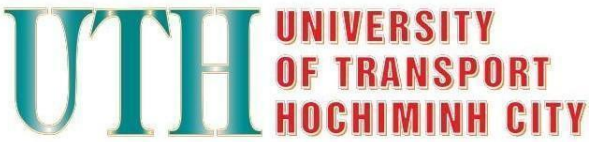
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH

Tên đề tài:

Xây dựng ứng dụng game ghép hình

Giảng viên hướng dẫn: Ths. Mai Thanh Tâm

Sinh viên thực hiện:

2051120231: Đinh Phi Hậu

2051120232: Nguyễn Xuân Hậu

Thành phố Hồ Chí Minh, năm 2024

I. Tổng quan

1. Chức năng chính:

- Game xếp hình:
    - Game sử dụng chuột để chơi không dùng bàn phím.
    - Game có menu chức năng lựa chọn.
    - Hiển thị bảng chính được chia thành từng ô trống.
    - Hiển thị bảng khác chứa các mảnh hình ảnh nhỏ để ghép vào bảng chính.
    - Hiển thị hình nhỏ ở góc để gợi ý người chơi.
    - Người chơi kéo thả các ảnh nhỏ ghép thành một ảnh lớn và hoàn thành trò chơi.
    - Bộ đếm thời gian để hoàn thành trò chơi.
    - Hiển thị ảnh kết quả khi hoàn thành game nếu thua ảnh sẻ bị mờ không nhìn rõ.

- Chức năng điều chỉnh kích thước màn hình game:
    - Game có 4 kích thước cửa sổ khác nhau để điều chỉnh cửa sổ game.
    - Các thành phần trong cửa sổ game tự điều chỉnh kích thước khi điều chỉnh màn hình.

- Chức năng tạo ra màn chơi dựa vào việc nhập các cấu hình của màn game.
    - Tính năng lưu tiến trình và dữ liệu game đã chơi được
    - Thêm xóa sửa các màn chơi đã có hoặc thêm mới
    - Thông tin người lập trình (2 thành viên)

2. Công nghệ:

- Nhận dạng và xử lý ảnh:

- Sử dụng thư viện PIL chỉnh sửa ảnh
- Pygame để xử lý ảnh, Surface của pygame được dùng để hiển thị.

- Lập trình:
  - Ngôn ngữ lập trình chính là python.
  - Thư viện Tkinter lập trình chương trình tạo màn chơi.
  - Thư viện pickle được dùng để lưu dữ liệu kiểu bytes và đọc dữ liệu game khi khởi động.

3. Giao diện:

- Thiết kế giao diện trực quan, dễ chơi.
- Thông tin được hiển thị rõ ràng, đầy đủ, dễ hiểu.
- Cho phép người dùng thao tác bằng chuột dễ dàng.
- Chương trình có nhiều cửa sổ để hiển thị từng chức năng riêng
- Có thể điều chỉnh kích thước màn hình

4. Khả năng mở rộng:

- Chương trình được viết theo mô hình hướng đối tượng có các tiêu chí: đóng gói, đa hình, kế thừa, định nghĩa biến toàn cục,…

5. Một số lưu ý:

- Đảm bảo tính năng của các game.
- Thử nghiệm và đánh giá ứng dụng trước khi triển khai.

II. Chi tiết về trò chơi ghép hình

1. Mô tả trò chơi

Game ghép hình là một trò chơi đơn giản, game được chia thành nhiều màn chơi, bạn sẽ nhận được một ảnh bất kì và game sẽ chia ảnh thành các ảnh nhỏ với kích thước 3x3 hoặc 4x4 tùy vào độ khó. Người chơi sử dụng chuột để chọn từng

mảnh trong một bảng và đưa các mảnh ghép vào bảng chính để hoàn thành một bức tranh hoàn chỉnh.

Mỗi màn chơi sẽ có các khoảng thời gian giới hạn để hoàn thành game nếu như không hoàn thành bức tranh trong thời gian đếm ngược sẽ thua cuộc. Nếu chiến thắng sẽ được chuyển đến màn tiếp theo.

Tính năng chính:

- Chọn mức độ khó của bức ảnh: Từng màn chơi sẽ có các mức độ khó khác nhau, ví dụ: chia ảnh thành 3x3, 3x4, 4x4, và tăng dần kích thước.
- Tính điểm: Người chơi phải ghép từng ảnh nhỏ thành một ảnh lớn hoàn chỉnh đúng kích thước và vị trí ảnh con để có thể hoàn thành được màn chơi theo đúng thời gian.
- Điều khiển đơn giản: Trò chơi sử dụng chuột điều khiển chủ yếu, rất dễ thao tác. Phù hợp với mọi đối tượng người chơi, từ trẻ em đến người lớn.

2. Chi tiết trò chơi

Mục đích của dự án là tạo ra một trò chơi đơn giản và giải trí, trong đó người chơi điều khiển chuột chọn các ảnh nhỏ để ghép thành một ảnh lớn trong thời gian quy định sẵn. Trò chơi này được thiết kế để cung cấp trải nghiệm giải trí và thử thách cho người chơi một cách dễ dàng và thú vị.

Giải thích chi tiết:

- Khởi tạo pygame trong một class App cùng với một số thành phần như tiêu để game, kích thước màn hình, một class Clock để thiết lập tốc độ khung hình trên một giây

```
class App:
  def __init__(self, title, screen_size = SCREEN_SIZE) -> None:
    os.environ['SDL_VIDEO_WINDOW_POS'] = "%d,%d" % (100,100)
    pg.init()
    self.running = False

    self.root = pg.display.set_mode(screen_size)
    self.title = title
    self.screen_size = screen_size
    pg.display.set_caption(self.title)


    self.clock = pg.time.Clock()
    self.controls = []

    self.menu = MenuScreen(self.screen_size)
    self.menu.event_menuclick = lambda obj: self.menu_click(obj)

    self.screen = self.menu
```

chạy chương trình trong hàm run().

```
app = App("Game xếp hình", level.current_size)
app.run()
```

- Khai báo biến toàn cục và các tài nguyên: Các biến toàn cục và các thư viện sẽ được khai báo tại một file riêng (trong chương trình là file setting.py) để đồng nhất các dữ liệu.

```python
import os
import pygame as pg
import random, cv2 as cv, time, pickle
from PIL import Image, ImageFilter
from tkinter.messagebox import *
#Định nghĩa các hằng số

SCREEN_SIZE = 640,360
DEFAULT_SIZE = 50,50
BUTTON_SIZE = 80,40
ZERO_P = 0.0,0.0
BLACK = 0,0,0
WHITE = 255,255,255
GRAY = 200,200,200
FPS_TICK=60
RED = 155,0,0
GREEN = 0,155,0
BLUE = 0,0,255
FONT_SIZE = 20
FONT_SVN_RETRON = "src/font/SVN-Retron 2000.otf"
FONT_PADDING_TOP = -4
TEXT_COLOR = 220,220,220
MENU_PADDING = 10,4
BOARD_SCALE = 0.45, 0.7
IMG_BOARD_DEMO = 0.1
```

từ sau đó chỉ cần import file và sử dụng các biến đã được định nghĩa mà không cần dài dòng

app.py > ...

```python
from package.setting import *

```

Tài nguyên hình ảnh và các thông tin của từng level game sẽ được lưu trong file game.data theo kiểu dữ liệu bytes và được đọc bằng cách sử dụng thư viện pickle và được xây dựng bằng hướng đối tượng.

```python
def read_level(self):
  # read game data
  try:
    with open('game.data', 'rb') as file:
      bien = pickle.load(file)
      self.index_level = bien.index_level
      self.current_size = bien.current_size
      self.current_level = bien.current_level
      self.game_data = bien.game_data
    print("Đọc dữ liệu game thành công")

  except Exception as ex:
    print("Đọc dữ liệu game thất bại", ex)
```

- Vòng lặp chính của trò chơi: Vòng lặp while sẽ chạy khi gọi hàm run().

Có 3 hàm chính sẽ được gọi đến:

```python
  self.menu = MenuScreen(self.screen_size)
  self.menu.event_menuclick = lambda obj: self.menu_click(obj)

  ##screen là kiểu biến cha có thể gán thành từ nhiều screen con khác
  #ví dụ screen được gán thành screen menu, screen game, screen config
  self.screen = self.menu

def event(self):···

def render(self):···

def update(self):···

def run(self):
  self.running = True

  while self.running:
    self.event()
    self.render()
    self.update()
```

  - Hàm event(): Hàm này sẽ lắng nghe các sự kiện như nhấn nút từ chuột hay nhấn phím trên bàn phím và được phát triển để lắng nghe các sự kiện như hover, click_down, click_up,… từ đó để các thành phần con bên trong class App có thể phát triển nhận được các sự kiện khi hover hay nhấn nút.

```
def event(self):
    for e in pg.event.get():
        if e.type == pg.QUIT:
            self.running = False
        elif e.type == pg.KEYDOWN:
            if e.key == pg.K_ESCAPE:
                self.running = False

        self.screen.event()
```

- Hàm update(): Hàm này sẽ thực hiện cập nhật các phần tử bên trong class App. Thiết lập chính các khung hình trên giây và cập nhật fps trên thanh tiêu đề. Cập nhật các phần tử con bên trong từng màn hình và bắt các sự kiện khi đã nhận từ hàm event(). Ví dụ khi screen đang là màn hình Menu thì thực hiện đổi màu nền của một nút trong menu thành xanh lá sẽ được thực hiện (vì đã nhận được sự kiện hover của chuột được bắt tại hàm event()).

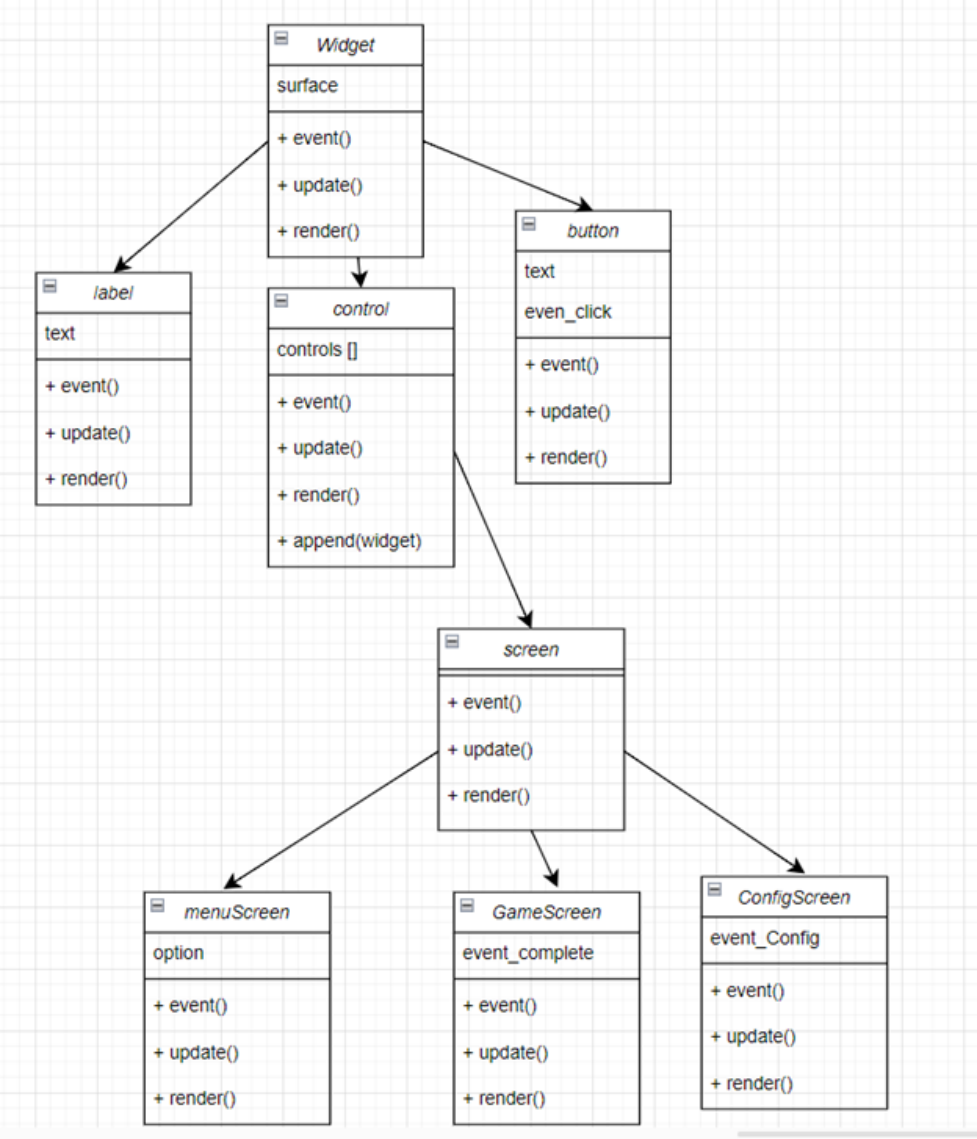
```
def update(self):
    self.screen.update()

    pg.display.flip()
    self.clock.tick(FPS_TICK)
    pg.display.set_caption(self.title + f" {self.clock.get_fps():.3f}")
```

- Hàm render(): Hàm này sẽ thực hiện fill thành màu đen trên từng khung hình. Và vẽ các màn hình con lên cửa sổ chính. Ví dụ: root là cửa sổ chính và hàm screen.render() sẽ vẽ các phần tử bên trong screen lên root từ đó đã có thể thấy được các thay đổi theo từng khung hình (ở đây là 60 khung hình, tức là trong một giây sẽ phải vẽ lại màn hình khoảng 60 lần).

```
def render(self):
    self.root.fill(BLACK)
    self.screen.render(self.root)
```

- Trò chơi được lập trình theo mô hình hướng đối tượng với các thành phần kế thừa như hình đơn giản bên dưới:



Mỗi một thành phần bên trong một màn hình sẽ có các sự kiện và kết hợp thao tác qua lại.

- Các hàm sự kiện và biểu thức lambda: Lambda Function còn được gọi là các hàm ẩn danh. Nó được gán cho một event bên ngoài một biến và gọi đến khi sự kiện bên trong biến đó được kích hoạt. Cụ thể, đoạn mã lambda obj: self.hoanthanh_level(obj) tạo ra một hàm nhận một đối số obj và gọi phương thức hoanthanh_level(obj). Lợi ích của việc sử dụng lambda trong trường hợp này là tạo ra một hàm nhanh chóng và ngắn gọn mà không cần định nghĩa một hàm riêng biệt. Điều này có thể hữu ích trong trường hợp chỉ cần sử dụng hàm này một lần duy nhất hoặc khi hàm không cần được tái sử dụng nhiều lần trong mã. Sau đó, hàm nặc danh này được gán cho thuộc tính sukien_hoanthanh_game của đối tượng game, để xử lý sự kiện hoàn thành game.

Trong Gamescreen event sự kiện khi hoàn thành game được khởi tạo bên trong class

```
class GameScreen(screen):
  def __init__(self, screen_size) -> None:
    super().__init__(screen_size)
    self.setlevel_game(screen_size)
    self.setposallcomponent(screen_size)

    self.img_select = None
    self.event_back_menu = None
    self.sukien_hoanthanh_game = None
```

Và được gán cho hàm khi sự kiện hoàn thành game bằng biểu thức lambda. thiết lập một sự kiện hoàn thành game. Khi sự kiện này xảy ra, phương thức hoanthanh_level được gọi với đối số tương ứng.

```
  game = GameScreen(self.screen_size)
  game.event_back_menu = lambda obj: self.backto_menu(obj)
  game.sukien_hoanthanh_game = lambda obj: self.hoanthanh_level(obj)

  self.screen = game
```

Khi hoàn thành game (thắng hoặc thua) game.sukien_hoanthanh_game sẽ được gọi đến để thông báo cho người dùng biết.

```
if self.is_complete():
    self.sukien_hoanthanh_game(True) if self.sukien_hoanthanh_game != None els
```

```
if self.time_limited <= 0:
    #kết thúc game
    self.sukien_hoanthanh_game(False) if self.sukien_hoanthanh_game != None
```

3. Chi tiết trò chơi và sử dụng

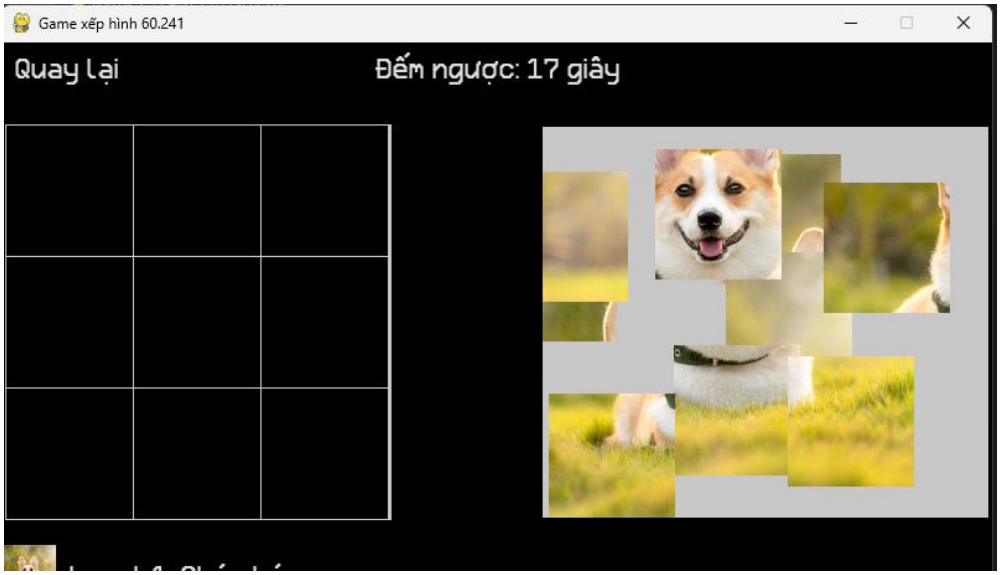
Điều kiện tiên quyết:

- Đọc kỹ file README.md trong thư mục game.
- Cài đặt python 3.9 trở lên để hạn chế lỗi.
- Cài đặt pygame: thư viện pygame được cài bằng pip: pip install pygame
- Cài đặt Pillow để xử lý ảnh với pip: pip install pillow
- Cài đặt pickle để đọc các tài nguyên với pip: pip install pickle
- Cài đặt tkinter để chạy chương trình cài đặt màn chơi: pip install tk

Cách chơi:

- Để bắt đầu trò chơi, nhấn bắt đầu. Màn hình game sẽ hiện ra.

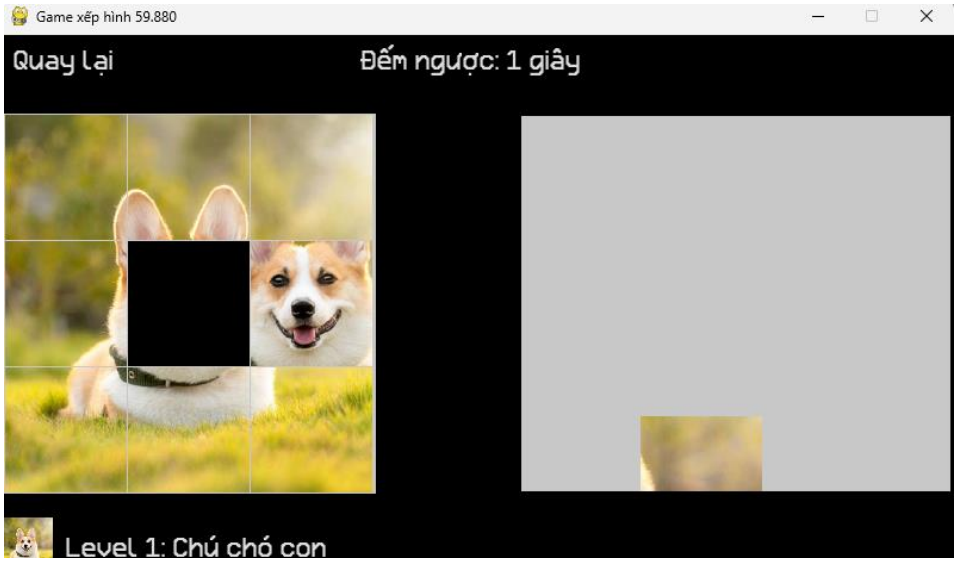
- Khi trò chơi bắt đầu, sẽ có 2 bảng. Bảng 1 chứa các gird, bảng 2 chứa các mảnh của một bức ảnh lớn.



- Người chơi dùng chuột chọn từng ảnh và ghép vào bảng bên trái.
- Phải hoàn thành trò chơi với thời gian quy định và tránh mất thời gian. Nếu hết thời gian mà chưa hoàn thành bạn sẽ thua cuộc.

- Khi trò chơi kết thúc, bức ảnh lớn sẽ bị mờ người chơi không nhìn rõ ràng được ảnh.
- Để chơi lại, dùng chuột nhấn vào chơi lại để thực hiện lại trò chơi.

- Nếu chọn sai vị trí người chơi có thể nhấn vào ảnh bị sai. Bức ảnh đó sẽ trở lại bảng bên phải hoặc chọn ảnh khác lấp vào ảnh bị sai, ảnh sai cũng sẽ xuất hiện lại bên bản bên phải



- Để thoát khỏi trò chơi, bạn phải quay lại menu và nhấn “Thoát” hoặc nhấn ‘X’ trên thanh tiêu đề.
- Khi hoàn thành game đúng giờ bạn sẽ chiến thắng, màn hình chiến thắng hiện ra nếu bạn muốn tiếp tục màn tiếp theo nhấn vào “Tiếp tục game”.

4. Chương trình chỉnh sửa màn chơi:

- Chương trình sử dụng file game.data để lưu các cài đặt và hình ảnh của game.
- Để tạo thêm game mới hãy nhấn "Tải ảnh", sau đó nhập thông tin như tên Level, số cột, số hàng, thời gian đếm ngược và nhấn "Thêm" sau đó "Lưu".
- Để sửa hoặc xóa ta chọn vào một level và nhập lại thông tin chọn ảnh giống với bước thêm và chọn "Sửa" để chỉnh sửa hoặc "Xóa" để loại bỏ level sau đó nhấn "Lưu" để lưu game. Sau đó vào trò chơi để tiếp tục chơi game.