HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

***

BÁO CÁO BÀI TẬP LỚN LẬP TRÌNH PYTHON

Đề tài:	**Thiết kế game "Learning Python with Plano Game"**
Giảng viên giảng dạy:	**Vũ Hoài Thư**
Sinh viên thực hiện:	**Vương Huy Long - B19DCCN400**
	Đoàn Anh Hiếu - B19DCCN239
	Nguyễn Hoàng Dương - B19DCCN153
Nhóm:	**07**

Hà Nội – 2021

Mục lục

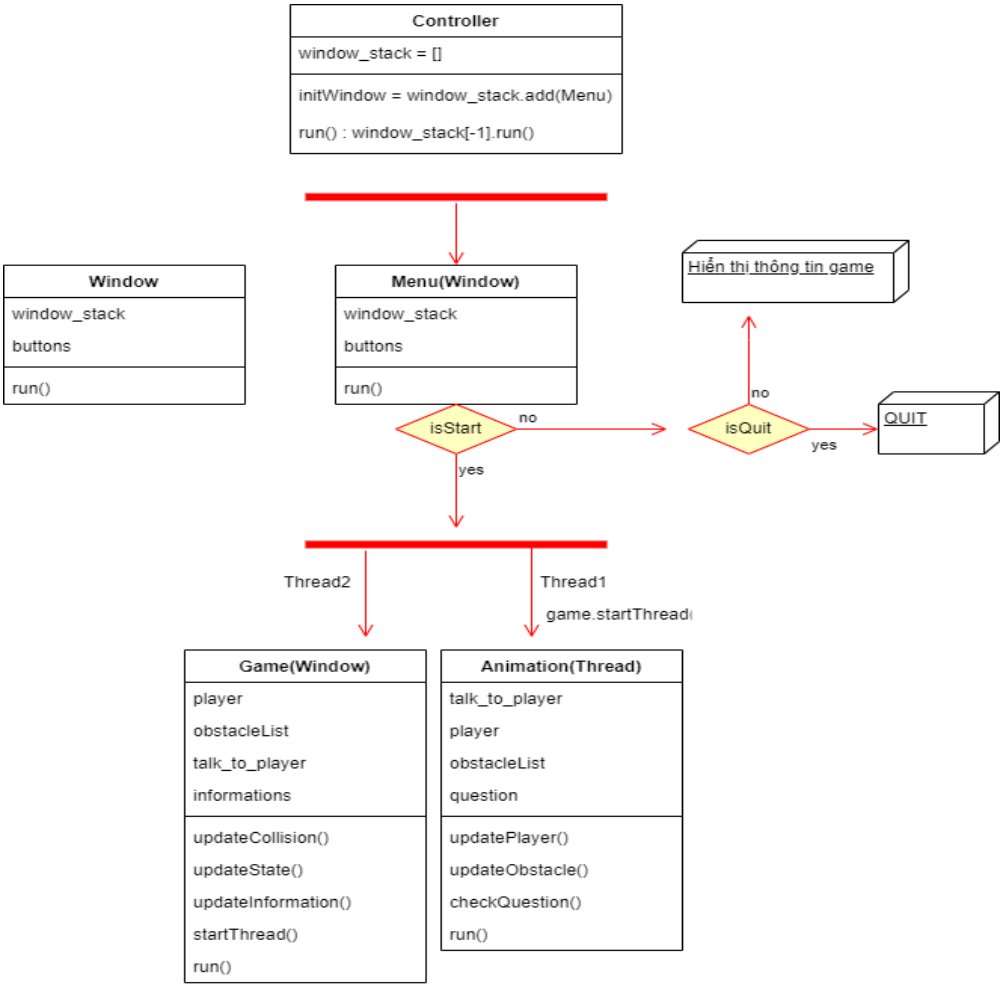
I. Mô tả trò chơi và mục đích thiết kế trò chơi

1. Mô tả trò chơi

Game cần phải có:

- Khởi tạo trò chơi: là một Frame cho phép người chơi biết một số thông tin cần thiết, cách thức điều khiển trò chơi.
- Người điều khiển tàu vũ trụ: người chơi sử dụng các nút trên bàn phím để điều khiển tàu.
- Chướng ngại vật: một điều không thể thiếu trong trò chơi này. Ở đây chướng ngại vật chính là những tàu bay địch và tên lửa của tàu bay địch bắn ra và các thiên thạch to nhỏ, nếu đâm vào nhau sẽ gây ra va chạm và tàu của người đó sẽ không thể bay tiếp. Khi đó, một cửa sổ câu hỏi sẽ hiển thị ra, người chơi phải trả lời câu hỏi để có thể được chơi tiếp.
- Điểm số: được hiển thị ở phía bên trên của cửa sổ chơi.
- Mức độ khó của trò chơi: Tự động nâng mức độ khó theo điểm số của người chơi tại vòng chơi đó. Điểm càng cao thì mật độ và tốc độ của máy bay và đạn ngày càng tăng.

2. Luồng chương trình



- Chương trình bắt đầu từ file main.py
- Window là lớp abstract, ý nghĩa là màn hình hiển thị, gồm các nút,nền ảnh,...
- Đầu tiên, khởi tạo đối tượng Controller có stack quan trọng là windowStack, thêm và xóa window theo cơ chế LIFO.
- Đối tượng Controller sẽ lấy window cuối để gọi hàm run(). Chạy liên tục hàm run() này bằng while loop cho tới khi gặp điều kiện kết thúc.
- Đối tượng Menu kế thừa từ Window sẽ được thêm vào stack đầu tiên và sẽ gọi đầu tiên.
- Nếu bấm nút Start thì sẽ vào window Select - một window để chọn Nhân vật.
- Sau khi chọn nhân vật xong, đối tượng game sẽ được khởi tạo.
- Game có 5 thuộc tính quan trọng là người chơi,chướng ngại vật,thông tin(điểm-tốc độ chạy - tốc độ chạy nền - state), và đối tượng animation kế thừa từ lớp Thread..

- Sau khi game được khởi tạo, sẽ gọi luồng animation sau đó mới chạy hàm run().

3. Mục đích thiết kế trò chơi

Trước tình hình dịch bệnh bùng phát, việc học tập và làm việc online trở nên quen thuộc và cần thiết đối với mọi người, đặc biệt với học sinh, sinh viên.

Với mong muốn đồng hành và phát triển về lĩnh vực giáo dục, việc phát triển những trò chơi mang tính giáo dục giúp việc học tập, tiếp thu kiến thức trở nên dễ dàng và thú vị hơn, tăng khả năng sáng tạo đồng thời cũng rất thân thiện với người dùng chúng tôi đã phát triển trò chơi có tên là "PlanoGame".

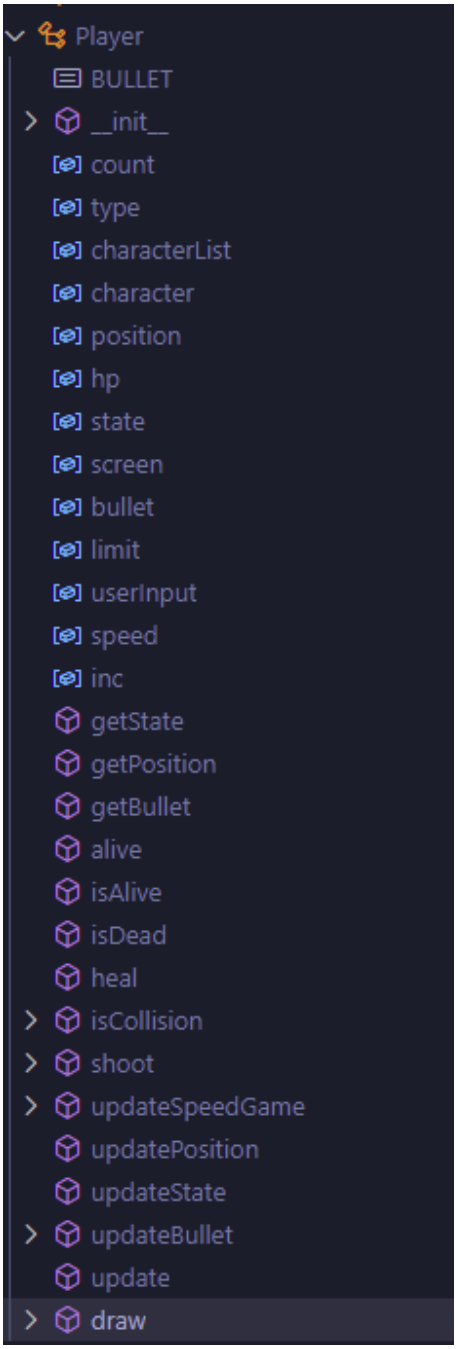
"PlanoGame" là tựa game mang tính giáo dục thông qua các câu hỏi được lồng ghép trong quá trình người chơi tham gia vào trò chơi.

II. Các lớp của chương trình

Để xây dựng game chúng tôi cần xây dựng 19 lớp.

1. Lớp Player:

Đây là lớp người chơi chính, tất cả trạng thái nhân vật sẽ được cập nhật tại đây(khởi tạo nhân vật, lấy các trạng thái của người chơi,update vị trí, tốc độ nhân vật,...)

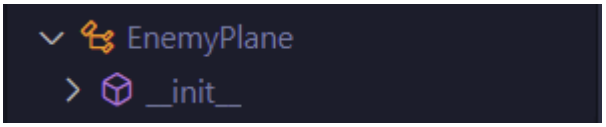


2. Lớp Bullet

Khởi tạo và cập nhật trạng thái đạn của nhân vật

3. Lớp EnemyPlane

Kế thừa từ lớp Obstacle, Lớp EnemyPlane kế thừa từ lớp Obstacle. Là 1 trong các Obstacle.

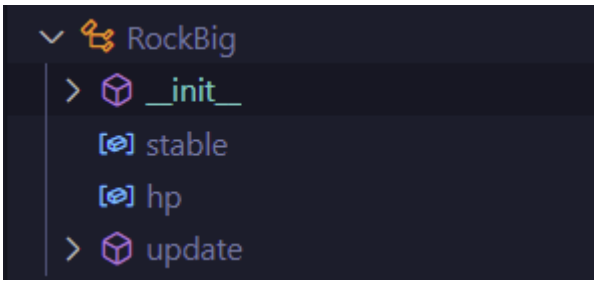


4. Lớp BulletEnemy

Kế thừa từ lớp Obstacle, Khởi tạo và cập nhật đạn của EnemyPlane

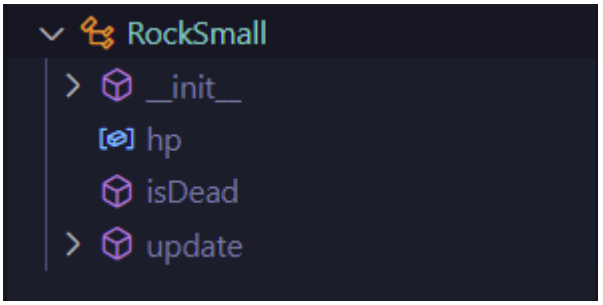
5. Lớp RockBig

Được hiển thị là các thiên thạch to. Là một trong các Obstacle.



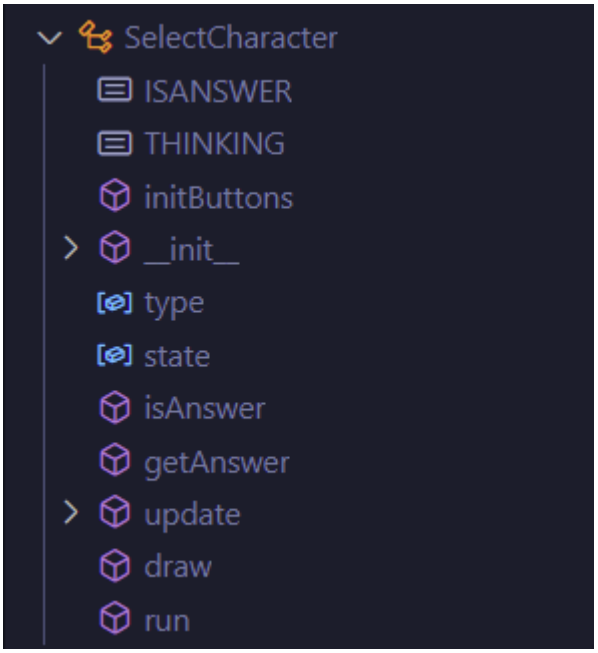
6. Lớp RockSmall

Được hiển thị là các thiên thạch nhỏ. Là một trong các Obstacle.



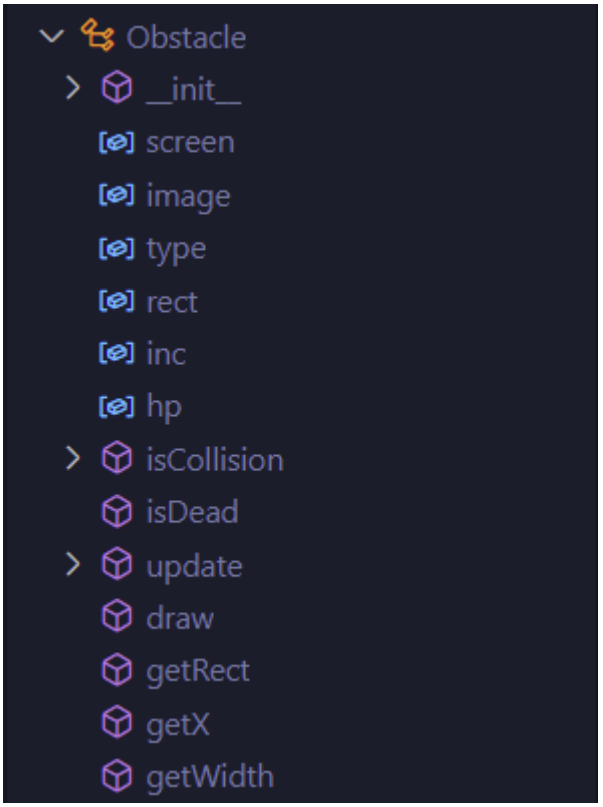
7. Lớp SelectCharacter

Được kế thừa từ lớp Window. Người chơi được lựa chọn nhân vật trước khi vào màn chơi



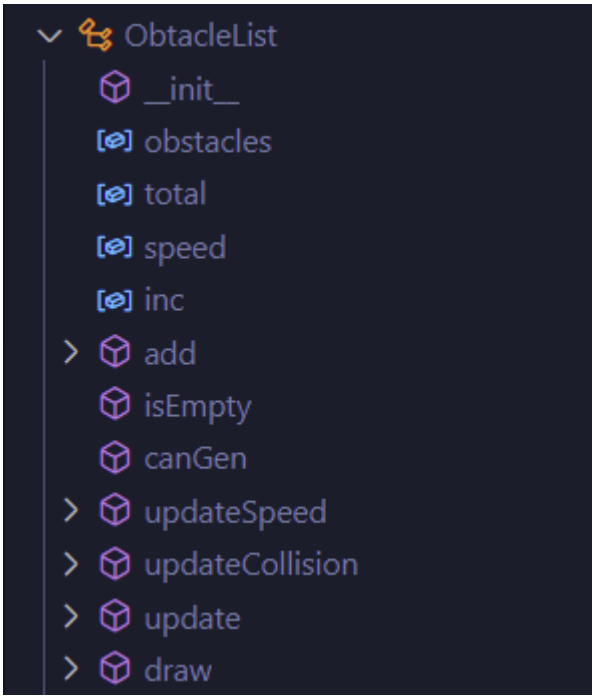
8 .Lớp Obstacle

Lớp cơ bản khởi tạo vật cản, trả về các trạng thái của vật cản.



9. Lớp ObstacleList

Các vật cản được quản lý tại đây, thêm và xóa.

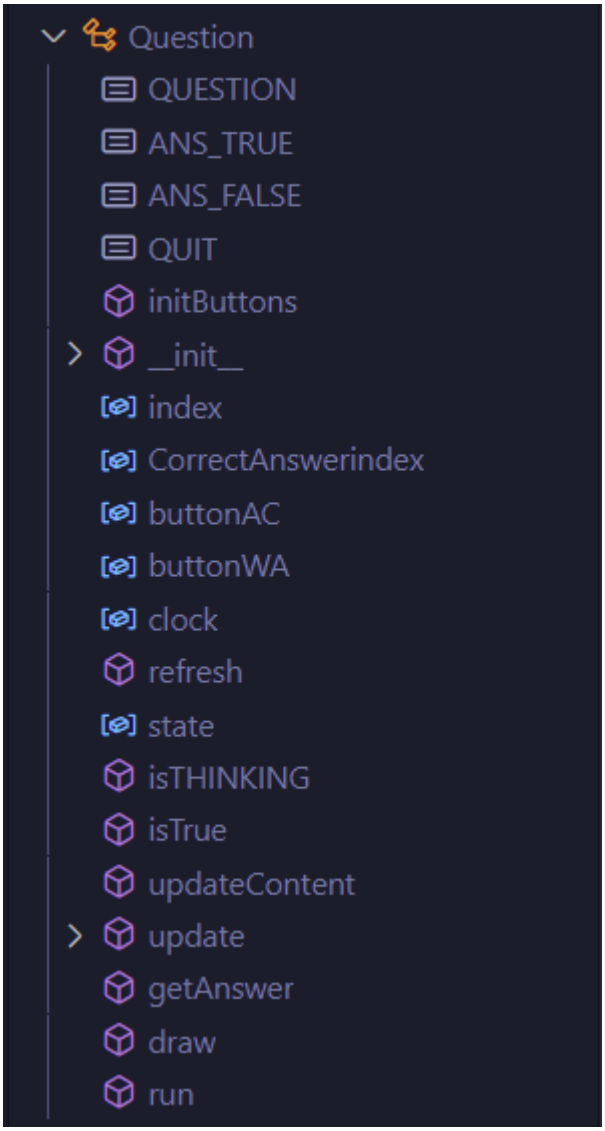


10. Lớp Communicate

Lớp cập nhật trạng thái giữa game với người chơi

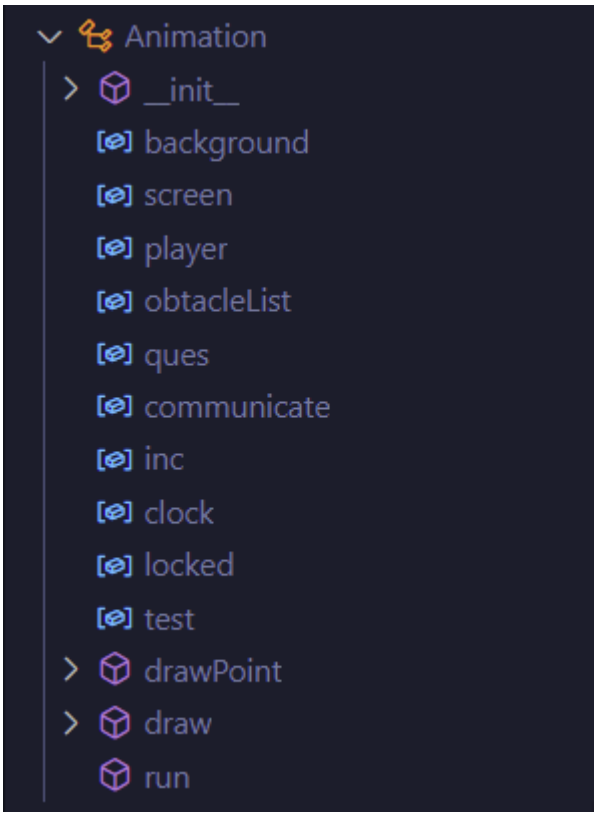
11. Lớp Question

Kế thừa từ Lớp Tab. Khởi tạo các câu hỏi, cập nhật câu hỏi và trả lời.



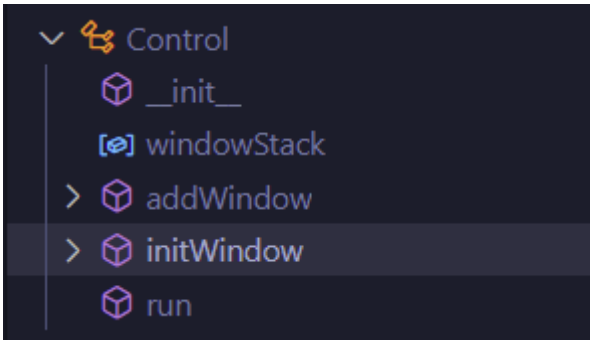
12. Lớp Animation

Lớp tương tác với người chơi. Các vật cản và người chơi sẽ được update trạng thái tại đây.



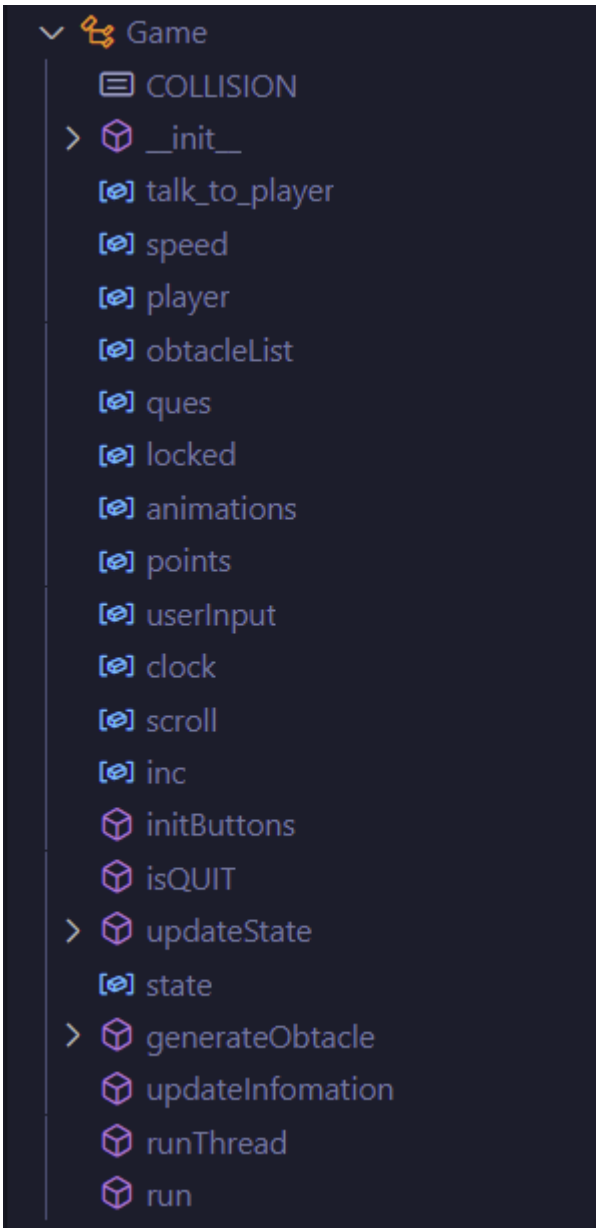
13. Lớp Control

Lớp khởi tạo cửa sổ, khởi chạy Menu và quản lí các tác vụ Window



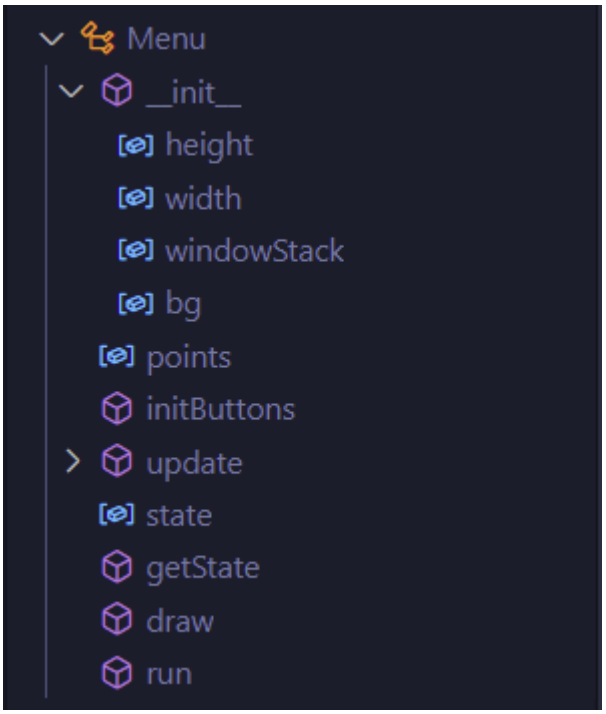
14. Lớp Game

Lớp quản lý game, cập nhật tất cả các trạng thái game, vị trí nhân vật và đạn của nhân vật



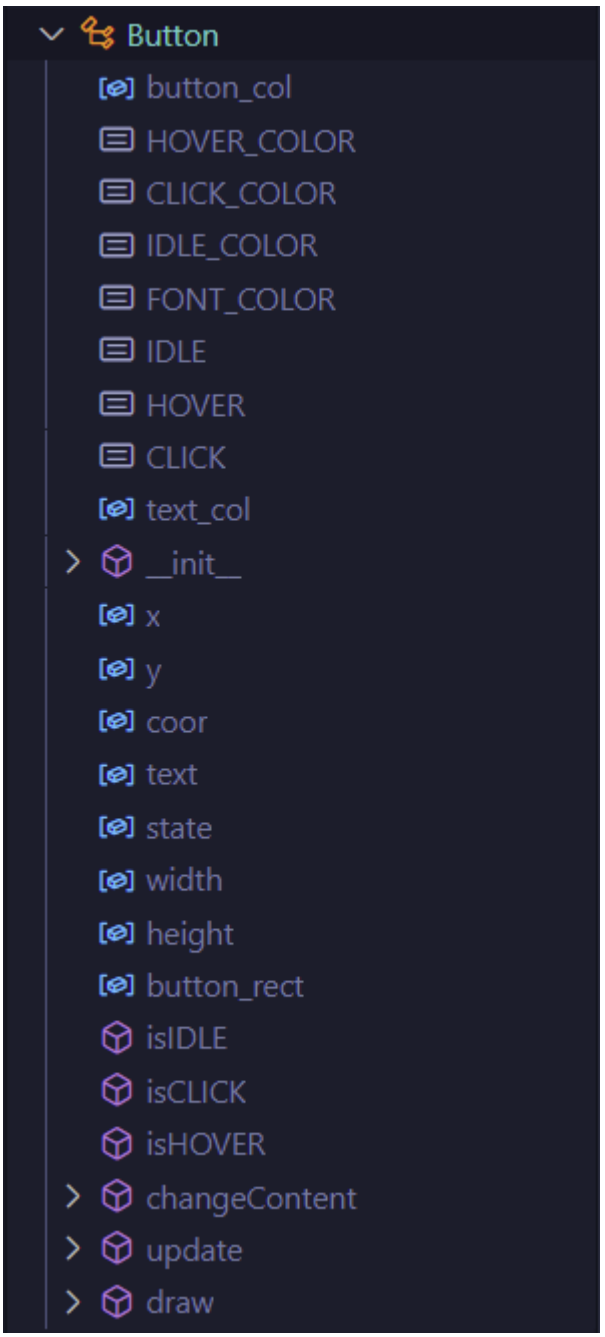
15. Lớp Menu

Lớp Menu kế thừa từ lớp Window, hiển thị Menu game



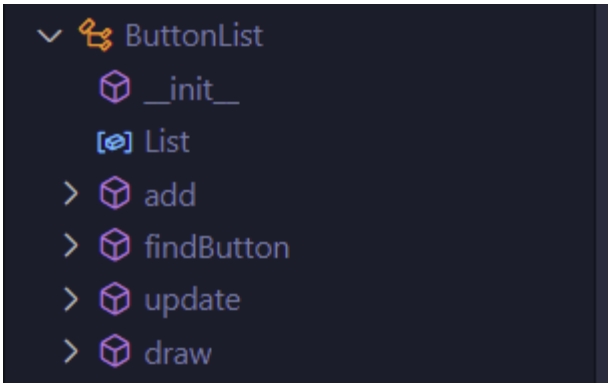
16. Lớp Button

Lớp Button khởi tạo các nút và chức năng khi người chơi tương tác với Button.


Button
button_col
HOVER_COLOR
CLICK_COLOR
IDLE_COLOR
FONT_COLOR
IDLE
HOVER
CLICK
text_col
__init__
x
y
coor
text
state
width
height
button_rect
isIDLE
isCLICK
isHOVER
changeContent
update
draw

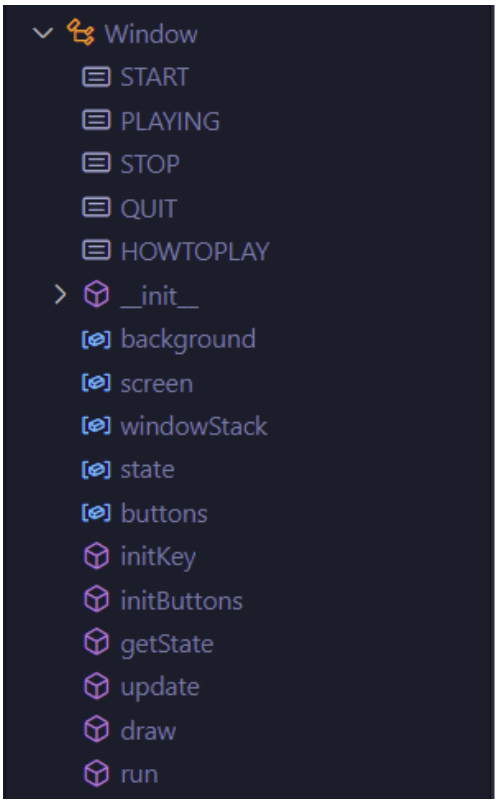
17. Lớp ButtonList

Lớp quản lý các nút



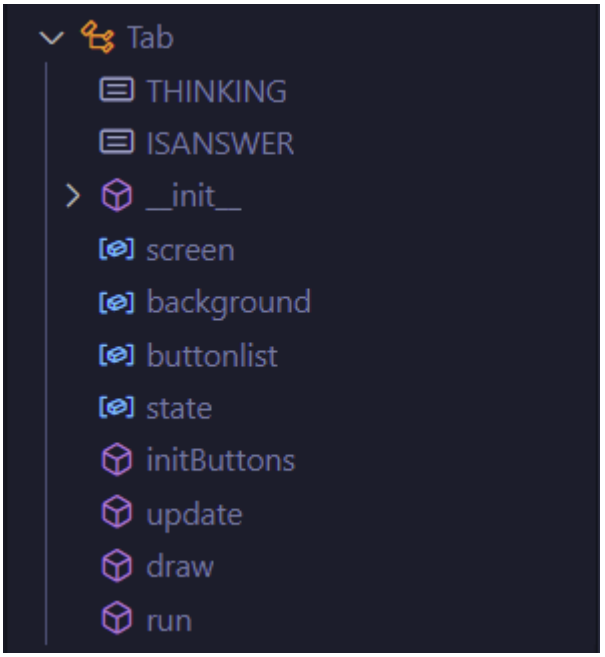
18. Lớp Window

Khởi tạo window, quản lý các thành phần hiển thị trên cửa sổ



19. Lớp Tab

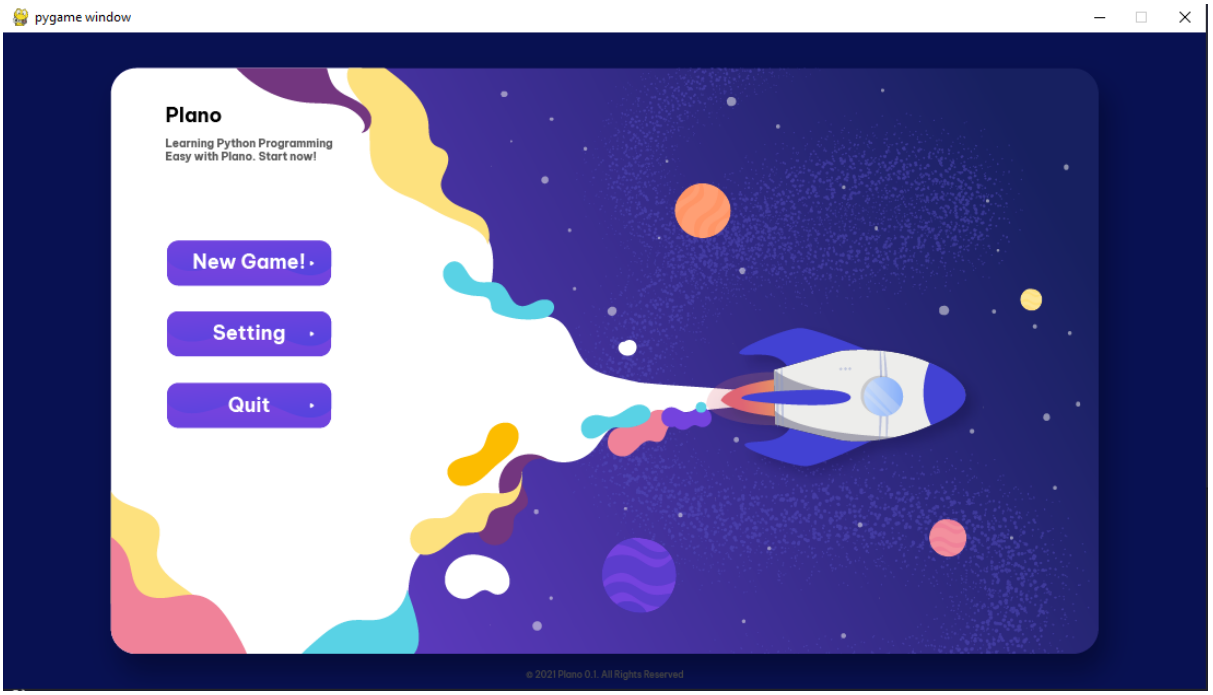
Lớp hiển thị chèn lên window như 1 tab riêng



III. Xây dựng chương trình

1. Khởi tạo trò chơi

Lớp Menu là lớp khởi tạo trò chơi. Lớp này có giao diện bắt mắt, dễ sử dụng. Ở đây người chơi biết cách điều khiển trò chơi, khởi tạo trò chơi mới và thoát khỏi trò chơi. Giao diện đơn giản chỉ cần vài lần click chuột là người chơi có thể tự tham gia trò chơi.




```
from Window import  Window
from Game import Game
import button as a
```

Để có được giao diện và hành động đó lớp Menu đã kế thừa từ lớp Window, ngoài ra còn import thêm 2 module Game và button.

```
SCREEN_HEIGHT = 600
SCREEN_WIDTH = 1100
SCREEN = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
```

Thao tác này sẽ khởi chạy một cửa sổ có kích thước đã được xác định từ trước (SCREEN_WIDTH, SCREEN_HEIGHT)..

```
BGcolor = (204, 102, 0)
RED = (255, 0, 0)
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
```

Khởi tạo các thông số màu dưới dạng tuple.

```
def __init__(self,height,width,windowStack,bg = BG):
    super().__init__(bg,height,width,windowStack)
    self.points = 0
    self.initButtons()
```

Phương thức __init__() trong lớp Menu khởi tạo cửa sổ giao diện.

Ta khởi tạo initButtons:

```
def initButtons(self):
    self.buttons.add("Start",a.Button(150,190,20,START[0],START[1]))
    self.buttons.add("Setting",a.Button(150, 255,20,SETTING[0],SETTING[1]))
    self.buttons.add("Quit",a.Button(150, 320,20,QUIT[0],QUIT[1]))
```

Những Button sau khi được khởi tạo sẽ được tạo bằng cách lấy file ảnh từ folder images tương ứng khi ta truyền vào, khi người dùng di chuyển chuột, ấn chuột, nhả chuột thì Button này cũng sẽ thay đổi giao diện và màu sắc. File ảnh sẽ được lưu trữ trong thư mục Assets.

```
START = [pygame.image.load(os.path.join("Assets/Other", "StartIdle.png")),
        pygame.image.load(os.path.join("Assets/Other", "StartHover.png"))]
SETTING = [pygame.image.load(os.path.join("Assets/Other", "SettingIdle.png")),
        pygame.image.load(os.path.join("Assets/Other", "SettingHover.png"))]
QUIT = [pygame.image.load(os.path.join("Assets/Other", "QuitIdle.png")),
        pygame.image.load(os.path.join("Assets/Other", "QuitHover.png"))]
BG_select = pygame.image.load(os.path.join("Assets/Other", "BG_ChoosePlayer.png"))
class Button():
    # colours for button and text
    IDLE = 1
    HOVER = 2
    CLICK = 3
    text_col = BLACK
    def __init__(self, x, y,sizeText = 30
                 ,img_idle = None,img_hover = None, text = None):
        self.x = x
        self.y = y
        self.font = pygame.font.SysFont('Constantia', sizeText)
        self.coor = [x,y]
        self.text = text
        self.state = self.IDLE
        self.img_idle = img_idle
```

```
    self.img_hover = img_hover
    self.width = self.img_idle.get_width()
    self.height = self.img_idle.get_height()
    self.button_rect = Rect(x,y,img_idle.get_width(),img_idle.get_height())
def isIDLE(self):
    return self.state == self.IDLE
def isCLICK(self):
    return self.state == self.CLICK
def isHOVER(self):
    return self.state == self.HOVER
def changeContent(self,text):
    self.text = text
def update(self):
    pos = pygame.mouse.get_pos()
    if self.button_rect.collidepoint(pos):  # check if the mouse cursor is in the button
        if pygame.mouse.get_pressed()[0] == 1:  # if the left mouse has been clicked to the button
            self.state = self.CLICK
        else:
            self.state = self.HOVER
    elif self.state != self.IDLE:
        self.state = self.IDLE
def draw(self,screen):
    if self.state == self.HOVER:
        screen.blit(self.img_hover,(self.x,self.y))
    else:
        screen.blit(self.img_idle,(self.x,self.y))
    if self.text != None:
        text_img = self.font.render(self.text, True, (0,0,0))
        text_len = text_img.get_width()
        screen.blit(text_img, (
        self.x + int(self.width / 2) - int(text_len / 2), self.y + 25))
def draw1(self,screen,img):
    if self.state == self.HOVER:
        screen.blit(img,(self.x,self.y))
    else:
        screen.blit(img,(self.x,self.y))
    if self.text != None:
        text_img = self.font.render(self.text, True, (0,0,0))
        text_len = text_img.get_width()
        screen.blit(text_img, (
```

```
            self.x + int(self.width / 2) - int(text_len / 2), self.y + 25))
```

Các Button được quản lý theo dạng dict (key = tên của nút, value = nút)

```
class ButtonList():
    def __init__(self):
        self.List = {}
    def add(self,name,button):
        self.List.update({name: button})
    def findButton(self,name):
        for i in self.List.items():
            if name == i[0]:
                return i[1] # return a button ( not name )
        return None
    def update(self):
        for button in self.List.values():
            button.update()

    def draw(self,screen):
        for button in self.List.values():
            button.draw(screen)
```

Ta cập nhật lại surface bằng hàm display.update() sau khi đã hoàn thành gọi xong các hàm vẽ.

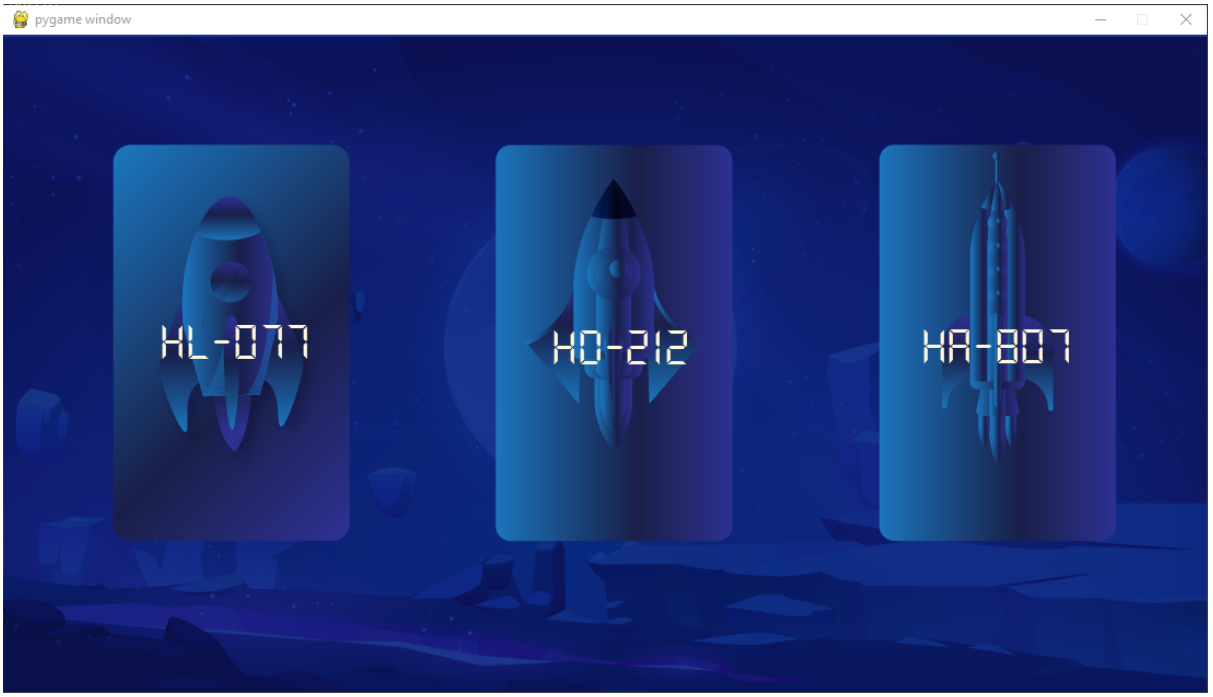
```
    def getState(self):
        return self.state
    def draw(self):
        SCREEN.blit(BG,(0,0))
        self.buttons.draw(self.screen)
    def run(self):
        self.update()
        self.draw()
        pygame.display.update()
```

Trong hàm update(), có có hàm event.get() để bắt sự kiện ở hàng đợi. Nếu click vào dấu X hoặc click vào nút Quit đã được khởi tạo ở trên thì sẽ thoát khỏi chương trình. Khi người chơi click vào button Start sẽ bắt đầu mở window Select. Lớp Game sẽ được gọi.

```
def update(self):
    self.buttons.update()
    key = pygame.key.get_pressed()
    for event in pygame.event.get():
        if event.type == pygame.QUIT or self.buttons.findButton("Quit").isCLICK():
            pygame.quit()
            sys.exit()
        elif self.buttons.findButton("Start").isCLICK():
            select = Select.SelectCharacter(BG_select,self.screen,self.windowStack)
            self.windowStack.append(select)
        elif self.buttons.findButton("About").isCLICK():
            self.state = self.SETTING
            self.background = BG_About
        elif key[pygame.K_BACKSPACE]:
            self.background = self.BGbefore
            self.state = self.PLAYING
```

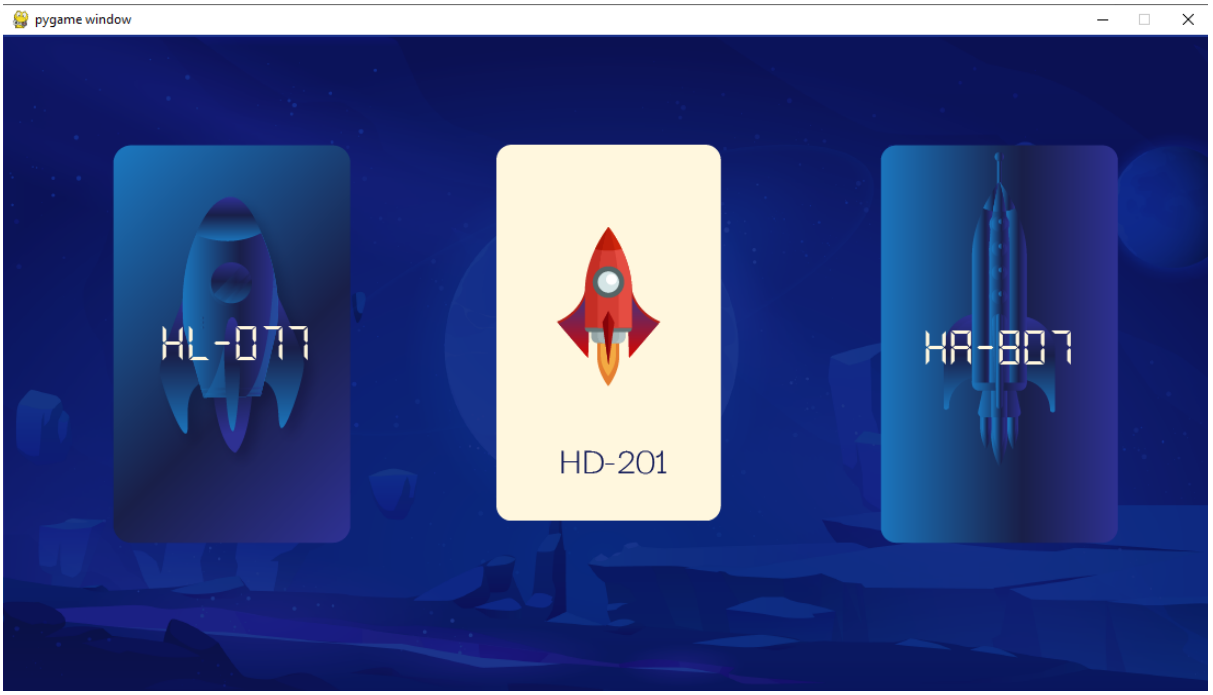
2. Bắt đầu trò chơi

Người chơi sẽ được lựa chọn nhân vật trước khi vào màn chơi.



```python
class SelectCharacter(Window):
    ISANSWER = 20
    THINKING = 21
    def initButtons(self):
        self.buttons.add("character1",button.Button(100,100,30,CHARACTER[0],CHARACTER[1]))
        self.buttons.add("character2",button.Button(450,100,30,CHARACTER[2],CHARACTER[3]))
        self.buttons.add("character3",button.Button(800,100,30,CHARACTER[4],CHARACTER[5]))
    def __init__(self,background,screen,windowstack):
        super().__init__(background,screen,windowstack)
        self.initButtons()
        self.type = -1
        self.state = self.THINKING
    def isAnswer(self):
        return self.state == self.ISANSWER
    def getAnswer(self):
        return self.type
    def update(self):
        self.buttons.update()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if self.buttons.findButton("character1").isCLICK():
                self.type = 0
            elif self.buttons.findButton("character2").isCLICK():
                self.type = 1
            elif self.buttons.findButton("character3").isCLICK():
                self.type = 2
            if self.type != -1:
                self.state = self.PLAYING
                game = Game(BG1, self.screen, self.windowStack, self.state, RUNNING, self.type)
                game.runThread()
                self.windowStack.append(game)
    def draw(self):
        self.screen.blit(self.background,(0,0))
        self.buttons.draw(self.screen)
    def run(self):
        if self.state == self.PLAYING:
            self.windowStack.pop()
        self.update()
        self.draw()
        pygame.display.update()
```

Khi di chuột vào từng ô, người chơi sẽ thấy được từng nhân vật được hiện lên. Từng nhân vật sẽ có những hình dáng và chức năng khác nhau.



Lớp Player và Bullet được khởi tạo. Khi đó ảnh đạn và máy bay của người chơi và địch, các thiên thạch sẽ được truyền vào.

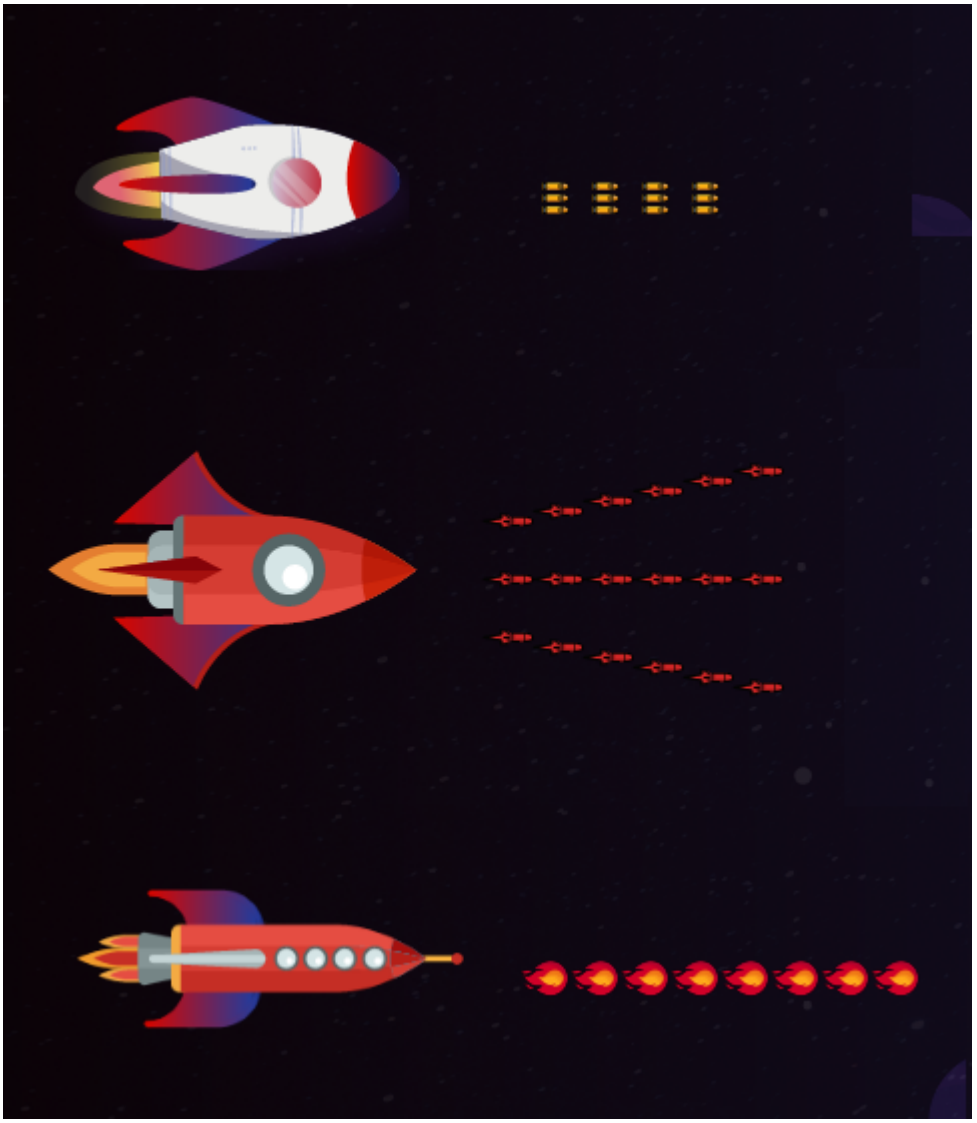
```
PLANE_ENEMY = [pygame.image.load(os.path.join("Assets/Obtacle", "Rocket_Enemy.png")),
               pygame.image.load(os.path.join("Assets/Obtacle", "Enemy2.png"))]
BULLET = [pygame.image.load(os.path.join("Assets/Dino", "Rocket_Bullet.png")),
          pygame.image.load(os.path.join("Assets/Dino", "Rocket_Bullet.png")),
          pygame.image.load(os.path.join("Assets/Dino", "Bullet_Rocket1.png"))]
BULLET_ENEMY = [pygame.image.load(os.path.join("Assets/Obtacle", "Bullet_enemy.png"))]
ROCKBIG = [pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_5_Big.png")),
        pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_1_Big.png")),
        pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_2_Big.png")),
        pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_3_Big.png")),
        pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_4_Big.png"))]
ROCKSMALL = [pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_5_Small.png")),
        pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_1_Small.png")),
        pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_2_Small.png")),
        pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_3_Small.png")),
        pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_4_Small.png")),
             pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_5_Medium.png")),
             pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_1_Medium.png")),
             pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_2_Medium.png")),
             pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_3_Medium.png")),
             pygame.image.load(os.path.join("Assets/Obtacle/Rock", "Rock_4_Medium.png"))
             ]
```

```
    BULLET = [pygame.image.load(os.path.join("Assets/Dino", "Bullet.png")),
              pygame.image.load(os.path.join("Assets/Dino", "Rocket_Bullet.png")),
              pygame.image.load(os.path.join("Assets/Dino", "Bullet_Rocket1.png"))]
```

Khi khởi tạo 1 nhân vật mới, đối tượng bullet sẽ được truyền type của nhân vật tương ứng:

```
def shoot(self):
    self.state = SHOOTING
    rect_x = self.position.x + self.character.get_width()//2
    rect_y = self.position.y + self.character.get_height() // 2
    if len(self.bullet) < self.limit:
        self.bullet.append(Bullet(self.speed,self.BULLET,self.screen,rect_x,rect_y,self.type))
        if self.type == 1:
            self.bullet.append(Bullet(self.speed, self.BULLET, self.screen, rect_x, rect_y + 4, self.type,5))
            self.bullet.append(Bullet(self.speed, self.BULLET, self.screen, rect_x, rect_y - 4, self.type,-5))
```

1 vài hình ảnh minh họa:



Trong lớp Bullet có các hàm như init(), isCollision, isDead,.. Sẽ dùng để lấy các tọa độ, tốc độ, giới hạn khung và gán tọa độ, tốc độ của đường đạn. Riêng hàm update() sẽ cập nhật lại tọa độ, tốc độ của đạn sau 1 khoảng thời gian.

```
    def update(self):
        if self.state == COLLISION and self.type == 2:
            self.position.y += 10*self.stable
        elif self.type == 1:
            self.position.y += self.inc
        self.position.x += (5*self.speed)
    def draw(self):
        self.screen.blit(self.bullet,(self.position))
```

Lớp Player có các hàm như init(), getState(), getPosition(), getBullet(), isAlive(), isDead(),... sẽ dùng để lấy các tọa độ, tốc độ, giới hạn khung và gán tọa độ,tốc độ như lớp Bullet.

Hàm updateState() và updateBullet() sẽ cập nhật tình trạng của người chơi.

```
def updateState(self):
    if self.state == COLLISION:
        self.hp -= 10
        self.state = ISALIVE
    elif self.state == SHOOTING:
        self.state = ISALIVE
    if self.isDead():
        self.state = DEAD
def updateBullet(self):
    for i in self.bullet:
        if i.isDead():
            self.bullet.remove(i)
        i.update()
def update(self):
    if self.inc != 0:
        self.updateBullet()
    self.updatePosition()
    self.updateState()
```

Vị trí của người chơi sẽ được cập nhật qua hàm updatePosition()

```
    def updatePosition(self):
        if self.userInput[pygame.K_s] and self.position.y <550:
            self.position.y += 4* self.speed *self.inc
        elif self.userInput[pygame.K_w] and self.position.y > -50:
            self.position.y -=4 * self.speed *self.inc
        elif self.userInput[pygame.K_d] and self.position.x <1100:
            self.position.x += 4 * self.speed * self.inc
        elif self.userInput[pygame.K_a] and self.position.x > -50:
            self.position.x -= 4 * self.speed * self.inc
```

Sau đó update sẽ cập nhật lại toàn bộ tình trạng trên của người chơi

```
    def update(self):
        if self.inc != 0:
            self.updateBullet()
        self.updatePosition()
        self.updateState()
```

Các EnemyPlane kế thừa từ Obstacle sẽ xuất hiện ngẫu nhiên cùng với bullet tương ứng

```
class EnemyPlane(Obstacle):
    def __init__(self,screen,image,type = 0):
        super().__init__(screen,image,type)
        rand = random.choice([100,150,200,250,300,350,400,450,500,550,600])
        self.rect.y = rand
        rand2 = random.choice([1200,1300,1400])
        self.rect.x = rand2
class BulletEnemy(Obstacle):
    def __init__(self,screen,image,position,type = 0):
        super().__init__(screen,image,type)
        rect_x = position.x + self.image[type].get_width()//2
        rect_y = position.y + self.image[type].get_height() // 2
        self.rect.x = rect_x
        self.rect.y = rect_y
    def update(self,speed):
        self.rect.x -= 5*speed
```

Bên cạnh đó còn có các thiên thạch RockBig và RockSmall

```
class RockBig(Obstacle):
    def __init__(self,screen,image,type = 0):
        super().__init__(screen,image,type)
        self.rect.y = random.randint(50,550)
        if (self.rect.y >= 300):
            self.stable = random.choice([-1,-2,-3])
        else:
            self.stable = random.choice([1,2,3])
        rand2 = random.randint(1100,1200)
        self.rect.x = rand2
        rand3 = random.choice([210,260,300,350,500])
        self.hp = rand3

    def update(self,speed):
        if self.rect.y >= 600 or self.rect.y <= 0:
            self.stable *= -1
        self.rect.x -= (speed+1)*self.inc
        self.rect.y += self.stable*self.inc
```

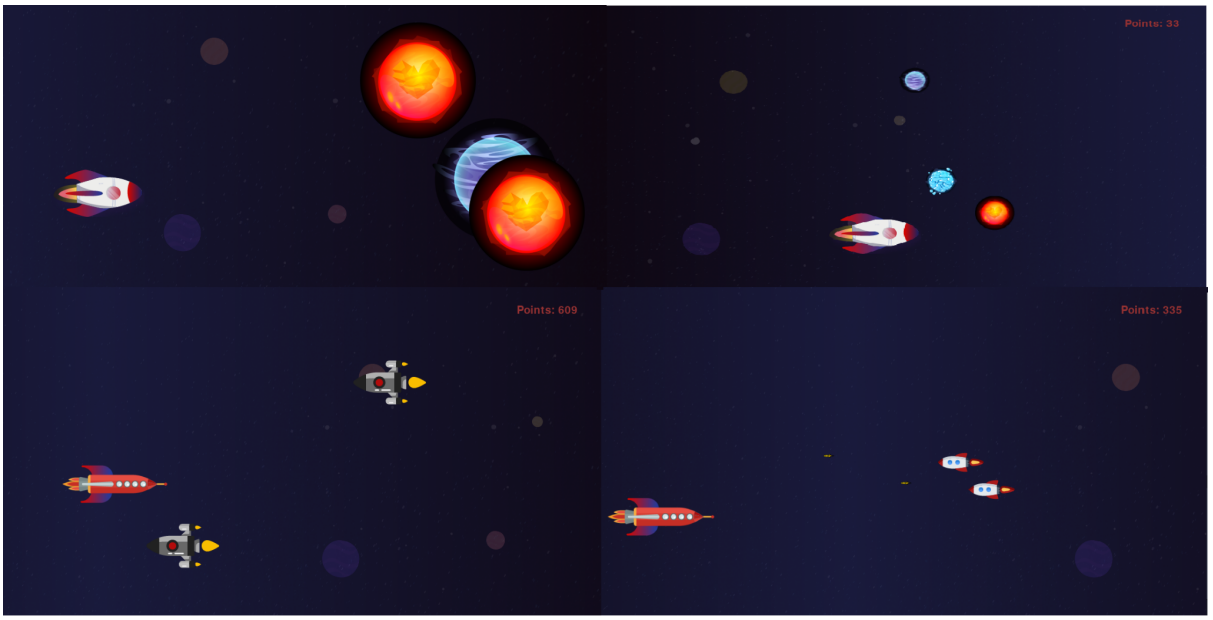
```
class RockSmall(Obstacle):
    def __init__(self,screen,image,type = 0):
        super().__init__(screen,image,type)
        if self.type >= 5:
            self.hp = 40
        rand = random.randint(-150,-30)
        self.rect.y = rand
        rand2 = random.randint(500,1100)
        if rand2 % 4 != 1:
            self.rect.x = rand2
        else:
            self.rect.x = rand2 - 400
    def isDead(self):
        return (self.hp <= 0 or self.rect.y >= 600)
    def update(self,speed):
        if self.type >= 5:
            speed -= 2
        self.rect.x -= (speed*2)*self.inc
        self.rect.y += speed*3*self.inc
```

Các thiên thạch cũng được tạo ngẫu nhiên trong khoảng 1 vị trí nhất định. RockSmall sẽ rơi từ trên xuống, còn RockBig sẽ bay ngang từ phải sang

Sau đó các Obstacle sẽ được tạo tại GenerateObstacle()

```
def generateObtacle(self):
    t = self.speed*89 + 34
    rand1 = random.randint(1,100)
    if t%3 == 0 and self.obstaclelist.canGen():
        rand = random.choice([0,1])
        obs = Obstacles.EnemyPlane(self.screen, PLANE_ENEMY,rand)
        self.obstaclelist.add(obs)
        rand = random.choice([1, 2])
        if rand == 1:
            self.obstaclelist.add(Obstacles.BulletEnemy(self.screen, BULLET_ENEMY, obs.getRect()))
    elif t%3 == 1 and self.obstaclelist.canGen():
        obs = Obstacles.RockBig(self.screen, ROCKBIG, rand1%5)
        self.obstaclelist.add(obs)
    elif t%3 == 2 and self.obstaclelist.canGen() :
        obs = Obstacles.RockSmall(self.screen, ROCKSMALL, rand1%10)
        self.obstaclelist.add(obs)
```

Hình ảnh về các Obstacle:



Khi người chơi bị các Obstacle đâm, 1 cửa sổ câu hỏi và câu trả lời sẽ hiện ra

Người chơi buộc phải chọn đáp án để có thể tiếp tục trò chơi. Nếu trả lời đúng sẽ câu trả lời sẽ hiện màu xanh lá cây, sai sẽ hiện màu đỏ; được thể hiện trong getAnswer()

```
def getAnswer(self):
        if self.state == self.ANS_TRUE:
            self.buttonAC.drawAnswer(self.screen,ANSWER[2])
        elif self.state == self.ANS_FALSE:
            self.buttonAC.drawAnswer(self.screen,ANSWER[2])
            self.buttonWA.drawAnswer(self.screen,ANSWER[3])
        if self.state != self.THINKING:
            pygame.time.delay(500)
        pygame.display.update()
```

Các câu hỏi được truyền vào từ file QA.txt list listQA

```
QA.txt
Cau lenh return dung de ? - Ket thuc vong lap - Tra lai gia tri - Ca A va B - Ket thuc ham va tra lai gia tri - 4
What is python ? - DTP software - a programming language - Spreadsheet software - Computer  - 2
Co bao nhieu class trong Plano code ? - 10 - 12 - 13 - Biet the quai nao duoc! - 4
Method nay dung de lam gi : game.runThread() ? - Start 1 luong Animation - Start Game - Start 2 luong Animation and Game - Cat nhat trang thai - 1
Method nao dung de Cat nhat va cham, thuoc thread va class nao? - updateCollision(player),thread game,class Player - updateCollision(player),thread animation,class Obtacle - upd
```

```
listQA = []
f = open("QA.txt")
for data in f:
    listQA.append(data.split("-"))
f.close()
```

Các câu hỏi được truyền vào dưới dạng list, khi đó nó sẽ được add vào QUESTION

```
QUESTION = ["","Ans1","Ans2","Ans3","Ans4"]
```

Từng câu hỏi và đáp án sẽ được add vào từng button để hiển thị ra màn hình

```
def initButtons(self):
    self.buttonlist.add("Ques", a.Button(200 + 40, 200,10,QUESTION[0],QUESTION[0],listQA[self.index][0]))
    self.buttonlist.add(self.QUESTION[1], a.Button(240 , 340,10,ANSWER[0],ANSWER[1],listQA[self.index][1]))
    self.buttonlist.add(self.QUESTION[2], a.Button(600 - 70, 340,10,ANSWER[0],ANSWER[1],listQA[self.index][2]))
    self.buttonlist.add(self.QUESTION[3], a.Button(240 , 420,10,ANSWER[0],ANSWER[1],listQA[self.index][3]))
    self.buttonlist.add(self.QUESTION[4], a.Button(600 - 70, 420,10,ANSWER[0],ANSWER[1],listQA[self.index][4]))
```

Trong đó, đáp án đúng sẽ được lấy từ vị trí thứ 5 trong từng câu hỏi ở QA.txt

```
    self.CorrectAnswerindex =  int(listQA[self.index][5])
```

Câu hỏi sẽ được cập nhật qua updateContent()

```
def updateContent(self):
    self.buttonlist.findButton("Ques").changeContent(listQA[self.index][0])
    self.buttonlist.findButton(self.QUESTION[1]).changeContent(listQA[self.index][1])
    self.buttonlist.findButton(self.QUESTION[2]).changeContent(listQA[self.index][2])
    self.buttonlist.findButton(self.QUESTION[3]).changeContent(listQA[self.index][3])
    self.buttonlist.findButton(self.QUESTION[4]).changeContent(listQA[self.index][4])
```

Note: Về Lớp Communicate và ManageAnimation: đây là 2 lớp tương tác trực tiếp với người chơi, lắng nghe các hoạt động, trạng thái của người chơi.

Như ta có thể thấy được tại hàm run() trong class ManageAnimation, các tương tác với người chơi ví dụ như khi người chơi thoát khỏi trò chơi, hoặc khi người chơi va chạm với các vật cản, self.ques.run() đóng vai trò gọi lên lớp Question tại class Game

self.ques = Question(self.screen,BG_ques)

Đồng thời với đó, nếu người chơi trả lời đúng, HP của người chơi sẽ được tăng lên self.player.heal()

```
def run(self):
    while self.communicate.isRunning():
        if self.communicate.isQuit():
            break
        if self.communicate.isCollision():
            self.ques.run()
            if self.ques.isTrue():
                self.player.heal()
            if not self.ques.isTHINKING():
                self.ques.refresh()
                self.ques.updateContent()
                self.communicate.setCollision(False)
                self.communicate.continue_game()
        self.obtacleList.update()
        self.player.update()
```

IV. Kết luận

Trong thời gian làm bài tập lớn, nhờ các kiến thức của cô Vũ Hoài Thư và sự giúp đỡ của các thành viên trong nhóm, nhóm chúng em đã thu được nhiều kết quả trong việc học lập trình Python.

Do thời gian và khả năng có hạn nên bài tập lớn của nhóm chúng em còn rất nhiều thiếu sót, chúng em rất mong nhận được sự góp ý, giúp đỡ của thầy và các bạn để bài tập của chúng em được hoàn thiện hơn.