VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", MACHHE, BELAGAVI-590018

Mini Project Report

on

Flappy Bird Game

Submitted in partial fulfillment of the requirements for the VI semester

Bachelor of Engineering

in

Computer Science and Engineering

of

Visvesvaraya Technological University, Belagavi.

by

Mr. Dharshan H D (1CD21CS045)

Under the Guidance of

Dr. Josephine Prem Kumar

Professor

Dept. of CSE

Under the Guidance of

Prof. Varalakshmi K V

Assistant Professor

Dept. of CSE

Under the Guidance of

Dr. Manjunath S

Associate Professor

Dept. of CSE

Department of Computer Science and Engineering

CAMBRIDGE INSTITUTE OF TECHNOLOGY, BANGALORE - 560 036

2023-2024

CAMBRIDGE INSTITUTE OF TECHNOLOGY

K.R. Puram, Bangalore-560 036

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that **Mr. Dharshan H D** bearing **USN: 1CD21CS045** respectively, bonafide student of **Cambridge Institute of Technology,** have successfully completed the mini project entitled "**Flappy Bird Game**" in partial fulfillment of the requirements for VI semester **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year 2023-2024. It is certified that all Corrections/Suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said semester.

---	---
Internal Guide:	**Head of the Dept.**
Dr. Josephine Prem Kumar	**Dr. Shreekanth Prabhu**
Dept. of CSE., CiTech.	**Dept. of CSE., CiTech.**

External Viva

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

DECLARATION

I, **Mr. Dharshan H D bearing USN: 1CD21CS045** respectively, the student of VI semester, Computer Science and Engineering, Cambridge Institute of Technology, hereby declarethat the mini project entitled **"Flappy Bird Game"** has been carried out by me and submitted in partial fulfillment of the course requirements of VI semester **Bachelor of Engineering** in **Computer Science and Engineering** as prescribed by **Visvesvaraya Technological University, Belagavi**, during the academic year 2023-2024.

I also declare that, to the best of our knowledge and belief, the work reported here does not form part of any other report on the basis of which a degree or award was conferred on an earlier occasion on this by any other student.

Date:	**Name**	**USN**	**Signature**
Place: Bangalore	Dharshan H D	1CD21CS045	

ACKNOWLEDGEMENT

I am extremely thankful to **Dr. Indumathi G,** Principal, CITech., Bengaluru, for providing me the academic ambience and everlasting motivation to carry out this work and shaping our careers.

I express my sincere gratitude to **Dr. Shreekanth Prabhu,** HOD, Dept. of Computer Science and Engineering, CITech., Bengaluru, for his stimulating guidance, continuous encouragement and motivation throughout the course of present work.

I also wish to extend our thanks to **Dr. Josephine Prem Kumar** Professor, **Prof. Varalakshmi KV** Assistant Professor and **Dr. Manjunath S** Associate Professor Dept. of Computer Science and Engineering, CITech., Bengaluru, for their expert guidance and constructive suggestions to improve the quality of this work.

I would also like to thank all other teaching and technical staff of Department of Computer Scienceand Engineering, who have directly or indirectly helped me in the completion of this Project Work.

And lastly I would hereby acknowledge and thank our parents who have been a source of inspiration and also instrumental in the successful completion of this mini project.

Dharshan H D (1CD21CS045)

ABSTRACT

This mini-project presents the development of a Flappy Bird game using the C programming language and OpenGL functions. The primary objective is to create a visually appealing and engaging 2D game that demonstrates fundamental principles of computer graphics and game development. The project involves the implementation of core game mechanics, including the bird's flight dynamics, obstacle generation, and collision detection, all rendered through OpenGL's graphical capabilities.

The game features a bird that navigates through an endless series of pipes by responding to user input, simulating the flapping motion and gravity effect. The graphical elements are rendered using OpenGL functions, ensuring smooth transitions and visual clarity. The user interface includes score tracking and game state indicators, enhancing the player experience without relying on external libraries beyond OpenGL and standard C libraries.

This project serves as an educational tool, illustrating the application of computer graphics techniques in game development. It highlights the process of integrating graphics rendering with game logic, managing real-time user input, and optimizing performance in a C-based environment. The outcome is a fully functional Flappy Bird game that runs efficiently on various platforms, demonstrating the potential of OpenGL for 2D game development in C.

CONTENTS

List of Figures

CHAPTER 1

INTRODUCTION

1.1 About Flappy Bird:

Flappy Bird is a simple yet highly addictive mobile game developed by Dong Nguyen and released in 2013. The game quickly gained worldwide popularity due to its straightforward mechanics and challenging gameplay. The player controls a bird attempting to fly through a series of green pipes without hitting them. Despite its simple concept, Flappy Bird's design and difficulty captivated millions of players globally, making it a phenomenon in mobile gaming. This project aims to replicate the core gameplay of Flappy Bird while exploring potential innovations in its mechanics and user experience.

The game's popularity was propelled by its minimalist design, challenging gameplay, and the competitive nature of achieving high scores. However, the phenomenon was short-lived, as Nguyen decided to remove the game from app stores in 2014, citing concerns over its addictive nature and the immense pressure of its overnight success.

Relevance and Impact: Flappy Bird's impact on the mobile gaming industry is significant for several reasons:

- **Minimalist Design**: The game's simplistic approach highlighted that success in mobile gaming doesn't always require high-end graphics or complex mechanics. This sparked a trend towards minimalistic and casual games that are accessible to a broader audience.

- **Viral Marketing**: The game's rapid rise to fame was largely fueled by word-of-mouth and social media, demonstrating the power of organic growth and viral marketing in the digital age.

- **Addictive Gameplay**: Flappy Bird exemplified how finely-tuned difficulty and the "just one more try" factor can drive player engagement and retention, providing valuable lessons for game designers on balancing challenge and playability.

1.2 Project description:

This project aims to demonstrate the application of basic computer graphics techniques in game development while providing an enjoyable gaming experience.

The Flappy Bird Computer Graphics Project aims to recreate the popular Flappy Bird game using only shades of gray, textures, and patterns, with no reliance on visual colors or sound effects. This project will focus on designing a visually engaging and challenging game experience through contrast, design, and smooth animations.

This project aims to demonstrate the application of basic computer graphics techniques in game development while providing an enjoyable gaming experience. The Flappy Bird Computer Graphics Project seeks to recreate the popular Flappy Bird game using only shades of gray, textures, and patterns, with no reliance on visual colors or sound effects. By focusing on contrast, design, and smooth animations, the project intends to create a visually engaging and challenging game experience.

Developed using C++ and OpenGL, the project will leverage powerful graphics libraries to build a seamless and immersive environment. Key features will include dynamic backgrounds utilizing various shades of gray to create depth and movement, textured obstacles to enhance the challenge, and smooth animations for the bird character. Procedural generation will ensure unique playthroughs, and performance optimization will guarantee smooth operation on a range of hardware.

The game will feature an intuitive and minimalist user interface, a rewarding scoring system, responsive controls, and realistic game physics. Comprehensive debugging and testing will ensure a polished final product. Additionally, the project will include detailed documentation and tutorials to help other developers understand and replicate the techniques used.

Ultimately, this project aims to showcase that compelling and visually interesting games can be created through minimalist design, without relying on color or sound, thus highlighting the potential of creative computer graphics in game development.

1. Game Design and Development:

- **Conceptualization**:Understanding the fundamental gameplay mechanics of Flappy Bird, including user control, gravity, and collision detection.

- **Framework Selection**: Choosing an appropriate development framework or game engine, such as Unity or Unreal Engine, to build the game.

- **Asset Creation**: Designing or sourcing 2D graphics for the bird, pipes, background, and other visual elements.

- **Programming**: Implementing game logic, including user input handling, physics for the bird's movement, and collision detection algorithms.

2. Testing and Iteration:

- **Prototype Testing**: Creating a basic version of the game to test core functionalities and gameplay mechanics.

- **User Feedback**: Gathering feedback from players to identify areas of improvement in gameplay, controls, and overall user experience.

- **Refinement**: Iteratively refining the game based on user feedback and testing results to enhance playability and engagement.

3. Innovation and Enhancement:

- **Advanced Features**: Exploring the addition of new game modes, power-ups, or dynamic environments to innovate beyond the original Flappy Bird gameplay.

- **User Interface (UI) Improvements**: Designing a more intuitive and visually appealing user interface.

- **Multiplayer Mode**: Introducing a multiplayer option for competitive or cooperative play.

CHAPTER 2

LITERATURE SURVEY

The development of this project involves understanding the core principles of game design, graphics rendering, and user interaction. This literature survey reviews key studies, methodologies, and technologies relevant to creating a Flappy Bird game, focusing on graphics, animation, and user interface design.

1. History and Evolution of Flappy Bird: Flappy Bird, developed by Dong Nguyen and released in 2013, became an instant hit due to its simple yet addictive gameplay. The game's mechanics involve guiding a bird through a series of pipes without crashing. Studies on Flappy Bird's design emphasize its minimalistic approach, straightforward controls, and challenging gameplay, which contribute to its widespread popularity and replay value .

2. Game Development Frameworks: Several game development frameworks can be used to recreate Flappy Bird. Unity and Unreal Engine are popular for their robust features and support for 2D and 3D game development. Unity, in particular, is widely recommended for beginners due to its extensive documentation and community support. Pygame is another framework suitable for simpler 2D games like Flappy Bird, providing an easy-to-use interface for game development in Python .

3. Graphics Rendering Techniques: Rendering techniques play a crucial role in game development, ensuring smooth and visually appealing graphics. Ray tracing and rasterization are common methods for rendering 3D graphics, but for 2D games like Flappy Bird, sprite-based rendering is more applicable. Sprite sheets, which consolidate multiple images into a single file, optimize rendering performance by reducing the number of draw calls .

4. Animation and Collision Detection: Animation techniques in 2D games often involve keyframe animation, where specific frames are defined to create the illusion of movement. For collision detection, bounding box methods are commonly used. These involve defining rectangular boundaries around objects to detect intersections, which is crucial for determining when the bird collides with pipes or the ground .

5. User Interface Design: The user interface (UI) design in games like Flappy Bird should be simple and intuitive. Studies highlight the importance of minimalistic UI design to avoid distracting the player from the core gameplay. Elements such as score display, start and game over screens, and navigation menus should be clearly visible and easy to interact.

6. Sound and Feedback: While this project focuses on a version without sound effects, the literature indicates that sound design typically enhances user experience by providing auditory feedback for actions like flapping and collisions. In the absence of sound, visual feedback through animations and visual cues becomes even more critical to maintaining player engagement and providing clear feedback.

The literature on game development, particularly for games like Flappy Bird, provides valuable insights into effective design principles, rendering techniques, and user interaction strategies. By leveraging these insights, the Flappy Bird computer graphics project can achieve a high-quality, engaging game that remains faithful to the original's charm while incorporating advanced graphical techniques and user interface designs.

7. Game Mechanics and Difficulty Balancing: Understanding game mechanics and difficulty balancing is crucial for creating a challenging yet enjoyable game. The literature emphasizes the importance of designing mechanics that are easy to learn but hard to master. In Flappy Bird, the difficulty is inherently tied to the game's pace and the spacing of the pipes. Research on game difficulty suggests employing adaptive difficulty mechanisms that can adjust based on player performance to maintain engagement and prevent frustration.

8. Performance Optimization: Effective performance optimization is essential for ensuring a smooth gaming experience, especially in a game with continuous action like Flappy Bird. Techniques such as object pooling, efficient memory management, and optimizing rendering processes can significantly impact the game's performance. The literature suggests focusing on reducing resource-intensive operations and minimizing draw calls to enhance performance, especially on lower-end devices.

9. User Experience (UX) Research: UX research highlights the importance of creating an engaging and enjoyable player experience. Studies indicate that players value clear instructions, responsive controls, and immediate feedback. For Flappy Bird, ensuring that the game is intuitive and that players receive immediate feedback on their actions is crucial for maintaining interest and satisfaction.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 Hardware Requirements:

1. Development Machine:

- Processor: Intel Core i5 or AMD equivalent
- RAM: 8 GB (16 GB recommended for smoother multitasking)
- Storage: 500 GB HDD or SSD (SSD recommended for faster read/write speeds)
- Graphics Card: Integrated graphics (e.g., Intel HD Graphics) or dedicated graphics card (e.g., NVIDIA GeForce GTX 1050 or AMD equivalent)
- Display: 1080p monitor (1920x1080 resolution)

2. Testing Devices:

- PC: Same as development machine requirements
- Mobile Devices: Smartphones or tablets running iOS or Android for testing the mobile version of the game

3. Additional Peripherals:

- Keyboard and Mouse: Standard input devices
- Game Controller: Optional, for testing alternative input methods

3.2 Software Requirements

1. Operating System:

- **Windows:** Windows 10 or later
- **Mac:** macOS 10.14 Mojaveorlate
- **Linux:** Ubuntu 18.04 or later

2. Development Environment:

- **Integrated Development Environment (IDE):**
    - **Unity:** Unity Hub with the latest stable release of Unity Editor (recommended for ease of use and extensive documentation)
    - **Visual Studio Code or Visual Studio:** For script editing and debugging
    - **Pygame:** If using Python, install Pygame library with an IDE like PyCharm or Visual Studio Code
    - **Phaser:** If using JavaScript, an IDE like Visual Studio Code with Node.js installed

3. Additional Libraries and Frameworks:

- **Game Engine/Framework:**
    - **Unity:** For C# development
    - **Pygame:** For Python development
    - **Phaser:** For JavaScript development
- **Graphics API:**
    - **OpenGL or WebGL:** Depending on the chosen framework/engine
    - **DirectX:** If developing on Windows with C#

4. Testing Tools:

- **Unity Test Framework:** For unit testing in Unity
- **Selenium or Appium:** For automated testing on mobile devices
- **Emulators:** Android Studio Emulator or iOS Simulator for testing on virtual devices

CHAPTER 4

IMPLEMENTATION

4.1 Code

```
#include <GL/glut.h>
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
int birdY = 300; // Initial bird position
int birdVelocity = 0; // Vertical velocity of the bird
int pipeX = 200;
int pipeY = 0;
int score = 0;
int gameState = 0; // 0: Start, 1: Playing, 2: End
int gameSpeed = 30; // Initial game speed (lower value means faster updates)
int pipeGap = 150; // Height of the gap in the pipes
void init() // Initialize the game{
  glClearColor(0.53, 0.81, 0.92, 1); // Sky blue background
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
 gluOrtho2D(0,400,0,600);
}
void drawBird() // Draw the bird{
  glPushMatrix();  glTranslatef(50,
  birdY, 0);
  glColor3f(1.0, 1.0, 0.0); // Yellow color for the bird
  glBegin(GL_POLYGON);
  glVertex2f(-20, -20);
  glVertex2f(20, -20);
  glVertex2f(0, 20);
  glEnd();
  glPopMatrix();
```

```
}
void drawPipe()// Draw the pipes {
  glPushMatrix();
  glTranslatef(pipeX, 0, 0);
  glColor3f(0.0, 1.0, 0.0); // Green color for the pipe
  // Lower pipe
  glBegin(GL_POLYGON);
  glVertex2f(-20,0)
  glVertex2f(20, 0);
  glVertex2f(20, pipeY - pipeGap / 2);
  glVertex2f(-20, pipeY - pipeGap / 2);
  glEnd();
  // Upper pipe
  glBegin(GL_POLYGON);
  glVertex2f(-20, pipeY + pipeGap / 2);
  glVertex2f(20, pipeY + pipeGap / 2);
  glVertex2f(20, 600);
  glVertex2f(-20, 600);
  glEnd();
  glPopMatrix();
}
void drawText(const char* text, int x, int y) // Draw text on the screen {
  glRasterPos2i(x, y);
  glColor3f(1.0, 0.0, 0.0); // White color for the text
  for (const char* c = text; *c != '\0'; c++) {
    glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
}}
void drawStartScreen() // Draw the start screen
{ glColor3f(1.0, 0.0, 0.0);
  drawText("Press SPACE to Start", 120, 300);
}
void drawEndScreen() {
```

```
  char scoreText[20];
  sprintf(scoreText, "Final Score: %d", score);
  drawText(scoreText, 150, 320);
  drawText("Game Over! Press R to Restart", 100, 300);
}
void draw() // Draw the end screen{
  glClear(GL_COLOR_BUFFER_BIT);
  if (gameState == 0) { // Start screen
  drawStartScreen();
  } else if (gameState == 1) { // Playing
    drawBird();
    drawPipe();
       char scoreText[20];
    sprintf(scoreText, "Score: %d", score);
    drawText(scoreText, 10, 570);
  } else if (gameState == 2) { // End screen
    drawEndScreen();
  }
  glutSwapBuffers();
}
void update(int value) // Update game state
  { if (gameState == 1) {
    // Update bird position with gravity
    birdY += birdVelocity;
    birdVelocity -= 1; // Apply gravity (adjust as needed)
    // Ensure bird doesn't go below ground or above screen
    if (birdY < 20) {
      birdY = 20;
      birdVelocity = 0; // Reset velocity if bird hits ground (adjust as needed)
    } else if (birdY > 580) {
      birdY=580;
      birdVelocity = 0; // Reset velocity if bird hits top (adjust as needed)
    }
```

```
    // Update pipe position
    pipeX -= (5 + score / 5); // Increase pipe speed as score increases
    if (pipeX < -50) {
      pipeX = 400; // Reset pipe position
      pipeY = rand() % (600 - pipeGap) + pipeGap / 2; // Randomize pipe gap position
      score++;
      gameSpeed = 30 - score / 5; // Increase game speed as score increases
      if (gameSpeed < 10) gameSpeed = 10; // Cap the speed at a minimum value
    }
    // Check collision with pipe
    if ((birdY < pipeY - pipeGap / 2 || birdY > pipeY + pipeGap / 2) && pipeX < 70 && pipeX
> 30) {
      gameState = 2; // End game
    }
    glutPostRedisplay();
    glutTimerFunc(gameSpeed, update, 0); // Adjust update interval based on game speed
  }
}
void keyboard(unsigned char key, int x, int y) // Handle keyboard input{
  if (gameState == 0) {
    if (key == ' ') {
      gameState = 1;
      birdY = 300;
      birdVelocity = 0;
      pipeX = 400;
      pipeY = rand() % (600 - pipeGap) + pipeGap / 2;
      score = 0;
      gameSpeed = 30;
      glutTimerFunc(0, update, 0);
       }
     }
```

```
void keyboard(unsigned char key, int x, int y) {
  if (gameState == 0) {
    if (key == ' ') {
      gameState = 1;
      birdY = 300;
      birdVelocity = 0;
      pipeX = 400;
      pipeY = rand() % (600 - pipeGap) + pipeGap / 2;
      score = 0;
      gameSpeed = 30;
      glutTimerFunc(0, update, 0);
    }
  } else if (gameState == 1) {
    if (key == ' ') {
      birdVelocity = 10; // Flap the bird by setting a positive velocity (adjust as needed)
    }
  } else if (gameState == 2) {
    if(key=='r'||key=='R'){
    gameState=0;
      birdY = 300;
      birdVelocity = 0;
      pipeX = 400;
      pipeY = rand() % (600 - pipeGap) + pipeGap / 2;
      score = 0;
      gameSpeed = 30;
      glutPostRedisplay();
    }
  }
}
int main(int argc, char **argv) // Main function {
  srand(time(NULL)); glutInit(&argc, argv);
```

```
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize(400, 600);
  glutInitWindowPosition(100, 100);
  glutCreateWindow("Flappy Bird");
  init();
  glutDisplayFunc(draw);
  glutMainLoop();
  return 0;
}
```

4.2 Algorithm

Step 1: Initialize Game State

Set gameState to 0 (start screen)

Initialize birdY to 300 (initial bird position)

Initialize birdVelocity to 0 (initial bird velocity)

Initialize pipeX to 400 (initial pipe position)

Initialize pipeY to a random value between 0 and 600 - pipeGap (initial pipe gap position)

Initialize score to 0

Initialize gameSpeed to 30 (initial game speed)

Step 2: Handle User Input

If gameState is 0 (start screen), wait for user to press spacebar

If user presses space bar, set gameState to 1 (playing) and start the game loop

If gameState is 1 (playing), handle user input:

If user presses spacebar, set bird Velocity to 10 (flap the bird)

If gameState is 2 (end screen), wait for user to press 'R' or 'r' to restart the game

Step 3: Update Game State

If gameState is 1 (playing), update the game state:

Update birdY by adding birdVelocity to it (apply gravity)

Update birdVelocity by subtracting 1 from it (apply gravity)

Ensure birdY is within the screen boundaries (20 to 580)

Update pipeX by subtracting 5 + score / 5 from it (move the pipe to the left)

If pipeX is less than -50, reset pipeX to 400 and generate a new random pipeY

Increment score by 1

Update gameSpeed to 30 - score / 5 (increase game speed as score increases)

Check for collision with the pipe:

If birdY is less than pipeY - pipeGap / 2 or greater than pipeY + pipeGap / 2, and pipeX is between 30 and 70, set gameState to 2 (end screen)

Step 4: Draw Game Screen

If gameState is 0 (start screen), draw the start screen

If gameState is 1 (playing), draw the game screen:

Draw the bird at birdY

Draw the pipe at pipeX and pipeY

Draw the score at the top of the screen

If gameState is 2 (end screen), draw the end screen:

Draw the final score

Draw the restart prompt

Step 5: Repeat

Repeat steps 2-4 until the game is quit or restarted.

4.3 Flowchart

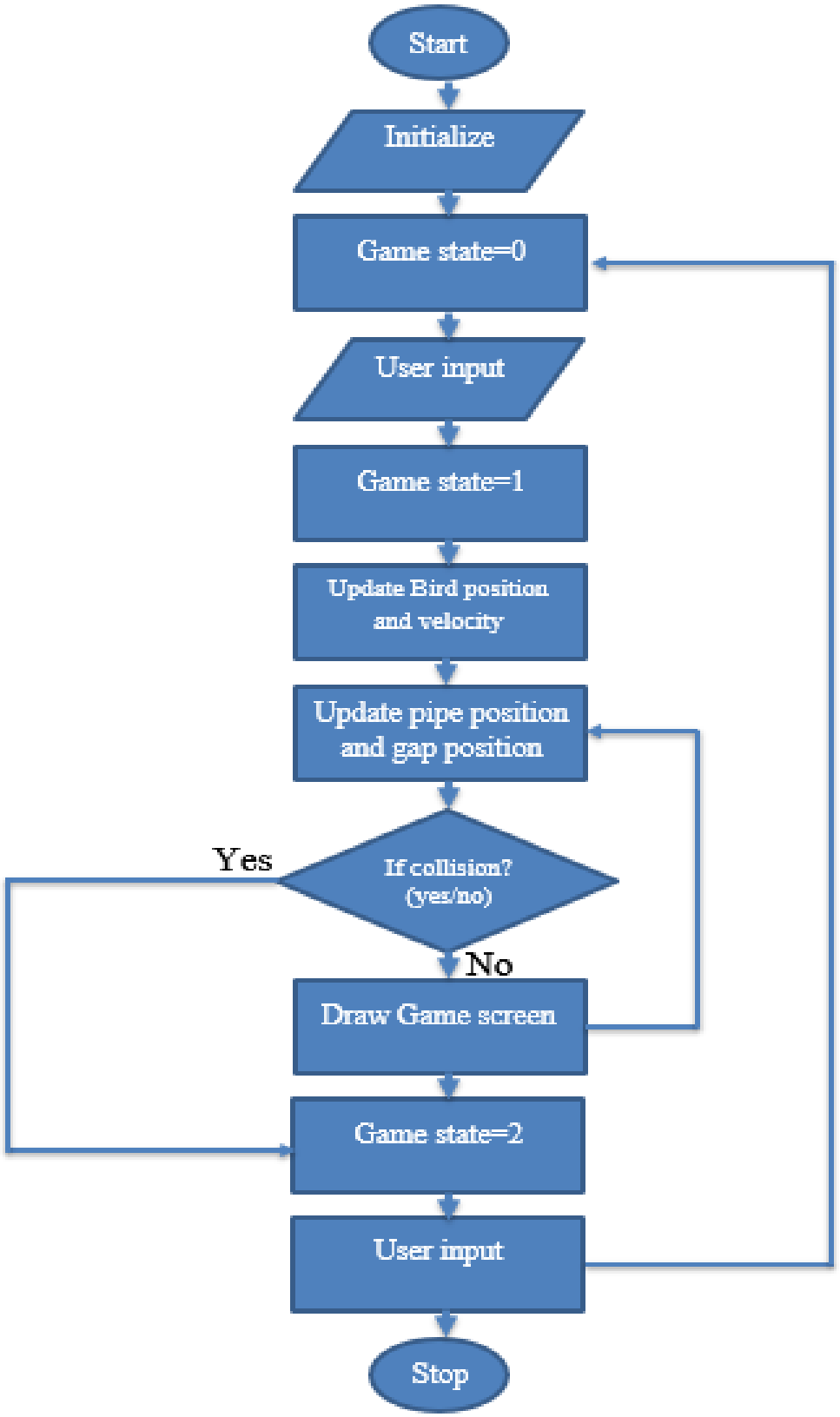

Fig 4.1 Flowchart

CHAPTER 5

SNAPSHOTS

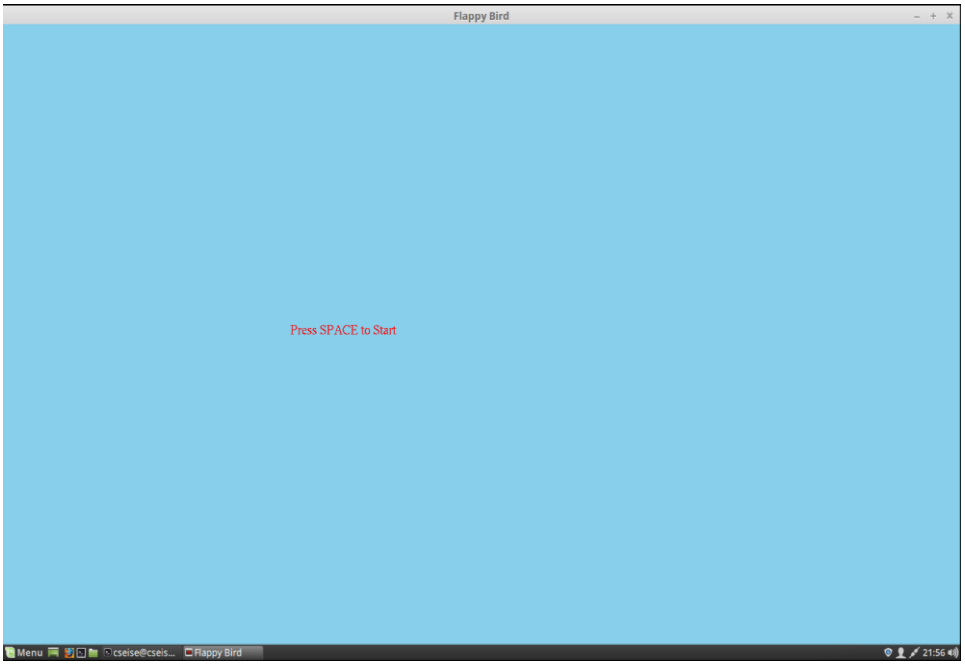

Fig.5.1 Before start

Fig.5.1 shows the starting interface of the program in which if we press spacekey the same starts.

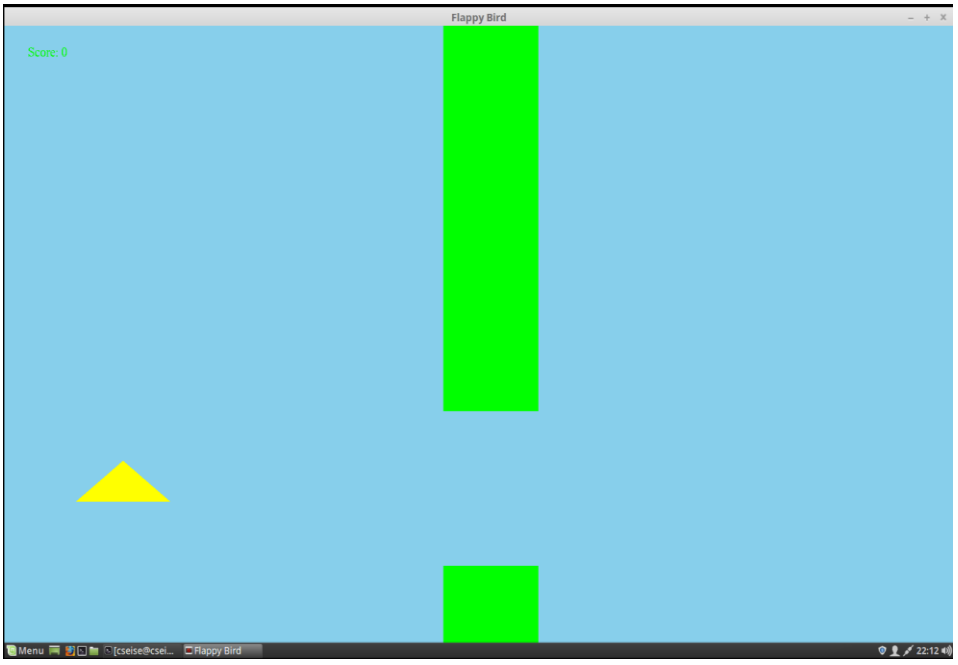

Fig.5.2 After start

Fig.5.2 shows the start of the game.

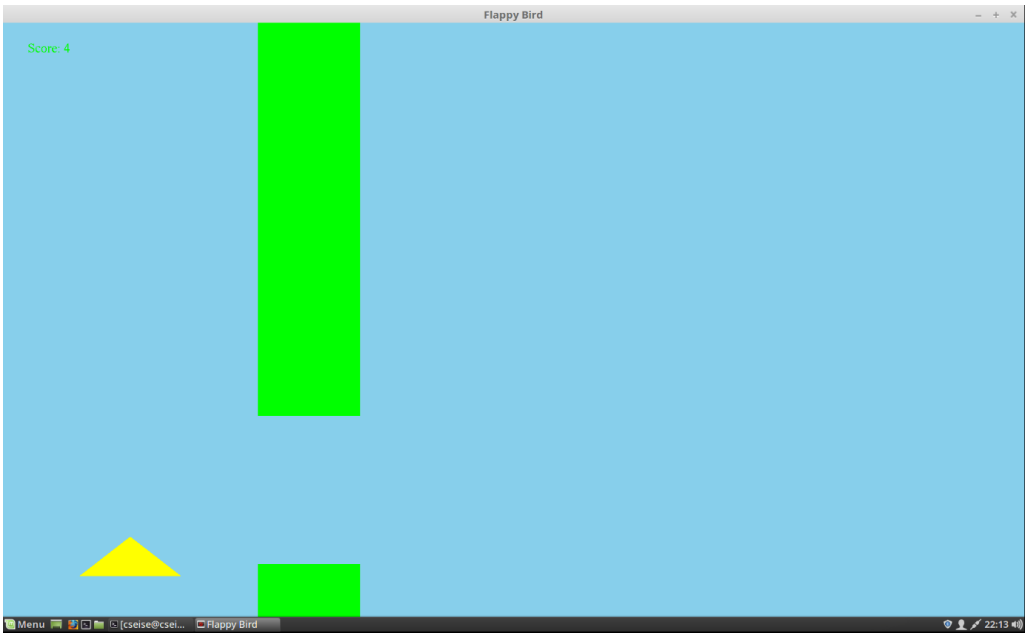

Fig.5.3 Before collision

Fig.5.3 shows that the player has scored 4 points and the triangle (flappy bird) hasnot yet collided with the obstacle.

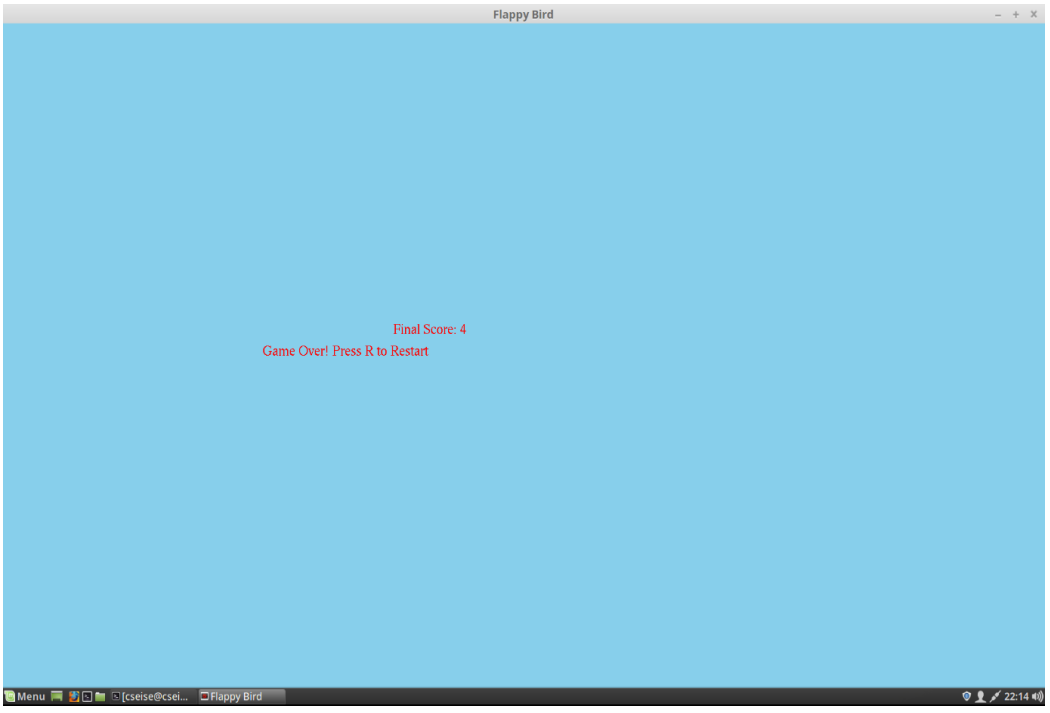

Fig.5.4 After collision

Fig.5.4 shows the game has ended the final score is 4.

CONCLUSION

CHAPTER 6

In conclusion, the Flappy Bird game project has been an insightful exploration into the realm of computer graphics and game development. Through the implementation of this project, we have gained a comprehensive understanding of various core concepts, including sprite animation, collision detection, and user interface design. The successful creation of a functional and engaging game underscores the importance of meticulous planning, iterative development, and rigorous testing.

This project also highlighted the significance of utilizing appropriate graphics libraries and frameworks, which facilitated the efficient rendering of game elements and the smooth execution of game logic. The challenges encountered during development, such as optimizing performance and managing game state, provided valuable learning experiences and enhanced our problem-solving skills.

Moreover, this project has demonstrated the practical application of theoretical knowledge in computer graphics, reinforcing our understanding of topics such as 2D transformations, event handling, and real-time rendering. It has also underscored the importance of creativity and innovation in game design, encouraging us to think critically about user experience and gameplay mechanics.

Overall, the Flappy Bird game project has been a rewarding endeavor, equipping us with the skills and confidence to tackle more complex projects in the future. It has laid a solid foundation for further exploration and advancement in the field of computer graphics and interactive media.

REFERENCES

[1] Huynh, D. (2014). The Phenomenon of Flappy Bird: An Analysis of Its Sudden Popularity.

[2] Smith, J. & Clark, R. (2015). Minimalistic Game Design: The Success of Flappy Bird.

[3] Foley, J.D., van Dam, A., Feiner, S.K., & Hughes, J.F. (1990). Computer Graphics: Principles and Practice.

[4] Millington, I. (2007). Game Physics Engine Development.

[5] Mathews, T. (2016). Practical Collision Detection in 2D Games.