HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG CƠ SỞ TẠI TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN II

BÁO CÁO CUỐI KỲ
MÔN: LẬP TRÌNH ỨNG DỤNG ĐA PHƯƠNG TIỆN

Đề tài:
Xây dựng Game FlappyBird bằng thư viện Pygame

Lớp: D22CQPTUD01-N ***Nhóm 5***

Họ tên sinh viên thực hiện: **1. Trần Nguyễn Bá Huy - N22DCPT037**
2. Nguyễn Đăng Khoa - N22DCPT045

GVHD: cô Phan Thị Thể

TP.HCM, 2025

MỤC LỤC

LỜI CẢM ƠN

Chúng em xin được bày tỏ lòng biết ơn sâu sắc và chân thành nhất tới cô Phan Thị Thể, giảng viên bộ môn Lập Trình Ứng Dụng Đa Phương Tiện. Trong suốt quá trình thực hiện đồ án "Flappy Bird", cô không chỉ là người hướng dẫn mà còn là nguồn cảm hứng lớn lao, giúp chúng em vượt qua mọi thử thách và hoàn thành dự án một cách tốt nhất.

Những buổi học trên lớp và đặc biệt là những buổi hướng dẫn ngoài giờ của cô đều vô cùng giá trị. Cô đã không ngừng cung cấp cho chúng em những kiến thức chuyên môn vững chắc, những phương pháp tiếp cận vấn đề hiệu quả và những lời khuyên thực tế, giúp chúng em áp dụng lý thuyết vào việc phát triển một ứng dụng đa phương tiện hoàn chỉnh. Sự nhiệt huyết, tận tâm trong từng lời giảng và những góp ý chi tiết, kịp thời của cô đã là động lực to lớn, thúc đẩy chúng em không ngừng học hỏi, tìm tòi và hoàn thiện sản phẩm của mình.

Đồ án này không chỉ là một bài tập học thuật mà còn là một hành trình trải nghiệm quý báu, nơi chúng em được thực hành quy trình phát triển phần mềm, rèn luyện kỹ năng làm việc nhóm, và đối mặt với các vấn đề thực tế trong lập trình game. Tất cả những điều này sẽ là hành trang vô cùng quan trọng cho con đường học tập và sự nghiệp tương lai của chúng em.

Chúng em xin trân trọng cảm ơn cô Phan Thị Thể đã dành thời gian, công sức và tâm huyết để dìu dắt chúng em. Kính chúc cô luôn dồi dào sức khỏe, tràn đầy năng lượng, hạnh phúc và gặt hái được nhiều thành công hơn nữa trong sự nghiệp giáo dục cao quý của mình.

CHƯƠNG I: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Mục tiêu đề tài:

Trong khuôn khổ môn học LTUD DPT, nhóm chúng em đã lựa chọn một đề tài vừa quen thuộc vừa đầy thử thách: *xây dựng lại tựa game Flappy Bird huyền thoại sử dụng thư viện Pygame trong ngôn ngữ lập trình Python*. Đây không chỉ là một dự án học thuật đơn thuần để hoàn thành yêu cầu môn học, mà còn là một hành trình thú vị để biến những kiến thức lý thuyết đã học thành một sản phẩm thực tế, có thể nhìn thấy, tương tác và mang lại trải nghiệm trực quan.

Flappy Bird, với lối chơi tưởng chừng đơn giản – điều khiển một chú chim bay qua các khoảng trống giữa những cặp ống – nhưng lại có sức gây nghiện đáng kinh ngạc. Sự thành công của nó nằm ở cơ chế gameplay lặp đi lặp lại, nhịp độ nhanh và độ khó tăng dần, tạo ra một thách thức liên tục cho người chơi. Việc tái tạo một trò chơi như vậy không chỉ dừng lại ở việc sao chép hình thức, mà còn đòi hỏi sự hiểu biết sâu sắc về các nguyên tắc cốt lõi của lập trình game: từ logic điều khiển nhân vật, xử lý đồ họa, tích hợp âm thanh, cho đến các thuật toán phát hiện va chạm và quản lý trạng thái trò chơi.

Dự án này là cơ hội lý tưởng để chúng em áp dụng trực tiếp kiến thức lập trình Python vào một môi trường thực tế và sinh động. Chúng em sẽ được thử thách trong việc tổ chức cấu trúc code một cách rõ ràng và hiệu quả, sử dụng các nguyên lý lập trình hướng đối tượng để xây dựng các thực thể game như chim, ống, môi trường, hay hệ thống tính điểm. Thay vì chỉ làm việc với những đoạn mã khô khan, chúng em sẽ được chứng kiến từng dòng lệnh biến thành chuyển động mượt mà, hiệu ứng âm thanh sống động và tương tác tức thì trên màn hình. Điều này không chỉ củng cố nền tảng lập trình mà còn khơi gợi niềm đam mê sáng tạo mãnh liệt.

Đặc biệt, trọng tâm của đồ án này là hiểu rõ được thư viện Pygame. Chúng em sẽ đi sâu vào việc tìm hiểu cách Pygame quản lý màn hình hiển thị, tải và thao tác với các hình ảnh (sprites) để tạo ra các nhân vật và vật thể trong game. Các kỹ năng quan trọng như xử lý sự kiện (nhận biết cú nhấp chuột hoặc nhấn phím để chim bay), thiết lập và duy trì vòng lặp game ổn định ở một tốc độ khung hình (FPS) nhất định, cũng như tính toán và phát hiện va chạm một cách chính xác giữa các đối tượng sẽ được rèn luyện tỉ mỉ. Hơn nữa, việc tích hợp các hiệu ứng âm thanh cho mỗi cú nhảy, tiếng va chạm, hay âm báo khi ghi điểm sẽ giúp trò chơi trở nên sống động và hấp dẫn hơn rất nhiều.

Với tất cả những yếu tố trên, đề tài không chỉ là một nhiệm vụ học thuật mà còn là một trải nghiệm học tập toàn diện, trang bị cho chúng em những kỹ năng và kinh nghiệm thực tế, làm nền tảng vững chắc cho con đường phát triển sự nghiệp trong lĩnh vực lập trình ứng dụng và phát triển game sau này.

1.2. Mục tiêu đề tài:

Khi bắt tay vào thực hiện đề tài, nhóm đã đặt ra được những mục tiêu như sau:

- Áp dụng và Củng cố Kiến thức Lập trình Python: Dự án là môi trường lý tưởng để chúng em thực hành các khái niệm cơ bản và nâng cao của Python như cấu trúc dữ liệu, thuật toán, lập trình hướng đối tượng (OOP) thông qua việc tổ chức code thành các lớp (class) như chim, ống cống, sàn nhà, v.v.
- Làm chủ Thư viện Pygame: Đây là trọng tâm chính của đề tài. Chúng em sẽ đi sâu vào việc tìm hiểu và ứng dụng Pygame để:
- Xử lý đồ họa: Tải và hiển thị hình ảnh (sprites), điều khiển vị trí, kích thước, và các hiệu ứng chuyển động của nhân vật, ống cống, nền trời.
    + Quản lý sự kiện: Bắt các sự kiện từ bàn phím (nhấn phím cách để chim bay), chuột, hoặc các sự kiện thời gian trong game.

+ Phát hiện va chạm: Đây là một phần quan trọng, đòi hỏi chúng em phải lập trình để xác định chính xác khi nào chim va vào ống, mặt đất, hoặc vượt qua giữa các ống để tính điểm.
+ Tích hợp âm thanh: Thêm nhạc nền, hiệu ứng âm thanh khi chim bay, va chạm, hoặc ghi điểm, nhằm tăng tính chân thực và hấp dẫn cho trò chơi.
+ Quản lý vòng lặp game và FPS (Frames Per Second): Đảm bảo trò chơi chạy mượt mà, ổn định trên các hệ thống khác nhau.
+ Phát triển Tư duy Logic và Giải quyết Vấn đề: Mỗi khía cạnh của trò chơi, từ việc tạo chuyển động rơi tự do và bay lên của chim, cách thức tạo và di chuyển các cặp ống ngẫu nhiên, cho đến việc tính điểm và xử lý trạng thái thua cuộc, đều là những bài toán yêu cầu phân tích, thiết kế thuật toán và tìm ra giải pháp tối ưu. Đây là quá trình rèn luyện khả năng tư duy phản biện và sáng tạo không ngừng.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu ngôn ngữ lập trình Python:

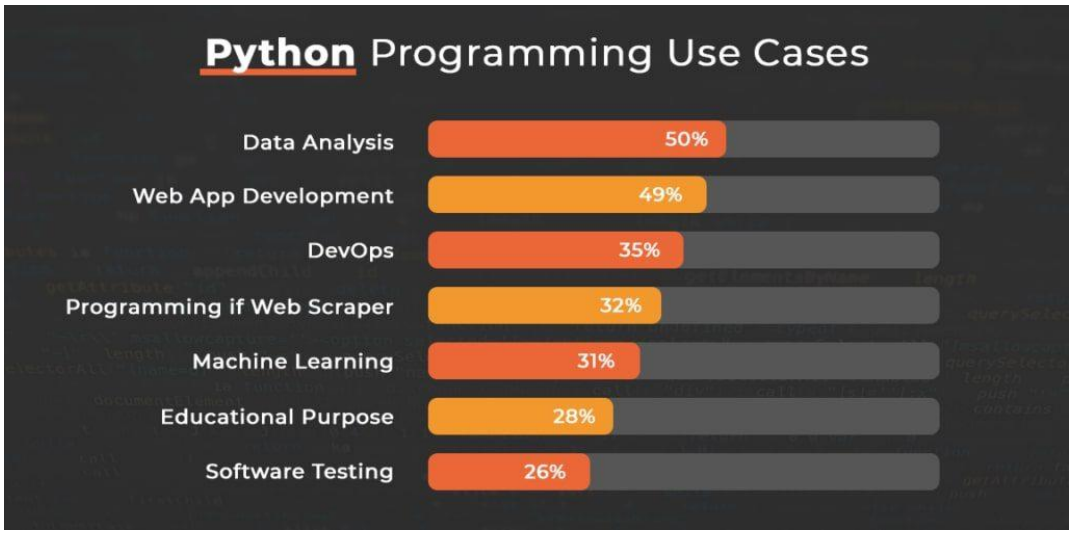
Python ***được thiết kế với tư tưởng giúp người học dễ đọc, dễ hiểu và dễ nhớ***; vì thế ngôn ngữ Python có hình thức rất clear, cấu trúc rõ ràng, thuận tiện cho người mới học. Cấu trúc của Python cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu, nói cách khác thì so với các ngôn ngữ lập trình khác, chúng ta có thể sử dụng ít dòng code hơn để viết ra một chương trình trong Python.

Python sử dụng một trình thông dịch để chạy mã. Khi bạn viết mã Python, bạn không cần phải biên dịch nó thành mã máy trước khi chạy. Thay vào đó, trình thông dịch sẽ đọc và thực thi mã của bạn trực tiếp, từng dòng một

Python là một ngôn ngữ lập trình đa mẫu hình, nó hỗ trợ hoàn toàn mẫu lập trình hướng đối tượng và lập trình cấu trúc; ngoài ra về mặt tính năng, Python cũng hỗ trợ lập trình hàm và lập trình hướng khía cạnh. Nhờ vậy mà Python có thể làm được rất nhiều thứ, sử dụng trong nhiều lĩnh vực khác nhau.

Python có sẵn các cấu trúc dữ liệu mạnh mẽ như list, dictionary, tuple, giúp bạn dễ dàng xử lý và lưu trữ dữ liệu. Ngoài ra bạn không cần phải khai báo kiểu dữ liệu cho các biến, Python sẽ tự động xác định kiểu dữ liệu dựa trên giá trị của biến.

Trong dự án này, Python được sử dụng làm ngôn ngữ chính để xây dựng trò chơi Flappy Bird thông qua thư viện Pygame. Với tính linh hoạt và cú pháp đơn giản, Python giúp nhóm dễ dàng tiếp cận, phát triển và triển khai trò chơi một cách hiệu quả.



Các lĩnh vực CNTT ứng dụng Python phổ biến nhất hiện nay

2.2. Tổng quan về thư viện Pygame:

Pygame là một thư viện của ngôn ngữ lập trình Python và là một tập hợp các mô-đun Python được thiết kế riêng để lập trình trò chơi. Pygame được viết bởi Pete Shinners thay thế cho chương trình PySDL sau khi quá trình phát triển dự án này bị đình trệ. Chính thức phát hành từ năm 2000, Pygame được phát hành theo phần mềm miễn phí GNU Lesser General Public License.

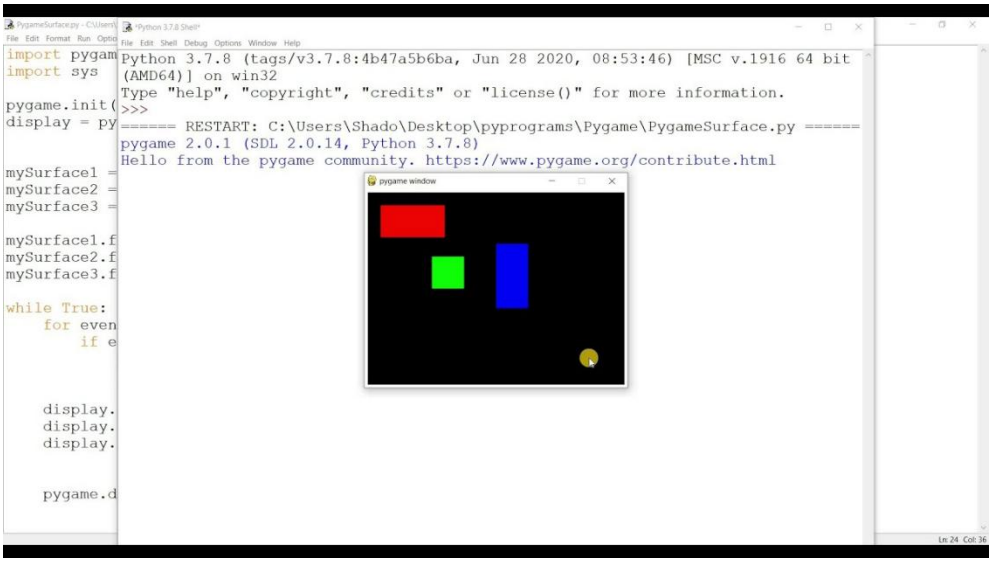
Pygame có thể chạy trên nhiều nền tảng và hệ điều hành khác nhau. Với thư viện pygame trong Python, các nhà phát triển có thể sử dụng công cụ và chức năng mở rộng để tạo ra các trò chơi nhập vai ấn tượng. Bởi vậy, Pygame đang ngày càng phổ biến với nhà phát triển vì tính đơn giản, linh hoạt, dễ sử dụng.

Dưới đây là một số đặc điểm nổi bật của Pygame:

- Pygame sử dụng Simple DirectMedia Layer (SDL), một thư viện phát triển đa nền tảng cho phép các nhà phát triển có thể truy cập vào phần cứng máy tính như đồ họa, âm thanh và thiết bị đầu vào.
- Xây dựng các trò chơi trên nhiều nền tảng khác nhau như Windows, Mac, Linux thậm chí là cả các thiết bị di động.
- Nhà phát triển có thể quản lý tất cả các yếu tố trong quá trình phát triển trò chơi. Đó có thể là các chức năng như xuất đồ họa, xử lý sự kiện, hoạt ảnh, hiệu ứng âm thanh và phát lại nhạc.
- Cung cấp nhiều chức năng mở rộng hỗ trợ nhà phát triển tập trung phát triển trò chơi.
- API trực quan và dễ hiểu, hỗ trợ người mới sử dụng hay cả những nhà phát triển có kinh nghiệm đều có thể truy cập được.
- Nguồn tài nguyên và tài liệu phong phú, các nhà phát triển có thể sử dụng các mã nguồn mở miễn phí để phát triển dự án của mình.
- Tính đa phương tiện giúp nhà phát triển có thể ứng dụng để xử lí hình ảnh hay video, mô phỏng, công cụ giáo dục….

Dưới đây là các thành phần cấu tạo nên Pygame:

- Pygame Module (Mô-đun Pygame): Đóng vai trò là khối xây dựng để giúp phát triển từng bước tạo ra trò chơi. Mô-đun hỗ trợ nhà phát triển các tác vụ như quản lý cửa sổ trò chơi, xử lý sự kiện và hiển thị đồ họa màn hình. Bằng cách kết hợp mô-đun Pygame vào code của mình, nhà phát triển sẽ tận dụng được tối đa tính năng này.
- Display Surface (Bề mặt hiển thị): Đóng vai trò rất lớn trong Pygame vì đây là nơi trò chơi của nhà phát triển hiển thị. Màn hình sẽ hiển thị lần lượt các đối tượng, hình ảnh, hoạt ảnh cũng như môi trường đồ hoạ trong trò chơi. Với chức năng này, nhà phát triển cũng có thể điều chỉnh kích thước, tiêu đề và các thuộc tính theo ý muốn của mình.



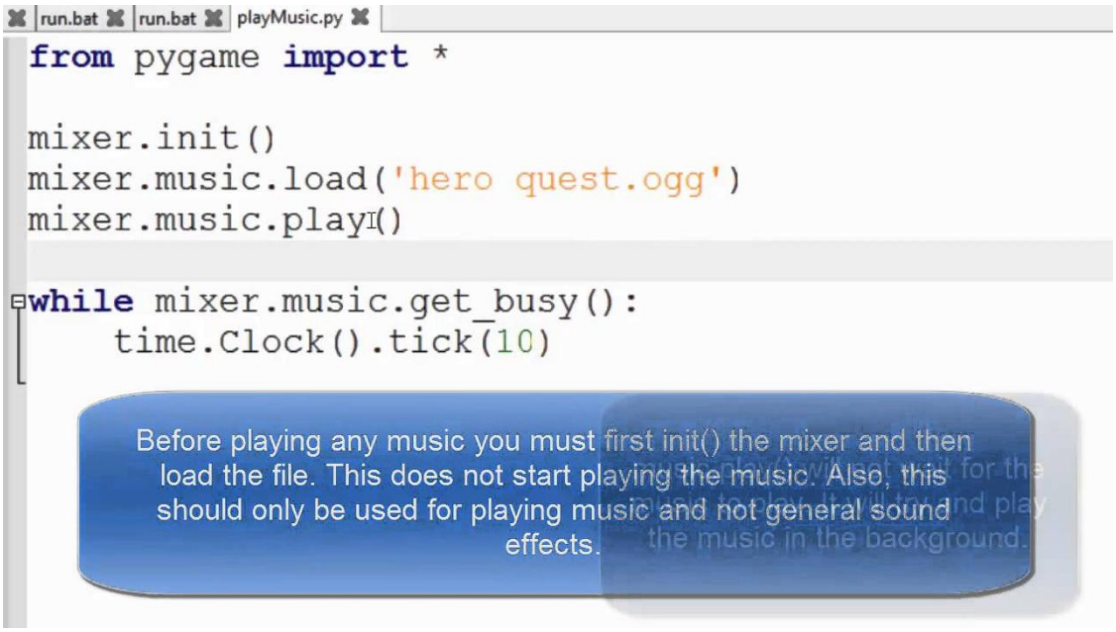
- Sprites (Nhân vật): Là một trong những yếu tố thiết yếu để phát triển trò chơi. Nhân vật sẽ đại diện cho thực thể trong thế giới trò chơi. Thư viện Pygame đơn giản hoá việc quản lý các nhân vật bằng cách cung cấp các nhân vật có sẵn được hiển thị trên màn hình. Pygame cũng cung cấp chức năng để nhà phát triển phát hiện được những xung đột giữa tương tác nhân vật và các hoạ tiết được thêm vào.

- Surface (Bề mặt): Các bề mặt trong Pygame là bơi hiển thị đồ hoạ như hình ảnh, văn bản… Với thư viện Pygame, nhà phát triển có thể tùy ý tải lên các hình ảnh khác nhau.

- Event Handling (Xử lí sự kiện): Thao tác này được Pygame đơn giản hóa bằng quá trình phát hiện và phản hồi các tương tác của người dùng. Bằng việc sử dụng mô-đun sự kiện do Pygame cung cấp, nhà phát triển có thể nắm bắt các sự kiện như nhấn phím, nhấp chuột và thay đổi kích thước cửa sổ. Chức năng này cho phép nhà phát triển tạo các trò chơi tương tác phản hồi linh hoạt, tạo điều kiện thuận lợi cho việc triển khai trò chơi và giao diện người dùng.

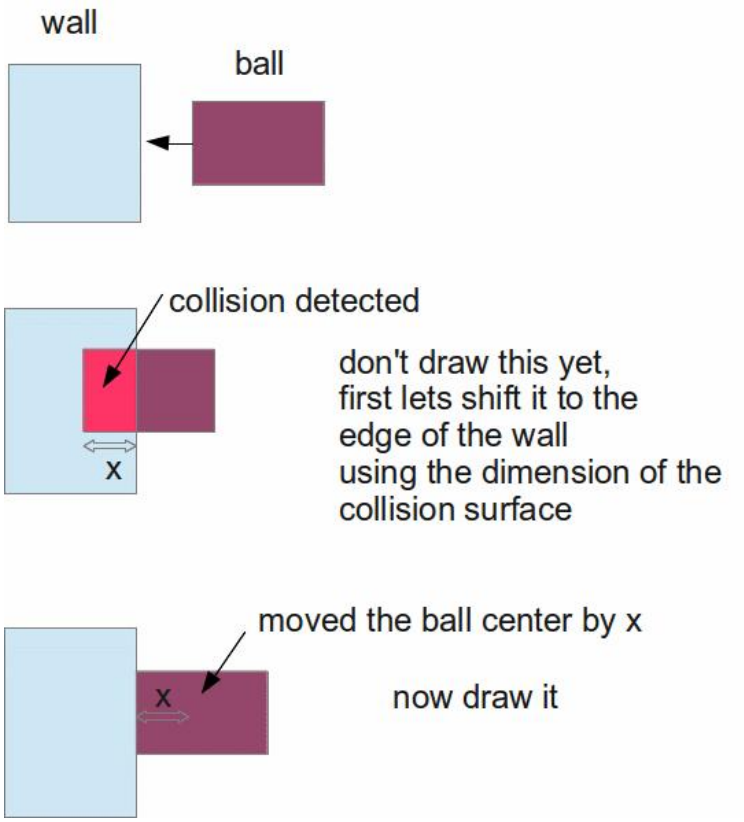
Sr No.	Event	Attributes
1.	KEYDOWN	key,mode,unicode
2.	KEYUP	key,mod
3.	MOUSEBUTTONUP	pos,button
4.	MOUSEBUTTONDOWN	pos,button
5.	MOUSEMOTION	pos,rel,buttons
6.	QUIT	-

- Sound and Music (Âm thanh và Âm nhạc): Đây là hai thành tố không thể thiếu trong trò chơi, Pygame cung cấp các chức năng để nhà điều khiển có thể tự mình tạo ra các âm thanh cho trò chơi của mình. Bên cạnh đó, mô-đun trộn trong Pygame còn cho phép nhà phát triển tải và phát hiệu ứng âm thanh, nhạc nền, điều chỉnh âm lượng… để giúp trò chơi sống động hơn



- Collision Detection (Phát hiện va chạm): Đây là một trong những khía cạnh quan trọng trong quá trình phát triển trò chơi. Pygame cung cấp cơ chế tích hợp để phát hiện va chạm giữa các họa tiết và các đối tượng bằng việc sử dụng tính năng phát hiện va chạm, nhà điều khiển có thể phát hiện và phản

hồi các va chạm, triển khai cơ chế chơi trò chơi như tính điểm và tạo tương tác thực tế giữa với các nhân vật trong thế giới của mình.



Việc liên tục tìm hiểu và ứng dụng các thành phần trong Pygame giúp nhà phát triển tối ưu được toàn bộ tính năng trong thư viện Pygame và ứng dụng biến những ý tưởng trò chơi thành hiện thực.

CHƯƠNG III: THIẾT KẾ

3.1. Phân tích và thiết kế Game:

Ý tưởng:

Flappy Bird là một trò chơi điện tử 2D do lập trình viên người Việt Nguyễn Hà Đông phát triển và phát hành lần đầu vào năm 2013. Dù chỉ là một trò chơi đơn giản, Flappy Bird lại tạo nên cơn sốt toàn cầu nhờ cách chơi dễ hiểu nhưng rất dễ "gây nghiện". Trò chơi nhanh chóng đạt hàng triệu lượt tải chỉ trong thời gian ngắn.

Trong game, người chơi điều khiển một chú chim nhỏ bay qua các cột ống nước. Mỗi lần nhấn phím, chim sẽ bay lên một đoạn, sau đó rơi xuống nếu không được giữ thăng bằng. Mục tiêu là bay càng xa càng tốt, vượt qua càng nhiều ống nước càng ghi được nhiều điểm. Nếu chim va vào ống hoặc rơi xuống đất, trò chơi kết thúc.

Nhóm chúng em chọn Flappy Bird làm đề tài vì gameplay đơn giản nhưng thú vị, rất phù hợp để luyện tập lập trình game với Python và thư viện Pygame. Ngoài việc tái

hiện lại trò chơi gốc, nhóm còn muốn thử thêm vào vài chức năng mới như chọn skin, để tăng phần hấp dẫn và sáng tạo cho sản phẩm cuối cùng.

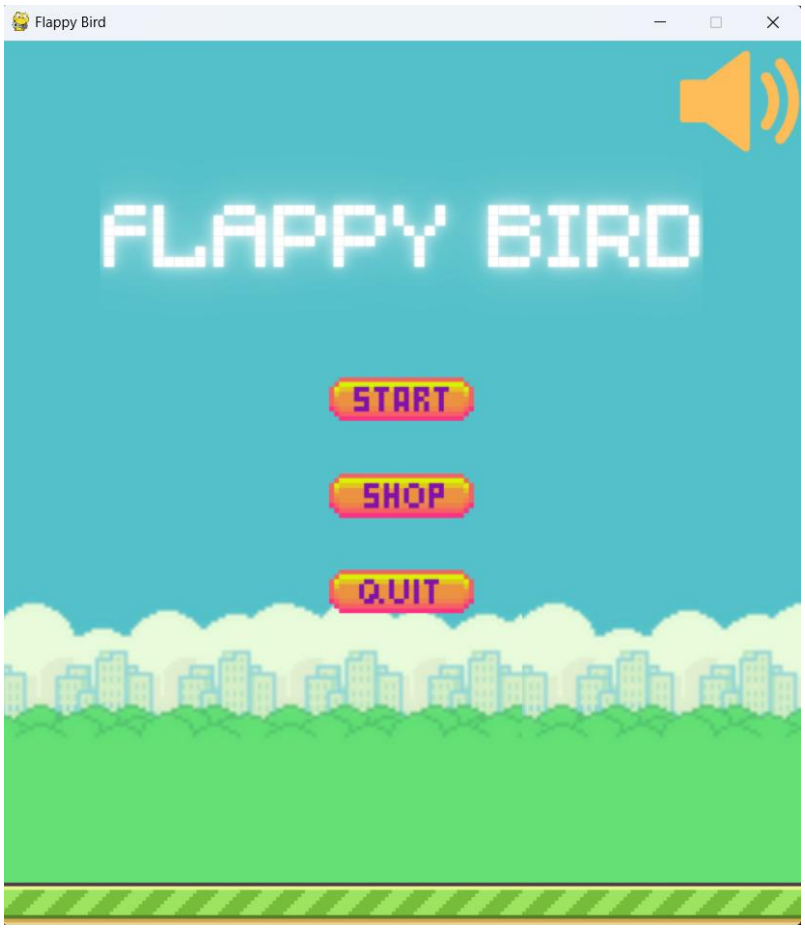
3.2. Các thành phần chính:

Màn hình chính (Main menu):

Màn hình chính là nơi đầu tiên người chơi nhìn thấy khi khởi động trò chơi. Giao diện được thiết kế đơn giản nhưng sinh động, với phong nền bầu trời xanh, mây trắng và thành phố phía xa, đúng phong cách pixel cổ điển của Flappy Bird.

Trên màn hình chính bao gồm 3 nút chức năng:

- START: Bắt đầu trò chơi.
- SHOP: Dẫn đến cửa hàng nơi người chơi có thể chọn skin cho chú chim.
- QUIT: Thoát khỏi trò chơi.
- TẮT / MỞ NHẠC: tắt mở nhạc nền tùy nhu cầu.

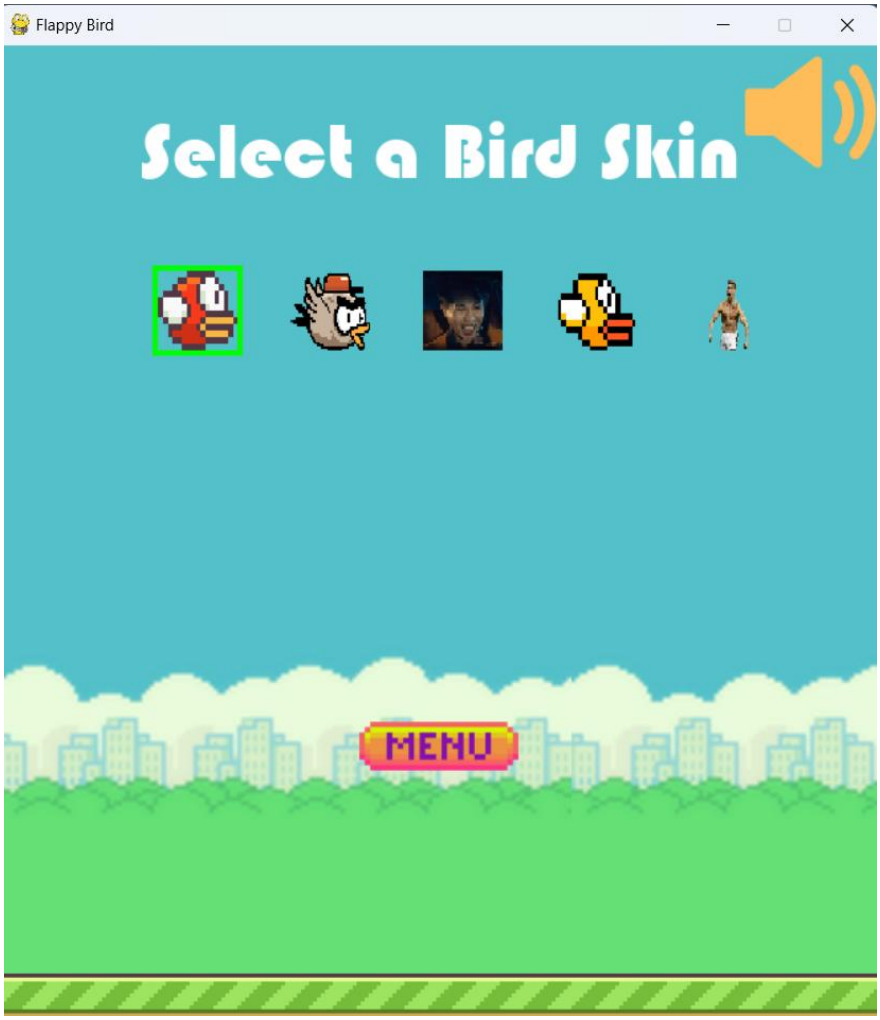


Giao diện chọn nhân vật

Sau khi người chơi nhấn nút SHOP từ màn hình chính, giao diện sẽ chuyển đến màn hình chọn nhân vật. Tại đây, người chơi có thể lựa chọn giao diện (skin) cho chú chim mà mình điều khiển trong game.

Các đặc điểm chính:

- Hiển thị nhiều lựa chọn skin: Các nhân vật được sắp xếp theo hàng ngang, dễ nhìn và dễ chọn.
- Khung viền xanh lá đánh dấu skin hiện tại đang được chọn.
- Nút MENU: Cho phép quay lại màn hình chính.

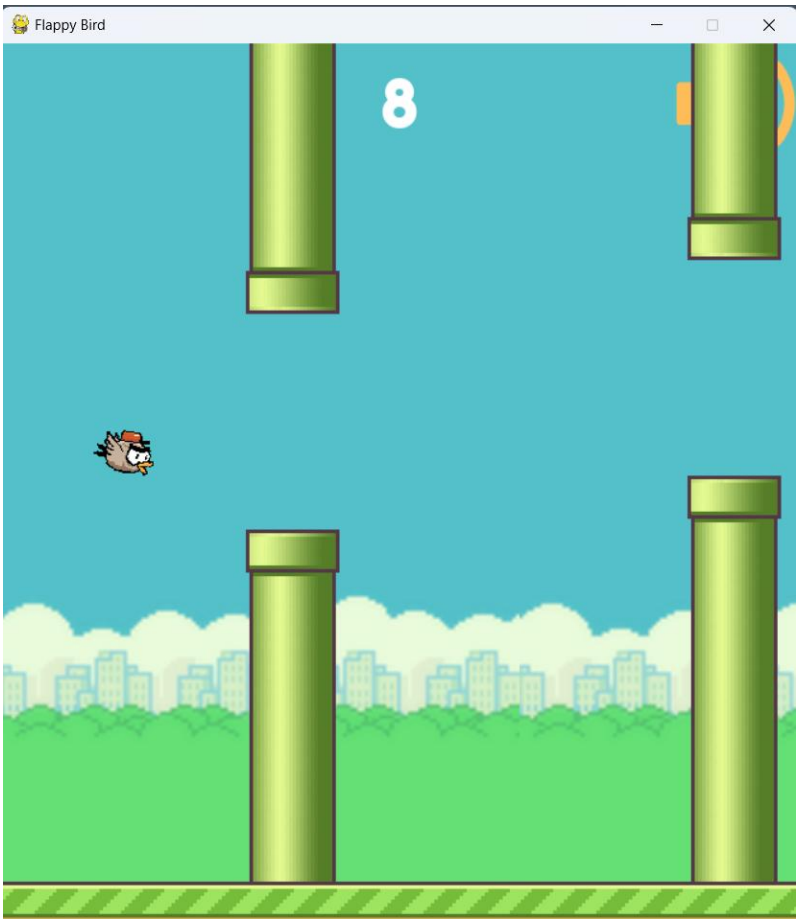


Gameplay

Đây là phần cốt lõi của trò chơi – nơi người chơi trực tiếp điều khiển chú chim bay vượt qua các chướng ngại vật.

Đặc điểm nổi bật:

- Cách chơi đơn giản: Người chơi nhấn phím (phím cách hoặc chuột) để giúp chim bay lên, sau đó chim sẽ rơi tự do xuống do trọng lực.
- Chướng ngại vật: Các cặp ống di chuyển từ phải sang trái tạo thành chướng ngại vật. Khoảng trống giữa các ống là nơi người chơi cần đưa chim bay qua.
- Tính điểm: Mỗi khi vượt qua một cặp ống thành công, người chơi sẽ được cộng thêm 1 điểm. Điểm số hiện tại được hiển thị rõ ràng ở giữa màn hình trên cùng.
- Va chạm: Nếu chim va vào ống nước hoặc rơi xuống đất, trò chơi sẽ kết thúc ngay lập tức.

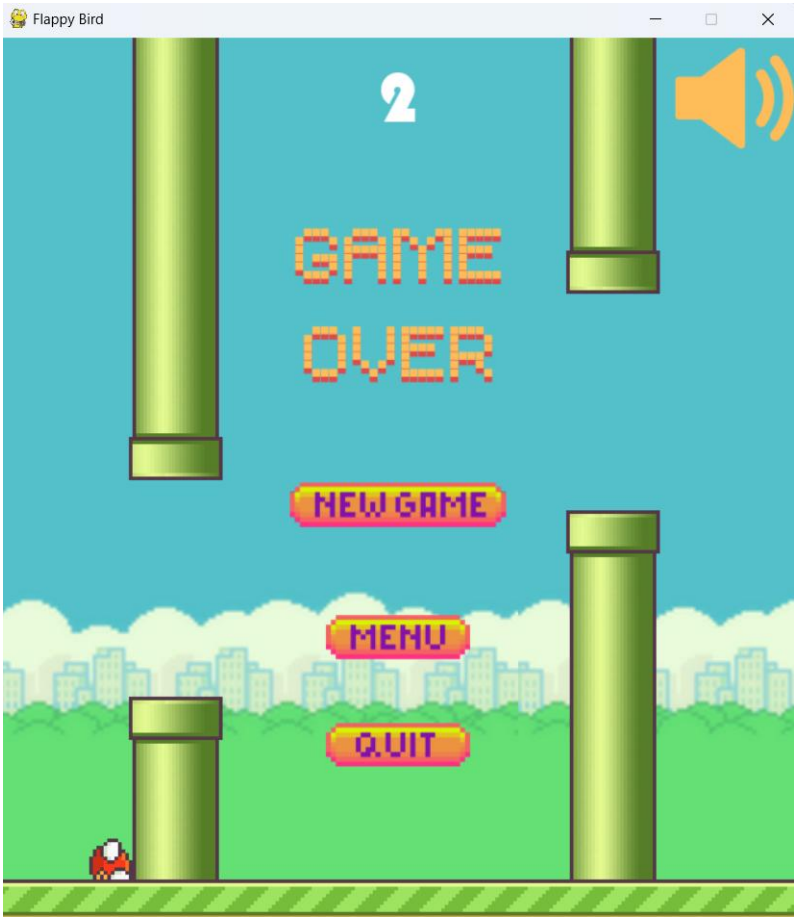


Giao diện kết thúc game

Màn hình Game Over xuất hiện khi người chơi thất bại, tức là chim đã va chạm với ống hoặc rơi xuống đất.

Các thành phần chính:

- Điểm số đạt được: Được hiển thị nổi bật ở phía trên cùng của màn hình, giúp người chơi dễ dàng nhận biết thành tích vừa đạt được.
- Thông báo "Game Over": Dòng chữ nổi bật với màu sắc bắt mắt nhằm thông báo rõ ràng rằng trò chơi đã kết thúc.
- Các lựa chọn tương tác:
- New Game: Bắt đầu một ván chơi mới ngay lập tức.
- Menu: Trở về màn hình chính (Main Menu).
- Quit: Thoát khỏi trò chơi.



CHƯƠNG IV: LẬP TRÌNH

4.1. Khởi tạo và cài đặt chung:

Thiết lập môi trường Pygame và các thông số cơ bản cho cửa sổ game:

```
import pygame
from pygame.locals import *
import random
import sys
import os

# init
pygame.init()
clock = pygame.time.Clock()
fps = 60

# screen
screen_width = 664
screen_height = 736
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption('Flappy Bird')
white = (255, 255, 255)
sound_on = True
```

Phần này đóng vai trò thiết lập môi trường hoạt động cho game. Bắt đầu bằng việc khởi động thư viện Pygame, tạo ra cửa sổ game với kích thước và tiêu đề cụ thể. Tại đây, chúng ta cũng định nghĩa các thông số cơ bản như tốc độ khung hình mục tiêu (FPS) để đảm bảo trải nghiệm chơi game mượt mà, và các hằng số màu sắc cần thiết cho việc hiển thị đồ họa…

4.2. Quản lý tài nguyên:

Phần này chịu trách nhiệm tải tất cả các tài nguyên bên ngoài mà game sử dụng, bao gồm hình ảnh và âm thanh. Việc tải tài nguyên được thực hiện một lần khi game khởi động để tối ưu hiệu suất. Các hàm quản lý âm thanh cũng được định nghĩa ở đây để quản lý việc phát nhạc nền và hiệu ứng âm thanh trong game.

```python
# img
bg = pygame.image.load('img/bg.png').convert()
ground_img = pygame.image.load('img/ground.png').convert_alpha()
button_img = pygame.image.load('img/newgame_btn.png').convert_alpha()
new_game_img = pygame.image.load('img/start_btn.png').convert_alpha()
exit_img = pygame.image.load('img/quit_btn.png').convert_alpha()
menu_img = pygame.image.load('img/menu_btn.png').convert_alpha()
game_over_img = pygame.image.load('img/gameover.png').convert_alpha()
game_logo_img = pygame.image.load('img/flappy_logo.png').convert_alpha()
shop_img = pygame.image.load('img/shop_btn.png').convert_alpha()
sound_on_img = pygame.image.load('img/sound_on.png').convert_alpha()
sound_off_img = pygame.image.load('img/sound_off.png').convert_alpha()

# skin
bird_skins = []
base_bird_img = pygame.image.load('img/bird1.png').convert_alpha()
base_size = base_bird_img.get_size()

skin_index = 1
while True:
    skin_path = f'img/bird{skin_index}.png'
    if not os.path.isfile(skin_path):
        break
    skin_img = pygame.image.load(skin_path).convert_alpha()
    skin_img = pygame.transform.scale(skin_img, base_size)
    bird_skins.append(skin_img)
    skin_index += 1

if len(bird_skins) == 0:
    bird_skins.append(base_bird_img)
```

4.3. Quản lý biến trạng thái (toàn cục):

Phần này khởi tạo và quản lý tất cả các biến toàn cục cần thiết để theo dõi và điều khiển trạng thái hiện tại của game. Các biến này quyết định xem game đang ở màn hình nào (menu, play, game over, shop), điểm số của người chơi, trạng thái của chim, và các thông số liên quan đến môi trường game như tốc độ game hay tần suất xuất hiện các cặp ống.

```
69  # state var
70  ground_scroll = 0
71  scroll_speed = 4
72  flying = False
73  game_over = False
74  pipe_gap = 180
75  pipe_frequency = 1500
76  last_pipe = pygame.time.get_ticks() - pipe_frequency
77  score = 0
78  pass_pipe = False
79  main_menu = True
80  selected_skin = 1
81  shop_menu = False
```

4.4. Các hàm hỗ trợ và công cụ:

Phần này chứa các hàm tiện ích được thiết kế để thực hiện các tác vụ chung, giúp source gọn gàng và dễ quản lý hơn. Hàm *draw_center_text* dùng để tạo các đoạn văn bản ở giữa màn hình một cách thuận tiện, trong khi *reset_game* đảm bảo game có thể được đưa về trạng thái khởi đầu sau mỗi ván chơi.

```
83  # func
84  def draw_center_text(text, font, color, x, y):
85      img = font.render(text, True, color)
86      rect = img.get_rect(center=(x, y))
87      screen.blit(img, rect)
88
89  def reset_game():
90      pipe_group.empty()
91      global flappy
92      flappy = Bird(100, screen_height // 2, selected_skin)
93      bird_group.empty()
94      bird_group.add(flappy)
95      return 0
```

4.5. Định nghĩa các Class cần thiết:

Phần này định nghĩa các Class cho các thực thể chính tương tác trong game: nhân vật (mặc định là Bird), các cặp ống làm chướng ngại vật, các Button dùng cho giao diện. Mỗi Class bao gồm các thuộc tính và phương thức để mô phỏng chức năng của chúng trong trò chơi.

```
# class
class Bird(pygame.sprite.Sprite):
    def __init__(self, x, y, skin_index):
        super().__init__()
        self.images = [bird_skins[skin_index - 1]]
        self.index = 0
        self.counter = 0
        self.image = self.images[self.index]
        self.rect = self.image.get_rect(center=(x, y))
        self.vel = 0
        self.clicked = False

    def update(self):
        if flying:
            self.vel += 0.5
            self.vel = min(self.vel, 8)
            if self.rect.bottom < 768:
                self.rect.y += int(self.vel)

        if not game_over:
            if pygame.mouse.get_pressed()[0] == 1 and not self.clicked:
                self.clicked = True
                self.vel = -10
            if pygame.mouse.get_pressed()[0] == 0:
                self.clicked = False

            self.counter += 1
            if self.counter > 5:
                self.counter = 0
                self.index = (self.index + 1) % len(self.images)
            self.image = pygame.transform.rotate(self.images[self.index], self.vel * -2)
        else:
            self.image = pygame.transform.rotate(self.images[self.index], -90)
```

Hình 4.5.1: Class Bird dành cho nhân vật

```
class Pipe(pygame.sprite.Sprite):
    def __init__(self, x, y, position):
        super().__init__()
        self.image = pygame.image.load('img/pipe.png').convert_alpha()
        self.rect = self.image.get_rect()
        if position == 1:
            self.image = pygame.transform.flip(self.image, False, True)
            self.rect.bottomleft = [x, y - pipe_gap // 2]
        else:
            self.rect.topleft = [x, y + pipe_gap // 2]

    def update(self):
        self.rect.x -= scroll_speed
        if self.rect.right < 0:
            self.kill()
```

Hình 4.5.2: Class Pipe dành cho các cặp ống

```python
class Button:
    def __init__(self, x, y, image):
        self.image = image
        self.original_image = image.copy()
        self.rect = self.image.get_rect(center=(x, y))
        self.clicked = False  # bien theo doi click

    def draw(self):
        action = False
        pos = pygame.mouse.get_pos()

        if self.rect.collidepoint(pos):
            bright_image = pygame.Surface(self.image.get_size()).convert_alpha()
            bright_image.fill((40, 40, 40, 0))
            hover_img = self.original_image.copy()
            hover_img.blit(bright_image, (0, 0), special_flags=pygame.BLEND_RGB_ADD)
            screen.blit(hover_img, self.rect)

            if pygame.mouse.get_pressed()[0] == 1 and not self.clicked:
                action = True
                self.clicked = True
            elif pygame.mouse.get_pressed()[0] == 0:
                self.clicked = False
        else:
            screen.blit(self.image, self.rect)

        return action
```

Hình 4.5.3: Class Button dùng cho các button ở Menu

4.6. Quản lý Sprite Group và Button:

Để quản lý các đối tượng game một cách hiệu quả, Pygame sử dụng khái niệm sprite group.

Phần này tạo ra các nhóm để chứa Bird và Pipe, giúp dễ dàng cập nhật và quản lý. Ngoài ra, tất cả các button dùng để tương tác trong giao diện game cũng được khởi tạo ở đây với kích thước và path cụ thể, sẵn sàng cho vòng lặp Main xử lý tương tác.

```python
# sprite group
bird_group = pygame.sprite.Group()
pipe_group = pygame.sprite.Group()
flappy = Bird(100, screen_height // 2, selected_skin)
bird_group.add(flappy)

# Buttons
restart_button = Button(screen_width // 2, screen_height // 2 + 20, button_img)
back_to_menu_button = Button(screen_width // 2, screen_height // 2 + 160, menu_img)
exit_button_gameover = Button(screen_width // 2, screen_height // 2 + 220, exit_img)
new_game_button = Button(screen_width // 2, screen_height // 2 - 70, new_game_img)
shop_button = Button(screen_width // 2, screen_height // 2 + 10, shop_img)
exit_button_mainmenu = Button(screen_width // 2, screen_height // 2 + 90, exit_img)
menu_button_gameover = Button(screen_width // 2, screen_height // 2 + 130, menu_img)
sound_button = Button(screen_width - 50, 50, sound_on_img)

```

4.7. Vòng lặp chính của game (Main loop):

Đây là phần quan trọng nhất của chương trình, hoạt động như bộ não của game. Vòng lặp chính liên tục kiểm tra các sự kiện (từ người dùng, từ thời gian), cập nhật trạng thái của tất cả các đối tượng trong game, và vẽ lại toàn bộ màn hình để hiển thị những thay đổi. Phần này quản lý flow giữa các màn hình (menu, play, game over, shop) trong game, xử lý logic điểm số, va chạm, và tạo random ra các chướng ngại vật mới.

```
192  run = True
193  while run:
194      clock.tick(fps)
195      screen.blit(bg, (0, 0))
196
197      #sound btn
198      sound_button.image = sound_on_img if sound_on else sound_off_img
199      sound_button.original_image = sound_button.image.copy()
200      if sound_button.draw():
201          sound_on = not sound_on
202          if sound_on:
203              if main_menu or shop_menu:
204                  play_music(menu_music)
205              else:
206                  play_music(game_music)
207          else:
208              stop_music()
209
```

Hình 4.7.1: Block code xử lý Button tắt/ bật âm thanh

```
210     # main menu
211     if main_menu and not shop_menu:
212         if current_music != 'menu':
213             play_music(menu_music)
214             current_music = 'menu'
215 
216         screen.blit(ground_img, (0, 700))
217         screen.blit(game_logo_img, game_logo_img.get_rect(center=(screen_width // 2, screen_height // 2 - 200)))
218 
219         if new_game_button.draw():
220             main_menu = False
221             shop_menu = False
222             flying = False
223             game_over = False
224             score = reset_game()
225 
226         if shop_button.draw():
227             shop_menu = True
228 
229         if exit_button_mainmenu.draw():
230             run = False
```

Hình 4.7.2: Block code Main menu

```
232     # shop Menu
233     elif shop_menu:
234         if current_music != 'menu':
235             play_music(menu_music)
236             current_music = 'menu'
237 
238         screen.blit(ground_img, (0, 700))
239         draw_center_text("Select a Bird Skin", font, white, screen_width // 2, 80)
240 
241         for i, skin in enumerate(bird_skins):
242             skin_scaled = pygame.transform.scale(skin, (60, 60))
243             rect = skin_scaled.get_rect(center=(150 + i * 100, 200))
244             screen.blit(skin_scaled, rect)
245 
246             if rect.collidepoint(pygame.mouse.get_pos()):
247                 pygame.draw.rect(screen, (255, 255, 0), rect.inflate(6, 6), 3)
248                 if pygame.mouse.get_pressed()[0]:
249                     selected_skin = i + 1
250                     flappy = Bird(100, screen_height // 2, selected_skin)
251                     bird_group.empty()
252                     bird_group.add(flappy)
253 
254             if selected_skin == i + 1:
255                 pygame.draw.rect(screen, (0, 255, 0), rect.inflate(8, 8), 4)
256 
257         if back_to_menu_button.draw():
258             shop_menu = False
259             main_menu = True
```

Hình 4.7.3: Block code Shop menu - nơi người chơi chọn skin để bắt đầu game

```
    # game
    else:
        if current_music != 'game':
            play_music(game_music)
            current_music = 'game'

        bird_group.draw(screen)
        bird_group.update()
        pipe_group.draw(screen)
        screen.blit(ground_img, (ground_scroll, 700))

        if len(pipe_group) > 0:
            bird = bird_group.sprites()[0]
            pipe = pipe_group.sprites()[0]
            if bird.rect.left > pipe.rect.left and bird.rect.right < pipe.rect.right and not pass_pipe:
                pass_pipe = True
            if pass_pipe and bird.rect.left > pipe.rect.right:
                score += 1
                pass_pipe = False

        draw_center_text(str(score), font, white, screen_width // 2, 50)

        if pygame.sprite.groupcollide(bird_group, pipe_group, False, False) or flappy.rect.top < 0:
            if not game_over:
                stop_music()
                gameover_music.play()
            game_over = True
            current_music = None

        if flappy.rect.bottom >= 700:
            if not game_over:
                stop_music()
                gameover_music.play()
            game_over = True
            flying = False
            current_music = None
```

Hình 4.7.4: Block code xử lý logic khi chơi game

```
        if not game_over and flying:
            time_now = pygame.time.get_ticks()
            if time_now - last_pipe > pipe_frequency:
                pipe_height = random.randint(-100, 100)
                pipe_group.add(Pipe(screen_width, screen_height // 2 + pipe_height, -1))
                pipe_group.add(Pipe(screen_width, screen_height // 2 + pipe_height, 1))
                last_pipe = time_now

            ground_scroll -= scroll_speed
            if abs(ground_scroll) > 35:
                ground_scroll = 0

            pipe_group.update()

        if game_over:
            screen.blit(game_over_img, game_over_img.get_rect(center=(screen_width // 2, screen_height // 2 - 150)))

            if restart_button.draw():
                game_over = False
                flying = False
                score = reset_game()

            if menu_button_gameover.draw():
                game_over = False
                flying = False
                main_menu = True
                shop_menu = False

            if exit_button_gameover.draw():
                run = False
```

Hình 4.7.5: Block code xử lý logic khi chơi game

```
329     # global events
330     for event in pygame.event.get():
331         if event.type == pygame.QUIT:
332             run = False
333         if event.type == pygame.MOUSEBUTTONDOWN and not flying and not game_over and not main_menu:
334             flying = True
335 
336     pygame.display.update()
337 
338 pygame.quit()
339 sys.exit()
```

Hình 4.7.5:Event toàn cục và kết thúc chương trình

CHƯƠNG V: KẾT LUẬN

5.1. Kết luận sau khi kết thúc đồ án:

Đồ án "Flappy Bird”, đã được Nhóm 5 của chúng em hoàn thành dưới sự hướng dẫn tận tình của cô Phan Thị Thể. Đây là một dự án không chỉ tái hiện một trò chơi kinh điển mà còn là cơ hội quý báu để chúng em áp dụng và củng cố kiến thức về phát triển ứng dụng đa phương tiện.

Trong quá trình thực hiện đồ án, nhóm đã thành công trong việc xây dựng một hệ thống game hoàn chỉnh, bao gồm các chức năng cốt lõi sau:

+ Xây dựng Logic Game Động: Chúng em đã triển khai thành công cơ chế chuyển động của chim với trọng lực và phản ứng nhảy linh hoạt. Đồng thời, hệ thống tạo và di chuyển ống cũng được phát triển một cách ngẫu nhiên, đảm bảo tính thử thách và khả năng chơi lại cho game. Cơ chế tính điểm và phát hiện va chạm chính xác cũng là điểm nhấn, tạo nên sự hấp dẫn cho mỗi ván chơi.

+ Thiết Kế Giao Diện Tương Tác: Đồ án chú trọng vào việc tạo ra một giao diện người dùng trực quan và dễ sử dụng. Từ màn hình menu chính với các tùy chọn rõ ràng (Chơi mới, Cửa hàng, Thoát), màn hình cửa hàng skin cho phép người chơi tùy chỉnh chú chim của mình, cho đến màn hình Game Over hiển thị điểm số và các lựa chọn tiếp theo, tất cả đều được thiết kế để mang lại trải nghiệm liền mạch cho người dùng.

+ Quản Lý Tài Nguyên Đa Phương Tiện: Chúng em đã thực hiện hiệu quả việc tích hợp các yếu tố đa phương tiện vào game. Các tài nguyên đồ họa như hình ảnh nền, chim, ống, và nút bấm được tải và tối ưu hóa để hiển thị mượt mà. Bên cạnh đó, hệ thống âm thanh với nhạc nền phù hợp và hiệu ứng âm thanh

rõ ràng đã góp phần tăng cường tính sống động và cuốn hút của trò chơi, đi kèm với tính năng bật/tắt tiện lợi.

+ Ứng Dụng Nguyên Lý Lập Trình Hướng Đối Tượng (OOP): Các thực thể trong game như chim, ống, và nút bấm được xây dựng dưới dạng các lớp đối tượng riêng biệt. Cách tiếp cận này không chỉ giúp tổ chức mã nguồn một cách khoa học, dễ hiểu mà còn tạo điều kiện thuận lợi cho việc bảo trì và mở rộng tính năng trong tương lai.

Thông qua đồ án này, Nhóm 5 đã tích lũy được nhiều kinh nghiệm thực tế trong việc phát triển game 2D bằng Python và Pygame, từ khâu thiết kế đến triển khai. Chúng em một lần nữa được xin chân thành cảm ơn cô đã tận tình hướng dẫn và cung cấp những kiến thức quý báu, giúp chúng em hoàn thành tốt đồ án này.

5.2. Hướng phát triển trong tương lai:

Mặc dù đồ án đã hoàn thiện các chức năng cơ bản của game, vẫn còn rất nhiều tiềm năng để phát triển và cải thiện trải nghiệm người dùng. Một số hướng phát triển tiềm năng mà trong tương lai mà nhóm đặt ra được bao gồm:

+ Hệ thống điểm cao (High Score): Lưu trữ và hiển thị điểm số cao nhất của người chơi, khuyến khích tính cạnh tranh.

+ Thêm nhiều skin và hiệu ứng: Phát triển thêm các skin chim độc đáo, hiệu ứng âm thanh và hình ảnh mới khi đạt mốc điểm nhất định hoặc khi chim va chạm.

+ Độ khó động: Tăng dần tốc độ bay của chim hoặc thay đổi khoảng cách giữa các ống khi điểm số tăng lên để tăng tính thử thách.

+ Chế độ chơi mới: Giới thiệu các chế độ chơi khác như chế độ tính thời gian, chế độ vượt chướng ngại vật đặc biệt.

+ Tích hợp cơ chế tiền tệ/cửa hàng: Cho phép người chơi thu thập tiền trong game để mở khóa các skin hoặc vật phẩm hỗ trợ.

+ Cải thiện đồ họa và animation: Nâng cấp chất lượng hình ảnh, thêm các animation mượt mà hơn cho chim và môi trường.

+ Tối ưu hóa hiệu suất: Tiếp tục tối ưu hóa mã nguồn để game chạy mượt mà hơn trên nhiều cấu hình máy tính.

BẢNG PHÂN CÔNG VÀ ĐÁNH GIÁ CÔNG VIỆC

Họ và tên	MSSV	Nhiệm vụ	Mức độ hoàn thành	Điểm đánh giá của nhóm	Điểm đánh giá của GV
Trần Nguyễn Bá Huy (leader)	N22DCPT037	Nhiệm vụ đã làm: + Xây dựng class Button + Xây dựng logic xử lý sound effect. + Xây dựng Main menu + Xây dựng Shop menu - chọn skin + Tìm kiếm và design các tài nguyên (sound effect, hình ảnh). + Tổng hợp, làm báo cáo, làm slide.	100%	10	
Nguyễn Đăng Khoa	N22DCPT045	Nhiệm vụ đã làm: + Xây dựng class Bird. + Xây dựng class Pipe. + Xây dựng hàm công cụ(reset game, text-center,..) + Xây dựng logic ingame. + Tìm kiếm các hình ảnh tài nguyên. + Hỗ trợ làm báo cáo, làm slide.	100%	10	

HẾT