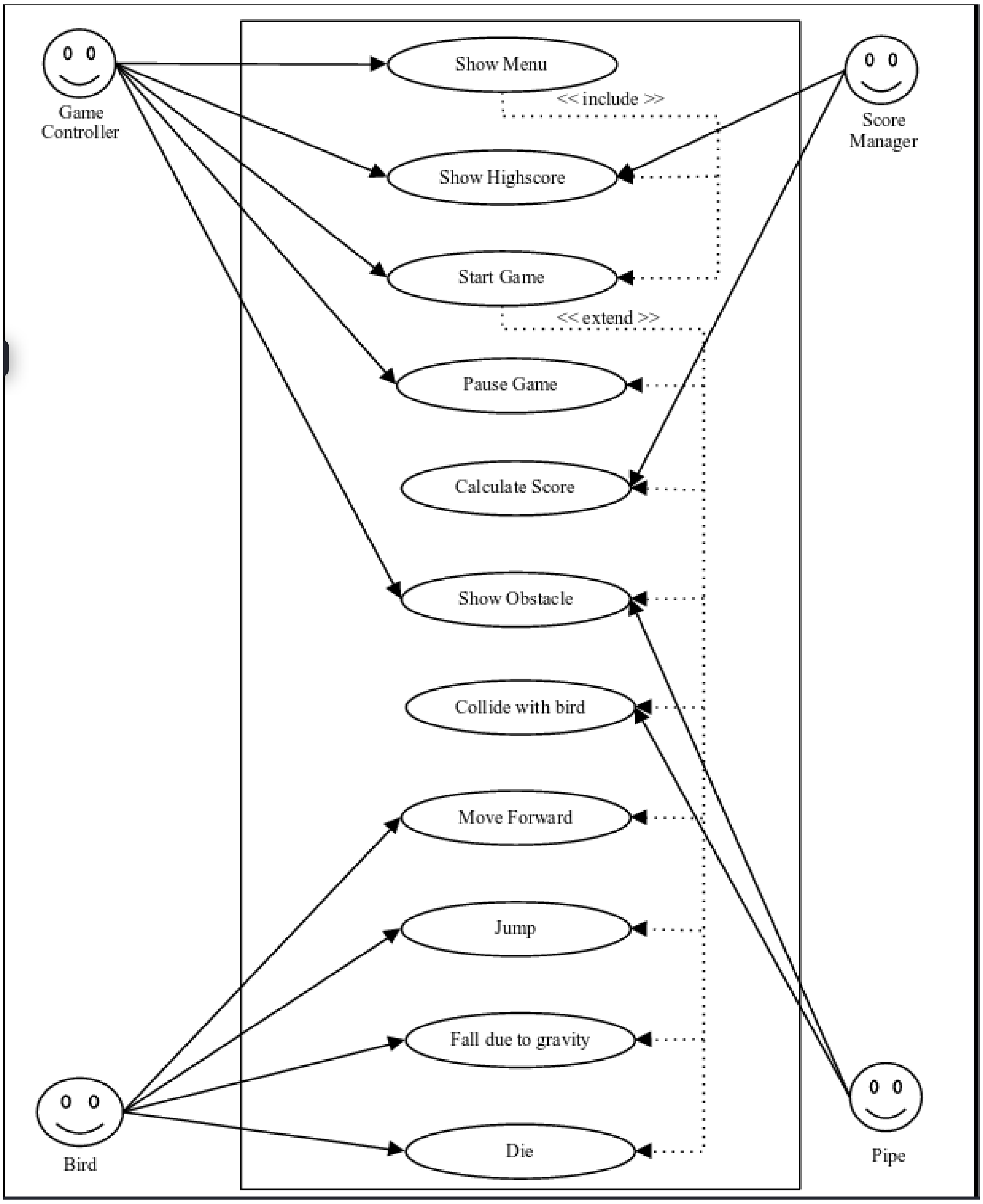

FIGURE 3.1.1.1: USE CASE DIAGRAM FOR THE PROJECT

There are four actors in our project; the bird, the pipe, the score manager and the game controller. Main objective of the bird is to get rid of all the obstacles by jumping, moving forward, falling due to gravity and die on contact with the obstacle pipe. Pipe is another entity of our game whose sole purpose is to collide with the bird while score manager keeps track of score when bird moves past a pipe.

Talking about our vital actor game controller, it does most of the heavy lifting like showing menu, highest score, pause game, and most importantly starting the game. All these tangle up to make a fine recipe for an exciting game.

Some major entities of our program with their functions are:

a) Show Menu: It shows the menu with highscore and option to start the game.

b) Start Game: It initializes the game and the various entities that extend from it.

c) Show Obstacle: It shows the obstacles that player has to dodge to score points.

d) Collide with Bird: It detects the collision between the bird and obstacle.

e) Die: It ends the game if the collision is registered.

f) Calculate Score: It calculates the score of the user in that session.

ii. NON-FUNCTIONAL REQUIREMENT

a) ACCESSIBILITY

One can easily gain access to our game on downloading it via internet. Users won't have a problem downloading it from devices such as a pendrive. It is free of cost and provides a great entertainment to lure away user's boredom.

b) PERFORMANCE

The game that we've built is restricted to 30 FPS because we believed that physics at higher or lower frame rates would feel slightly off. Other than that, it should run fine in your computer screen without lags or glitches.

c) APPEARANCE

Flappy bird might give you vibes about Super Mario because of the pipes that look similar. Users who have grown playing Nintendo games can get the stimulus and environment of it and maybe even get back their happy good old moments

3.1.2 FEASIBILITY ANALYSIS

i. TECHNICAL FEASIBILITY

Our project is technically feasible because the code is written in C++ so it doesn't require much investment except to learn OpenGL which will not take much time.

ii. OPERATIONAL FEASIBILITY

Our project is operationally feasible because it doesn't require any investment to run normally it only requires that you have the file for the game which is easily available.

iii. ECONOMIC FEASIBILITY

Our game doesn't cost anything to either play or run so it is economically feasible to anyone who is interested in our game.

iv. SCHEDULE FEASIBILITY

Our game only required a week to program so it doesn't require much time investment hence it is feasible in any schedule.

3.2 SYSTEM DESIGN

3.2.1 ALGORITHM

```
STEP 1.   Start
STEP 2.   Show the welcome screen.
STEP 3.   Ask for the key press.
STEP 4.   Show obstacle.
STEP 5.   If bird collides with pipe go to STEP 7.
          If bird does not collide with pipe go to STEP 6.
STEP 6.   Increase score counter by 1 & go to STEP 4.
STEP 7.   Show welcome screen with the highest score recorded up to that point.
STEP 8.   End
```

3.2.2 FLOWCHART

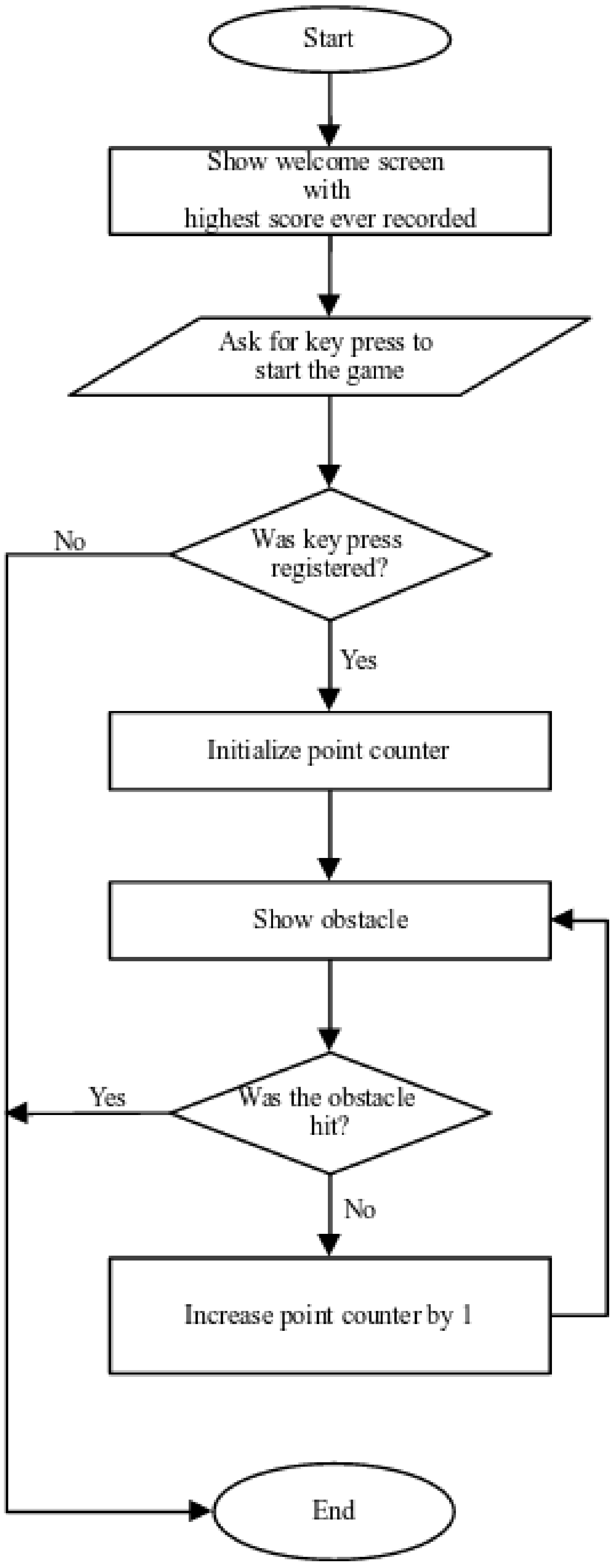
Start
Show welcome screen with highest score ever recorded
Ask for key press to start the game
Was key press registered?
No
Yes
Initialize point counter
Show obstacle
Was the obstacle hit?
Yes
No
Increase point counter by 1
End

FIGURE 3.2.2.1: FLOWCHART OF THE PROJECT

4. RESULT AND DISCUSSION

4.1 RESULT

The final output of our program has following outputs:

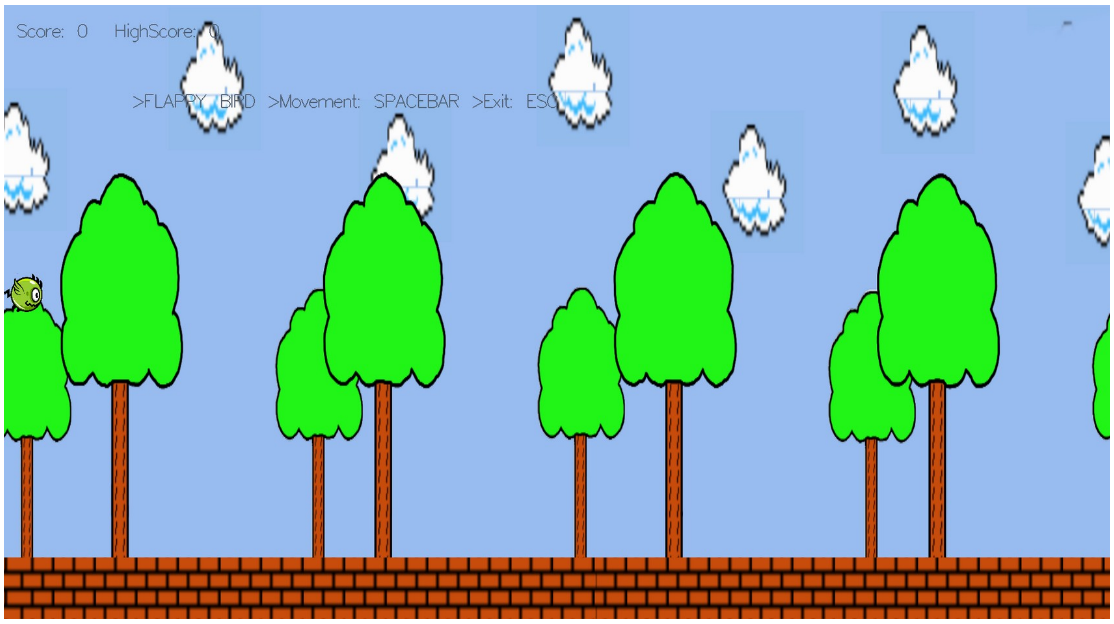

FIGURE 4.1.1: SCREENSHOT OF THE MENU SCREEN

The code we wrote was successful enough to produce an average menu screen which at first glance looks minimalistic and ostentatious. Although the bragging rights go fully to the libsoil library which had all the tools to load the menu screen. Since we ported the game to a pc version, we wanted the game to cover the entire screen and stretching the original design of the flappy bird game to cover the entire screen or infact multiple sizes of screens has its downsides most notable of which would have been the fact that it would look very blocky, specially the main character fappy.

The new design that we implemented, although decidedly not as good looking as the original menu screen it is, however, minimalistic and functional so it did the job for us. The texture might be reminisceint to Mario because the textures utilized were from the Mario palette that Nintendo used in their game.

And we went way past our expected outcome on this one as the game screen looks way better than this:

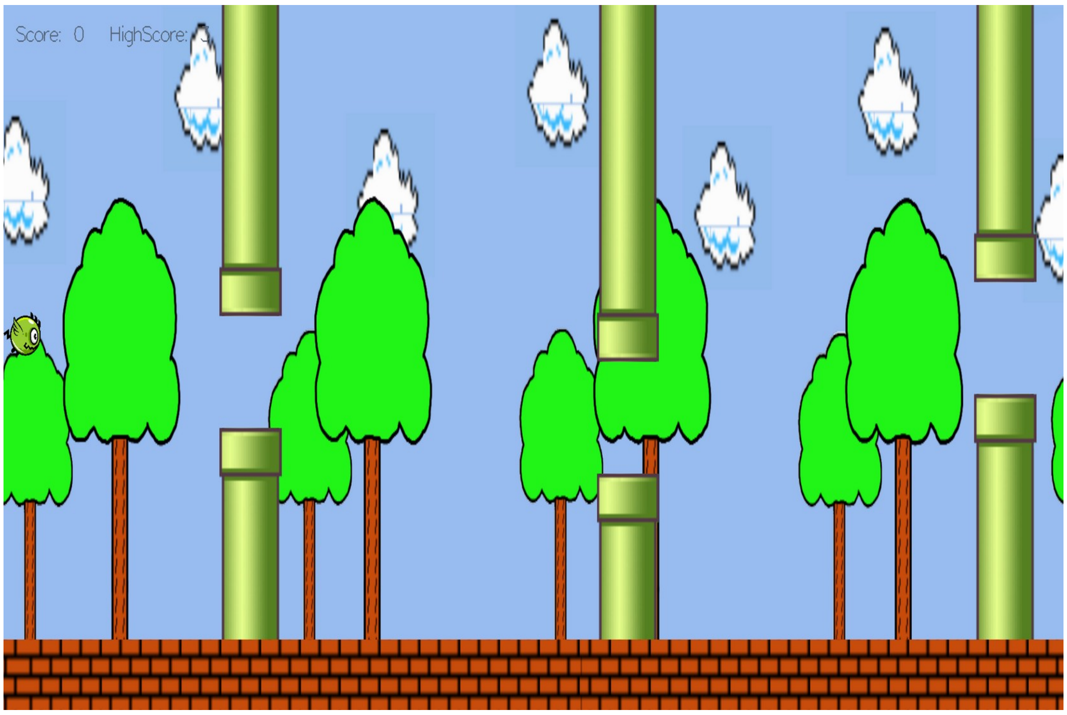

FIGURE 4.1.2: SCREENSHOT OF THE GAME SCREEN

Game screen looks similar to menu screen but one can sense the parallax effect on the screen which would make it more fun for the users to play. The game screen is however not to the point as we failed to maintain the flow of the game which Dong Nguyen perfected. The most drastic change that can be felt in our game as opposed to the original version is that our flying effects and bird sprite control is not upto par and somewhat difficult to control and the upward motion of the sprite looks somewhat laggy and slow.

But all things considered, our result is very competent given the skillset and experience that we had going into the project.

4.2 TEST CASES

Test Case ID	LOAD_001		
Test Case Description	Showing of main screen before starting the game		
Pre Requisites	a. Bash Terminal b. G++ Compiler		
Test Scenario	On Opening exec file		
Test Data	Desired Screen Size (1920 x 1080)		
Steps	Expected Result	Actual Result	Pass/Fail
a. Open exec file b. Provide desired screen size	Main Screen should be shown	Main Screen was shown	Pass

TABLE 4.2.1: TEST CASE 1

Test Case ID	LOAD_002		
Test Case Description	Showing of the main screen after starting the game		
Pre Requisites	a. Runing instance of the game		
Test Scenario	On Ending the game		
Test Data			
Steps	Expected Result	Actual Result	Pass/Fail
a. Move the bird b. Collide with object	Main Screen should be shown	Main Screen was shown	Pass

TABLE 4.2.2: TEST CASE 2

Test Case ID	MOVE_001		
Test Case Description	Movement on keypress of space key only		
Pre Requisites	a. Soil Image Loader Library b. FreeGLUT Library c. G++ Compiler		
Test Scenario	On key press of space key		
Test Data	< Space >		
Steps	Expected Result	Actual Result	Pass/Fail
a. Run the game b. Start the game c. Press < Space >	Fappy should move	Fappy moved	Pass

TABLE 4.2.3: TEST CASE 3

Test Case ID	MOVE_002		
Test Case Description	Movement on keypress other than space key		
Pre Requisites	a. Soil Image Loader Library b. FreeGLUT Library c. G++ Compiler		
Test Scenario	On keypress other than space key		
Test Data	< Up Cursor >		
Steps	Expected Result	Actual Result	Pass/Fail
a. Run the game b. Start the game c. Press key other than < Space>	Fappy shouldn’t move	Fappy didnt move	Pass

TABLE 4.2.4: TEST CASE 4

Test Case ID	SCORE_001		
Test Case Description	Increasing score on dodging obstacle		
Pre Requisites	a. FreeGLUT Library b. G++ Compiler		
Test Scenario	On dodging obstacle		
Test Data	< Space >		
Steps	Expected Result	Actual Result	Pass/Fail
a. Run the game b. Start the game c. Dodge the obstacle	The score counter should increase	The score counter increased	Pass

TABLE 4.2.5: TEST CASE 5

Test Case ID	SCORE_002		
Test Case Description	High score check after ending the game		
Pre Requisites	a. FreeGLUT Library b. G++ Compiler		
Test Scenario	On ending the game after certain score		
Test Data			
Steps	Expected Result	Actual Result	Pass/Fail
a. Run the game c. Dodge the obstacles d. End the game with certain score	The score should be recorded in highscore.	The score was recorded in highscore.	Pass

TABLE 4.2.6: TEST CASE 6

Test Case ID	ANIMATION_001		
Test Case Description	Tumbling animation on death		
Pre Requisites	a. Soil Image Loader Library b. FreeGLUT Library c. G++ Compiler		
Test Scenario	On collision with obstacle or on death		
Test Data			
Steps	Expected Result	Actual Result	Pass/Fail
a. Run the game b. Start the game c. Collide with obstacle	Fappy should show tumble animation	Fappy showed tumble animation	Pass

TABLE 4.2.7: TEST CASE 7

Test Case ID	ANIMATION_002		
Test Case Description	Animation trees with parallax effect		
Pre Requisites	a. Soil Image Loader Library b.FreeGLUT c. G++ Compiler		
Test Scenario	On running the game		
Test Data			
Steps	Expected Result	Actual Result	Pass/Fail
a. Open the exec file b. Run the game	Trees should move with repsect to your viewpoint.	Trees moved with respect to your viewpoint	Pass

TABLE 4.2.8: TEST CASE 8

5. LIMITATION

“Imperfection is sometimes way too perfect.”, The saying had to be brought up while casting some light on our not so flawless game, “Flappy Bird”. Although the game was built to be better than the original, “Flappy Bird” with the newly added parallax effect, improved controls to make users more addictive by maintaining science of flow, animation on death and so on, it still lacks few things.

1.Vsync not enabled: The game mode used to implement VSync could not find required extension in most of our test devices which had intel HD driver. With this, we could have watched the monitor’s max refresh rate. If the FPS of our game was equal to or higher than the refresh rate, VSync would have been enabled.

2. Scre tearing on Nvdia GPU: This project also failed to prevent screen tearing on Nvidia GPU, and that explains why it didn’t run on Macbook Air 2010 (one of our test device) which had Nvidia GPU. Not only that, also due to unavailability of free Glut, the game failed to run in the terminal.

3. Highscore initialized to 0 after every session: Achievements are something to brag about if you’re a human and if a human ever plays Flappy Bird, they have to show others if they get past double digits or not. Unfortunately, after a session ends on the terminal, our game is initialize the high score to 0 littering all your progress in thin air. Had there been works done on file handling, this could have been prevented.

4. Absence of Wakelocks: The game fails to prevent the processor from sleeping or dimming because there is no addition of Wakelocks.

Other than these flaws we don’t know of any other limitations of our game Flappy Bird.

6. FUTURE WORK

After working with flappy bird, we've come a long way to get some understanding about OpenGL and game development. We can now go ahead to work with some game engines to add more animations, and produce a 3D version of the game.

1.3D version of the game: Previously, players of this game played the game on third person view and had a very little feeling of what it means to be the bird Fapy. The game in the future can promise you with the perspective of the bird which would further more enhance your experience with the game.

2. Networking involved: Furthermore, we can create a community of the game lovers after we work on the networking so that you could know about the global leaders of the game.

3. Addition of Wakelocks: We can also work with PowerManager Wakelocks to prevent your device's processor from sleeping or dimming. The game runs on terminal and this might prevent the processor from dimming for now but we can't be satisfied here.

4.Flapping of wings: Animations is another aspect of our game that we failed at completely. While there was a death animation, we can show that the bird Fapy is flapping its wings doing all sorts of crazy stuffs while getting past the obstacles. This is achievable in the future and we are looking to bring changes in this aspect of our game.

5. Vsync enabled: Implementation of VSync would let us to know if FPS of our game was higher than monitor's max refresh rate or not. We are therefore also thinking about implementation of VSync.

6. High score initialization with file handling: Highscores in this project have been a headache for us as it initia zero on each session. That should'nt be the case after file handling is implemented here. Or there are some other aspects we've not looked into that keeps changing the highscore after every session.

7. RECOMMENDATION AND CONCLUSION

7.1 RECOMMENDATION

To those people who might take our report as reference we have some suggestions and recommendations:

First Recommendation: Dont use OpenGL. It might seem counter intuitive to say that considering what this report is on, but trust us, instead of using OpenGL that has many precompiled libraries and functions to make a game, try to make your own game engine. This will not only help to make you more familiar with the programming language of your choice, it also helps to learn the many aspects of game development and programming that might be lost to you if you use OpenGL.

Second Recommendation: Be sure of your expectations vs what you can do. When we started this project, we had many expectations from ourselves and from the game as a whole. Many things like rendering custom font for Score and Highscore display was not possible because the strike font library of OpenGL only has Roman font and using any other fonts like bitmap instead of strike font would ruin the game animation and game texture. So, before you have expectations be sure that your expectations are ahievable in the time that you have.

7.2 CONCLUSION

Those who are willing to start game development with C++ have a lots to look into from game engines to APIs, normal physics to vast camera movements and so much more. Just getting a grasp of basics of simple 2D games is a good way to go for it. We would advice you to get a firm knowledge on the concept of class and objects before starting a project like this. This is a long journey and in it comes a lesson on each turns.

In a nutshell, it was an exceptional project for us as we were working with graphics in C++ for the very first time. There were a lot of frustrations as we hit the dead end a lot of times on various aspects of the development. However, all those grinding gave us some knowledge about API and object oriented programming. This was indeed a great experience for our team.

REFERENCES

[1] G. Stevens, “Why you can’t stop playing Flappy Bird | The Daily Dot”, 2 March 2020. [Online]. Available: https://www.dailydot.com/parsec/psychology-flappy-bird-addiction/. [Accessed 27 March 2021].

[2] “LearnOpenGL”, LearnOpenGL, 12 May 2017. [Online]. Available: https://learnopengl.com/Getting-started/Review. [Accessed 27 March 2021]

[3] M. Peckman, “Time”, Times, 03 February 2014. [Online]. Available: https://time.com/4034/no-you-dont-have-to-play-flappy-bird/. [Accessed 03 April 2021]

[4] S. Roth, Clean C++: Sustainable Software Development Patterns and Best Practices. Bad Schwartau, Schelswig-Holstein: Apress 2017

[5] “Vector Maths for Game Dev Beginners”,GameDev.net, 16 May 2020. [Online]. Available: https://gamedev.net/tutorials/programming/math-and-physics/vector-maths-for-game-dev-beginners-r5442/. [Accessed: 06 April 2021]

[6] “’Flappy Bird’ holds the key for figuring out the perfect difficulty in video games”, *Technical.ly Brooklyn,* 31 August 2016. [Online]. Available: https://technical.ly/brooklyn/2016/08/30/nyu-game-center-flappy-bird-case-study/. [Accessed: 06 April 2021]