BÁO CÁO ĐỒ ÁN

Người chịu trách nhiệm dự án: Nguyễn Minh Trí (MSSV: 20120219).

Báo cáo quá trình phát triển game:

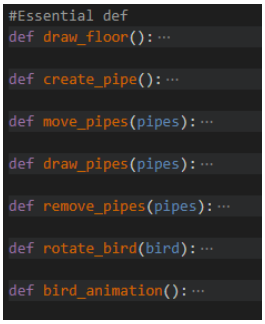
- Có 03 dự án games được phát triển: Connect4, FlappyBirdPygame và SYMBOLS Memory Game.
- Trạng thái các games: Đã hoàn thành.
- Bảng tóm tắt quá trình xây dựng:

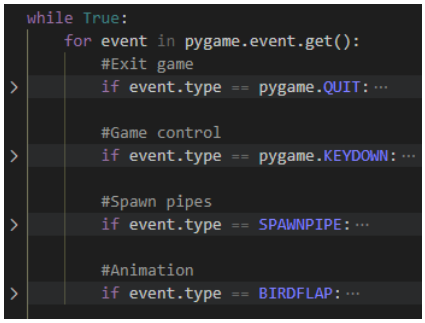
Tuần	Tóm tắt quá trình
Tuần 1	Tìm hiểu module Pygame và cách sử dụng. Viết và chạy thử FlappyBirdPygame.
Tuần 2	Thêm / thử một số tính năng và hoàn thiện FlappyBirdPygame. Bắt đầu xây dựng SYMBOLS Memory Game.
Tuần 3	Tiếp tục xây dựng SYMBOLS Memory Game.
Tuần 4	Thêm một số tính năng và hoàn thiện SYMBOLS Memory Game. Xây dựng trò chơi Connect4.
Tuần 5	Thêm giao diện và hoàn thiện Connect4. Tạo Demo giới thiệu game.

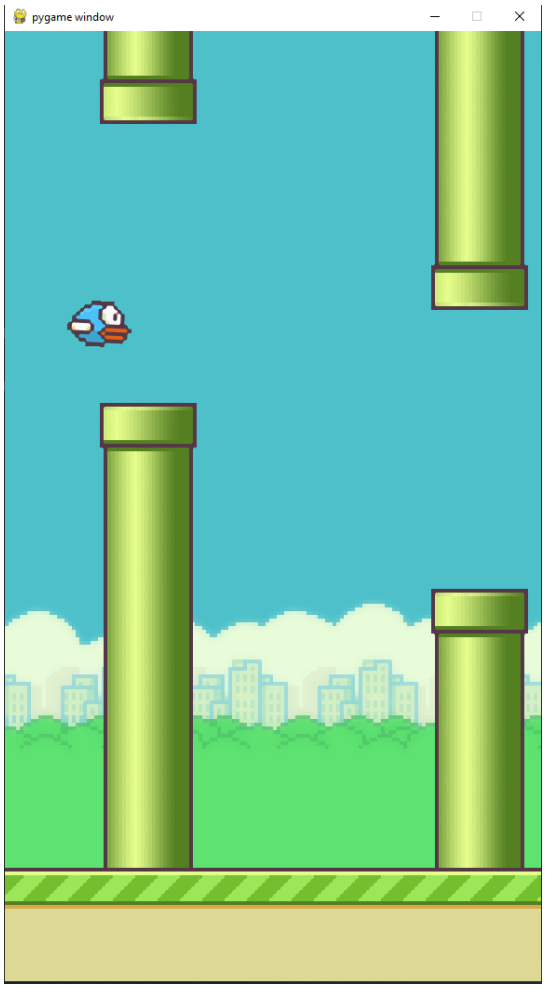
Chi tiết quá trình xây dựng:

Tuần 1:

- Tìm hiểu các công cụ làm game theo sự hướng dẫn của GVHD, xem các videos hướng dẫn, demo trên Youtube.
- Quyết định chọn ngôn ngữ và engine để viết game (Python và moudle Pygame) và công cụ viết, thử và chạy game (Visual Studio Code).
- Tìm hiểu về cách sử dụng moudle Pygame (www.pygame.org/docs/).
- Cài đặt moudle và thực hành, áp dụng viết game FlappBirdPygame.
- Tạo được các hàm cơ bản như xử lý event từ user, xuất (render) các thành phần cần thiết nhất trong game.







Hình 1, 2 và 3: Các hàm cơ bản và giao diện ban đầu.

Tuần 2:

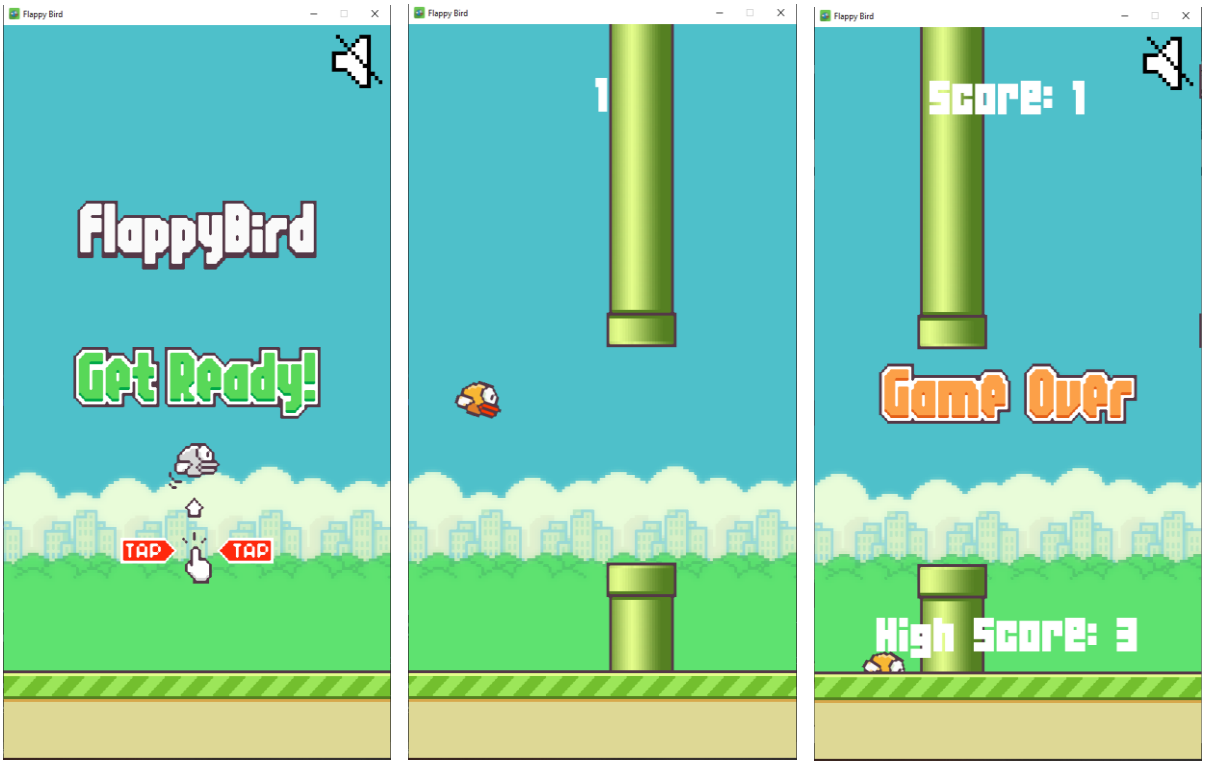
- Thêm một số tính năng cho FlappyBirdPygame như file config, menu screen, game over screen, thêm âm thanh và viết các hàm để vận hành trong game.
- Tinh chỉnh các thông số, chạy thử và đóng gói game sử dụng PyInstaller.

```
#READ AND LOAD THE CONFIG
config = configparser.ConfigParser()
config.read("data/config.ini")
```

```
for event in pygame.event.get():
    #Exit game
    if event.type == pygame.QUIT: ···

    #Game control
    if event.type == pygame.KEYDOWN and LOCK_KEYBOARD == False: ···

    if event.type == pygame.MOUSEBUTTONDOWN:
        x, y = event.pos
        if speaker_rect.collidepoint(x, y):
            IS_MUTED = not IS_MUTED

    #Spawn pipes
    if event.type == SPAWNPIPE: ···

    #Animate the bird
    if event.type == BIRDFLAP and GAMEACTIVE == True: ···
```

```
#FUNTIONS
def render_ground(): ···

def spawn_pipe(): ···

def move_pipes(pipes): ···

def render_pipes(pipes): ···

def remove_pipes(pipes): ···

def rotate_bird(bird): ···

def bird_animation(): ···

def isCollide(pipes): ···

def update_score(score, high_score): ···

def display(game_state): ···
```

```
# Gameplay
# Check sound
if IS_MUTED: ···
else: ···

# INIT checking is it the
# first launch of the game
if INIT: ···

# if GAMEACTIVE == True
# game is playing. Else
# => lose the game.
elif GAMEACTIVE: ···

else: ···
```

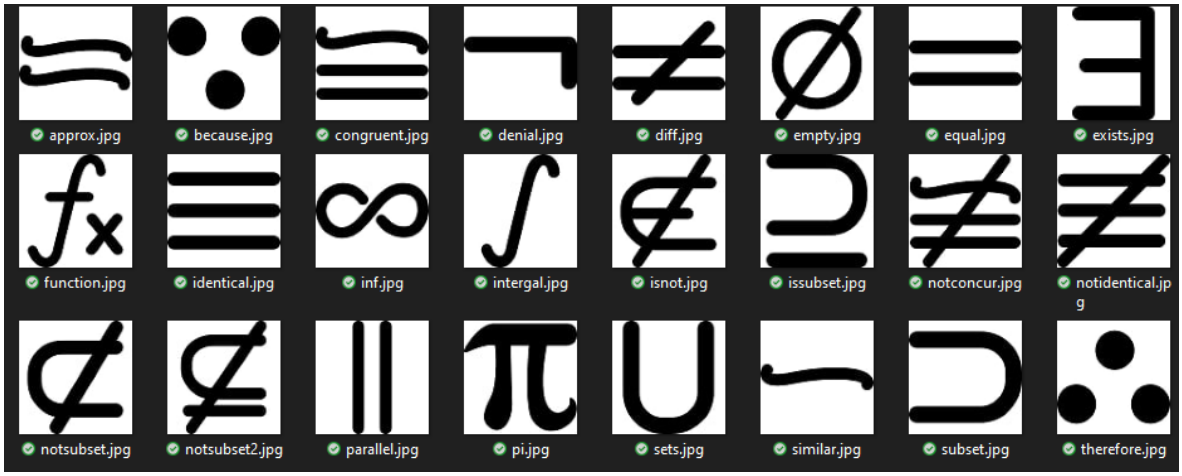


Hình 4, 5, 6, 7, 8, 9 và 10: Hoàn thiện các hàm và xử lý sự kiện, giao diện hoàn chỉnh của trò chơi.

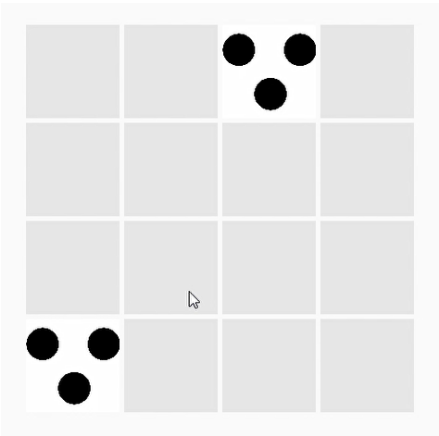
- Với kiến thức có được sau khi viết trò chơi FlappyBirdPygame, bắt đầu thiết kế và dựng SYMBOLS Memory Game.

- Với ý tưởng tạo một trò chơi giải đố tuy quen thuộc mà khó chơi, tôi đã sử dụng bộ biểu tượng của Toán học, với lối chơi tìm hai hình giống nhau truyền thống.

+ Chuẩn bị assets.



+ Dựng gameplay cơ bản.



+ Viết cơ bản các hàm xử lý event từ user và các class để xử lý các sự kiện trong game.

```
COLOR0 = (250, 250, 250)
COLOR1 = (230, 230, 230)
COLOR2 = (100, 100, 100)
COLOR3 = (120, 120, 120)

class Card:···

class GameBoard:···

class GameStats:···

def convertMillisecondsToMinAndSec(milliseconds):···

class ImageLibrary:···
```

```
COLOR1 = (90, 90, 90)
COLOR2 = (120, 120, 120)

class Button:···

class ActionButton(Button):···

class BoardSizeButton(Button):···
```

```
for event in pygame.event.get():
    if event.type == QUIT:···
    if event.type == pygame.MOUSEMOTION:···
    if event.type == MOUSEBUTTONDOWN:···

if gameState == 'game_not_started' and mouse_clicked:···

if gameState == 'game_started' and mouse_clicked:···

if gameState == 'show_info' and mouse_clicked:···

if gameState == 'game_won' and mouse_clicked:···

if mouse_clicked and row is not None and column is not None
and (row, column) != tempRowCol and not game_board[row][column].isMatched:
```

Hình 13, 14 và 15: Các class xử lý bảng hình, xử lý các sự kiện của nút và xử lý event user input.

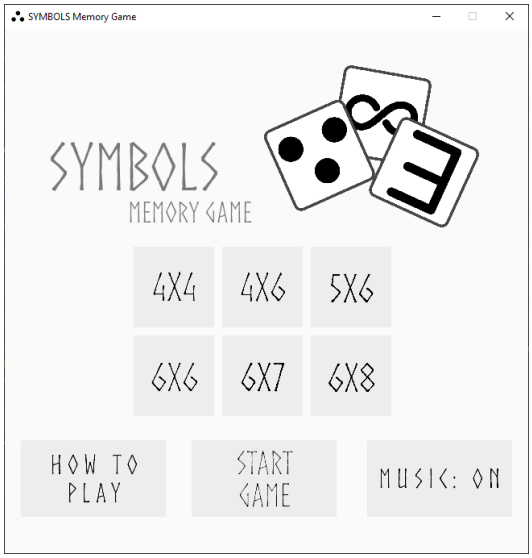
Tuần 3:

- Tiếp tục xây dựng SYMBOLS Memory Game.
  + Tạo màn hình Menu
  + Tạo các kích thước bảng chơi khác nhau
  + Tạo màn hình sau khi giải xong câu đố
  + Tạo hiệu ứng khi nhấn vào các ô
  + Thiết kế các nút để định vị trong trò chơi

Hình 16, 17, 18 và 19: Giao diện game và gameplay của game

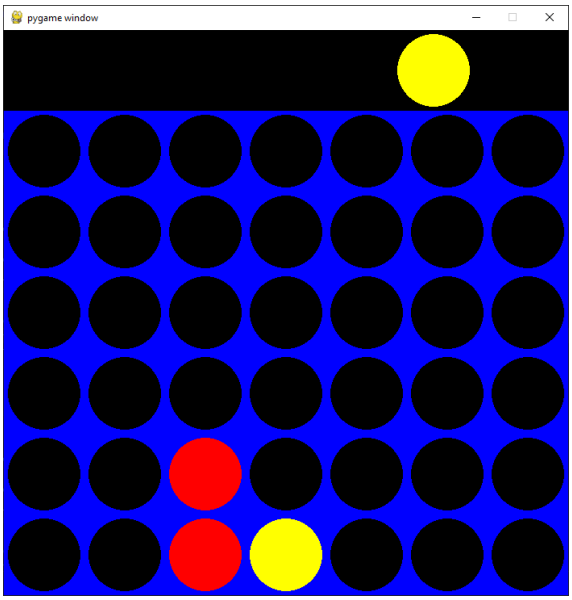
Tuần 4:

- Hoàn thiện trò chơi SYMBOLS Memory Game.
  + Thêm hiệu ứng âm thanh cho trò chơi
  + Thiết kế thông tin và các nút chức năng như How to play, Sound On / Off
  + Sửa lỗi, tối ưu, chạy thử và đóng gói game sử dụng PyInstaller.



Hình 20 và 21: Giao diện hoàn chỉnh của game

- Với mục đích ban đầu là tạo ra 2 game thuần tuý, tôi quyết định viết thêm Connect4, một trò chơi tuy không nổi tiếng ở Việt Nam nhưng quen thuộc với những người nước ngoài với lối chơi giống Caro nhưng độc đáo hơn, mang lại trải nghiệm mới lạ cho người chơi. Đây là tựa game chiến thuật 2 người chơi, mỗi lượt chơi người chơi sẽ thả những "token" của mình xuống những ô trống, người nào có 4 "token" liên tục với nhau (theo hàng thẳng, hàng dọc hoặc hàng xéo) sẽ chiến thắng.
  + Dựng gameplay cơ bản của trò chơi
  + Viết hàm xử lý event input từ người chơi
  + Viết hàm kiểm tra khả năng thắng của người chơi



Hình 22: Giao diện ban đầu của trò chơi.

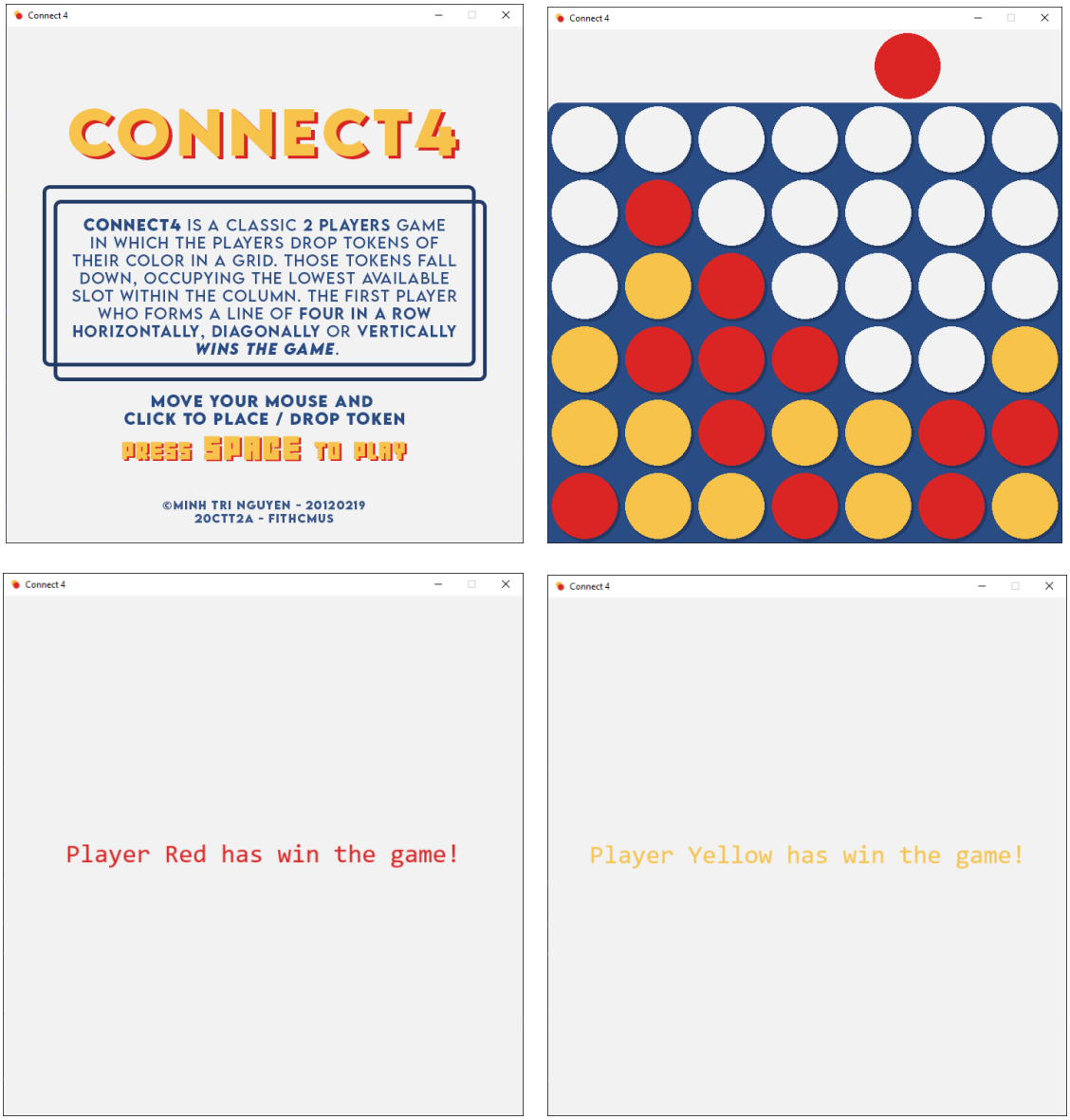
```
def create_board():
def drop_piece(board, row, col, piece):
def is_valid_location(board, col):
def get_next_open_row(board, col):
def print_board(board):
def winning_move(board, piece):
def draw_board(board):
```

```
while not game_over:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
        if event.type == pygame.MOUSEMOTION:
        if event.type == pygame.MOUSEBUTTONDOWN:
```

Hình 23 và 24: Các hàm xử lý gameplay và xử lý user input của trò chơi.

Tuần 5:

- Hoàn thiện trò chơi Connect 4.
  + Thiết kế lại giao diện trò chơi.
  + Thêm màn hình Khởi động và Thắng trò chơi.
  + Thiết kế màn hình chính.
  + Thêm hiệu ứng khi thắng trò chơi.
  + Sửa lỗi, tối ưu, chạy thử và đóng gói game sử dụng PyInstaller.
- Tạo demo các game.
- Báo cáo nhóm, họp bàn và hoàn thành đồ án.



Hình 25 và 26: Giao diện chính thức của trò chơi