ALGORITHM AND PROGRAMMING
FINAL PROJECT REPORT

Written by:
Catherine Isabelle Ong - 2802501035

I : TABLE OF CONTENTS

II: PROJECT SPECIFICATION

This Flappy Bird project is a simple and interactive game inspired by the popular mobile game with the same name. It is a game made to challenge its players to try and navigate a bird through a row of vertically aligned popes without touching them. This project fully uses Python programming to make the game. These include the game mechanics itself, the physics of the game, the event-driven programming and the graphical rendering.

OBJECTIVES:

The main goal of this project is to create a fun, visually appealing and entertaining game while demonstrating Python programming. This project focuses on developing a responsive control system for the bird so that players can feel as if they are in control of what will happen in the game. This game also focuses on making sure that its obstacles are unpredictable, making sure that every time the player were to start the game, it would give varied gameplay experiences. Implementing realistic game physics is also an important part of the project as it helps to stimulate gravity for the bird making its movements more realistic. This project also implements user feedback by adding a scoring system to keep the players motivated and engaged in the game. The last objective of this project is also to make sure that there are smooth transitions between game states such as starting, restarting and ending the game.

KEY FEATURES:

Key features of the game include:

1. **Player-controlled flying bird**
   The main focus of the game is to navigate this bird through the given obstacles (pipes). The bird's movement is controlled by both user input and physics forces such as gravity.
2. **Random obstacle generation**
   The pipes serve as obstacles for the bird in the game. They appear at regular intervals however the y position of the gap varies with every generation of the pair of pipes. This feature offers the player with varying gameplays every time.
3. **Scoring mechanism**
   The players gain points by successfully navigating the bird through the gaps in between the pipes. Every time the bird successfully passes through the center of a pipe, 1 point is added to the score that is displayed at the top-center part of the screen.
4. **Game states**
   The game is split into different game states such as game start, active gameplay and game-over to enhance the player's experience.
5. **Engaging visuals**
   The game itself includes the usage of vibrant graphics and animations which help to make it more eye-catching for the players.

EDUCATIONAL PURPOSE:
This project serves an educational purpose for programmers as it demonstrates the usage of object-oriented programming and event-driven programming which is frequently used in interactive applications and games. This project can also teach people about designing algorithms for game physics and procedural generation.

On the other hand, this project can also help teach players of the game to stay focused to achieve a certain task. In the case of this game, that certain task is to gain as many points as possible. Players are taught to persevere even if they fail in the game as they are able to restart the game easily by clicking the restart button.

TARGET AUDIENCE:
This project's target audience include people who are learning Python programming and are looking to learn about game development using Python and Pygame. It could also be an entertaining project for young kids as well if they want to play the game, however the programming aspect of the project may be too complex for them to understand.

III: SOLUTION DESIGN

My final project program consists of 4 files and 1 folder containing the images I used for the game itself. It was divided into modular components with each file handling a different functionality. These 4 files include the main.py file, the settings.py file, the bird.py file and the pipe.py file.

1. **MAIN MODULE (main.py)**
   Purpose:
   This module was made first and it serves as the entry point for the game. In order for the game to start running, you will need to run this file.

   Functionalities:
   Pygame is initialized at the start of this file and the other files are imported into it. Then I started to set up the game environment by loading the assets and resizing them to the desired width and height. The main module also contains the button class that helps with starting and restarting the game. This file manages the game loop which updates the state of the game, renders the images and processes user input. The state of the game is updated when it is either running, game over or restarting to make sure that the player's experience is smooth. A scoring system is also implemented in this file by checking when the bird successfully passes through the center of a pair of pipes. When this is done successfully, a point is added into the score. However if the bird hits the pipe, the ground or the top of the screen, this will lead to the bird falling down, dying and the game to be over. This is done by checking if the bird has collided with either one of the pipe images

or if the bird has reached a specific height in the screen. The pair of pipes are also made to regenerate at random heights once every set interval that is specified in the settings.py file. The image of the ground is made to scroll so that an illusion is created that the bird is flying when in fact it just stays in place and jumps up and down avoiding the pipes.

2. **BIRD MODULE (bird.py)**

Purpose:

This bird module contains all the properties and behaviors related to the bird which is the main player of the game. The bird has attributes such as position (x and y coordinates), velocity (jumping movement) and animation frames (creates the illusion that the bird's wings are flapping).

Functionalities:

This module responds to user input by making the bird fly upwards when the player presses the spacebar. Within the bird class, gravity is implemented to constantly add a downward force on the bird's velocity making it fall down if the space bar isn't pressed. However this is not triggered until the game is started by pressing the spacebar. This module also deals with the animation of the bird which is composed of three sprite images. The difference between each image is the position of the bird's wing. So as the animation iterates through these three images, it creates an illusion that the bird's wings are flapping up and down. This overall helps improve the visual experience of the player.

3. **PIPE MODULE (pipe.py)**

Purpose:

This pipe module is responsible for creating, managing and rendering the pipes on screen so that they can act as the obstacles in the game for the bird to avoid in order to continue the game and gain points. The pipes each have their own position, size and gap between them for the bird to pass through.

Functionalities:

This module generates pairs of pipes (a bottom pipe and a top pipe) at regular intervals which ensures a consistent challenge for the player. The y position of the pair of pipes are randomly generated between a set range to make sure that the gap is placed at different heights everytime a new set of pipes is generated. The pipes are also moved along its x position at a constant speed which also further helps in creating the illusion that the bird is the one flying forward when in fact it is just the pipes and the ground scrolling (the ground scrolling is controlled in the main.py file). When the pair of pipes reach off-screen, they are removed to optimize memory usage.

4. **SETTINGS MODULE (settings.py)**

   Purpose:

   This settings module centralizes all configurable variables making a single point of control for the game's settings. These variables include screen dimensions, frame rate, gravity strength, pipe spacing, bird properties, etc.

   Functionalities:

   This module allows for easy adjustments to the game parameters for testing and fine tuning purposes. It makes experimentation with difficulty levels much easier as variables can be found more quickly and therefore modifying the values can be done faster too.

GRAPHICS AND ASSETS (images folder):

This flappy bird remake project uses pre-designed image assets for the bird, pipes, background, ground and the other elements such as the title image, the game over image and the start/restart buttons. There are three different bird images which serve as animation frames for the bird to create a smooth wing flapping animation. The pipes and the ground elements are also made to scroll but at different speeds. The ground is made up of one image that is then placed one after the other as it scrolls to create an illusion that it is an endless image. However, the pair of pipes are generated once every set interval.

IV: WHAT WAS IMPLEMENTED & HOW IT WORKS

ALGORITHMS:

1. **Bird movement:**

   The movement of the bird is done by implementing a physics algorithm. The gravity adds a constant downwards velocity on the bird so that it will fall when the spacebar isn't pressed. However when the spacebar is pressed, it applies an upwards force on the bird which counteracts the constant gravity that was applied to the bird in the first place. Once these user inputs are done, the position of the bird is updated based on its velocity. A positive velocity means that the bird is moving down and a negative velocity means that the bird is moving up.

2. **Generate pipes:**

   A pair of pipes are generated once every set interval which is 1200 milliseconds. This is done using a timer mechanism. The vertical position of the gaps between the pipes are randomized but the size of the gap is maintained to keep the game consistent and to make sure that the gaps are always big enough for the bird to pass through. This is done by using the random function to pick a random height between a determined range for the height of the pipes. The pipes move at a constant speed horizontally to create an illusion that the bird is flying forwards. This is done by subtracting the position of the pipes by the scroll speed that was determined in the settings.py file.

3. **Collision detection:**
   This is done by using bounding boxes to detect if there is an overlap between the bird and the pipes. If the rectangular bounds of the bird overlap with the rectangular bounds of pipe or the ground/top of the screen, a collision will be detected and the game will end.

4. **Scoring system:**
   The scoring system implemented here adds the score whenever the center of the bird successfully passes through the center x position of the gap between the pipes. This score is then updated in the settings.py file and displayed onto the screen.

<u>SOLUTION SCHEME:</u>

1. **Object-oriented programming:**
   Each major component is encapsulated in a class (Bird, Pipe and Button class) which simplifies the code by reducing code duplications. The bird and pipe classes are also separated into its own files to further organize the code so that it is easily found as compared to if it was placed in the same file as the main.py file. However, the button class was kept in the main.py file as it was easier to be accessed this way.

2. **Event-handling:**
   This keeps an eye for user input such as key presses and translates them into actions in the game. Examples of these include the pressing of the spacebar which translates into the bird flying upwards, the hovering of the cursor on the start/restart buttons which translates into the button increasing in size and the click of the start/restart buttons which brings the player to a different game state which in this case is the actual game itself.

3. **Game loop:**
   This makes sure that the game has consistent timing. It also makes sure that the game updates, renders and processes events at the same time.

<u>DATA STRUCTURES:</u>

1. **Classes:**
   The classes I made in this program include the Bird class, Pipe class and the Button class. The Bird and Pipe class have their own files while the Button class is located inside the main.py file. These classes contain properties such as velocity and position and behaviors such as movement and collision detection for the objects in the game.
2. **Lists:**
   These lists are used to manage the pipes that are currently on screen and active. For every new pipe that appears on screen, it is appended onto the list and once it is off the screen it is removed to conserve memory.

GRAPHICS RENDERING:

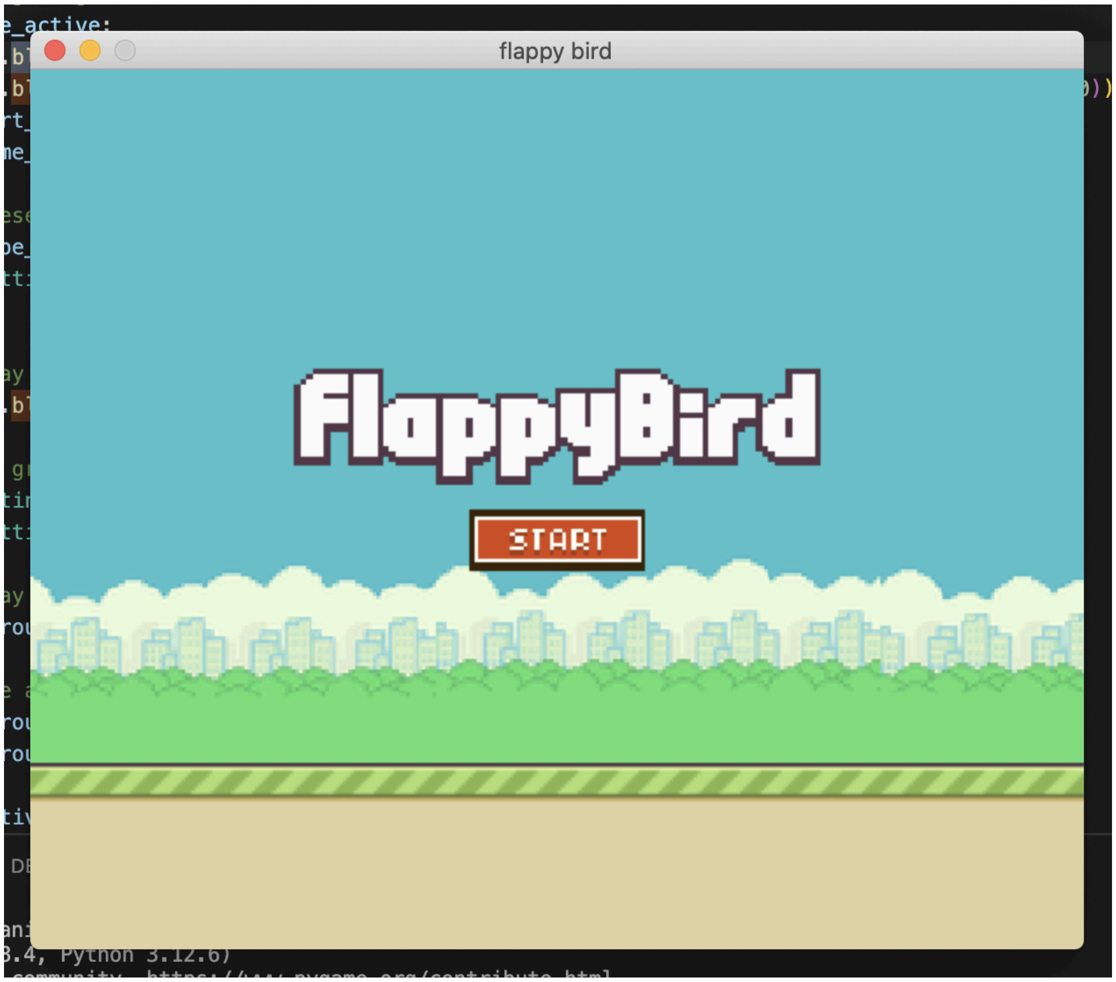
To render the images in this program, I used the blit function in Pygame. First the image is loaded and then it is placed onto a specific coordinate on the screen using the blit function. The images are layered to make sure that when it renders, the bird is in front of the pipes so that if it hits the pipe midway through, the bird will be visible as it falls and hits the ground.

PERFORMANCE OPTIMIZATION:

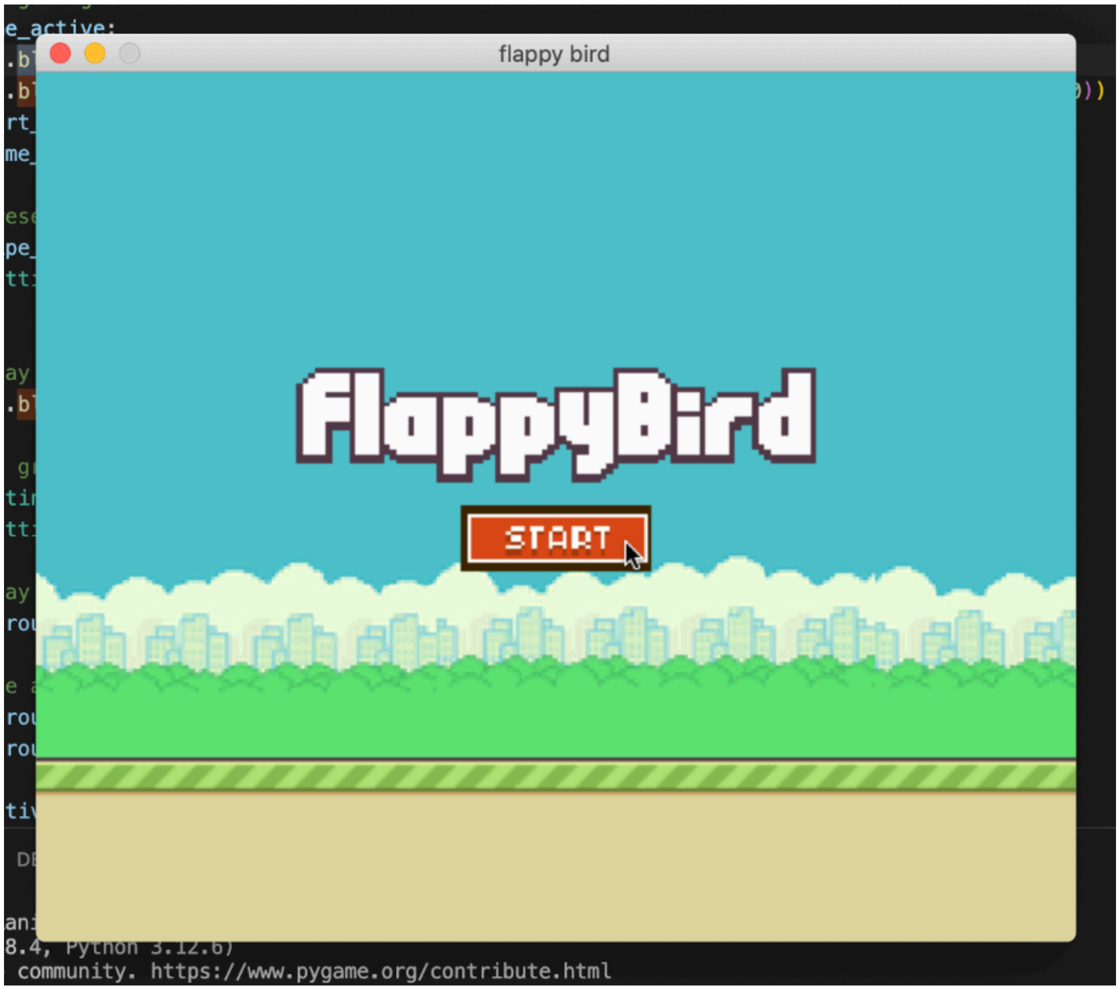
The pipes are managed so that for every pipe that goes on screen, it is appended onto a list and once the pipe reaches off the screen, it is removed from the list. This deletion process saves memory as it removes parts of the game that are no longer necessary.

V: EVIDENCE OF WORKING PROGRAM

1. Image of the homepage of the game. This appears when you first run the main.py file. It consists of the game title, background and ground images and the start button which will lead you to the main game itself.

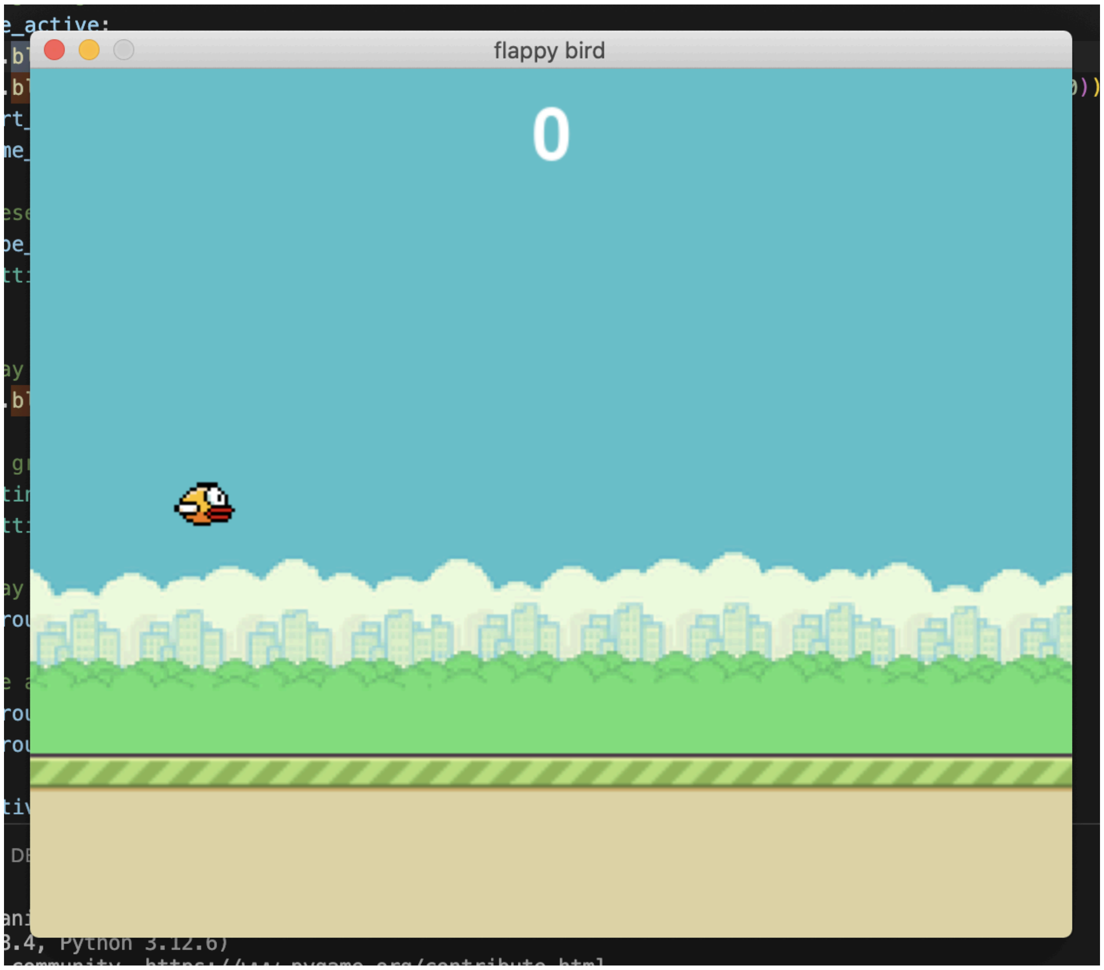


2. This is an image of the start button when the mouse is hovering over it, if you compare it with the previous image of the home page, the size of the start button is smaller in that photo compared to this. This is because I made sure that the button would increase in size whenever the cursor was hovering on it.

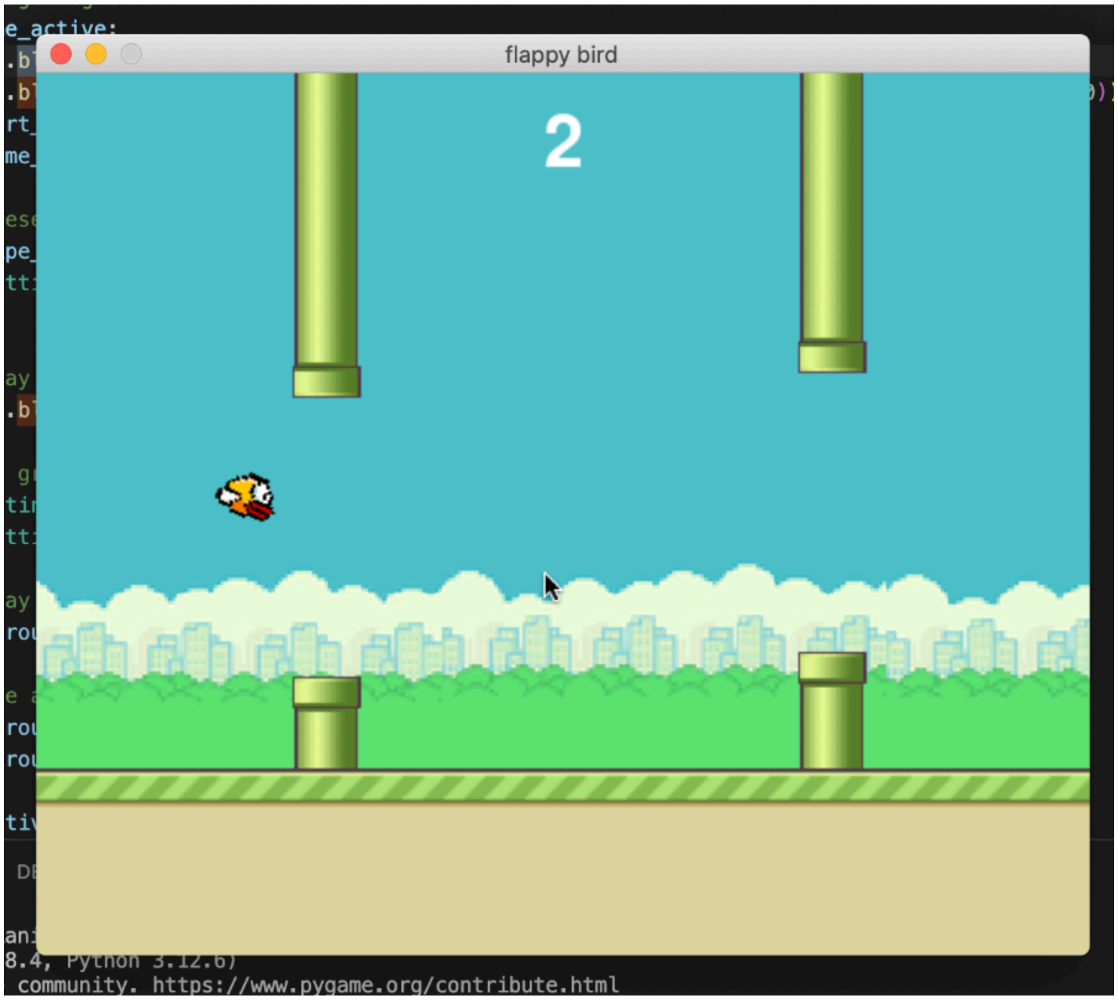


```
8.4, Python 3.12.6)
 community. https://www.pygame.org/contribute.html
```

3. This image shows the main game itself once the start button has been clicked. It shows the bird, the background and the ground without any pipes yet. In order for the pipes to start generating, the player will need to click the spacebar to start flying the bird.

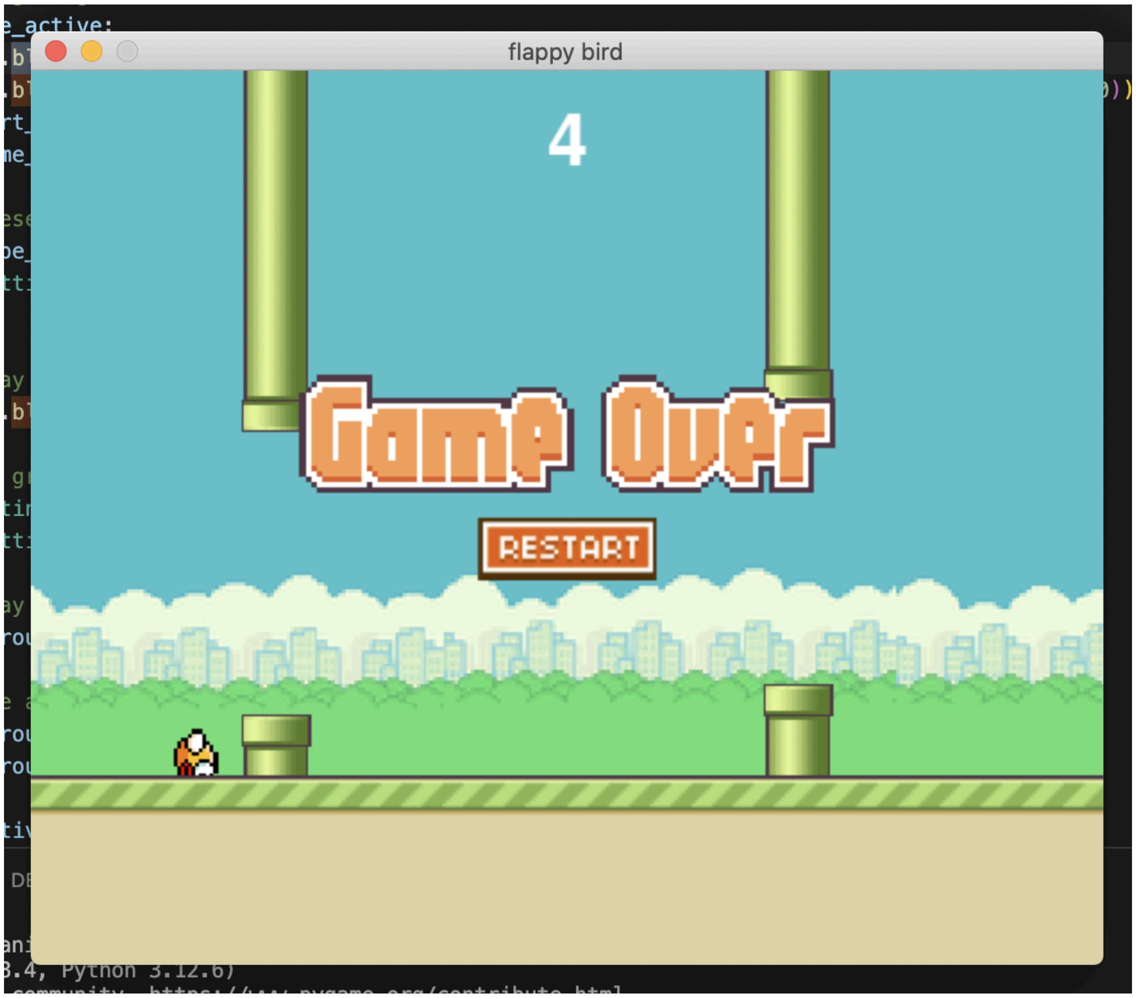


4. This image shows the game in action. This shows the bird flying and trying to go through the gap between the pipes to gain points.

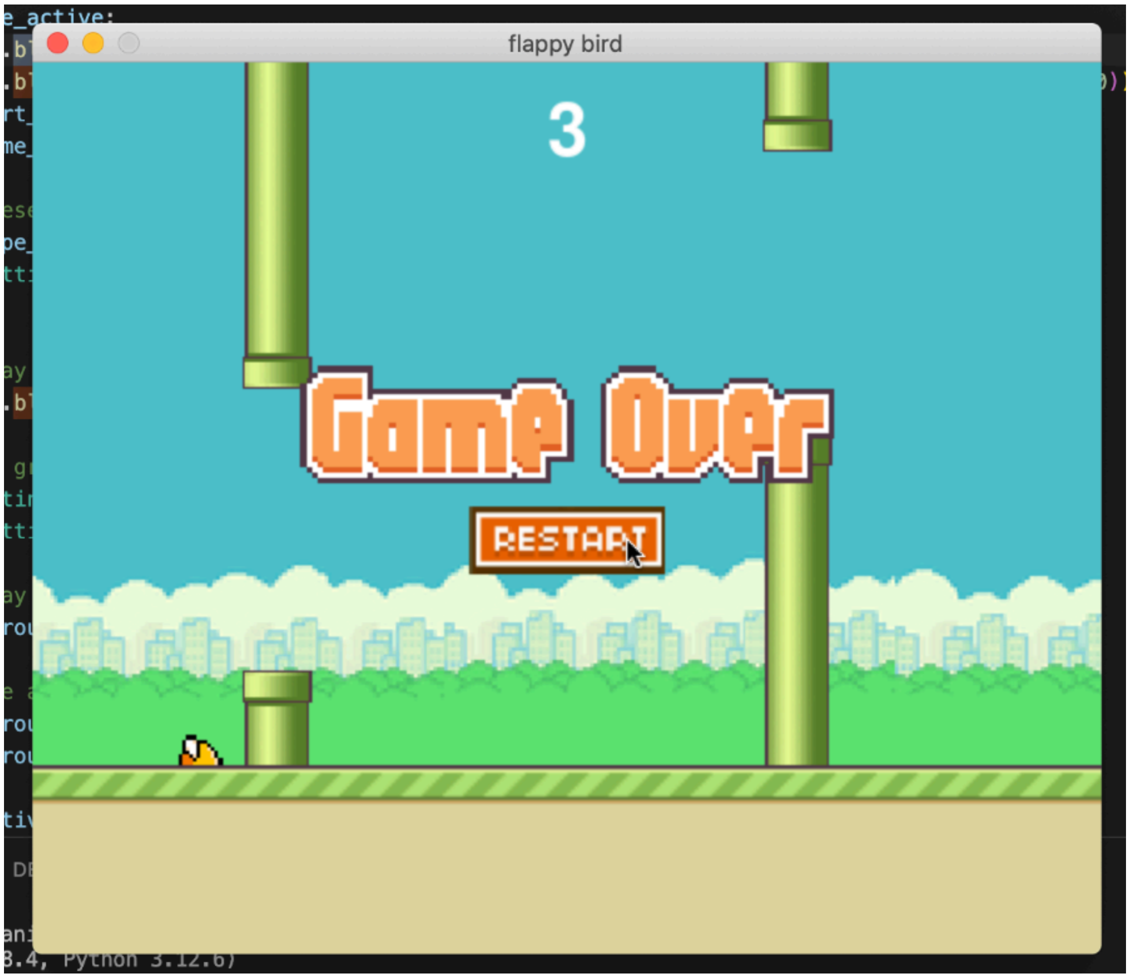


```
e_active:
8.4, Python 3.12.6)
 community. https://www.pygame.org/contribute.html
```

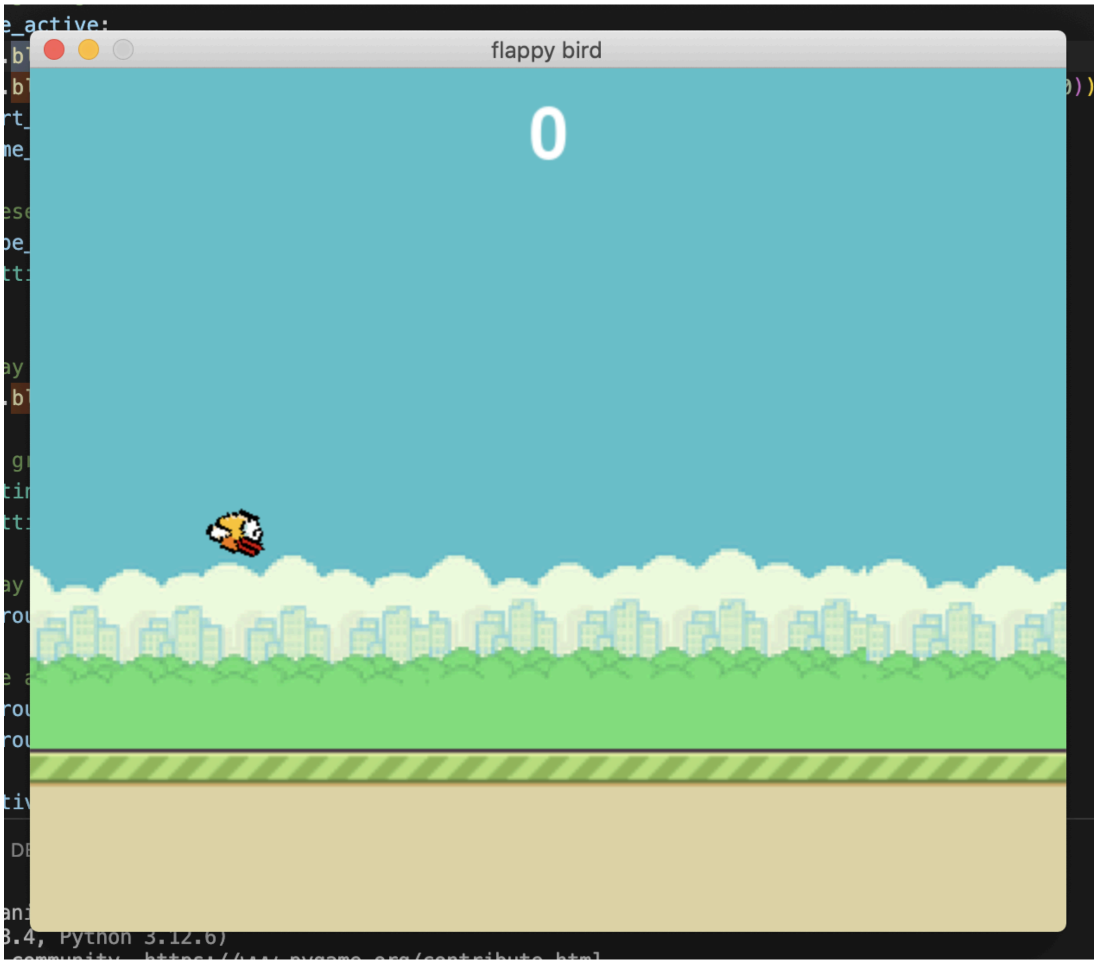
5. This image shows the game over page once the bird has hit either the ground, pipe or the top of the screen. In this case the bird has just hit a pipe. This page shows the game over text and the restart button over the main game page keeping all of the main game page attributes as it was after the bird falls to the ground and dies.



6. This is an image of the restart button when the mouse is hovering over it. If you compare it with the previous page when the cursor wasn't hovering on the button yet, the size of the button increased as I made it so that when you hover on top of the button, the button increases in size. This is the same feature as the start button in the homepage.



7. This is an image of the page that it brings you to after you click the restart button. It is very similar to the page that it brings you to when you first start the game but the only difference is that the bird is slightly tilted downwards.


flappy bird
0