



VIT<sup>®</sup>  
BHOPAL



## Flappy Bird Project Report (Manny Bird)

**Title:** Manny Bird – A Flappy Bird Clone

**Author:** Mann Sharma

**Registration no:** 25BAI10379

**Course:** [CSE]

**Date:** [21/11/25]

## **1. introduction**

this project recreates Flappy Bird in Python using Pygame. aim was to learn game loops, collision detection, and event handling.

## **2. problem statement**

make a simple bird game where player avoids pipes, scores points, and game ends on collision.

## **3. functional requirements**

- bird moves with keyboard
- pipes spawn with random gaps
- collision detection
- score tracking
- reset after game over

## **4. non-functional requirements**

- **smooth fps (~60)**
- **responsive controls**
- **simple code structure**
- **lightweight assets**

## **5. System architecture**

- **main loop: events + rendering**
- **bird class: position + image**
- **pipe class: position + scoring**
- **event system: key presses + timers**

## **6. design decisions**

- scaled images manually
- pipe timer with pygame.USEREVENT
- score increments by 0.5 per pipe and there are two pipes per
- used random package for pipe spawn randomly

## **7. implementation details**

- window: 360x640

- gravity: 0.4, jump: -5
  - pipe spawn: every 1.5s

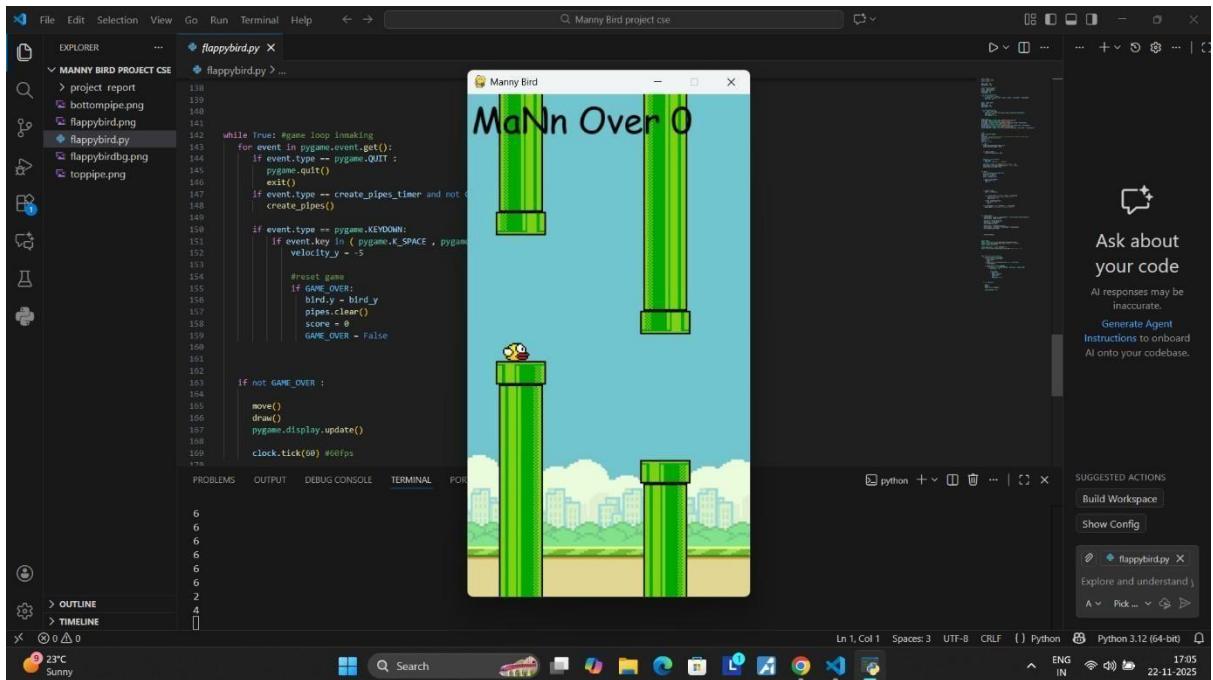
## 9. screenshots & results

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows the project structure under "MANNY BIRD PROJECT CSE".
- Editor:** The file "flappybird.py" is open, displaying Python code for a Flappy Bird game.
- Preview:** A central window titled "Manny Bird" shows a screenshot of the game. It features a yellow bird in the center, flying between two green pipes. The background is blue with white clouds, and there are green platforms at the bottom.
- Bottom Right:** A floating AI interface from GitHub Copilot is visible, with the text "Ask about your code".
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and a SUGGESTED ACTIONS section.
- Bottom Status Bar:** Shows file information (Ln 1, Col 1), spaces used (Spaces: 3), and the Python version (Python 3.12 (64-bit)).

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The title bar reads "Manly Bird project cse". The left sidebar displays a file tree for "MANNY BIRD PROJECT CSE" containing files like "flappybird.py", "project report", "bottompipe.png", "flappybird.png", "flappybirdbg.png", and "toppipe.png". The main editor area shows the content of "flappybird.py". The code defines a "Bird" class and a "Pipe" class, both derived from Pygame's Rect class. It also includes a "game\_images" section. Below the editor are tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS". The terminal tab shows the output: "Hello from the pygame community. https://www.pygame.org/contribute.html". The status bar at the bottom indicates "PS C:\python\Manly Bird project cse" and "Python 3.12 (64-bit)". On the right side, there are several floating panes: "SUGGESTED ACTIONS" with "Build Workspace" and "Show Config"; a "Search" pane; and a "CodeLens" pane titled "flappybird.py" with the message "Explore and understand". A sidebar on the right says "Ask about your code" with links to "Inaccurate", "Generate Agent", and "Instructions to onboard". The bottom right corner shows system icons for battery, signal, and time.

```
flappybird.py
1 import pygame
2 from pygame import exit
3 import random
4
5 game_width = 360
6 game_height = 640
7
8 # setting manlys bird y0
9 bird_x = game_width/10
10 bird_y = game_height/2
11 bird_width = 34
12 bird_height = 24
13
14 class Bird(pygame.Rect):
15     def __init__(self, img):
16         pygame.Rect.__init__(self, bird_x, bird_y, bird_width, bird_height)
17         self.img = img
18
19
20 pipe_x = game_width
21 pipe_y = 0
22 pipe_width = 64
23 pipe_height = 512
24
25
26 class Pipe(pygame.Rect):
27     def __init__(self, img):
28         pygame.Rect.__init__(self, pipe_x, pipe_y, pipe_width, pipe_height)
29         self.img = img
30         self.passed = False
31
32
33 #game images
```



## 10. testing approach

Manually played it for some time tested every possible way and yeah initially there were some bugs like the score wasn't increasing ,there was a time when it couldn't detect if the bird touched the ground or not but used a bit of youtube and google and fixed it .

## 11. challenges faced

Looping was very hard and settings pole were very hard to used a bit of youtube . It was quite challenging it took me like 2days to understand hpw pygame works but yeah with the help of yt and google I managed to get it done . Not gonna lie I used yt but I wrote every line of code by myself

## **12. learnings**

Learned how newtons is applicable in coding lol like gravity. Also learned about how to use pygame , how to use random function ,how to use include external png's , how to resize them etc etc.

## **13. future enhancements**

add sounds, menu, high scores, bird animation, mobile controls.This was a basic game in future I might make it 3d using unreal engines maybe

## **14. references**

Youtube tutorials and Github examples.