FLAPPING BIRD GAME FOR MOBILE APPLICATION

A CAPSTONE PROJECT
Submitted By

N Vamsi Chowdari
192210317
B Durga Prasad
192210417

In Partial Fulfillment for the completion of the course

CSA0911
PROGRAMMING IN JAVA FOR AN NEURAL PLATFORM
SEPT 2024

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

CHENNAI - 602105
TAMIL NADU, INDIA

SAVEETHA
INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES
(Declared as Deemed to be University under Section 3 of UGC Act 1956)

BONAFIDE CERTIFICATE

This is to certify that the project report entitled FLAPPING BIRD GAME FOR MOBILE APPLICATIONS submitted by N.Vamsi chowdari - 192210317,B.Durga prasad-192210417 to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of bonafide work carried out by him/her under my guidance. The project fulfills the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

Dr.Ganesh Ramachandran
COURSE FACULTY
Department of Deep Learning.
Saveetha School of Engineering,
SIMATS, Chennai - 602105

ACKNOWLEDGEMENT

This project work would not have been possible without the contribution of many people.

It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor **Dr. N M VEERAIYAN**, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics **Dr. DEEPAK NALLASWAMY,** SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director **Dr. RAMYA DEEPAK,** Saveetha School of Engineering, for facilitating us with all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to **Dr. B RAMESH,** Principal, Saveetha School of Engineering for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty **S GODWIN**, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire faculty of the Department of Design, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

INDEX

BONAFIDE CERTIFICATE 1

ACKNOWLEDGEMENT 2

1. ABSTRACT 4

2. INTRODUCTION 4

3. ARCHITECTURE DIAGRAM 5

4. FLOWCHART 7

5. UML DIAGRAM 9

6. CLASS DIAGRAM 10

7. CODE IMPLEMENTATION .... 12 7.1 JAVA CODE 12

7.2 HTML CODE 15

7.3 CSS CODE 15

8. OUTPUT SCREENSHOT 18

9. CONCLUSION 21

10. REFERENCES 22

1. ABSTRACT

This capstone project focuses on the development of a Flappy Bird game using Java for desktop applications. The goal is to create an engaging and interactive game where players control a bird to navigate through a series of pipes. The system architecture adopts a modular approach, with a Java backend responsible for game logic, collision detection, and scoring. The frontend utilizes Java's Swing framework, offering a user-friendly interface and incorporating custom animations and visual effects. Key functionalities include multiple levels, increasing difficulty, responsive controls, and score tracking. This project demonstrates the application of Java programming skills in creating a polished and enjoyable desktop game, showcasing proficiency in object-oriented design, graphical user interface (GUI) development, and event-driven programming.

2. INTRODUCTION

The Flappy Bird game is a popular arcade-style game known for its simplicity and challenging gameplay. This capstone project aims to develop a modern, engaging version of Flappy Bird using Java, offering users an immersive and enjoyable gaming experience.

The project is organized around several key components. The backend is implemented in Java, managing game logic, collision detection, scoring, and level progression. The frontend utilizes Java's Swing framework to create a dynamic and responsive graphical user interface (GUI). Custom graphics and animations are integrated to enhance the visual appeal and overall immersion of the game. Additionally, sound effects are included to enrich the gaming experience.

In the game, players control a bird attempting to navigate through a series of pipes by tapping the screen to flap. The objective is to avoid colliding with the pipes while aiming to achieve the highest score possible. Each successful navigation through the pipes earns points, and the game progresses by increasing the difficulty, challenging players to improve their skills and reflexes. One of the primary challenges in developing this game was ensuring smooth and responsive controls. Managing real-time interactions between the bird and the pipes required precise collision detection and physics calculations. The application leverages Java's event-handling capabilities to deliver accurate control and fluid animations.

In conclusion, this capstone project showcases the application of Java programming skills in game development, demonstrating the creation of a polished and engaging Flappy Bird game. The project highlights technical proficiency in object-oriented design, graphical user interface (GUI) development, and event-driven programming. By integrating Java for both backend logic and frontend presentation, the project exemplifies the effective use of technology to deliver a cohesive and enjoyable gaming experience. This endeavor not only underscores technical competencies but also addresses practical challenges in game design and user interaction.

3. ARCHITECTURE DIAGRAM

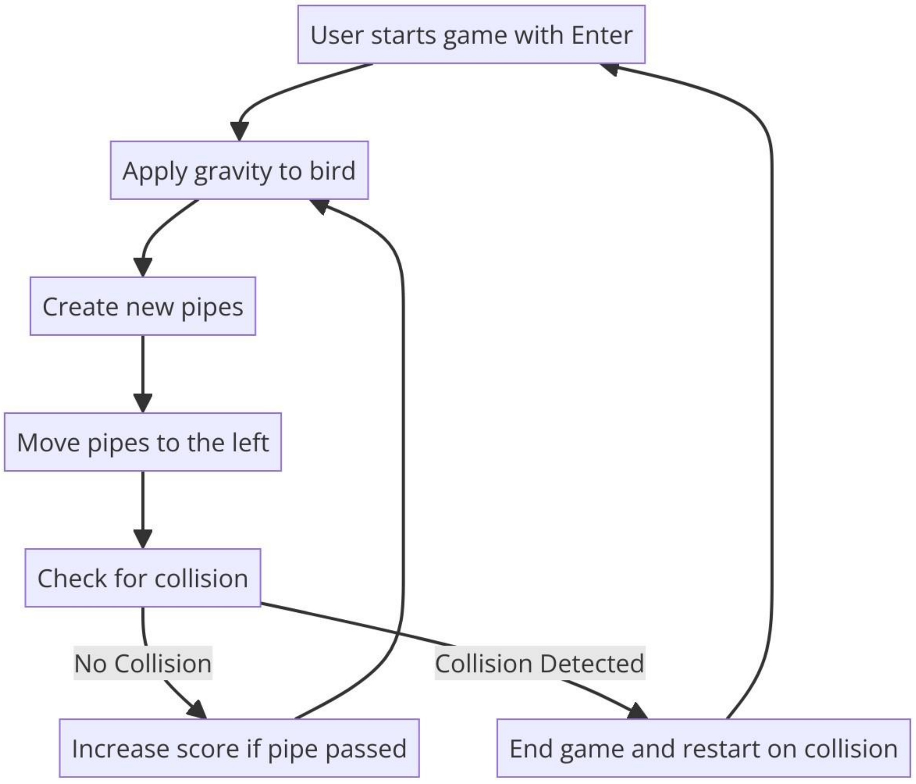


Figure 3.1 : Architecture Diagram

This architecture diagram outlines the structure of the Flappy Bird game, illustrating how various components interact to provide a seamless gaming experience. The architecture is designed around modular components, each responsible for specific aspects of the game, ensuring a smooth, responsive, and enjoyable experience for players.

- **Game Logic**: The core of the architecture, implemented in Java, handles game mechanics, collision detection, scoring, and level progression. This component ensures that the game runs consistently and accurately.

- **Frontend Interface:** Built using Java's Swing framework, this component provides the graphical user interface (GUI) of the game. It includes custom animations, visual effects, and interactive elements to engage players.
- **Event Handling:** Java's event-handling capabilities manage user inputs and interactions, such as flapping the bird and navigating through pipes. This ensures responsive controls and smooth gameplay.
- **Graphics and Animations:** Custom graphics and animations enhance the visual appeal of the game, making it more immersive and engaging for players.
- **Sound Effects:** Integrated sound effects contribute to the overall gaming experience by providing audio feedback for actions such as scoring points and collisions.
- **Performance Management:** The game is optimized for smooth performance on various devices, with careful attention to collision detection and physics calculations to ensure accurate and responsive gameplay.
- **Scalability:** While designed for a single-player experience, the modular architecture allows for future expansion, such as adding additional levels or features.

This architecture is well-suited for creating a polished and engaging desktop game, offering flexibility in development and maintenance. Each component can be developed and tested independently, making it easier to manage and enhance the game over time.

4. FLOWCHART

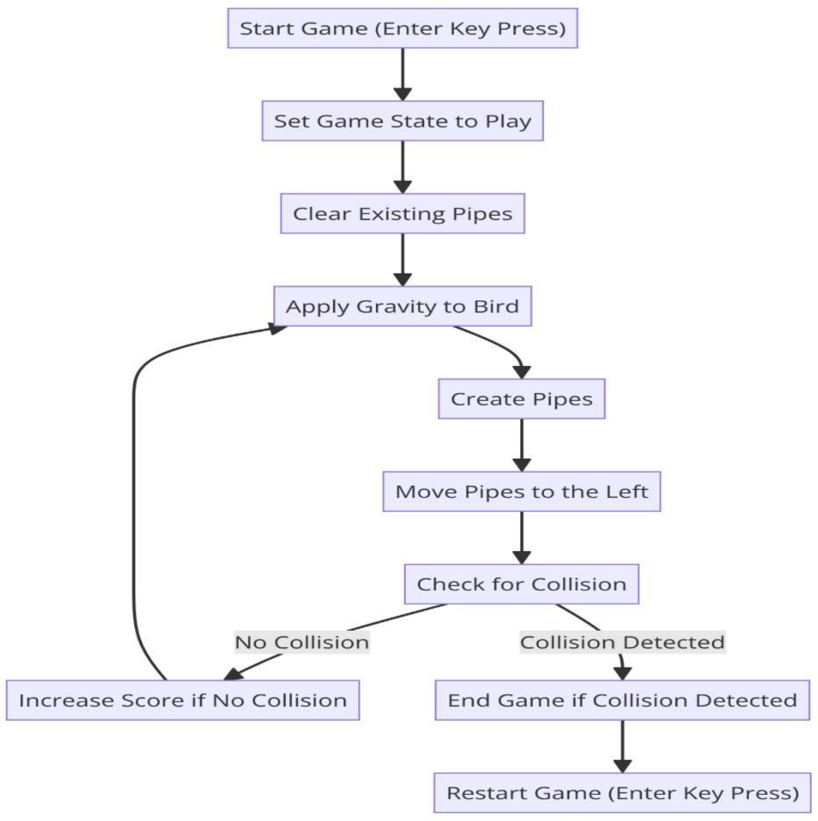


Figure 4.1 : Flowchart

This flowchart illustrates the core gameplay loop of the Flappy Bird game. The game begins when the player starts a new session and continues as long as the player successfully navigates the bird through the pipes without colliding with them. The gameplay revolves around controlling the bird’s flight by tapping the screen to keep it airborne, avoiding obstacles to achieve the highest score possible.

- **Game Start:**
    - The game begins when the player initiates a new session.
    - This could be triggered by selecting "Start" from the main menu or after initializing the game environment.

- **Play Game**:
    - The main gameplay loop is initiated, where the player taps the screen to control the bird's flight.
    - The game continues as long as the player avoids collisions with pipes and the ground.

- **Bird Flies**:

    - The bird moves horizontally across the screen while the player controls its vertical movement by tapping to make it flap.
    - Gravity continuously pulls the bird downward, requiring the player to maintain a balance between ascending and descending.

- **Pipe Collision (Decision Point)**:
    - The game checks whether the bird has collided with a pipe or the ground.
    - If the bird collides with an obstacle or the ground, the game moves to the next step of checking the game-over condition.

- **Score Point:**
    - When the bird successfully passes between two pipes without colliding, the player earns a point. o The game continues, with the bird flying toward the next set of pipes.

- **Game Over (Decision Point)**:
    - If the player fails to navigate the bird past a pipe (i.e., the bird collides with a pipe or the ground), the game ends with a "Game Over" screen.
    - The player can then choose to restart the game or return to the main menu.

This flowchart encapsulates the essence of the Flappy Bird game's lifecycle, highlighting key interactions and decision points that influence the player's progression and overall experience. The flowchart demonstrates the simplicity and challenge of the game, where each tap and decision can significantly impact the outcome.

5. UML DIAGRAM

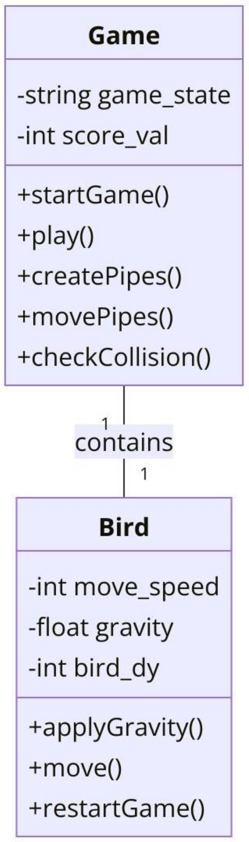


Figure 5.1 : UML DIAGRAM

The UML diagram provides a visual representation of the classes and their relationships within the Flappy Bird game.

- **Player:** The actor who initiates and controls the game. The player starts the game, controls the bird's flight by tapping the screen, and navigates through the pipes to avoid collisions.

- **Game:** Represents the overall game management system. It handles the initialization of game components, score tracking, collision detection, and game over conditions. It coordinates the interactions between the different game objects.
- **Bird:** The central game object controlled by the player. The bird moves horizontally across the screen, with its vertical movement controlled by the player's taps. The bird's position and movement are critical to avoiding collisions with pipes and the ground.

- **Pipe:** Represents the obstacles that the bird must navigate through. The pipes move horizontally across the screen from right to left. If the bird collides with a pipe, the game checks for a game-over condition.

- **Ground:** Another game object that the bird must avoid colliding with. If the bird hits the ground, the game also checks for a game-over condition.

- **Score:** Tracks the player's score, which increases each time the bird successfully passes between a pair of pipes. The score is updated and displayed continuously during the game.

A sequence diagram could be used to show the step-by-step flow of control between these objects during gameplay. It would illustrate how the player interacts with the game components (bird, pipes, and ground) and how the game handles the core gameplay loop, collision detection, scoring, and game-over conditions in the Flappy Bird game.

6. CLASS DIAGRAM

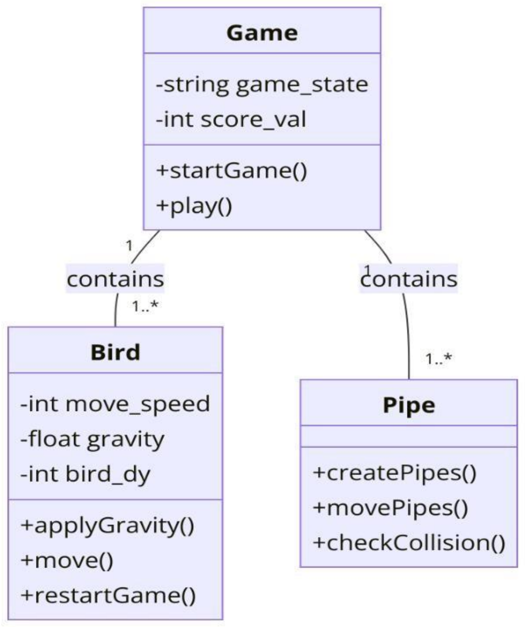


Figure 6.1 : CLASS DIAGRAM

The class diagram specifies the structure of the Flappy Bird game by illustrating the system's classes, their attributes, methods, and the relationships among the objects.

- **Game:** ○ Attributes: score, is Game Over, and difficulty Level.

- Methods: start Game(), pause Game(), resume Game(), end Game(), and update Score().

- **Bird:**
  - Attributes: position (x, y), velocity, and gravity.
  - Methods: flap(), move(), check Collision(), and update Position().

- **Pipe:**
  - Attributes: position (x, y), width, height, and is Passed.
  - Methods: move(), check Collision With Bird(), and reset Position().

- **Ground:** o Attributes: position (x, y), width, and height. o Methods: check Collision With Bird().

- **Score:**
  - Attributes: current Score.
  - Methods: increment Score(), reset Score(), and display Score().

- **Level:**
  - Attributes: pipe Set, ground, and bird.
  - Methods: load Level(), reset Level(), and is Level Complete().

- **Input Handler:**
  - Attributes: current Input.
  - Methods: handle Input() and process Tap().

- **UI:**
  - Attributes: score Display, game Over Screen, and start Screen.
  - Methods: update Score Display(), show Game Over Screen(), and show Start Screen().

7. CODE IMPLEMENTATION

7.1 JAVA CODE
```
let move_speed = 3; let gravity = 0.5; let bird = document.querySelector('.bird'); let
background = document.querySelector('.background').getBoundingClientRect(); let score_val =
document.querySelector('.score_val'); let message = document.querySelector('.message'); let
score_title = document.querySelector('.score_title'); let game_state = 'Start'; let bird_dy = 0; let
pipe_seperation = 0; let pipe_gap = 35;

// Start the game on Enter key press document.addEventListener('keydown', (e) => {     if
(e.key == 'Enter' && game_state != 'Play') {
document.querySelectorAll('.pipe_sprite').forEach(e => e.remove());        bird.style.top =
'40vh';        game_state = 'Play';        message.innerHTML = '';        score_title.innerHTML
= 'Score : ';        score_val.innerHTML = '0';        play();
  }
});

// Main game function function play() {
function move() {        if (game_state !=
'Play') return;

    document.querySelectorAll('.pipe_sprite').forEach(element => {
      let  pipe_sprite_props  =  element.getBoundingClientRect();                    let
bird_props = bird.getBoundingClientRect();

      if (pipe_sprite_props.right <= 0) {               element.remove();
      }     else     {
if (
          bird_props.left < pipe_sprite_props.left + pipe_sprite_props.width &&                  bird_props.left +
bird_props.width > pipe_sprite_props.left &&                  bird_props.top < pipe_sprite_props.top +
pipe_sprite_props.height &&                  bird_props.top + bird_props.height > pipe_sprite_props.top
        ) {
          game_state = 'End';                  message.innerHTML =
'Press Enter To Restart';                  message.style.left = '28vw';
return;              } else {                  if (
                    pipe_sprite_props.right < bird_props.left &&                    pipe_sprite_props.right +
move_speed >= bird_props.left &&                    element.increase_score == '1'
                ) {
                    score_val.innerHTML = +score_val.innerHTML + 1;
                }
                element.style.left = pipe_sprite_props.left - move_speed + 'px';
            }
        }
    });

    requestAnimationFrame(move);
  }
  requestAnimationFrame(move);

  function apply_gravity() {
    if (game_state != 'Play') return;

    bird_dy += gravity;
document.addEventListener('keydown', (e) => {          if (e.key
== 'ArrowUp' || e.key == ' ') {              bird_dy = -7.6;
        }
    });

    let bird_props = bird.getBoundingClientRect();

    if (bird_props.top <= 0 || bird_props.bottom >= background.bottom) {
game_state = 'End';           message.innerHTML = 'Press Enter To Restart';
message.style.left = '28vw';           return;
    }
    bird.style.top = (bird_props.top + bird_dy) + 'px';       requestAnimationFrame(apply_gravity);
  }
  requestAnimationFrame(apply_gravity);

  function create_pipe() {      if (game_state
!= 'Play') return;
```

```
        if (pipe_seperation > 115) {            pipe_seperation = 0;
            let pipe_posi = Math.floor(Math.random() * 43) + 8;             let
pipe_sprite_inv = document.createElement('div');
pipe_sprite_inv.className = 'pipe_sprite';             pipe_sprite_inv.style.top
= (pipe_posi - 70) + 'vh';             pipe_sprite_inv.style.left = '100vw';
document.body.appendChild(pipe_sprite_inv);

            let pipe_sprite = document.createElement('div');
pipe_sprite.className = 'pipe_sprite';             pipe_sprite.style.top = (pipe_posi
+ pipe_gap) + 'vh';             pipe_sprite.style.left = '100vw';
pipe_sprite.increase_score = '1';
document.body.appendChild(pipe_sprite);
        }
        pipe_seperation++;         requestAnimationFrame(create_pipe);
    }
    requestAnimationFrame(create_pipe);
```

7.2 HTML CODE

```
<!DOCTYPE html>
<html> <head>
<link rel="stylesheet" href="style.css">
</head>
<body>
<div class="background">
<h1 class="game-title">Flappy Bird Game</h1>
<img class="bird" src="logo.png" alt="bird-img">
 <div class="message">
Press Enter To Start Game
</div>
<div class="score">
<span class="score_title"></span>
 <span class="score_val"></span>
 </div>
 </div>
<script src="gfg.js"></script>
</body> </html>
```

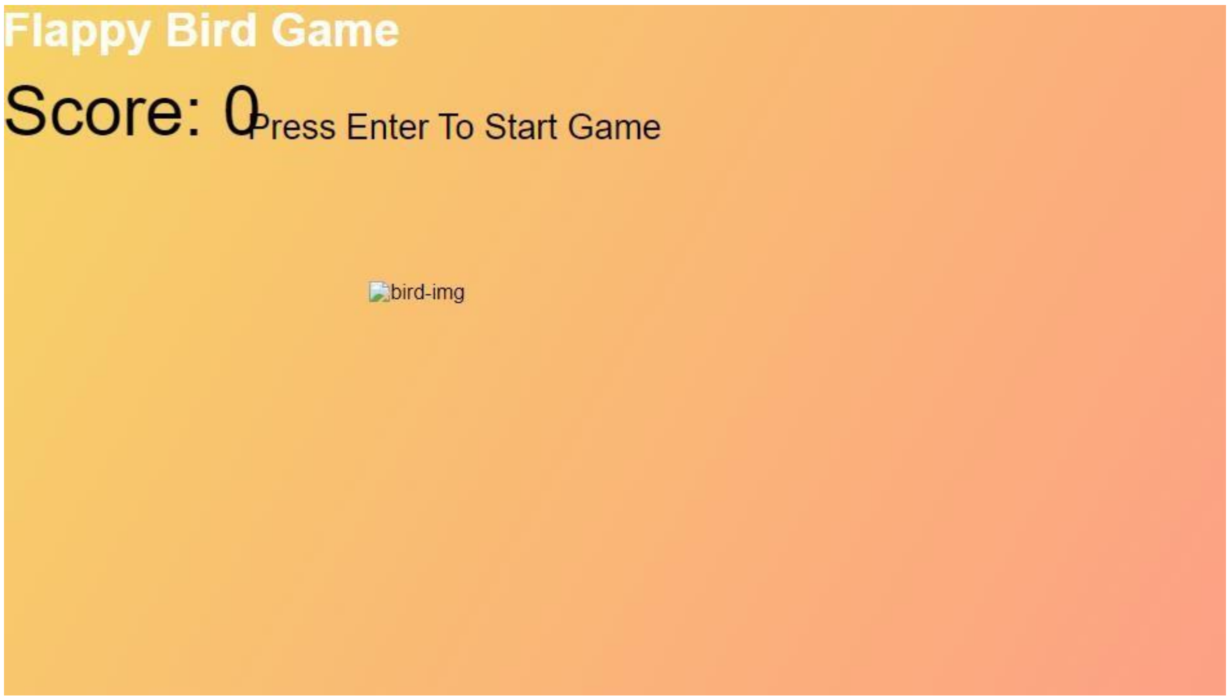
7.3 CSS CODE

```
/* styles.css */

body {   font-family: "Arial", sans-serif;   background: linear-gradient(120deg,
#f6d365 0%, #fda085 100%);   margin: 0;   display: flex;   justify-content: center;
align-items: center;   height: 5vh;
}

.container {   text-
align:center;   max-width:
600px;   width: 100%;
margin: auto;
}
```

```
h1 {   color: #ffffff;   font-
size: 36px;   margin-bottom:
10px;
}

#gameCanvas {   background:
#70c5ce;   border: 2px solid
#fff;   display: block;
margin: 0 auto 20px;
border-radius: 10px;
}

.controls {   display: inline;
justify-content: center;
}

#startButton {   padding: 10px 20px;   font-size:
18px;   background-color: #ff6f61;   color: #fff;
border: none;   border-radius: 5px;   cursor:
move;   transition: background-color 0.3s ease;
}

#startButton:hover {   background-color: #ff3d3d;
}

@media (max-width: 600px) {
  h1  {           font-size:
15px;
  }

  #gameCanvas {
width: 60%;     height:
auto;
  }
  #startButton {    font-size: 16px;
  }
}
* {   margin: 0;   padding: 0;
box-sizing: border-box;
}

body {   height:
100vh;   width:
100vw;
}

.background {   height: 100vh;
width: 100vw;   background-color:
reddish;
}

.bird {   height:
100px;   width:
160px;   position:
fixed;   top: 40vh;
left: 30vw;   z-
index: 100;
}

.pipe_sprite {   position: fixed;
top: 40vh;   left: 100vw;   height:
70vh;   width: 6vw;
background-color: green;
}

.message {   position:
Fixed;   z-index: 10;
height: 5vh;   font-size:
```

```
5vh;   font-weight:
100;   color: black;
top: 15vh;   left: 20vw;
text-align:center;
}

.score {   position:
fixed;   z-index: 10;
height: 5vh;   font-size:
10vh;   font-weight:
100;   color: black;
top: 30;   left: 0; }

.score_val  {       color:
black;
}
```

8. OUTPUT SCREENSHOT


Flappy Bird Game

Score: 0Press Enter To Start Game

Figure 8.1 : Start the game.

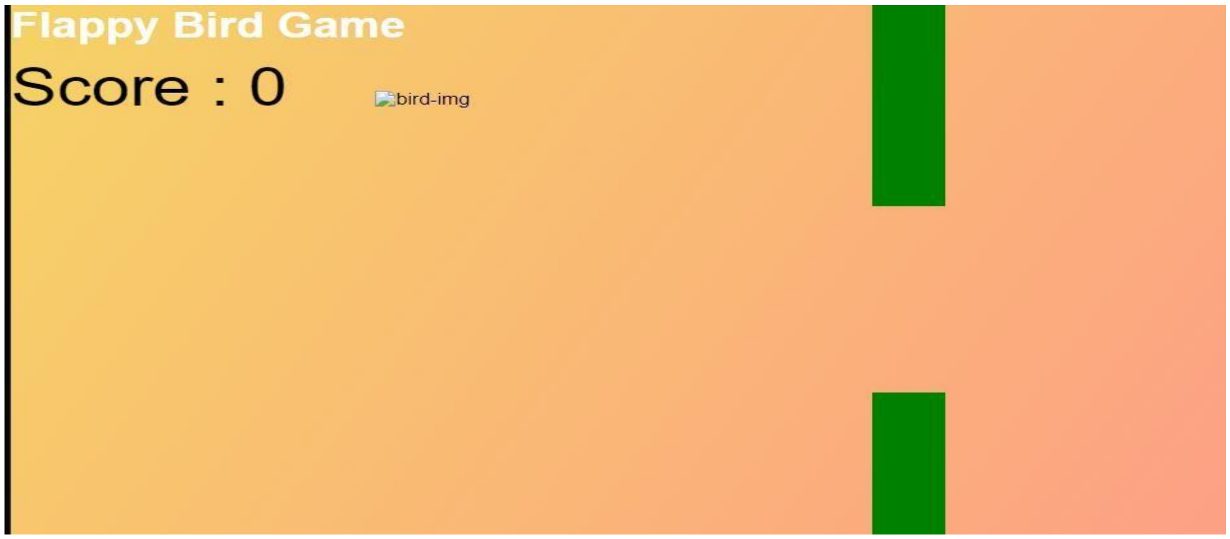


Figure 8.2 : Move the bird

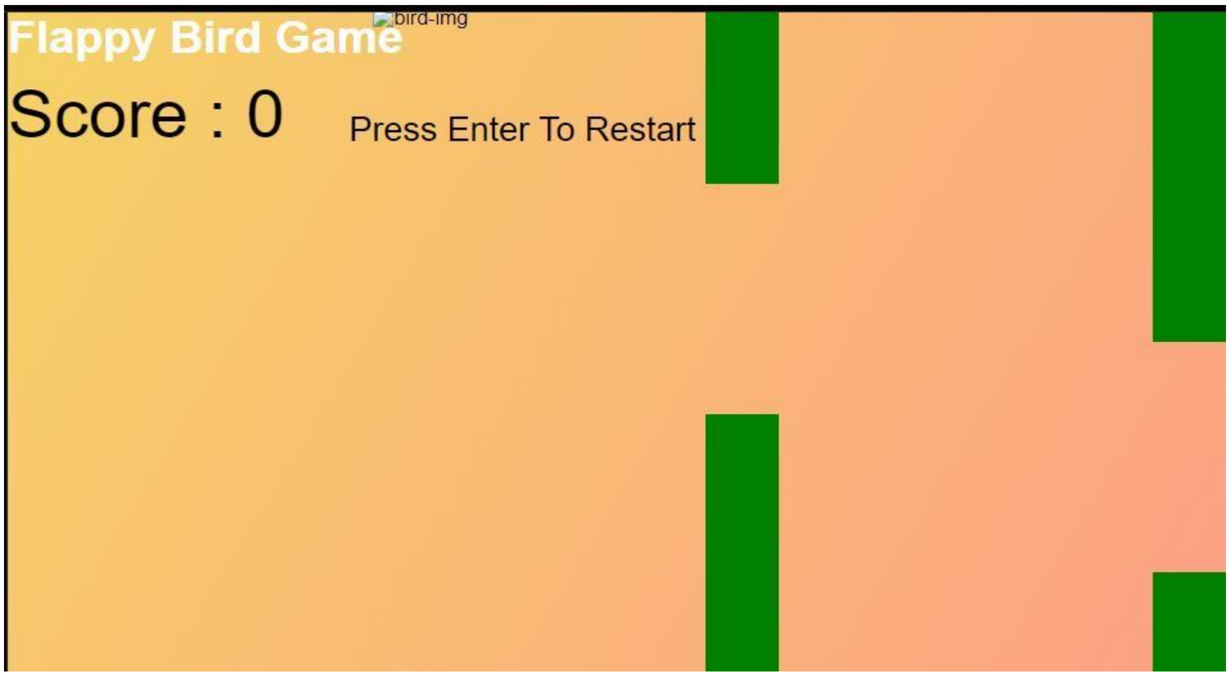


Figure 8.3 : Continuing the game

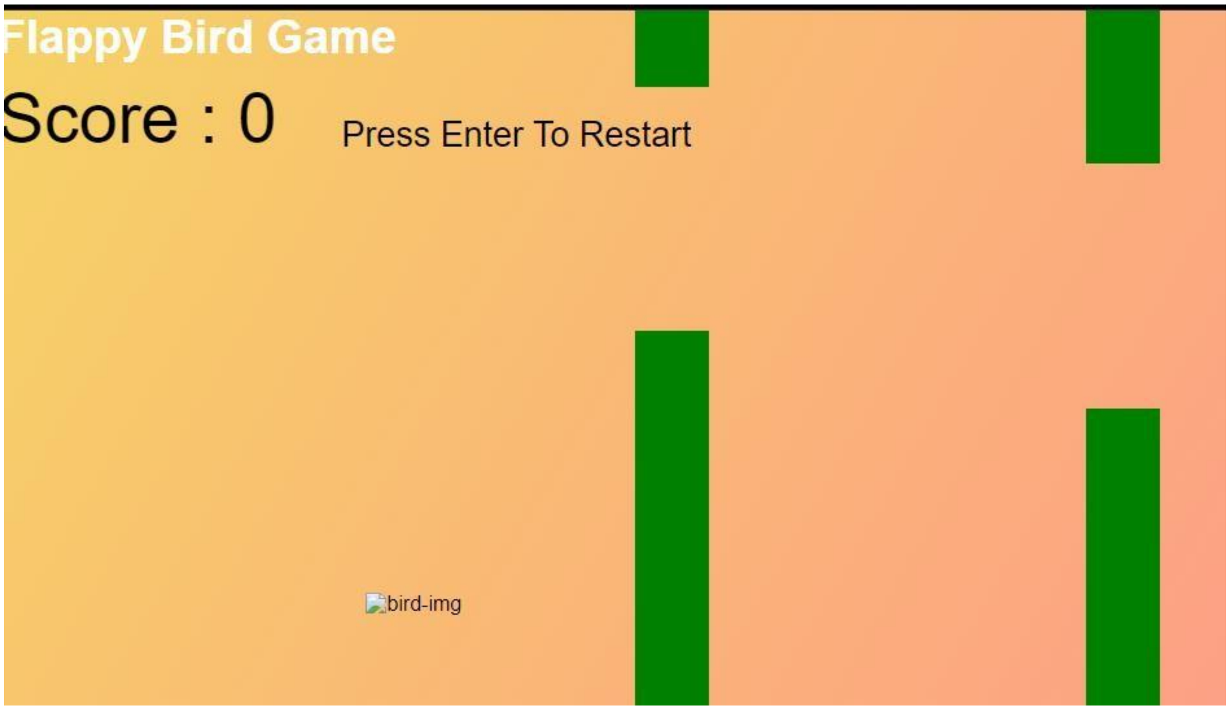


Figure 8.4 : You lost the game

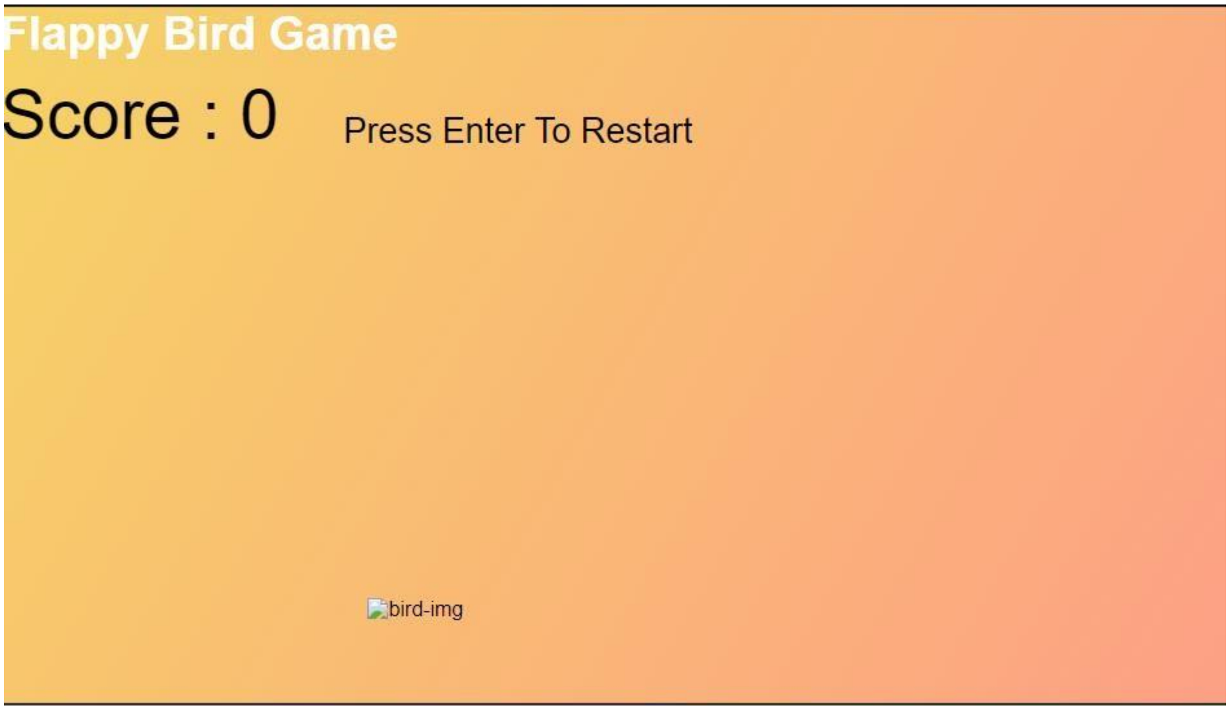


Figure 8.5 : Start the game again

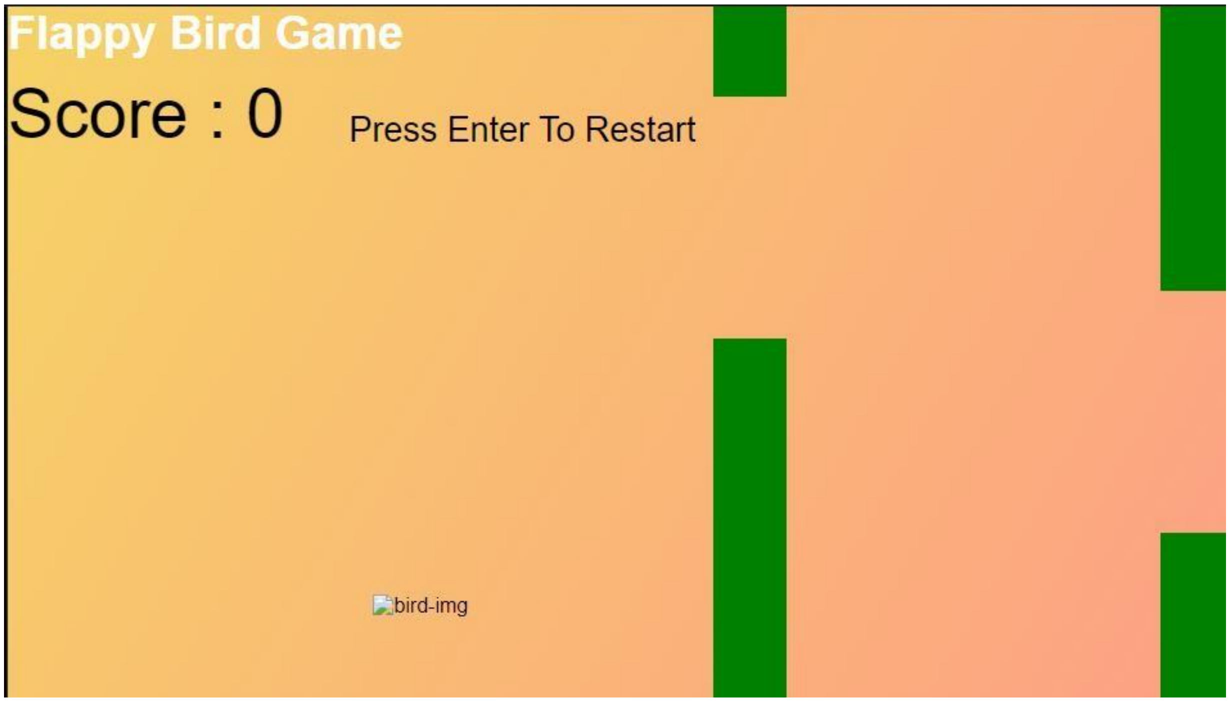


Figure 8.6 : You won the game

9. CONCLUSION

This capstone project successfully developed the Flappy Bird game, effectively addressing the need for an engaging and interactive gaming experience. The application integrates real-time game mechanics with a responsive user interface, ensuring both functionality and user enjoyment. By combining a robust Java backend for game logic with a visually appealing frontend built using Java's Swing framework, the project exemplifies the effective use of modern technologies in game development. The challenges of creating dynamic gameplay, managing game states, and designing an intuitive user interface were met with creative solutions, resulting in a polished and entertaining game. This project highlights the importance of programming skills and game design principles in creating immersive experiences, showcasing the potential of software development to entertain and engage users in the digital age.

10. REFERENCES

1. Yannakakis, G.N., Togelius, J.: Artificial Intelligence and Games. Springer, New York (2018.

2. Rhiannon, W.: What is Flappy Bird? The game taking the App Store by storm. The Daily Telegraph, 30 January 2014.

3. Hsu, F.-H.: IBM's deep blue chess grandmaster chips. IEEE Micro **19**(2), 70–81 (1999).

4. Mnih, V., et al.: Playing Atari with deep reinforcement learning. In: NIPS Deep Learning Workshop (2013).

5. Hausknecht, M., Lehman, J., Miikkulainen, R., Stone, P.: A neuroevolution approach to general Atari game playing. IEEE Trans. Comput. Intell. AI Games **6**(4), 355–366 (2014).

6. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search.
Nature **529**(7587), 484–489 (2016)..

7. Silver, D., et al.: Mastering the game of go without human knowledge. Nature **550**, 354 (2017).

8. Appiah, N., Vare, S.: Playing FlappyBird with Deep Reinforcement Learning.

9. Qiang, W., Zhongli, Z.: Reinforcement learning model, algorithms and its application. In: International Conference on Mechatronic Science, Electric Engineering and Computer, 19–22 August 2011.

10. Risi, S., Togelius, J.: Neuroevolution in games: state of the art and open challenges. IEEE Trans. Comput. Intell. AI Games **9**(1), 25–41 (2017).