

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO MÔN HỌC
PHÁT TRIỂN ỨNG DỤNG PHẦN MỀM
MÃ NGUỒN MỞ

ĐỀ TÀI

LẬP TRÌNH GAME ĐỐI KHÁNG

GV hướng dẫn: Thầy Từ Lăng Phiêu

Sinh viên thực hiện:

Họ và tên
Nguyễn Minh Tuấn

MSSV
3120410588

Liên hệ:

Email: minhhtuannguyen0504@gmail.com

Tp, Hồ Chí Minh, 2024

Mục lục

1	LỜI MỞ ĐẦU	2
2	Giới thiệu về đề tài	3
2.1	Lý do chọn đề tài	3
2.2	Mô tả đề tài	3
2.3	Đối tượng và phạm vi nghiên cứu	3
3	Cơ sở lý thuyết	4
4	Thiết kế ứng dụng	5
4.1	Mô tả thiết kế, ý nghĩa các bảng dữ liệu và các trường trong bảng (nếu có). Trình bày cấu trúc mã nguồn, mô hình ứng dụng, các tính năng được xây dựng, flowchart,...	5
4.2	Main.py	5
4.3	Class Fighter	8
5	Hiện thực	14
6	Cách thức cài đặt ứng dụng, môi trường chạy ứng dụng,	15
6.1	Cài đặt Python trên Widown hoặc Linux	15
6.2	Cài đặt Visual Studio Code	18
6.3	Mở source code dự án vào Visual Studio Code	18

1 LỜI MỞ ĐẦU

Hiện nay, ngành Công nghệ thông tin đang phát triển nhanh chóng và ứng dụng lan rộng trong mọi lĩnh vực. Công nghệ thông tin là một phần không thể thiếu, đóng vai trò quan trọng trong công cuộc công nghiệp hóa, hiện đại hóa và phát triển đất nước. Việc ứng dụng khoa học và công nghệ vào đời sống và công tác là vô cùng thiết yếu. Sự kết hợp giữa công nghệ thông tin và truyền thông là một yếu tố quan trọng trong hoạt động của các công ty và tổ chức, góp phần thay đổi suy nghĩ và lối tư duy của con người, giúp con người trở nên năng động và kết nối nhanh chóng ở bất kỳ đâu, từ đó tăng cường hiệu quả và năng suất làm việc.

Python là một ngôn ngữ lập trình hướng đối tượng cao cấp, được sử dụng để phát triển website và các ứng dụng đa dạng. Python được tạo ra bởi Guido van Rossum và phát triển trong dự án mã nguồn mở. Với cú pháp đơn giản và dễ hiểu, Python là lựa chọn hoàn hảo cho người mới học lập trình. Nó có thể được sử dụng để giải quyết các vấn đề về số học, xử lý văn bản, tạo game và nhiều công việc khác.

Trong quá trình tìm hiểu, em rất quan tâm đến các ứng dụng game được phát triển và lập trình bằng Python sử dụng thư viện Pygame. Pygame là một bộ công cụ tiện ích trong ngôn ngữ lập trình Python và đã tạo ra nhiều trò chơi huyền thoại từ thời ban đầu. Do đó, em đã quyết định sử dụng thư viện Pygame của Python để xây dựng trò chơi Fighting (game đối kháng).

2 Giới thiệu về đề tài

2.1 Lý do chọn đề tài

Trò chơi đối kháng là một đề tài phù hợp để áp dụng các kiến thức và kỹ năng lập trình. Vừa là cơ hội học hỏi về cách thiết kế game, đồ họa và âm nhạc.

2.2 Mô tả đề tài

Trong trò chơi đối kháng, hai người chơi sẽ sử dụng các phím trên bàn phím để điều khiển nhân vật của mình di chuyển và chiến đấu với nhau cho đến khi 1 trong 2 nhân vật bị hạ gục thì ván đấu đó sẽ kết thúc và bắt đầu một ván đấu mới.

2.3 Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: trò chơi đối kháng.

Phạm vi nghiên cứu:

- Cơ chế trò chơi
- Đồ họa trò chơi
- Âm thanh trong chơi

3 Cơ sở lý thuyết

- Thư viện đã sử dụng trong đề tài: Pygame

Ưu điểm:

1. Đơn giản và dễ học: Pygame cung cấp một cách tiếp cận dễ dàng cho việc phát triển trò chơi và ứng dụng đa phương tiện trên nền tảng Python.
2. Đa nền tảng: Pygame hoạt động trên nhiều hệ điều hành như Windows, MacOS, Linux, iOS và Android, tạo điều kiện thuận lợi cho việc phát triển ứng dụng đa nền tảng.
3. Hỗ trợ đa phương tiện: Pygame cung cấp các công cụ để xử lý âm thanh, đồ họa và đầu vào người dùng, giúp người dùng tạo ra các ứng dụng đa phương tiện phức tạp.
4. Cộng đồng mạnh mẽ: Có một cộng đồng nhiệt tình và sẵn sàng chia sẻ kiến thức, ví dụ như hướng dẫn, mã nguồn mẫu và phần mềm miễn phí.

Nhược điểm:

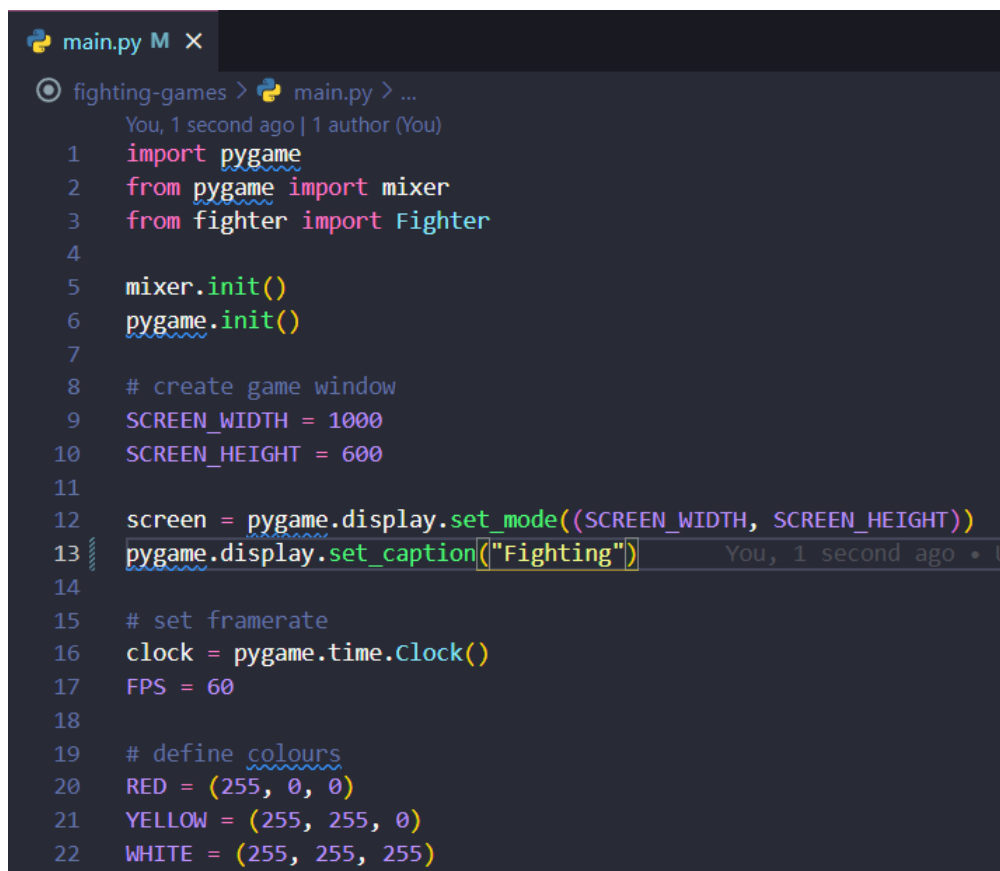
1. Hiệu suất hạn chế: Đối với các ứng dụng đa phương tiện phức tạp, Pygame có thể đạt đến giới hạn hiệu suất do sử dụng ngôn ngữ Python không tối ưu cho xử lý đồ họa và âm thanh.
2. Thiếu tính linh hoạt: So với các thư viện đồ họa và game engine mạnh mẽ khác, Pygame có thể thiếu các tính năng linh hoạt và công cụ phân tích mạnh mẽ hơn.
3. Hạn chế cho các trò chơi lớn quy mô: Pygame chỉ với thư viện cơ bản, có thể gặp hạn chế đối với việc phát triển các trò chơi lớn quy mô với yêu cầu cao về hiệu suất và tính năng.
4. Còn tiềm ẩn một số lỗi và rủi ro an ninh: Do sự phát triển không ngừng, Pygame vẫn có thể chứa đựng một số lỗi và rủi ro an ninh mà cần được cập nhật và vá sau này.

4 Thiết kế ứng dụng

4.1 Mô tả thiết kế, ý nghĩa các bảng dữ liệu và các trường trong bảng (nếu có). Trình bày cấu trúc mã nguồn, mô hình ứng dụng, các tính năng được xây dựng, flowchart,...

4.2 Main.py

- Đây là file chính trong cấu trúc thư mục dự án, nơi thực hiện các xử lý như:



```
main.py M X
fighting-games > main.py > ...
You, 1 second ago | 1 author (You)
1 import pygame
2 from pygame import mixer
3 from fighter import Fighter
4
5 mixer.init()
6 pygame.init()
7
8 # create game window
9 SCREEN_WIDTH = 1000
10 SCREEN_HEIGHT = 600
11
12 screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
13 pygame.display.set_caption("Fighting")
14
15 # set framerate
16 clock = pygame.time.Clock()
17 FPS = 60
18
19 # define colours
20 RED = (255, 0, 0)
21 YELLOW = (255, 255, 0)
22 WHITE = (255, 255, 255)
```

Khởi tạo màn hình game (thiết lập các width, height, fps, định nghĩa biến lưu mã màu)

```
# define fighter variables
WARRIOR_SIZE = 162
WARRIOR_SCALE = 4
WARRIOR_OFFSET = [72, 56]
WARRIOR_DATA = [WARRIOR_SIZE, WARRIOR_SCALE, WARRIOR_OFFSET]
WIZARD_SIZE = 250
WIZARD_SCALE = 3
WIZARD_OFFSET = [112, 107]
WIZARD_DATA = [WIZARD_SIZE, WIZARD_SCALE, WIZARD_OFFSET]
```

Định nghĩa các biến như vị trí nhân vật, kích thước hình ảnh

```
# load music and sounds
pygame.mixer.music.load("assets/audio/music.mp3")
pygame.mixer.music.set_volume(0.5)
pygame.mixer.music.play(-1, 0.0, 5000)
sword_fx = pygame.mixer.Sound("assets/audio/sword.wav")
sword_fx.set_volume(0.5)
magic_fx = pygame.mixer.Sound("assets/audio/magic.wav")
magic_fx.set_volume(0.75)
```

Đọc các file âm thanh tấn công, nhạc nền

```
# load background image
bg_image = pygame.image.load("assets/images/background/background.jpg").convert_alpha()

# load spritesheets
warrior_sheet = pygame.image.load(
    "assets/images/warrior/Sprites/warrior.png"
).convert_alpha()
wizard_sheet = pygame.image.load(
    "assets/images/wizard/Sprites/wizard.png"
).convert_alpha()

# load victory image
victory_img = pygame.image.load("assets/images/icons/victory.png").convert_alpha()
```

Load ảnh nền và hình ảnh nhân vật trong game

```
# function for drawing text
def draw_text(text, font, text_col, x, y):
    img = font.render(text, True, text_col)
    screen.blit(img, (x, y))

# function for drawing background
def draw_bg():
    scaled_bg = pygame.transform.scale(bg_image, (SCREEN_WIDTH, SCREEN_HEIGHT))
    screen.blit(scaled_bg, (0, 0))

# function for drawing fighter health bars
def draw_health_bar(health, x, y):
    ratio = health / 100
    pygame.draw.rect(screen, WHITE, (x - 2, y - 2, 404, 34))
    pygame.draw.rect(screen, RED, (x, y, 400, 30))
    pygame.draw.rect(screen, YELLOW, (x, y, 400 * ratio, 30))
```

Xây dựng các hàm tạo background, text và thanh progress tượng trưng cho lượng máu của nhân vật.

```
# create two instances of fighters
fighter_1 = Fighter(
    1, 200, 310, False, WARRIOR_DATA, warrior_sheet, WARRIOR_ANIMATION_STEPS, sword_fx
)
fighter_2 = Fighter(
    2, 700, 310, True, WIZARD_DATA, wizard_sheet, WIZARD_ANIMATION_STEPS, magic_fx
)
```

Khởi tạo đối tượng Fighter từ class Fighter và truyền vào các đối số cần thiết.

4.3 Class Fighter

- Class đại diện cho đối tượng là nhân vật trong game, bao gồm:

```
main.py M  fighter.py M X
fighting-games > fighter.py > Fighter > __init__
You, 1 second ago | 1 author (You)
1 import pygame
You, 1 second ago | 1 author (You)
2 class Fighter:
3     def __init__(self, player, x, y, flip, data, sprite_sheet, animation_steps, sound):
4         self.player = player
5         self.size = data[0]
6         self.image_scale = data[1]
7         self.offset = data[2] You, 23 hours ago • update load fighter image
8         self.flip = flip
9         self.animation_list = self.load_images(sprite_sheet, animation_steps)
10        self.action = 0
11        self.frame_index = 0
12        self.image = self.animation_list[self.action][self.frame_index]
13        self.update_time = pygame.time.get_ticks()
14        self.rect = pygame.Rect((x, y, 80, 180))
15        self.vel_y = 0
16        self.running = False
17        self.jump = False
18        self.attacking = False
19        self.attack_type = 0
20        self.attack_cooldown = 0
21        self.attack_sound = sound
22        self.hit = False
23        self.health = 100
24        self.alive = True
```

Hàm khởi tạo có tham số

```
def load_images(self, sprite_sheet, animation_steps):  
    # extract images from sprite_sheet  
    animation_list = []  
    for y, animation in enumerate(animation_steps):  
        temp_img_list = []  
        for x in range(animation):  
            temp_img = sprite_sheet.subsurface(  
                x * self.size, y * self.size, self.size, self.size  
            )  
            temp_img_list.append(  
                pygame.transform.scale(  
                    temp_img,  
                    (self.size * self.image_scale, self.size * self.image_scale),  
                )  
            )  
        animation_list.append(temp_img_list)  
    return animation_list
```

Hàm load hình ảnh của nhân vật tương ứng với hành động hiện tại mà người chơi đang điều khiển

```
def move(self, screen_width, screen_height, surface, target, round_over):
    SPEED = 10
    GRAVITY = 2
    dx = 0
    dy = 0
    self.running = False
    self.attack_type = 0

    # get keypresses
    key = pygame.key.get_pressed()

    # can only perform other actions if not currently attacking
    if self.attacking == False and self.alive == True and round_over == False:
        # check player 1 controls
        if self.player == 1:
            # movement
            if key[pygame.K_a]:
                dx = -SPEED
                self.running = True
            if key[pygame.K_d]:
                dx = SPEED
                self.running = True
            # jump
            if key[pygame.K_w] and self.jump == False:
                self.vel_y = -30
                self.jump = True
            # attack
            if key[pygame.K_j] or key[pygame.K_k]:
                self.attack(target)
                # determine which attack type was used
                if key[pygame.K_j]:
                    self.attack_type = 1
                if key[pygame.K_k]:
                    self.attack_type = 2
```

```
# check player 2 controls
if self.player == 2:
    # movement
    if key[pygame.K_LEFT]:
        dx = -SPEED
        self.running = True
    if key[pygame.K_RIGHT]:
        dx = SPEED
        self.running = True
    # jump
    if key[pygame.K_UP] and self.jump == False:
        self.vel_y = -30
        self.jump = True
    # attack
    if key[pygame.K_KP1] or key[pygame.K_KP2]:
        self.attack(target)
        # determine which attack type was used
        if key[pygame.K_KP1]:
            self.attack_type = 1
        if key[pygame.K_KP2]:
            self.attack_type = 2

# apply gravity
self.vel_y += GRAVITY
dy += self.vel_y

# ensure player stays on screen
if self.rect.left + dx < 0:
    dx = -self.rect.left
if self.rect.right + dx > screen_width:
    dx = screen_width - self.rect.right
if self.rect.bottom + dy > screen_height - 110:
    self.vel_y = 0
```

Hàm quy định các phím có thể điều khiển nhân vật, mỗi phím là một hành động khác và xử lý va chạm.

```
# handle animation updates
def update(self):
    # check what action the player is performing
    if self.health <= 0:
        self.health = 0
        self.alive = False
        self.update_action(6) # 6:death
    elif self.hit == True:
        self.update_action(5) # 5:hit
    elif self.attacking == True:
        if self.attack_type == 1:
            self.update_action(3) # 3:attack1
        elif self.attack_type == 2:
            self.update_action(4) # 4:attack2
    elif self.jump == True:
        self.update_action(2) # 2:jump
    elif self.running == True:
        self.update_action(1) # 1:run
    else:
        self.update_action(0) # 0:idle

    animation_cooldown = 50
    # update image
    self.image = self.animation_list[self.action][self.frame_index]
    # check if enough time has passed since the last update
    if pygame.time.get_ticks() - self.update_time > animation_cooldown:
        self.frame_index += 1
        self.update_time = pygame.time.get_ticks()
    # check if the animation has finished
    if self.frame_index >= len(self.animation_list[self.action]):
        # if the player is dead then end the animation
        if self.alive == False:
            self.frame_index = len(self.animation_list[self.action]) - 1
```

Cập nhật hình ảnh tương ứng với hành động hiện tại

```
def attack(self, target):
    if self.attack_cooldown == 0:
        # execute attack
        self.attacking = True
        self.attack_sound.play()
        attacking_rect = pygame.Rect(
            self.rect.centerx - (2 * self.rect.width * self.flip),
            self.rect.y,
            2 * self.rect.width,
            self.rect.height,
        )
        if attacking_rect.colliderect(target.rect):
            target.health -= 10
            target.hit = True

def update_action(self, new_action):
    # check if the new action is different to the previous one
    if new_action != self.action:
        self.action = new_action
        # update the animation settings
        self.frame_index = 0
        self.update_time = pygame.time.get_ticks()

def draw(self, surface):
    img = pygame.transform.flip(self.image, self.flip, False)
    surface.blit(
        img,
        (
            self.rect.x - (self.offset[0] * self.image_scale),
            self.rect.y - (self.offset[1] * self.image_scale),
        ),
    )
```

Các hàm cập nhật hành động hiện tại của nhân vật, tạo rectagle ảo để kiểm tra va chạm và xử lý tấn công.

5 Hiện thực

- Giao diện của trò chơi



- Hướng dẫn cách chơi:

+ Player 1: Phím A (lùi), phím D (tiến), phím W (nhảy), J hoặc K để tấn công.

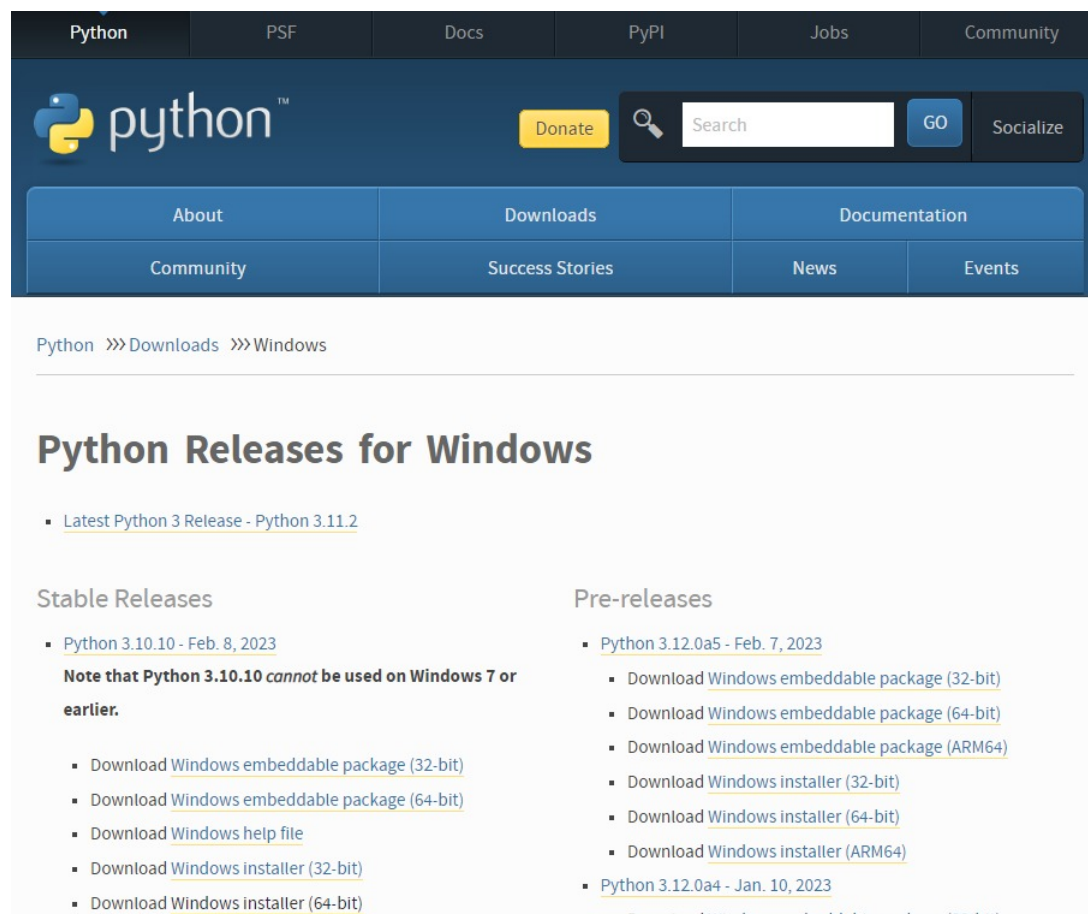
+ Player 2: Hướng điều khiển di chuyển tương ứng với các phím mũi tên trên bàn phím, nhấn số 1 hoặc 2 để tấn công.

6 Cách thức cài đặt ứng dụng, môi trường chạy ứng dụng,

6.1 Cài đặt Python trên Windows hoặc Linux

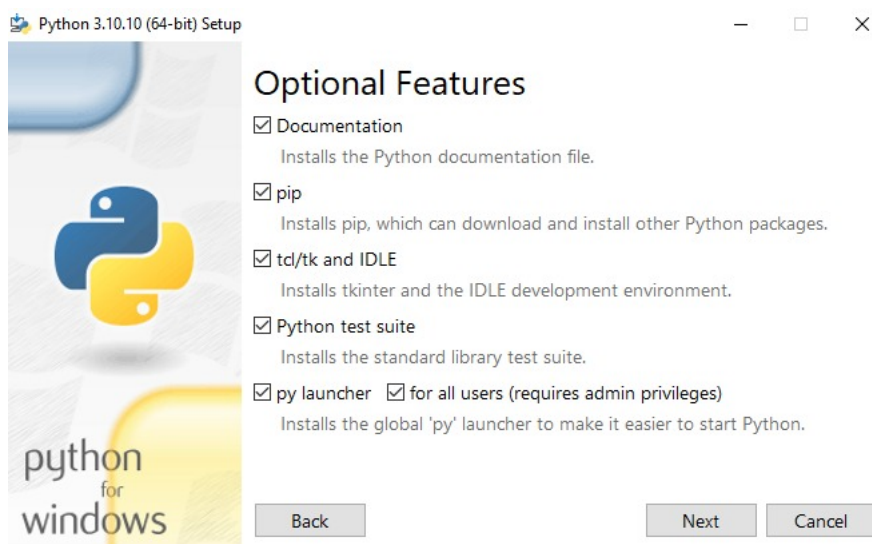
Bước 1: Vào trang chủ python chọn phiên bản cần và tải xuống trình cài đặt

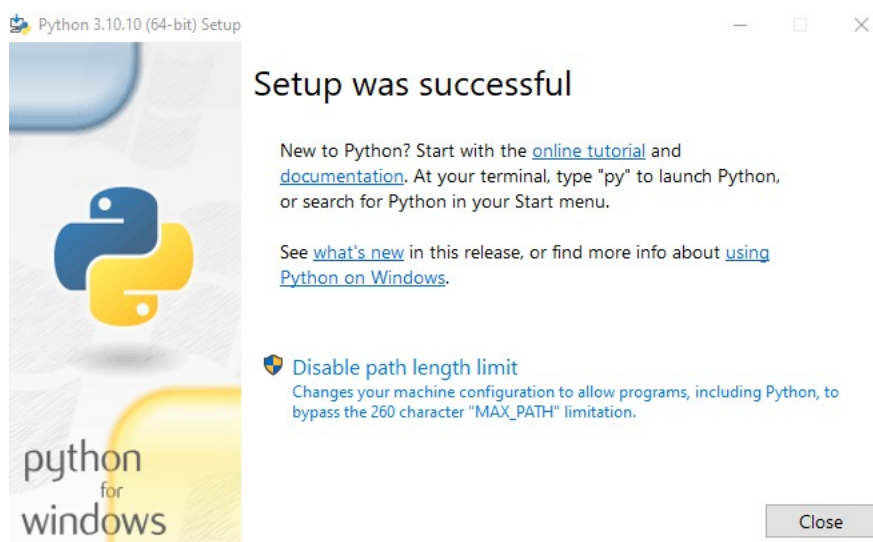
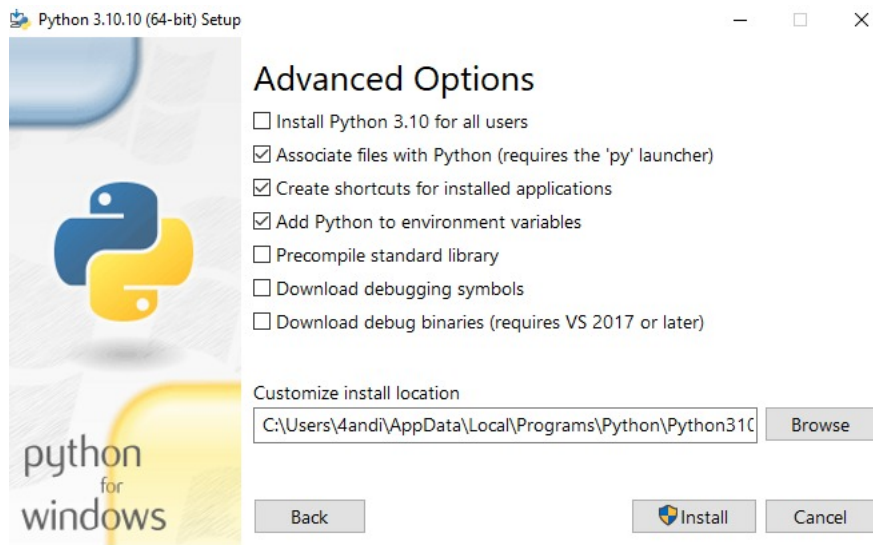
- <https://www.python.org/downloads/>



The screenshot shows the Python.org website. The top navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation bar is a search bar and a 'Donate' button. The main content area is titled 'Python Releases for Windows'. It lists the latest Python 3 release as Python 3.11.2. Under 'Stable Releases', it shows Python 3.10.10 from Feb. 8, 2023, with a note that it cannot be used on Windows 7 or earlier. It provides download links for Windows embeddable packages (32-bit and 64-bit), Windows help file, and Windows installers (32-bit and 64-bit). Under 'Pre-releases', it shows Python 3.12.0a5 from Feb. 7, 2023, and Python 3.12.0a4 from Jan. 10, 2023, with download links for Windows embeddable packages and installers.

Bước 2: Chạy trình cài đặt thực thi





Bước 3: Thêm Python vào Biến môi trường

Bước 1: Đi tới **Start** và nhập advanced system settings vào thanh tìm kiếm.

Bước 2: Nhấp vào **View advanced system settings**.

Bước 3: **In the System Properties dialog box**, nhấp vào **Advanced** và sau đó nhấp vào **Environment Variables**.

Bước 4: Tùy thuộc vào cài đặt của bạn:

- Nếu bạn đã chọn **Install for all users** trong khi cài đặt, hãy chọn **Path from the System Variables** và nhấp **Edit** .

- Nếu bạn không chọn **Settings for all users** trong khi cài đặt, hãy chọn **Path from the System Variables** và nhấp **Edit** .

Bước 5: Bấm vào **New** và nhập đường dẫn thư mục Python, sau đó bấm **OK** cho đến khi tất cả các hộp thoại được đóng lại.

Bước 4: Kiểm tra Python đã cài đặt chưa bằng lệnh

Đi tới **Start** và nhập **cmd** vào thanh tìm kiếm. Bấm vào **cmd**.

Gõ và lệnh: `python --version`

Nếu thấy hiện version hiện tại của python thì đã cài đặt thành công.

6.2 Cài đặt Visual Studio Code

- Truy cập trang chủ của vscode, chọn tải về cho hệ điều hành tương ứng và tiến hành cài đặt.

- <https://code.visualstudio.com/download>

6.3 Mở source code dự án vào Visual Studio Code

Bước 1: Tải thư viện pygame vào terminal gõ lệnh **pip install pygame**

Bước 2: Mở project trong VSCode

Bước 3: Run file main và bắt đầu chơi game



Tài liệu

- [1] Source “link: <https://www.youtube.com/watch?v=s5bd9KMSSW4>”