|  |
| --- |
| HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG |
|  |
| *Báo cáo bài tập lớn*  **Ngôn ngữ lập trình PYTHON** |
| Sinh viên thực hiện:  Nguyễn Huy Hoàng – B19DCCN278  Đỗ Mai Ly – B19DCCN413  Nguyễn Thế Điệp – B19DCCN182 |
| HÀ NỘI, THÁNG 11/2021 |

***Mục lục***

***Phần 1: Giới thiệu và nội dung về game Dinosaur***

***Phần 2: Các chức năng và cách điều khiển nhân vật trong game***

***Phần 3: Thực hành tạo ra game***

***+)Part 1: Cài đặt và hiệu ứng của khủng long***

***.)* 1.Thiết lập pygame và những hình ảnh Dino cần thiết**

***.)* 2. Cài đặt thư viện và hiệu ứng hình ảnh của khủng long**

***+)Part 2: Xây dựng hàm Duck() Jump() Run() trong class Dinosaur.***

***+) Part 3: Thêm Background và Score.***

***+) Part 4: Tạo lớp Obstacle(lớp tạo các chướng ngại vật)***

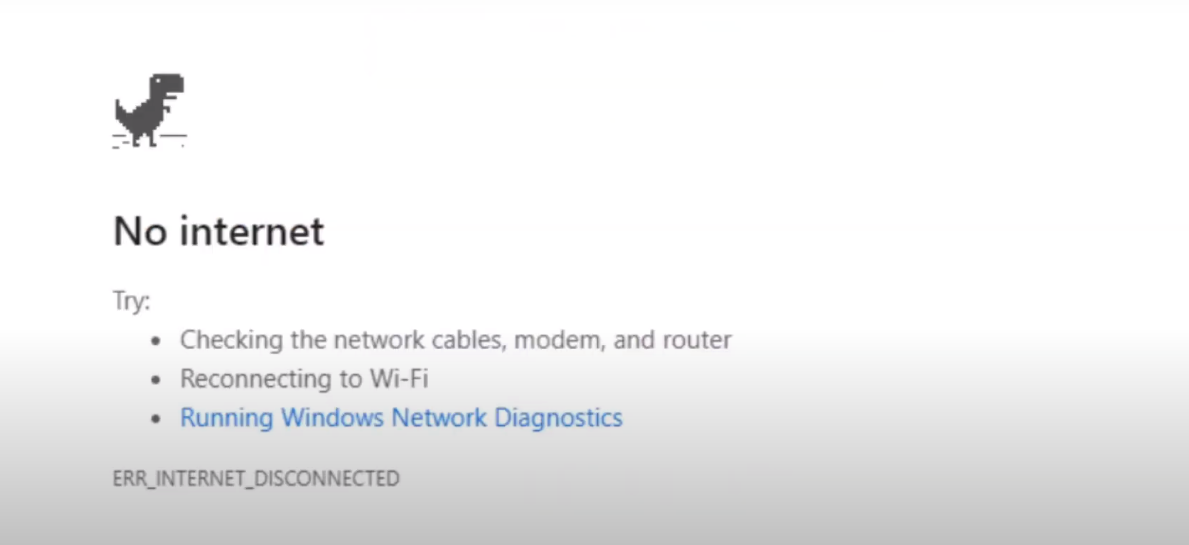
***+) Part 5: Tạo lớp vẽ xương rồng nhỏ và xương rồng lớn***

***+)* Part 6: Tạo lớp vẽ Bird có hoạt ảnh bay qua màn hình**

**+)*Part 7: Tạo menu sau khi death***

***Chủ đề: Lập trình game Dinosaur***

***Phần 1: Giới thiệu và nội dung về game Dinosaur***



***Phần 2: Các chức năng và cách điều khiển nhân vật trong game***

* Người chơi điều khiển nhân vật qua các phím
* Nhấn phím hoặc Space để nhảy
* Nhấn phím để cúi
* Nhấn phím từ 1 -> 8 để thay đổi các style của khủng long

***Phần 3: Thực hành tạo ra game***

***Part 1: Cài đặt và hiệu ứng của khủng long***

**1.Thiết lập pygame và những hình ảnh Dino cần thiết**

- Game Dinosaur gồm 4 đối tượng chính bao gồm cả nhân vật, các chướng ngại vật và hình ảnh trang trí đó là: Khủng long Dino, Chim, Cây xương rồng và mây.

\* Khủng long Dino

- Khủng long đứng



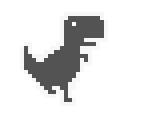
- Khủng long cúi

- Khủng long nhảy



- Khủng long chạy

- Khủng long khi bắt đầu



-

\* Chim (Chướng ngại vật)

\*Cây xương rồng(Chướng ngại vật)

\*Các hình ảnh khác

+) Mây : 

+) Track:



**2. Cài đặt thư viện và hiệu ứng hình ảnh của khủng long**

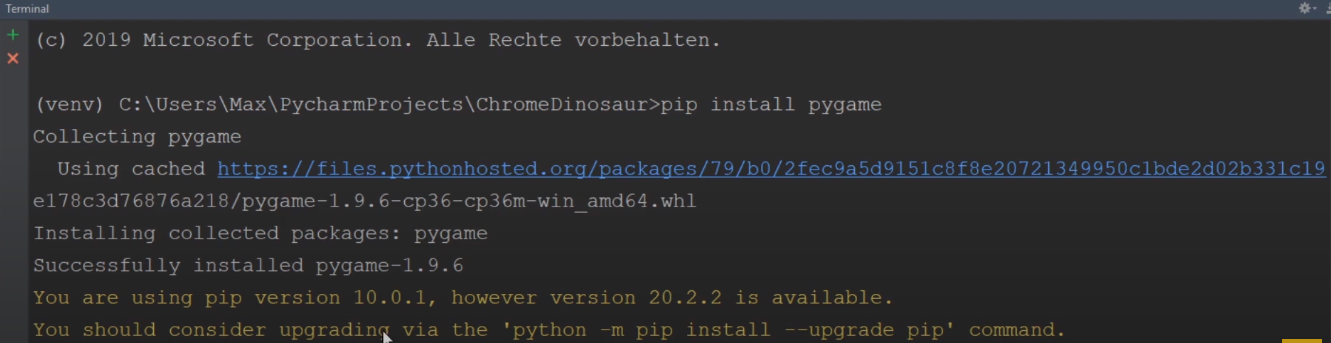
Bước 1. Cần đưa các hình ảnh đã thu thập được vào một mục riêng có tên Assets.

Bên trong assets có các folder riêng là Bird, Dino, Catus và Other

Tạo new project trong VS code và kéo thả folder Asset vào project vừa tạo

Bước 2. Cài đặt thư viện pygame

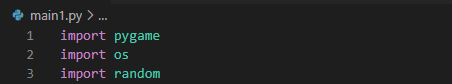
- Mở terminal (Windown + R) sau đó ấn “pip install pygame” như hình ảnh dưới đây.



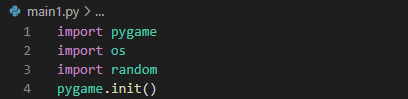
Bước 3. Bắt đầu làm game

- Tạo một python file trong project vừa tạo đặt tên là main.py

- Khai báo các thư viện.

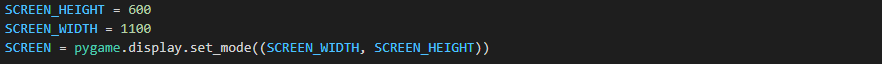


-Khởi tạo trò chơi



+) Dòng pygame.init() sẽ khởi tạo các thiết lập cài đặt cho màn hình nền mà Pygame cần để hoạt động bình thường.

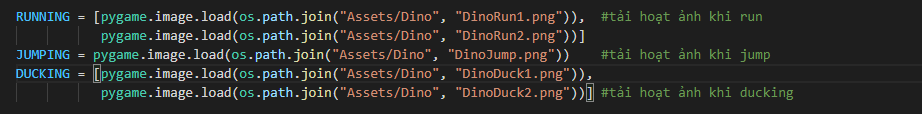
- Tạo cửa sổ màn hình cho trò chơi



+) Bằng cách truyền hai biến SCREEN\_HEIGHT và SCREEN\_WIDTH vào hàm pygame.display.set\_mode(), ta có thể tạo một cửa sổ có chiều rộng là 1100 pixel vào chiều cao là 600 pixel (Bạn có thể điều chỉnh số liệu khác sao cho phù hợp với màn hình máy tính cá nhân của bạn)

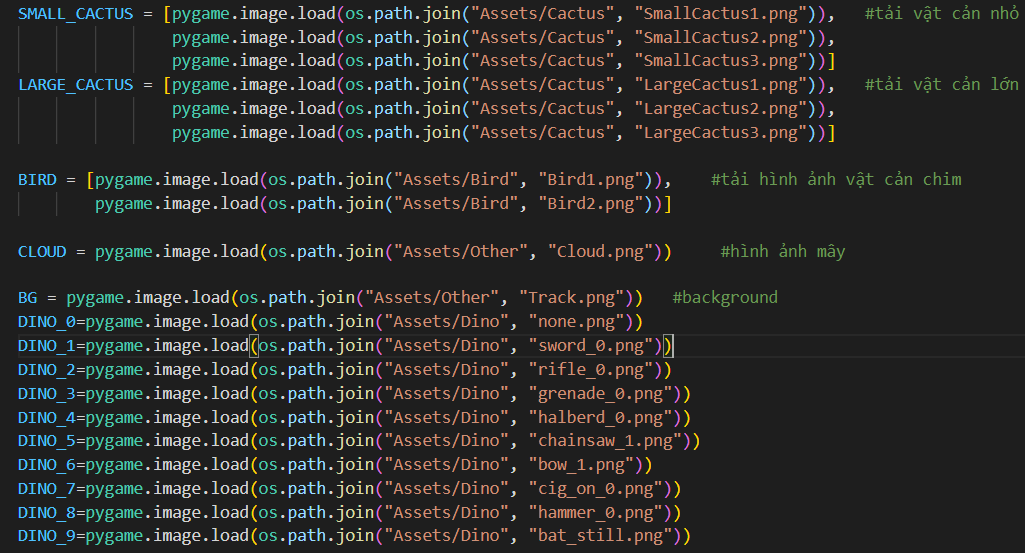
- Tải tất cả các hình ảnh đã chuẩn bị để xây dựng trò chơi

+) Tạo các list chứa tất cả các hình ảnh khủng long đang hoạt động trong trò chơi(chạy, cúi, nhảy, … )



+) Để tải hình ảnh, ta sẽ gọi hàm **pygame.image.load()**, thêm đường dẫn làm đối số. Ví dụ như hình ảnh trên có nghĩa là Khủng long đang nằm trong thư mục có tên là Dino và lần lượt nằm trong thư mục có tên là Assets

- Tương tự như trên thì tiếp tục tạo các list lưu trữ các hình ảnh cần thiết khác ở trong game(Cây xương rồng lớn, nhỏ; chim và những đám mây và cả hình nền của game)



- Trò chơi được điều khiển bởi vòng lặp while, chứa vòng lặp các sự kiện và đoạn mã quản lý các lần thay đổi cập nhật cho màn hình. Sự kiện là một hành động mà người dùng thực hiện trong khi chơi, chẳng hạn như nhấn phím hoặc di chuyển chuột. Để làm cho chương trình phản hồi các sự kiện, ta sẽ viết một vòng lặp sự kiện để lắng nghe từng sự kiện và thực hiện một tác vụ tương ứng tùy thuộc vào kiểu sự kiện đã xảy ra.

-Tạo ra hàm main()

+) Tạo biến run làm công tắc đóng mở cho vòng lặp while

+) Tạo biến clock để làm đồng hồ thời gian cho trò chơi

 run = True  #công tắc cho vòng lặp

 clock = pygame.time.Clock()  #đồng hồ cho trò chơi

+) Tạo vòng lặp game để bắt các sự kiện

while run:

        for event in pygame.event.get():                 #dấu X thoát trò chơi

            if event.type == pygame.QUIT:      #Khi người chơi click vào dấu x vòng lặp sẽ dừng

                run = False

+) Khởi tạo nhân vật khủng long trong hàm main là 1 đối tượng trong lớp Dinosaur(class này chúng ta sẽ tạo ngay sau đó)

 player = Dinosaur()

+) Trong vòng lặp game ta sẽ vẽ lên màn hình khủng long và nền màn hình màu trắng,tạo biến để lưu lại các input đầu vào từ người chơi

userInput = pygame.key.get\_pressed()   #biến đầu vào người dùng

        player.draw(SCREEN)           #vẽ lên màn hình

        player.update(userInput #cập nhật khủng long với mỗi sự kiện tương tác của người chơi

player.draw(SCREEN)             #vẽ khủng long lên màn hình

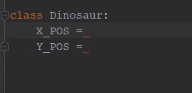
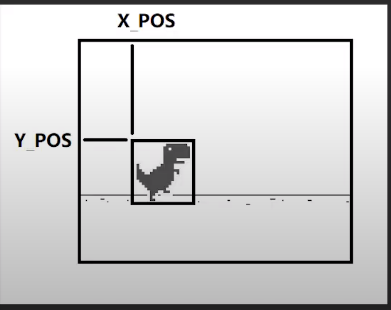
        player.update(userInput)          #cập nhật khủng long với mỗi sự kiện tương tác của người chơi

clock.tick(30)       #cài đặt fps

        pygame.display.update()    #đưa lên màn hình

**2)Tạo ra class Dinosaur:**

+)tạo biến xác định tọa độ của khủng long

+) Tạo hàm init() để tạo ra khủng long bất cứ khi nào một đối tượng của lớp này được tạo, phương thức init() trước tiên bao gồm tất cả các hình ảnh của khủng long đang chạy, nhảy hay cúi người.

def \_\_init\_\_(self):        #khởi tạo khủng long bất cứ khi nào nó được tạo lại

       self.duck\_img = DUCKING         #hình ảnh khủng long cúi

       self.run\_img = RUNNING            #hình ảnh khủng long chạy

       self.jump\_img = JUMPING           #hình ảnh khủng long nhảy

       self.dino\_duck = False

       self.dino\_run = True                #cài đặt mặc định khủng long running

       self.dino\_jump = False

       self.dino\_style = DINO\_0

       self.step\_index = 0                  #chỉ số bước chân = 0

       self.image = self.run\_img[0]               #hình ảnh đầu tiên khi con khủng long được chạy

       self.dino\_rect = self.image.get\_rect()      #hình chữ nhật làm khung cảm ứng va chạm cho hình ảnh khủng long

       self.dino\_rect.x = self.X\_POS         #tọa độ x và y của hình chữ nhật = tọa độ khủng long

       self.dino\_rect.y = self.Y\_POS

+) Tạo hàm update() để cập nhật khủng long qua mỗi vòng lặp game và kiểm tra trạng thái của khủng long, với tham số truyền vào là input từ người chơi

def update(self, userInput): #hàm cập nhật khủng long mỗi lần lặp lại vòng while

        if self.dino\_duck:          #Kiểm tra trạng thái khủng long

            self.duck()

        if self.dino\_run:

            self.run()

        if self.dino\_jump:

            self.jump()

        if self.step\_index >= 10:   #chỉ số bước mỗi 10 bước thì reset về 0

            self.step\_index = 0

        if (userInput[pygame.K\_UP] or userInput[pygame.K\_SPACE]) and not self.dino\_jump:    #khi bấm up hoặc Space mà khủng long không jump thì khủng long jump

            self.dino\_duck = False

            self.dino\_run = False

            self.dino\_jump = True

        elif userInput[pygame.K\_DOWN] and not self.dino\_jump:   #khi bấm down mà khủng long không duck thì khủng long duck

            self.dino\_duck = True

            self.dino\_run = False

            self.dino\_jump = False

        elif not (self.dino\_jump or userInput[pygame.K\_DOWN]):  #nếu không jump hay duck thì run

            self.dino\_duck = False

            self.dino\_run = True

            self.dino\_jump = False

        if userInput[pygame.K\_1]: self.dino\_style = DINO\_1  # Nếu người chơi bấm phím 1 khủng long sẽ thay đổi style thành DINO1

        if userInput[pygame.K\_2]: self.dino\_style = DINO\_2  # Nếu người chơi bấm phím 2 khủng long sẽ thay đổi style thành DINO2

        if userInput[pygame.K\_3]: self.dino\_style = DINO\_3  # Nếu người chơi bấm phím 3 khủng long sẽ thay đổi style thành DINO3

        if userInput[pygame.K\_4]: self.dino\_style = DINO\_4  # Nếu người chơi bấm phím 4 khủng long sẽ thay đổi style thành DINO4

        if userInput[pygame.K\_5]: self.dino\_style = DINO\_5  # Nếu người chơi bấm phím 5 khủng long sẽ thay đổi style thành DINO5

        if userInput[pygame.K\_6]: self.dino\_style = DINO\_6  # Nếu người chơi bấm phím 6 khủng long sẽ thay đổi style thành DINO6

        if userInput[pygame.K\_7]: self.dino\_style = DINO\_7  # Nếu người chơi bấm phím 7 khủng long sẽ thay đổi style thành DINO7

        if userInput[pygame.K\_8]: self.dino\_style = DINO\_8  # Nếu người chơi bấm phím 8 khủng long sẽ thay đổi style thành DINO8

+) Tạo hàm draw() để vẽ khủng long và các style của khủng long lên màn hình game

 def draw(self, SCREEN):          #hàm vẽ lên màn hình

        SCREEN.blit(self.image, (self.dino\_rect.x, self.dino\_rect.y))  #Vẽ hình ảnh của khủng long lên màn hình game

        if(self.dino\_style == DINO\_1):

            SCREEN.blit(self.dino\_style, (self.dino\_rect.x-30, self.dino\_rect.y-180)) #dino 1

        elif(self.dino\_style == DINO\_2):

            SCREEN.blit(self.dino\_style, (self.dino\_rect.x-20, self.dino\_rect.y-180 )) #dino 2

        elif(self.dino\_style == DINO\_3):

            SCREEN.blit(self.dino\_style, (self.dino\_rect.x-40, self.dino\_rect.y-180 )) #dino 3

        elif(self.dino\_style == DINO\_4):

            SCREEN.blit(self.dino\_style, (self.dino\_rect.x-40, self.dino\_rect.y-180 )) #dino 4

        elif(self.dino\_style == DINO\_5):

            SCREEN.blit(self.dino\_style, (self.dino\_rect.x-40, self.dino\_rect.y-170 )) #dino 5

        elif(self.dino\_style == DINO\_6):

            SCREEN.blit(self.dino\_style, (self.dino\_rect.x-20, self.dino\_rect.y-170 )) #dino 6

        elif(self.dino\_style == DINO\_7):

            SCREEN.blit(self.dino\_style, (self.dino\_rect.x-25, self.dino\_rect.y-160 )) #dino 7

        elif(self.dino\_style == DINO\_8):

            SCREEN.blit(self.dino\_style, (self.dino\_rect.x-45, self.dino\_rect.y-170 )) #dino 8

        elif(self.dino\_style == DINO\_9):

            SCREEN.blit(self.dino\_style, (self.dino\_rect.x-30, self.dino\_rect.y-140 )) #dino 9

        else:

            SCREEN.blit(self.dino\_style, (self.dino\_rect.x-45, self.dino\_rect.y-170 )) #dino 0

Và đây là kết quả sau các bước mà chúng ra đã làm ở trên khi run code:

******

***Part 2: Xây dựng hàm Duck() Jump() Run() trong class Dinosaur.***

1. **Hàm Run()**

def run(self):

        self.image = self.run\_img[self.step\_index // 5] #hình ảnh tương ứng của khủng long(khi số bước chân tứ 0 - 5 hình ảnh 1 hiển thị , từ 5 -10 hình ảnh 2 hiển thị vượt quá 10 chỉ số được đặt lại)

        self.dino\_rect = self.image.get\_rect()       #tọa độ hình chữ nhật của hình ảnh khủng long

        self.dino\_rect.x = self.X\_POS            #đặt hình chữ nhật vào vị trí khủng long hiển thị

        self.dino\_rect.y = self.Y\_POS

        self.step\_index += 1                      #tăng chỉ số bước chân lên 1

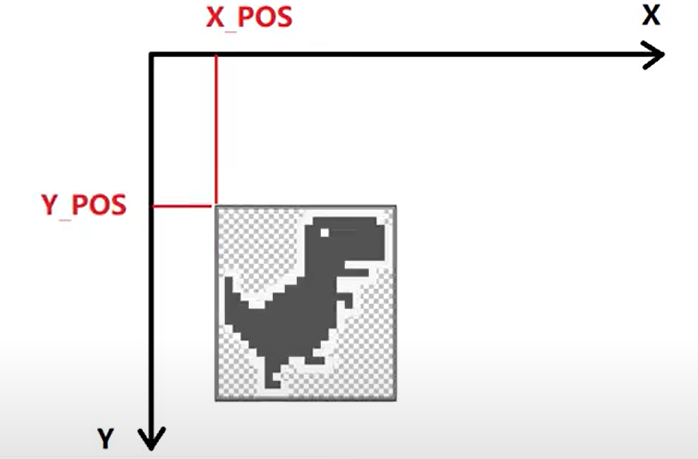
1. **Hàm Duck()**

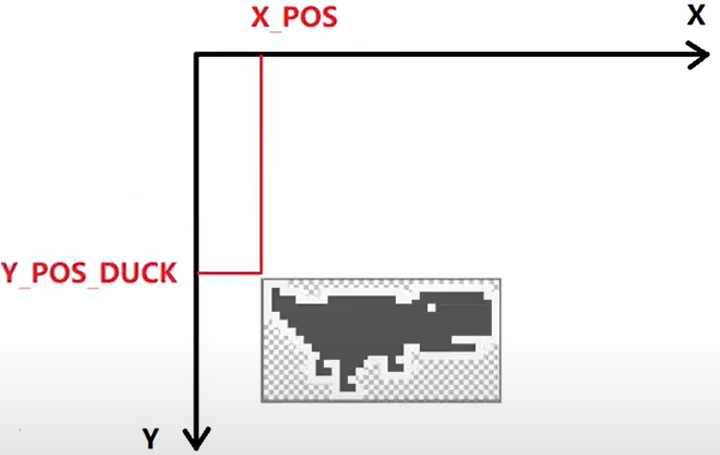
* Ta cần thêm 1 hằng số trong lớp dinosaur là :

Y\_POS\_DUCK = 340 #tọa độ Y khi khủng long duck

Vì khi duck tọa độ Y của khủng long từ 310 - > 340

*Khi khủng long run: Khi khủng long duck:*





- Xây dựng hàm Duck():

def duck(self):

        self.image = self.duck\_img[self.step\_index // 5]   #hình ảnh tương ứng của khủng long(khi số bước chân tứ 0 - 5 hình ảnh 1 hiển thị , từ 5 -10 hình ảnh 2 hiển thị vượt quá 10 chỉ số được đặt lại)

        self.dino\_rect = self.image.get\_rect() #tọa độ hình chữ nhật của hình ảnh khủng long

        self.dino\_rect.x = self.X\_POS  #đặt hình chữ nhật vào vị trí khủng long hiển thị

        self.dino\_rect.y = self.Y\_POS\_DUCK x

        self.step\_index += 1      #tăng chỉ số bước chân lên 1

1. **Hàm Jump()**

* Tương tự như hàm Duck() ta cũng cần thêm 1 hằng số trong lớp dinosaur() là :

JUMP\_VEL = 8.5   #tọa độ Y thay đổi khi khủng long jump

self.jump\_vel = self.JUMP\_VEL   #khởi tạo thuộc tính jump\_vel trong hàm init bằng 1 hằng số JUMP\_VEL mặc định

* Hàm jump()

def jump(self):

        self.image = self.jump\_img      #thay đổi hình ảnh của khủng long

        if self.dino\_jump:     #nếu trạng thái nhảy của khủng long được kích hoạt

            self.dino\_rect.y -= self.jump\_vel \* 4 #tọa độ y của khủng long sẽ thay đổi dựa vào chỉ số jump\_vel

            self.jump\_vel -= 0.8    #Giảm dần chỉ số jump\_vel để hạ độ cao dần

        if self.jump\_vel < - self.JUMP\_VEL: #Khi khủng long trở về vị trí ban đầu trước lúc jump

            self.dino\_jump = False         # trạng thái Jump được tắt

            self.jump\_vel = self.JUMP\_VEL   #cài đặt lại thuộc tính jump\_vel về ban đầu

***Part 3: Thêm Background và Score.***

* Để bắt đầu ta cần import module random bằng câu lệnh: import random
* Trong hàm Main() ta cần khởi tạo 1 biến toàn cục có tên là game\_speed (Tốc độ của game)

global game\_speed = 14

1. **Xây dựng class Cloud**:

class Cloud:             #Mây

    def \_\_init\_\_(self):

        self.x = SCREEN\_WIDTH + random.randint(800, 1000)  #vị trí x của mây = độ rộng của màn hình + random(800 -> 1000)

        self.y = random.randint(50, 100)   #vị trí y của mây = random(50 -> 100)

        self.image = CLOUD           #hình ảnh mây

        self.width = self.image.get\_width()   # độ rộng của mây

    def update(self):

        self.x -= game\_speed            #vị trí x của mây trừ theo tốc độ của game

        if self.x < -self.width:      #khi mây vượt khỏi màn hình thì tạo lại mây

            self.x = SCREEN\_WIDTH + random.randint(2500, 3000)

            self.y = random.randint(50, 100)

    def draw(self, SCREEN):               #vẽ mây lên màn hình

        SCREEN.blit(self.image, (self.x, self.y))

* Tiếp theo tạo 1 đối tượng cloud từ lớp Cloud trong hàm main
* Trong vòng lặp game ta sẽ gọi tới 2 hàm draw(SCREEN) và update() để vẽ mây lên cửa sổ game

Và đây là kết quả ta đạt được: Mây sẽ xuất hiện ngẫu nhiên trong khoảng vị trí mà ta đã đặt cho nó

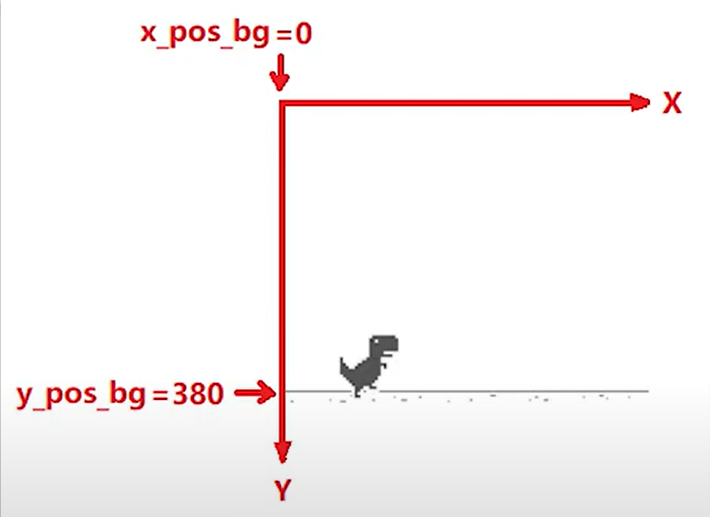


1. **Tạo ra Background.**

* Khởi tạo 2 biến global

 global x\_pos\_bg, y\_pos\_bg

* Khởi tạo x\_pos\_bg = 0 và y\_pos\_bg = 380



* Tạo ra hàm Background() trong Main():

def background():

        global x\_pos\_bg, y\_pos\_bg

        image\_width = BG.get\_width()          #chiều dài image = độ dài bg

        SCREEN.blit(BG, (x\_pos\_bg, y\_pos\_bg))  #vẽ background lên màn hình

        SCREEN.blit(BG, (image\_width + x\_pos\_bg, y\_pos\_bg)) #vẽ nối thêm 1 backgroud nữa

        if x\_pos\_bg <= -image\_width:                 #khi bg chạy quá gần hết màn hình thì vẽ lại

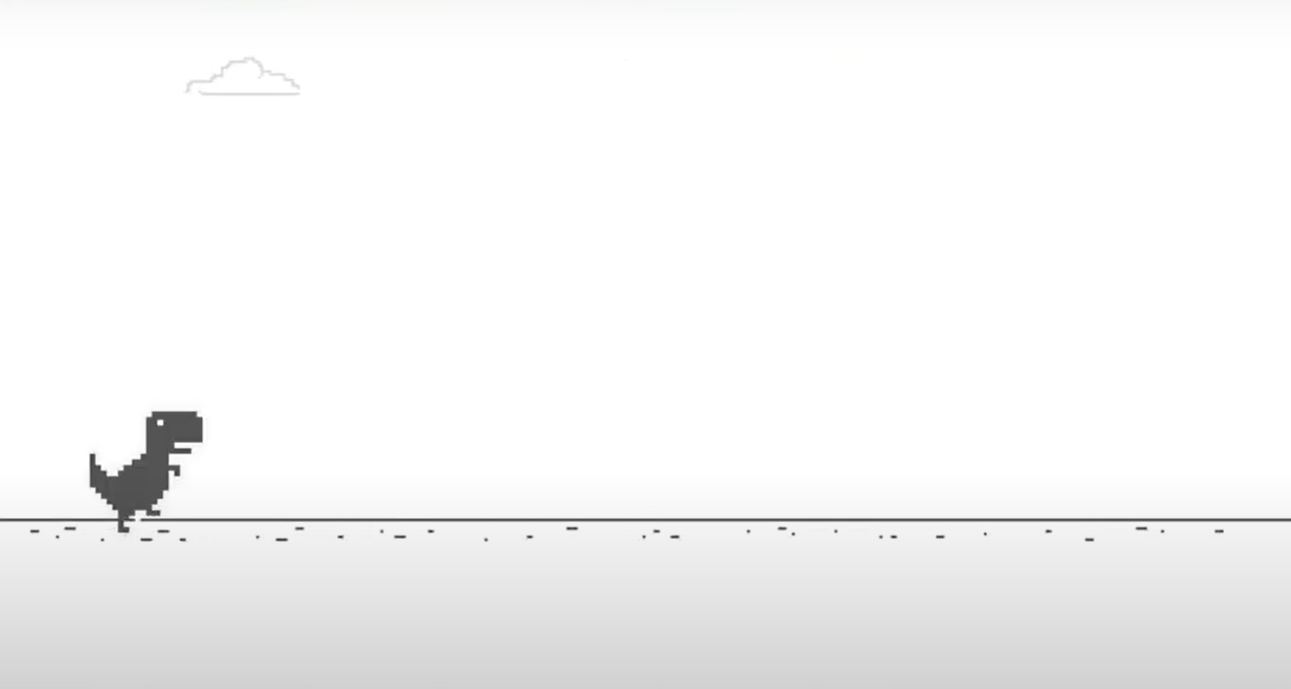
            SCREEN.blit(BG, (image\_width + x\_pos\_bg, y\_pos\_bg))

            x\_pos\_bg = 0

        x\_pos\_bg -= game\_speed #vị trí của backgroud giảm dần theo game\_speed

* Trong vòng lặp game gọi tới hàm background()

Và đây là kết quả ta có được:



1. **Tạo ra SCORE**

* Khởi tạo 1 biến global points = 0 trong hàm main().
* Gán font của score bằng câu lệnh:

font = pygame.font.Font('freesansbold.ttf', 20)   #font cho chữ score

* Tạo ra hàm score()

def score():

        global points, game\_speed

        points += 1             #cộng 1 point theo tg

        if points % 100 == 0:

            game\_speed += 1       #mỗi 100 point thì tăng tốc gốc

        text = font.render("Points: " + str(points), True, (0, 0, 0))       #text sẽ in ra point và điểm

        textRect = text.get\_rect()       #lấy hình chữ nhật để bao quanh score

        textRect.center = (1000, 40)     #vị trí của score

        SCREEN.blit(text, textRect)     #vẽ point lên màn hình

* Trong vòng lặp game ta gọi tới hàm Score()

Và đây là kết quả chúng ta đạt được:



***Part 4: Tạo lớp Obstacle(lớp tạo các chướng ngại vật):***

Gồm có 3 phương thức

1. Init
2. Update
3. draw

Text

Description automatically generated

1)Phương thức init

+phương thức init của lớp obstacle sẽ nhận 2 đối số là image(hình ảnh của chướng ngại muốn hiển thị), và type(loại chướng ngại như hình sau )

        self.image = image

        self.type = type

Calendar

Description automatically generated

Ta sẽ đặt type của chướng ngại có giá trị từ 0 - 2

Sau đó lấy tọa độ hình chữ nhật của hình ảnh đang hiển thị

        self.rect = self.image[self.type].get\_rect()

sau đó đặt tọa độ x của chướng ngại vật bằng với chiều rộng màn hình, có nghĩa là bất cứ khi nào chướng ngại vật được tạo ra thì nó sẽ chỉ ra khỏi mép bên phải màn hình

ta được câu lệnh sau:

    def \_\_init\_\_(self, image, type):

        self.image = image                         #hình ảnh

        self.type = type                         #loại vật cản

        self.rect = self.image[self.type].get\_rect()           #tọa độ hình chự nhật của vật cản

        self.rect.x = SCREEN\_WIDTH         #tọa độ vật cản = độ ngang màn hình

2) phương thức update

Update() giúp di chuyển các chướng ngại vật trên màn hình , bằng cách là giảm tọa độ X của chướng ngại vật trừ theo tốc độ của game(game\_speed)

        self.rect.x -= game\_speed

sau đó thêm câu lệnh if vào update()

if sẽ xóa bỏ chướng ngại vật khi chướng ngại vật chạy ra khỏi khung hình bên trái

        if self.rect.x < -self.rect.width

            obstacles.pop()

ta được câu lệnh

    def update(self):

        self.rect.x -= game\_speed

        if self.rect.x < -self.rect.width:

            obstacles.pop()

3)phương thức draw

Vẽ hình ảnh lên màn hình

    def draw(self, SCREEN):

        SCREEN.blit(self.image[self.type], self.rect)

***Part 5: Tạo lớp vẽ xương rồng nhỏ và xương rồng lớn:***

Đầu tiền tạo lớp smallcactus(xương rồng nhỏ) kế thừa từ lớp Obstacle

class SmallCactus(Obstacle):

lớp này chỉ có 1 phương thức init lấy hình ảnh làm tham số

sau đó đặt type là số nguyên ngẫu nhiên từ 0 – 2

    def \_\_init\_\_(self, image):

        self.type = random.randint(0, 2)

sau khi đặt kiểu thì gọi phương thức init của lớp Obstacle

và đặt tọa độ y nơi cây xương rồng được hiển thị là 325

        super().\_\_init\_\_(image, self.type

        self.rect.y = 325

ta được câu lệnh sau:

class SmallCactus(Obstacle):

    def \_\_init\_\_(self, image):

        self.type = random.randint(0, 2)

        super().\_\_init\_\_(image, self.type)

        self.rect.y = 325

tiếp theo tạo lớp xương rồng lớn tương tự như lớp xương rồng nhỏ

nhưng tọa độ y là 300

class LargeCactus(Obstacle):

    def \_\_init\_\_(self, image):

        self.type = random.randint(0, 2)

        super().\_\_init\_\_(image, self.type)

        self.rect.y = 300

**Part 6: Tạo lớp vẽ Bird có hoạt ảnh bay qua màn hình:**

Bird cũng kế thừa từ obstacle và có 2 phương thức:

class Bird(Obstacle):

vì Obstacle có biến type nhưng chim chỉ có 1 loại

nên ta sẽ đặt type của Bird là 0

sau đó gọi lớp init của Obstacles giống như Cactus

đặt tọa độ Y bay của chim là 200

sau đó them 1 chỉ mục self.index và đặt là 0

    def \_\_init\_\_(self, image):

        self.type = 0

        super().\_\_init\_\_(image, self.type)

        self.rect.y = 200

        self.index = 0

tạo thêm 1 hàm vẽ đè lên draw Obstacle để Bird có animate

dùng câu lệnh if nếu self.index >=9 thì sẽ đặt lại self.index=0 khi nó đạt đến giá trị 9

sau đó vẽ hình ảnh lên màn hình và tăng self.index lên 1

    def draw(self, SCREEN):

        if self.index >= 9:

            self.index = 0

        SCREEN.blit(self.image[self.index//5], self.rect

        self.index += 1

ta sẽ có khi self.index từ 0 – 4 và từ 5 – 9 thì dẽ vẽ đè lên các hình sau

A picture containing table

Description automatically generated

Việc này sẽ lặp đi lặp lại khiến cho chim như đang bay

**Sau đó ở hàm main chúng ta sẽ thêm biến obstacle để lưu trữ chướng ngại vật và list obstacle**

Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Để obstacle xuất hiện trên màn hình

Thêm câu lệnh If vào hàm main để nếu độ dài của list obstacle =0 tức là không còn chướng ngại vật thì ta sẽ thêm chướng ngại vật ngẫu nhiên vào list obstacle

        if len(obstacles) == 0:

            if random.randint(0, 2) == 0:

                obstacles.append(SmallCactus(SMALL\_CACTUS))

            elif random.randint(0, 2) == 1:

                obstacles.append(LargeCactus(LARGE\_CACTUS))

            elif random.randint(0, 2) == 2:

                obstacles.append(Bird(BIRD))

sau đó gọi hàm Draw và Update cho chướng ngại vật trong list obstacle:

        for obstacle in obstacles:

            obstacle.draw(SCREEN)

            obstacle.update()

để vẽ và cập nhật cho chướng ngại vật và thêm vào câu lệnh if để phát hiện va chạm nếu(hình chữ nhật của hình ảnh khủng long chạm vào hình chữ nhật của vật cản(obstacle) ) thì chờ 2s sau đó hiện ra menu sau khi death

        for obstacle in obstacles:

            obstacle.draw(SCREEN)

            obstacle.update()

            if player.dino\_rect.colliderect(obstacle.rect):

                pygame.time.delay(2000)

                death\_count +=

                menu(death\_count)

***Part 7: Tạo menu sau khi death***

Đầu tiên đặt biến đếm death =0 trong hàm main

A screenshot of a computer

Description automatically generated with medium confidence

Sau đó tạo hàm menu với biến truyền death\_count

def menu(death\_count):

vì muốn hiện thị số score

nên lấy     global points

đặt công tắc run = true

sau đó tạo vòng lặp nếu run = true

    while run:

tạo nền trắng cho menu nếu menu chạy và tạo font chữ

        SCREEN.fill((255, 255, 255))

        font = pygame.font.Font('freesansbold.ttf', 30)

nếu số lần chết =0 (death\_count = 0) tức là đang ở lúc vừa bật game thì in ra màn hình

 if death\_count == 0:

            text = font.render("Press any Key to Start and Press 0-8 to choose style", True, (0, 0, 0))

nếu đã chơi vài lần và số lần chết >0 thì in ra màn hình

  elif death\_count > 0:

            text = font.render("Press any Key to Restart and Press 0-8 to choose style", True, (0, 0, 0))

            score = font.render("Your Score: " + str(points), True, (0, 0, 0))

sau đó thì viết câu lệnh đặt ra giữa màn hình

            scoreRect = score.get\_rect()

            scoreRect.center = (SCREEN\_WIDTH // 2, SCREEN\_HEIGHT // 2 + 50)

            SCREEN.blit(score, scoreRect)

Phần text cũng đặt ra giữa màn hình(đề bên ngoài if)

        textRect = text.get\_rect()

        textRect.center = (SCREEN\_WIDTH // 2, SCREEN\_HEIGHT // 2)

        SCREEN.blit(text, textRect)

Và vẽ thêm 1 con khủng long với hình ảnh chạy

        SCREEN.blit(RUNNING[0], (SCREEN\_WIDTH // 2 - 20, SCREEN\_HEIGHT // 2 - 140))

sau đó update lên màn hình và thêm nút tắt game và thêm event nếu bấm phím bất kỳ thì game sẽ chạy

        pygame.display.update()

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                run = False

            if event.type == pygame.KEYDOWN:

                main()

ta được cả câu lệnh hàm menu như sau

def menu(death\_count):

    global points            #lấy point

    run = True

    while run:

        SCREEN.fill((255, 255, 255))               #nền trắng

        font = pygame.font.Font('freesansbold.ttf', 30)     #font chữ

        if death\_count == 0:        #số lần chết = 0 thì bấm để bắt đầu

            text = font.render("Press any Key to Start and Press 0-8 to choose style", True, (0, 0, 0))

        elif death\_count > 0:

            text = font.render("Press any Key to Restart and Press 0-8 to choose style", True, (0, 0, 0))                  #số lần chết lớn hơn 0 in bấm để chơi lại

            score = font.render("Your Score: " + str(points), True, (0, 0, 0))  #in ra số điểm đã đạt được

            scoreRect = score.get\_rect()

            scoreRect.center = (SCREEN\_WIDTH // 2, SCREEN\_HEIGHT // 2 + 50)

            SCREEN.blit(score, scoreRect)

        textRect = text.get\_rect()

        textRect.center = (SCREEN\_WIDTH // 2, SCREEN\_HEIGHT // 2)

        SCREEN.blit(text, textRect)

        SCREEN.blit(RUNNING[0], (SCREEN\_WIDTH // 2 - 20, SCREEN\_HEIGHT // 2 - 140))

        pygame.display.update()

        for event in pygame.event.get():                        #nếu bấm nút thoát thì sẽ out game

            if event.type == pygame.QUIT:

                run = False

            if event.type == pygame.KEYDOWN:               #bấm nút thì tiếp tục choi

                main()

**Cuối cùng gọi hàm menu để cho game bắt đầu chạy khi death\_count = 0**

menu(death\_count=0)