MINOR PROJECT REPORT

On

Flappy Bird

Submitted in Partial fulfillment for the Award of the degree of

Bachelor of Technology

In

INFORMATION TECHNOLOGY

Submitted to

RAJIV GANDHI PROUDYOGIKI VISHWAVIDHYALAYA, BHOPAL (M.P)

Submitted by

Vivek Waryani (0126IT211125)

Sakshi Rajput (0126IT211094)

Divyansh Choudhary (0126IT223D01)

Shilpa Shakya (0126IT211104)

Rahul Acharya (0126IT211076)

Under the guidance of

PROF. MOHD. IQBAL

Assistant Professor

ORIENTAL COLLEGE OF TECHNOLOGY, BHOPAL

Department of Information Technology

Approved by AICTE New Delhi & Govt. of MP

Affiliated to Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal

JULY - DEC 2023

Phone No.-0755-2529015, 2529016
Fax: 0755-2529472
E-mail: directoroct@oriental.ac.in
Website: http://www.oriental.ac.in/oct-bhopal/

ORIENTAL COLLEGE OF TECHNOLOGY, BHOPAL

Approved by AICTE, New Delhi & Govt. of M.P. Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal
Oriental Campus, Raisen Road, Bhopal-462021 (MP) INDIA

DEPARTMENT OF INFORMATION TECHNOLOGY

CANDIDATE'S DECLARATION

I hereby declare that the Minor Project report on "**Flappy Bird**" which is being presented here for the partial fulfillment of the requirement of Degree of ***"Bachelor of Technology"*** has been carried out at **Department of Information Technology. Oriental College of Technology Bhopal**. The technical information provided in this report is presented with due permission of the authorities from the training organization.

Signature of Students

Vivek Waryani (0126IT211125)
Divyansh Choudhary (0126IT223D01)
Shilpa Shakya (0126IT211104)
Sakshi Rajput (0126IT211094)
Rahul Acharya (0126IT211076)

Phone No.-0755-2529015, 2529016
Fax: 0755-2529472
E-mail: directoroct@oriental.ac.in
Website: http://www.oriental.ac.in/oct-bhopal/

ORIENTAL COLLEGE OF TECHNOLOGY, BHOPAL

Approved by AICTE, New Delhi & Govt. of M.P. Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal
Oriental Campus, Raisen Road, Bhopal-462021 (MP) INDIA

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE OF INSTITUTE

This is to certify that Mr. / Ms. **Vivek Waryani, Divyansh Choudhary, Shilpa Shakya, Sakshi Rajput, Divyansh Choudhary, Rahul Acharya** of B. Tech. Information Technology Department Enrollment No. **0126IT211125, 0126IT223D01, 0126IT211104, 0126IT211094, 0126IT211076** has successfully **completed** his / her Minor Project during the academic year 2023-2024 as partial fulfilment of the Bachelor of Technology in Information Technology.

Mohd. Iqbal

Assistant Professor

Sujeet Kumar Jha

Head OF Department

ACKNOWLEDGEMENT

Acknowledgments extend to the Pygame community, open-source contributors, and resources that facilitated the development process. We would like to express our gratitude to our Mentor, Prof. Iqbal Sir, who was a continual source of inspiration. She pushed us to think imaginatively and urged us to do this homework without hesitation. Her vast knowledge, extensive experience, and professional competence in Game Development enabled us to successfully accomplish this project. This endeavour would not have been possible without her help and supervision. We could not have asked for a finer mentor in our studies. This initiative would not have been a success without the contributions of each and every individual. We were always there to cheer each other on, and that is what kept us together until the end.

Vivek Waryani (0126IT211125)
Sakshi Rajput (0126IT211094)
Divyansh Choudhary (0126IT223D01)
Shilpa Shakya (0126IT211104)
Rahul Acharya (0126IT211076)

List of Figures

List of Symbols & Abbreviations

Py	**Python**
Pygame	**Python Game**

Executive Summary:

Pygame is a set of Python modules designed for writing video games. Pygame adds functionality on top of the excellent SDL library. This allows you to create fully featured games and multimedia programs in the python language.

The Flappy Bird Project represents a comprehensive exploration into the development and analysis of the iconic Flappy Bird game. The project encompasses various facets, from the initial concept and design to the implementation of game mechanics and subsequent evaluation.

Objective:

The primary goal of the Flappy Bird Python Game is to emulate the essence of the original game, where players control a bird that must navigate through a series of pipes by tapping the screen to make the bird flap its wings. The objective is to achieve the highest possible score by successfully passing through as many pipes as possible without colliding.

Key Features:

1. Game Mechanics: The game mechanics closely mimic the original Flappy Bird, requiring players to tap to make the bird ascend and navigate through openings in pipes. Gravity and physics are simulated to create a realistic flying experience for the bird.

2. Scoring System: A scoring system tracks the player's progress, incrementing with each successfully passed pipe. The game provides immediate feedback on the player's performance, encouraging repeated play.

3. Graphics and User Interface: The game features a simple yet visually appealing design, leveraging Python libraries such as Pygame for graphics and user interface elements. Intuitive controls enhance user experience, allowing players to focus on gameplay without unnecessary distractions.

4. End-Game Handling: The game includes appropriate handling for collisions with pipes, providing feedback to the player and allowing them to restart the game. An end-game screen displays the final score, encouraging players to strive for improvement in subsequent attempts.

5. Customization and Extensibility: The codebase is structured in a modular and extensible manner, enabling developers to easily modify game parameters, add features, or customize the visual elements.

The Flappy Bird Python Game project successfully recreates the beloved Flappy Bird experience using Python, offering users a nostalgic and entertaining gameplay environment. With its faithful emulation of the original game mechanics, scoring system, and user interface, this project serves as a testament to the versatility and capabilities of Python in game development.

TABLE OF CONTENTS

1. Introduction:

- This project is about the “Flappy Bird Game” which uses py-game module to get organised correctly an efficient manner.

- This module contains functions of every basic asset which are crucial for building any game, it contains- physics, blitting, FPS, clock, 2D and much more.

- We used such functions and classes to create Flappy Bird where user can give input to which our avatar reacts and the goal is to pass through all the random generated pipes as far as the user can and get good scores.

- But there is a twist, that every time when the avatar passes any pipe the user will get a gradual increment in the speed of the avatar.

Creating a Flappy Bird game using Python can be an exciting and educational project for those looking to delve into game development. In this introduction, we will cover the basics of the Flappy Bird game, discuss the Python programming language, and provide an overview of the libraries used for game development. Additionally, we'll touch upon the fundamental concepts of game design and how they apply to creating a Flappy Bird clone.

1.1 Python Programming Language:

Python is a versatile and beginner-friendly programming language widely used in various domains, including web development, data science, artificial intelligence, and game development. Known for its readability and concise syntax, Python is an excellent choice for beginners and experienced developers alike.

In the context of game development, Python offers several libraries and frameworks that simplify the process. One popular library for 2D game development is Pygame. Pygame provides functions and tools for handling graphics, sound, and user input, making it well-suited for creating simple games like Flappy Bird.

1.2 Pygame Library:

Pygame is a cross-platform set of Python modules designed for writing video games. It builds on top of the Simple DirectMedia Layer (SDL) library, providing a straightforward way to handle graphics, sound, and user input. Pygame is widely used in the Python game development community due to its simplicity and ease of use.

1.3 Basic Game Components:

Before diving into the implementation, let's outline the fundamental components of our Flappy Bird game:

1. Game Loop:
The game loop is the core structure that repeatedly updates the game state and renders the graphics. It ensures a continuous and interactive experience for the player.

2. Bird Character:
The main character is the bird that the player controls. It responds to user input (in this case, tapping) to move up and avoid obstacles.

3. Obstacles (Pipes):
The obstacles in Flappy Bird are vertical pipes with gaps for the bird to pass through. They move from right to left, creating a challenging environment for the player.

4. Collision Detection:
To make the game challenging, we need to detect collisions between the bird and the pipes. If a collision occurs, the game ends.

5. Scoring System:
Implement a scoring system based on the number of pipes successfully passed. This adds a competitive element to the game.

1.4 Game Design Concepts:

Understanding basic game design concepts is crucial for creating an enjoyable gaming experience. Here are a few concepts to keep in mind:

1. User Interface (UI):
Design a clean and intuitive user interface. Display the score, game over screen, and any other relevant information.

2. Game Physics:
Implement realistic physics for the bird's movement. Gravity should constantly pull the bird downward, and user input should provide upward acceleration.

3. Feedback and Animation:
Provide visual and auditory feedback for user actions. Use animations to make the game visually appealing.

4. Difficulty Progression:
Gradually increase the game's difficulty as the player progresses. In Flappy Bird, this could involve speeding up the pipes or narrowing the gaps.

1.5 Implementation Overview:

Now that we have a clear understanding of the game's components and design concepts, let's provide a brief overview of the implementation steps:

1. Initialize the Game:
Set up the game window, define constants, and initialize Pygame.

2. Load Assets:
Load the bird sprite, background image, pipe images, and any other graphics or sounds required for the game.

3. Create the Game Loop:
Implement the game loop that updates the game state, handles user input, and renders the graphics.

4. Implement Bird Movement:
Allow the bird to respond to user input for upward movement. Apply gravity to simulate a natural falling motion.

5. Generate Pipes:
Create a mechanism to generate pipes at regular intervals, and move them from right to left.

6. Collision Detection:
Check for collisions between the bird and pipes. If a collision occurs, end the game.

7. Scoring:
Keep track of the number of pipes successfully passed by the bird and update the score accordingly.

8. User Interface:
Design and display a user interface that includes the score and any relevant information. Show a game over screen when the player loses.

9. Game Over Handling:
Implement a mechanism to handle game over scenarios, allowing the player to restart the game.

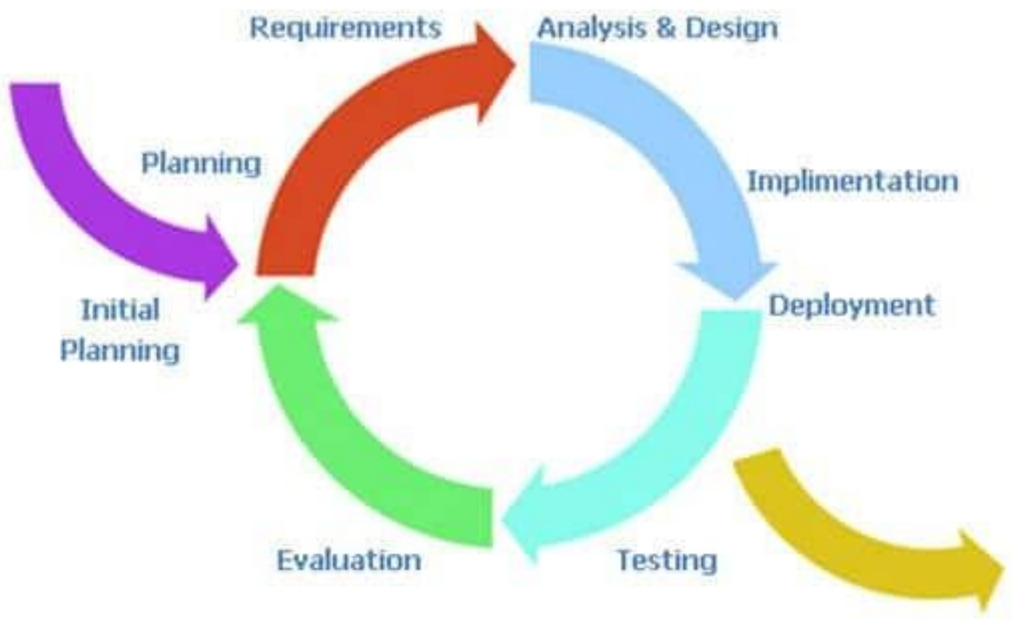
2. Literature Review:

2.1 Problem Statement:

- Design and implement a Flappy Bird game using the Pygame library, considering the challenges involved in creating a responsive and engaging gaming experience. The goal is to create a simple yet entertaining game with user-friendly controls and obstacles that progressively increase in difficulty.

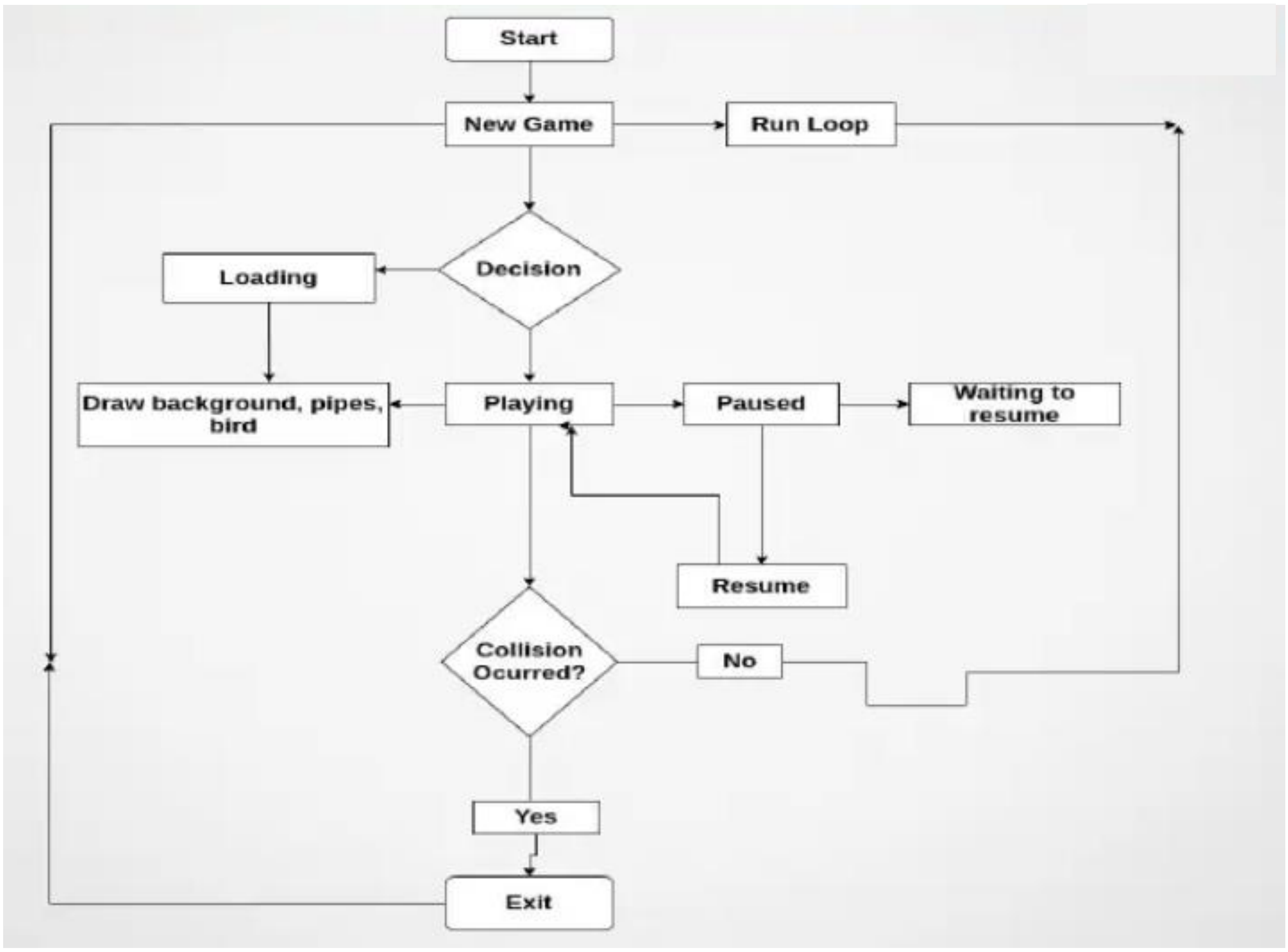
2.2 Methodology Used:

- The development of the Flappy Bird game involves using the Pygame library in Python. Pygame is a set of Python modules designed for writing video games.
- The game logic includes handling user input for controlling the bird's movement, managing collision detection with pipes, and implementing a scoring system. The gameplay is based on the classic Flappy Bird mechanics, where the player must navigate a bird through a series of pipes by tapping on the screen to make the bird jump.
- Here we used RAD (Rapid Application Development) model to create the software, RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element based construction approach.
- If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.
- Prototyping and early, reiterative user testing of designs



(Fig 2.1)

2.3 Flowchart Of Flappy Bird:



(Fig 2.2)

2.4 Creating the "__main__" function:

1. Now let's create the main function where our game will start and we have to initialize all pygame modules using pygame.init(). We also create fps_clock variable to help us track time at a moment using+ pygame.tick.Clock() function.

2. Then we will give a title to our main game window and store all the images in a tuple with first, which we are then assigning to the 'numbers' key in the game_images dictionary.

3. We use pygame.image.load() with paths of the images as arguments along with convert_alpha() to change the pixel format of an image including per pixel alphas.

4. Similarly, we add the images of the message, base, pipe, background, player, and title, into the dictionary using various keys. For pipe, we also added an inverted pipe image by

using pygame.transform.rotate() function and rotating the image by 180 degrees. We then add the sounds to the game_sounds dictionary using various keys.

5. It is similar to what we did for images but here we pygame.mixer.Sound() function with the paths for various sounds as the argument for storing the sounds. Then we start a loop calling the welcomeScreen() and mainGame() functions which will be defined in the later sections.

2.5 Creating "welcomeScreen" function:

1. Now, we define our welcomeScreen() function which will display the welcome screen on starting the game. We start by assigning the values of the x-coordinate and y-coordinate for the player, message, and title images.

2. We have selected the arguments by hit and trial method and you can alter the values that suit you the best. We also give the x-coordinate of base here. Then, we start a while loop which will always be True and thus will start a loop that will not stop unless the control says quit.

3. Here we make use of a for loop for analyzing all the events taking place throughout the game using pygame.event.get() function. Then we check that whenever a quit type of event is encountered by pressing the escape key, the game window will close.

4. We will check the next condition i.e. whether we clicked the up key or the space button. If yes, we will return from the function and start the game. And if no key or button is pressed, the welcome screen is displayed. For that, we will place the background, message, player, base, and title images using screen.blit() function.

5. Finally, we will update our window using pygame.display.update() and will update our clock variable with fps value as argument to show just 30 frames per second.

2.6 Creating the "mainGame()" function:

Now we define our mainGame() function by first initializing the variable score with 0, and also give the coordinates for player image and base again.
Then we create 2 pipes for blitting on the screen using getRandomPipe() which we will be defined later. Then we create a list of upper pipes (inverted ones) and lower pipes with their x and y coordinates.

Again we have chosen values by hit and trial method. Then, we declare variables for velocities in different directions for the bird. We also provide an acceleration variable. playerFlapVel is the velocity while flapping and playerFlapped is set to false (which is true only if the bird flaps). Then again we check for events.

1. First for exiting the game and exit the game if true.

2. Then we check if the up key or spacebar is pressed. If yes, we check if the player is below the screen top and if yes, we make some updates and play the sound of the wing using .play().

3. After this, we check if we have crashed using the isCollide() function we will define soon. If true, we will return from the function.

4. Then, we will check and update the scores. Using the player's, mid position, and the positions of the pipes, we increase the score if we cross a pipe and print it in the console.

Also, we play the point sound for crossing each pipe. Then if the player velocity in y-direction has not yet become max, we will provide the acceleration .

2.7 Simulation Parameters Used:

- Gravity: Simulating the effect of gravity on the bird's movement.

- Pipe Generation: Dynamic generation of pipes at regular intervals to create obstacles.

- User Input Handling: Capturing user input (keyboard) to control the bird's movement.

- Collision Detection: Detecting collisions between the bird and pipes to determine game over conditions.

- Scoring System: Implementing a scoring mechanism based on the number of pipes successfully passed.

2.8 Tools / Language / Simulation Environment:

- Language: Python
- Library: Pygame
- IDE: Any Python IDE (e.g., VSCode, PyCharm)
- Simulation Environment: Local machine running Python with Pygame installed.

2.9 Advantages/Disadvantages:

Advantages:

- Simplicity: Pygame simplifies game development, making it accessible for developers of various skill levels.

- Community Support: Active community and ample documentation for Pygame.

- Cross-Platform Compatibility: Games developed with Pygame can be run on multiple platforms.

Disadvantages:

- Performance: Pygame may have limitations in handling complex and resource-intensive games.

- Graphics Quality: While suitable for 2D games, Pygame might not be the best choice for high-end graphics.

2.10 Analysis:

- To learn control policies from high-dimensional image data, a Deep Q-Network agent with a
- computer vision model can perform much higher scores than humans.

- Training is very long but can be shortened with a wellchosen preprocessing method.

- Rewards start improving very late.

- Preprocessing helps the model learn policies faster: as a reward becoming positive means that the
- bird successfully passes the first obstacle, the model with preprocessing learns policies with
- 8,000-9000 games, the model without preprocessing learns with 11,000 games.

- There is a very high variance in the results.

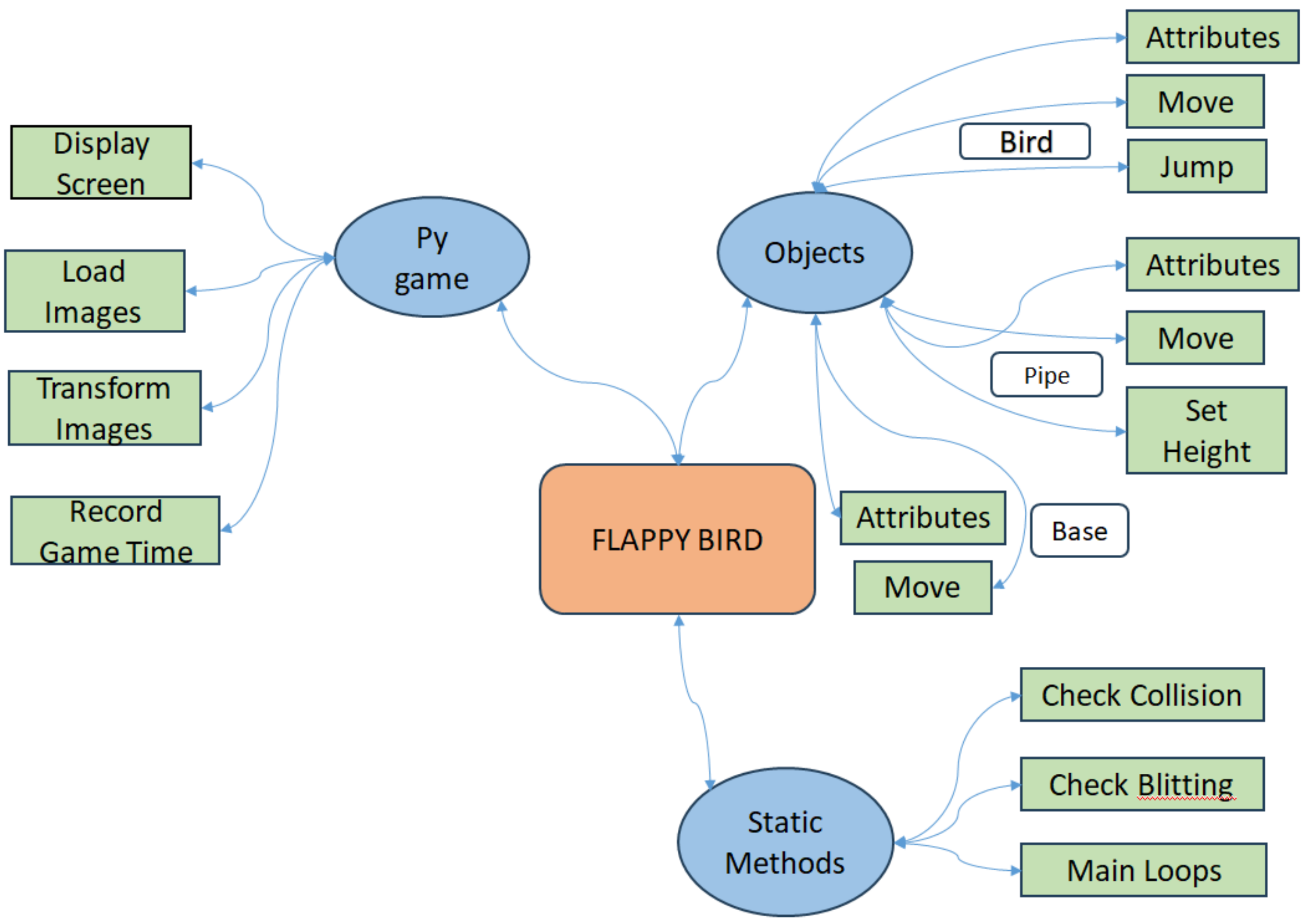
- Variance can be due to the random bird color (red, but also blue or yellow, which could be in some case hardly distinguished from the background). We could preprocess the image in a way that helps the model identify the bird's position to solve this issue.

2.11 Stability:

Evaluating a strategy during training is not easy for reinforcement learning, as policy evolves dynamically and we might not have the possibility to run different policies on the same test games in order to evaluate our model during training. Our evaluation metric is thus the average reward on recent games, which is very noisy as small changes in our model's weights can modify actions in a way that leads our agent to visit unvisited types of states where the model does not perform that well.

For example, a small change in weights could lead the bird to fly too often and visit high-altitude states where the model does not perform that well as it did not visit a lot of them.

2.12 Methodology Through E.R. Diagram:



(Fig 2.3)

2.13 Possible improvements:

The first thing we notice when we test the model is the very high variance between games. Our bird can miss the first obstacle for a play, or it can pass more than 80 obstacles. Our assumption is that this variance is due to 2 elements:

1. There is a high difference between environments, depending on the initialization. These cases might need a different kind of policy: if there is a huge altitude difference, the bird might have to fly two or three consecutive times; if the altitude difference is low, it might not need to fly more than once before the next obstacle. This can be solved by training the agent on more game plays, to enable it to learn situation-specific models.

2. The bird does not anticipate very well schemes that could lead it to fail passing an obstacle that is still far away. We could test this hypothesis by working on the discount factor, setting it at 0.999 or higher in order to build an agent that anticipates better future rewards.

3. We could also work on the reward definition, for example by computing an additional reward every time the bird successfully passes an obstacle. It might imply more work to find the right moment to add this reward, but might give our bird a better understanding of the game structure.

4. As a single wrong action can make you lose, keeping a 0.1 epsilon after the exploration phase can be a bad choice. We could test training the agent with a 0.1 to 0.01 linear annealing epsilon.

To make training faster and more accurate, the next steps would be:

1. To build a hand-made pipeline that deduces relevant features from the image. For example, pre-processed images make it possible to detect quite easily the altitude of a bird, the space between the bird and the next obstacle and the altitude of the space where the bird can pass the obstacle.

2. To find a way to reduce the game observations' dimension. A first idea would be to resize the image, or to resize the image after a pre-processing in order to keep only very relevant feature (in a 40x40 or a 20x20 pixels dimension, for example). The dimension might be well chosen, in order to keep relevant information.

3. Another way to reduce dimension before training would be to generate thousands of random games and to build a Principal Component Analysis space on these images, in a lower dimension that keeps most of the variance. We would then train the model on the observations projected on the PCA space.

4. We might also work on the structure of the network. There might be a lighter network structure that performs equivalent scores after the same training.

2.14 Conclusion/Results Based on Used Parameters:

The Flappy Bird game, developed using Pygame, demonstrates successful implementation of essential gaming elements. The gravity simulation, dynamic pipe generation, and user input handling contribute to an engaging and challenging gameplay experience. The collision detection mechanism ensures that the game responds appropriately to user actions, providing a satisfying and immersive experience.

2.15 Future Scope:

- Enhanced Graphics: Improving the visual aesthetics of the game by incorporating more sophisticated graphics and animations.

- Multiplatform Support: Adapting the game for deployment on various platforms, such as mobile devices or web browsers.

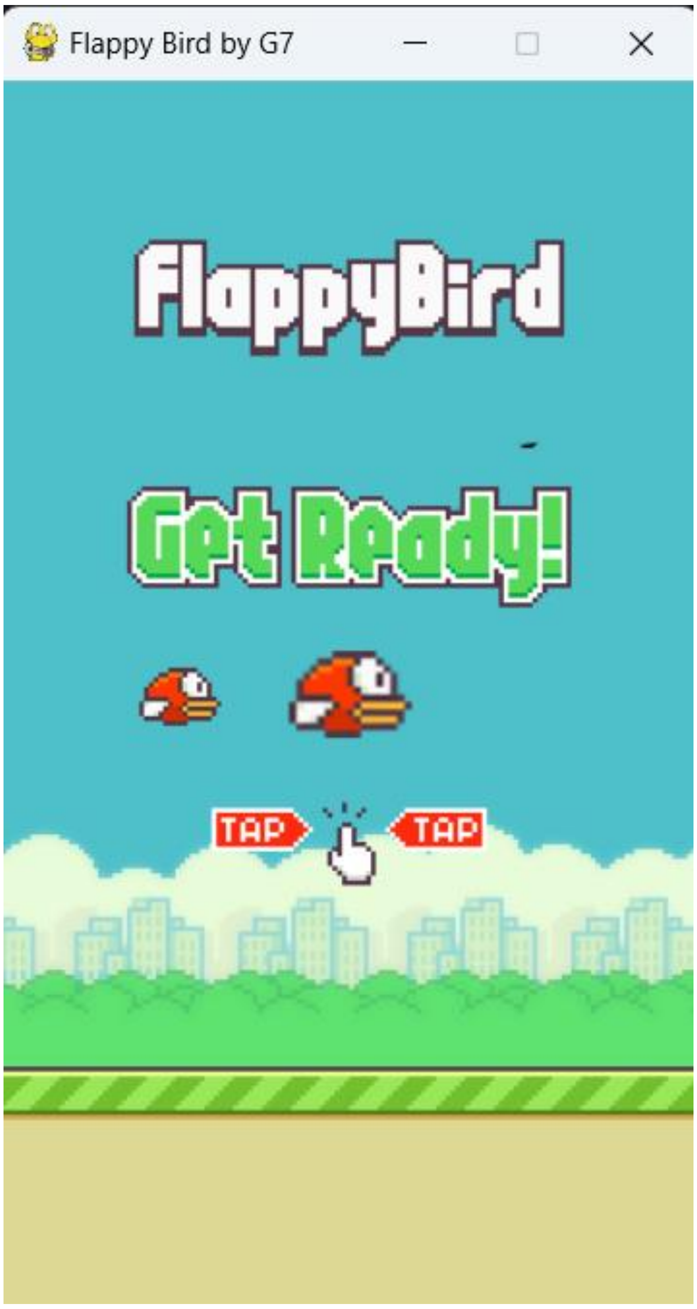
- Additional Game Features: Introducing new obstacles, power-ups, or levels to keep the game fresh and exciting for players.

3. Development and Implementation:

The core of the project involved the practical implementation of the Flappy Bird game using the Pygame library in Python. Various components, such as gravity simulation, dynamic pipe generation, and user input handling, were meticulously integrated to create a cohesive and engaging gaming experience.

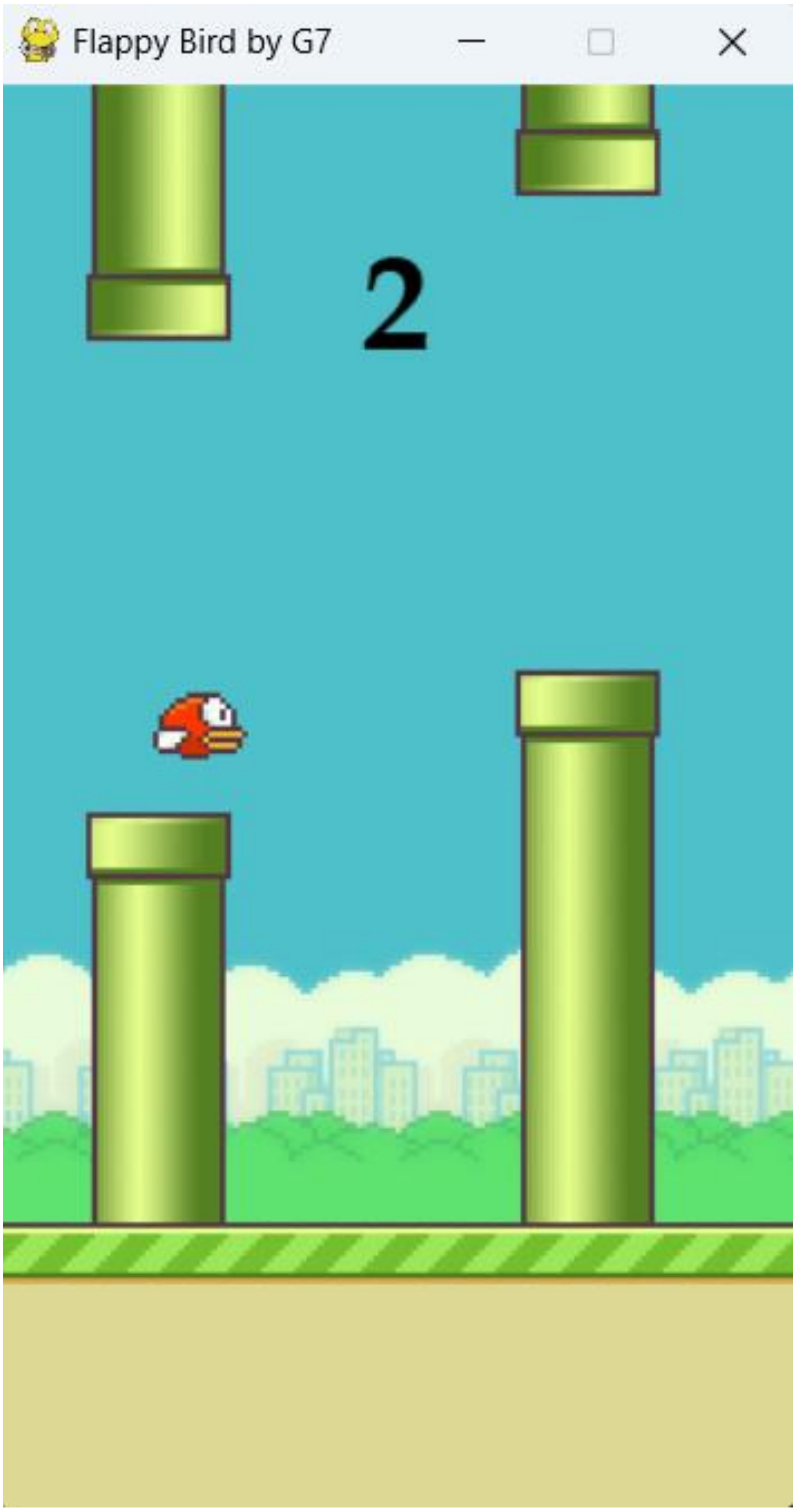
3.1 Screenshots of the program:

Welcome screen:



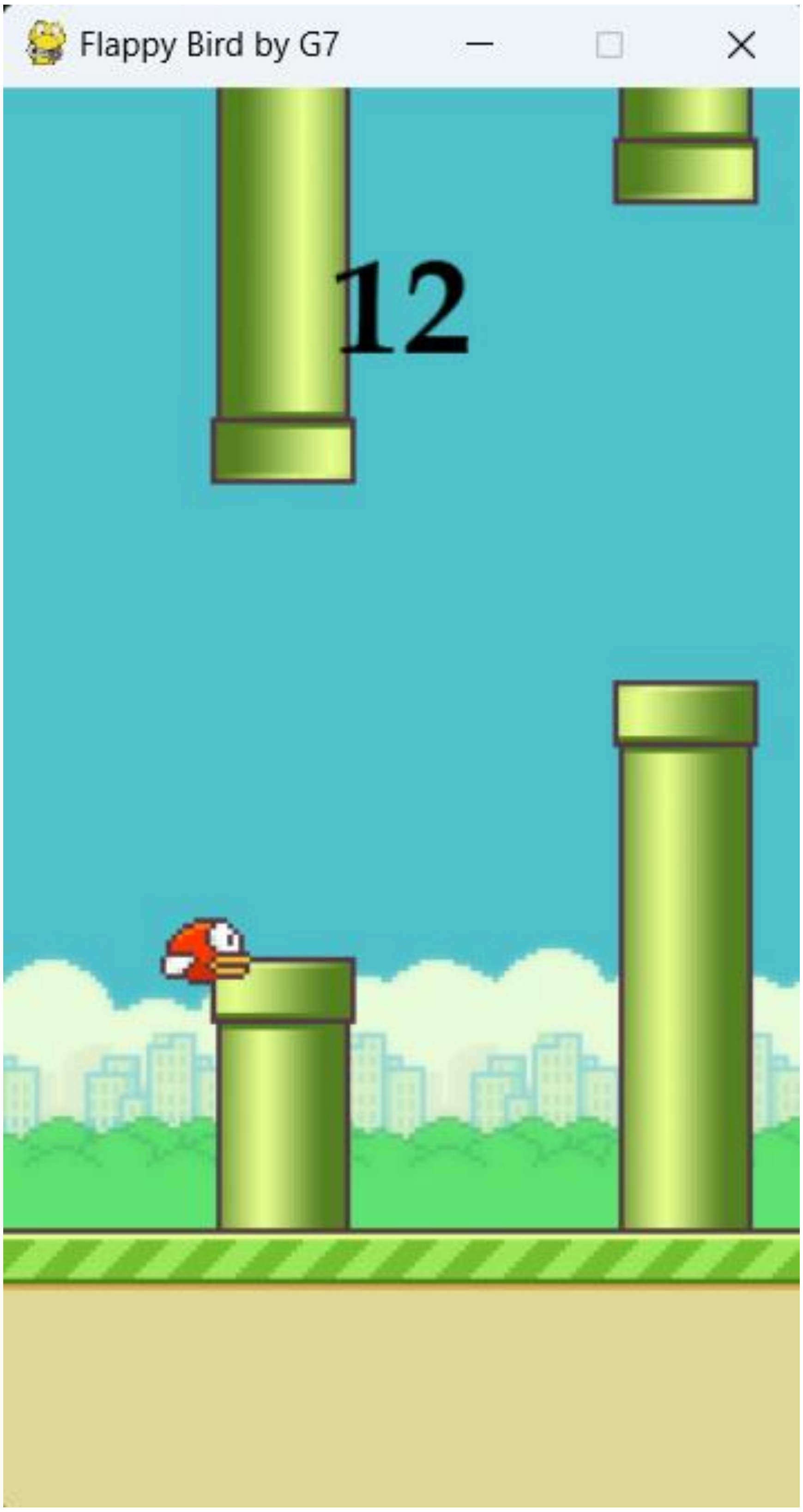
(Fig 3.1)

Main Game Screen:



(Fig 3.2)

Collision Detection:



(Fig 3.3)

4. Results and Discussion:

The project yielded promising results, showcasing a functional and enjoyable rendition of the Flappy Bird game. Results were discussed in the context of user experience, gameplay mechanics, and technical aspects. Notable findings included the successful integration of a scoring system and dynamic difficulty progression.

5. Conclusion and Future Scope:

In conclusion, the Flappy Bird Project demonstrated the successful development of a classic game while laying the groundwork for future enhancements. The game's simplicity and appeal were highlighted, and avenues for future improvements were discussed. The project not only served as an exploration of game development but also as a springboard for potential innovations and expansions. The Flappy Bird game, developed using Python, can be expanded and enhanced in various ways to provide a richer and more engaging experience for players. Here are some potential future scopes and features for the Flappy Bird game:

5.1 Multiplatform Support:

Adapt the game to run on various platforms, including mobile devices (iOS and Android) and web browsers, by leveraging frameworks like Kivy for mobile or Flask/Django for web deployment.

5.2 Multiplayer Mode:

Implement a multiplayer mode where players can compete in real-time or asynchronously. This could involve racing against each other to see who can achieve the highest score within a given time or collaborative modes where players work together to navigate through challenging levels.

5.3 Leaderboards and Achievements:

Integrate a system for global leaderboards to showcase the top scores of players worldwide. Additionally, introduce achievements that players can unlock by reaching specific milestones or accomplishing challenging tasks.

5.4 Enhanced Graphics and Animation:

Upgrade the visual elements of the game by incorporating more advanced graphics, animations, and visual effects. This could include dynamic backgrounds, improved bird animations, and visually appealing transitions between screens.

5.5 Level Editor:

Create a level editor that allows players to design and share their custom levels. This could lead to a community-driven repository of user-generated content, increasing the game's replayability.

5.6 Power-Ups and Obstacles:

Introduce power-ups that provide temporary advantages or modify gameplay dynamics. Conversely, include additional types of obstacles and challenges to diversify the gameplay and keep players engaged.

5.7 Sound and Music Integration:

Enhance the audio experience by incorporating background music, sound effects, and interactive sound scapes that respond to in-game events. This can contribute significantly to the overall immersive experience.

5.8 In-App Purchases and Monetization:

Implement in-app purchases for cosmetic items, themes, or additional features. Explore different monetization strategies, such as ad integration or premium versions, to generate revenue from the game.

5.9 Virtual Reality (VR) Support:

Explore the possibility of adapting the game for virtual reality platforms, providing a more immersive and interactive experience for players.

5.10 Machine Learning Integration:

Experiment with machine learning algorithms to create adaptive gameplay. The game could learn from the player's behavior and adjust difficulty dynamically to provide a personalized and challenging experience.

5.11 Localization:

Translate the game into multiple languages to reach a broader audience, making it more accessible and enjoyable for players around the world. By incorporating some or all of these features, the Flappy Bird game can evolve into a more sophisticated and feature-rich project, catering to a diverse audience and ensuring long-term engagement.

6. Recommendations:

Recommendations for future work include enhancing graphics, exploring multi-platform support, incorporating additional game features, and integrating online elements such as leaderboards and social media interaction. These suggestions aim to elevate the gaming experience and broaden the project's scope. While I cannot recommend specific Flappy Bird games developed with Python due to the lack of a live browsing capability, I can guide you on how to find quality resources and tutorials for creating a Flappy Bird game in Python using Pygame. Here are some recommendations:

6.1 Official Pygame Documentation:

Start by exploring the official Pygame documentation (https://www.pygame.org/docs/). It provides comprehensive information on Pygame's features, functions, and usage. This documentation is an excellent reference as you work on your game.

6.2 YouTube Tutorials:

YouTube is a rich source of tutorials for game development. Search for "Flappy Bird Python Pygame tutorial" to find step-by-step guides. Channels like "Tech With Tim" often cover Pygame and game development in Python.

6.3 Online Courses:

Platforms like Udemy, Coursera, and edX offer courses on Python game development using Pygame. Look for courses that cover Flappy Bird or similar games. Check reviews and ratings to ensure the course meets your needs.

6.4 GitHub Repositories:

Explore GitHub for open-source Pygame projects, including Flappy Bird implementations. Many developers share their code for educational purposes. Search for repositories using terms like "Flappy Bird Python Pygame" on GitHub.

6.5 Community Forums:

Engage with the Pygame community on forums such as Stack Overflow (https://stackoverflow.com/questions/tagged/pygame), Reddit (https://www.reddit.com/r/pygame/), or the Pygame community forum (https://www.pygame.org/news) to ask questions, seek advice, and share your progress.

6.6 Books:

Consider reading books on game development with Pygame. "Making Games with Python & Pygame" by Al Sweigart is a popular choice for beginners. Books can provide in-depth explanations and examples.

6.7 Code Snippet Websites:

Explore websites like Gist (https://gist.github.com/) for smaller code snippets related to Flappy Bird game mechanics in Pygame. This can be helpful for understanding specific aspects of the game.

7. References:

7.1 Pygame Documentation:

Pygame is a popular library for game development in Python. The official Pygame documentation (https://www.pygame.org/docs/) provides a wealth of information, tutorials, and examples that can help you get started with creating a Flappy Bird game.

7.2 YouTube Tutorials:

Many developers share their knowledge through video tutorials on platforms like YouTube. Search for "Flappy Bird Python Pygame tutorial" to find step-by-step guides and walkthroughs that demonstrate how to build the game from scratch.

7.3 GitHub Repositories:

Explore GitHub for open-source Flappy Bird projects implemented in Python. You can find complete source code, and examining others' code can be a valuable learning experience. Use search terms like "Flappy Bird Python Pygame" on GitHub.

7.4 Online Courses:

Websites like Udemy, Coursera, and edX often offer courses on Python game development using Pygame. Look for courses that specifically cover creating Flappy Bird or similar games.

7.5 Game Development Forums:

Participate in forums and communities dedicated to game development. Platforms like Stack Overflow, Reddit (e.g., r/pygame), and the Pygame community forum can be valuable for getting help, asking questions, and sharing your progress.

7.6 Books on Pygame:

Consider reading books that focus on Pygame and game development in Python. "Making Games with Python & Pygame" by Al Sweigart is a popular choice for beginners.

7.7 Code Repositories on Websites:

Explore code-sharing websites other than GitHub, such as GitLab and Bitbucket, to find Python Flappy Bird implementations. Remember to check the publication or update dates of the resources you find, as technology and libraries can evolve, and newer tutorials may be more aligned with the latest practices and versions of Python and Pygame.

The project drew upon a variety of references, including academic literature, programming documentation, and relevant online resources, ensuring a well-informed and robust development approach.

Following is the list of references we used:-

- **https://pngtree.com/so/1-to-10**
- **https://www.freepik.com/free-photos-vectors/game-background**
- **https://www.zapsplat.com/sound-effect-category/game-sounds/**

- **https://studio.youtube.com/channel/UC-PYa26rN_bbQZgNRmIYqTQ/music**
- **https://www.pngwing.com/en/search?q=birds**
- **https://ieeexplore.ieee.org/document/8924807**
- **https://ieeexplore.ieee.org/document/9763893**
- D. Barber. Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012.
- J. Read. Lecture III - Structured Output Prediction and Search. INF581 Advanced Topics in Artificial Intelligence, 2018.
- Mnih et al., Human-level control through deep reinforcement learning Nature 518, pages 529533 (26 February 2015)
- O. Vinyals et al. StarCraft II: A New Challenge for Reinforcement Learning. https://arxiv.org/abs/1708.04782, 2017.

- Chen et al., DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving http://openaccess.thecvf.com/content iccv 2015/ papers/Chen DeepDriving Learning Affordance ICCV 2015 paper.pdf
- Appiah et al., Playing FlappyBird with Deep Reinforcement Learning http://cs231n.stanford.edu/reports/2016/pdfs/111 Report.pdf