

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Đề tài:

Trò chơi bắn xe tăng

Giảng viên hướng dẫn: Từ Lãng Phiêu
Nhóm thực hiện: Nhóm 18
Sinh viên thực hiện: 3120410620 – Lê Thanh Vũ
3120410426 – Nguyễn Thành Quang
3120410635 – Đặng Huỳnh Như Y

TP. HỒ CHÍ MINH, THÁNG 4/2024

DANH SÁCH THÀNH VIÊN NHÓM 18

MSSV	Họ tên	Gmail
3120410620	Lê Thanh Vũ	thanhvu270202@gmail.com
3120410426	Nguyễn Thanh Quang	Quang.kasumi@gmail.com
3120410635	Đặng Huỳnh Như Y	nhuy.danghuynh2002@gmail.com



Mục lục

1 TỔNG QUAN GIAO DIỆN	4
1.1 Các thành phần trên giao diện	4
1.2 Tạo đối tượng button và xử lí sự kiện	5
2 MODE GAME VƯỢT MƯA THIÊN THẠCH	8
2.1 Giới thiệu	8
2.2 Hướng dẫn cách chơi	8
2.3 Ý tưởng	10
2.4 Mô hình chương trình	11
2.5 Quy trình thực hiện	12
2.5.1 Thư viện cần dùng	12
2.5.2 Cài đặt một số biến cơ bản	12
2.5.3 Định dạng hình ảnh và vị trí	12
2.5.4 Cơ chế điều khiển xe tăng	13
2.5.5 Tăng tốc độ của trò chơi	13
2.5.6 Xử lí khi xe tăng nhặt hộp quà	13
2.5.7 Xử lí khi xe tăng va trúng thiên thạch	14
2.5.8 Xử lí cửa sổ game	15
2.5.9 Hiệu ứng âm thanh	16
3 MODE TANK PVP OFFLINE	17
3.1 Hướng dẫn cách chơi	18
3.2 Ý tưởng thực hiện	20
3.3 Mô hình hóa chương trình	23
3.3.1 Mô tả chi tiết các thành phần cần xử lí	23
3.3.2 Cơ chế quá trình hoạt động	23
3.4 Quy trình thực hiện	26
3.4.1 Thư viện cần dùng	26
3.4.2 Căn chỉnh cửa sổ	26
3.4.3 Thiết kế giao diện menu game	27
3.4.3.a Khởi tạo và hiển thị menu game	28
3.4.3.b Xử lý sự kiện thoát khỏi game và chọn tùy chọn trong menu	30
3.4.4 Thiết lập màn hình Select Tank	32
3.4.5 Định dạng xe tăng	37
3.4.5.a Định dạng và đưa hình ảnh 2 tank lên cửa sổ	38
3.4.5.b Cơ chế điều khiển tank của người chơi	40
3.4.6 Thiết lập chỉ số tank	41
3.4.7 Thiết lập Map chiến đấu	43
3.4.8 Thiết lập và xử lí trận đấu	44
3.4.8.a Xử lý sự kiện	45
3.4.8.b Xử lý bắn đạn và thay đổi sprite	46
3.4.8.c Kiểm tra va chạm và tank nổ khi trúng đạn	47
3.4.8.d Cập nhật và hiển thị	48
3.4.8.e Kiểm tra điều kiện kết thúc ván đấu	48
3.4.8.f Thiết lập vị trí máu của player	49
3.4.9 Thiết lập màn hình cuối, trận chiến kết thúc	50
3.4.9.a Thiết lập màn hình sau trận đấu	50



3.4.9.b	Hiển thị kết quả và hướng dẫn	53
3.4.9.c	Cập nhật điểm số và quay lại menu chính	54
3.4.9.d	Khởi tạo các biến và font chữ	55
3.4.10	Thêm tiếng nhạc, bắn đạn, nổ	56
4	MODE TANK PVP ONLINE	58
4.1	Server cho game PVP	58
4.2	Game client	60
5	Web demo và tải game	67



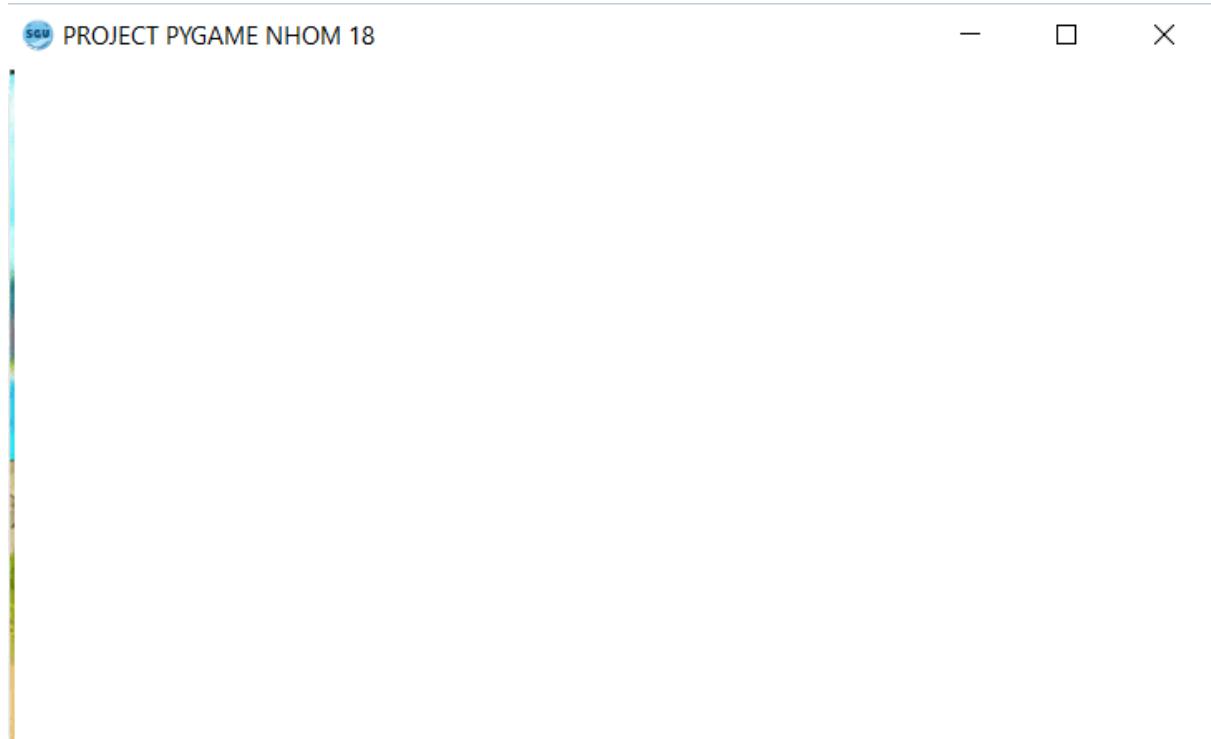
1 TỔNG QUAN GIAO DIỆN

1.1 Các thành phần trên giao diện

Giao diện được thiết kế bằng PyQt5 bao gồm một cửa sổ hiển thị hình ảnh giới thiệu chương trình, với 3 nút để chuyển đổi giữa các chế độ và nút thoát để đóng giao diện, cũng như có phần mô tả và biểu tượng của chương trình.

Phương thức ‘`__init__`’ được sử dụng để khởi tạo đối tượng của một lớp. Nó gọi phương thức khởi tạo của lớp cha, thiết lập các thuộc tính như ‘`title`’, ‘`left`’, ‘`top`’, ‘`width`’, ‘`height`’, và gọi phương thức ‘`initUI`’ để khởi tạo giao diện người dùng.

```
def __init__(self):
    super().__init__()
    self.title = 'PROJECT PYGAME NHOM 18' #thiết lập tiêu đề
    cửa sổ chính
    self.left = 700
    self.top = 300
    self.width = 728
    self.height = 410
    self.initUI()
```



1.2 Tạo đối tượng button và xử lý sự kiện

```
# Tạo 3 button
btn1 = QPushButton('Tank PvP', self)
btn1.move(90, 370)
btn1.setStyleSheet("QPushButton:hover { background-color:
red }") #đổi màu nền button
btn1.clicked.connect(self.on_click_game)

btn2 = QPushButton('Crazy Tank', self)
btn2.move(300, 370)
btn2.setStyleSheet("QPushButton:hover { background-color:
yellow }")
btn2.clicked.connect(self.on_click_game)

btn3 = QPushButton('Thoát', self)
btn3.move(510, 370)
btn3.clicked.connect(self.on_click_exit)

self.show()
```



Ta tạo nút nhấn có tên “**Tank PvP**” tương ứng với chế độ cùng tên và set vị trí chúng cho phù hợp trên trục tọa độ Oxy (**90,370**), và tạo hàm xử lí để xử lí sự kiện này, từ đó ta thực hiện các chế độ khác tương tự. Để thu hút người chơi thì nhóm đã tạo thêm hiệu ứng cho các button khi click vào.



```
@pyqtSlot()
def on_click_game(self):
    self.sound.stop()
    sender = self.sender() # Xác định đối tượng gửi sự kiện,
    khi user nhấn nút để chọn trò chơi

    game_process = None # biến game_process sẽ chứa quy trình
    subprocess được tạo ra khi một trò chơi được khởi động

    if sender.text() == 'Tank PvP':
        game_process = subprocess.Popen(['python',
'Project_Pygame_Nhom18/BangBang/Tank2P/TankPvP.py'])
    elif sender.text() == 'Crazy Tank':
        game_process = subprocess.Popen(['python',
'Project_Pygame_Nhom18/BangBang/CrazyTank.py'])

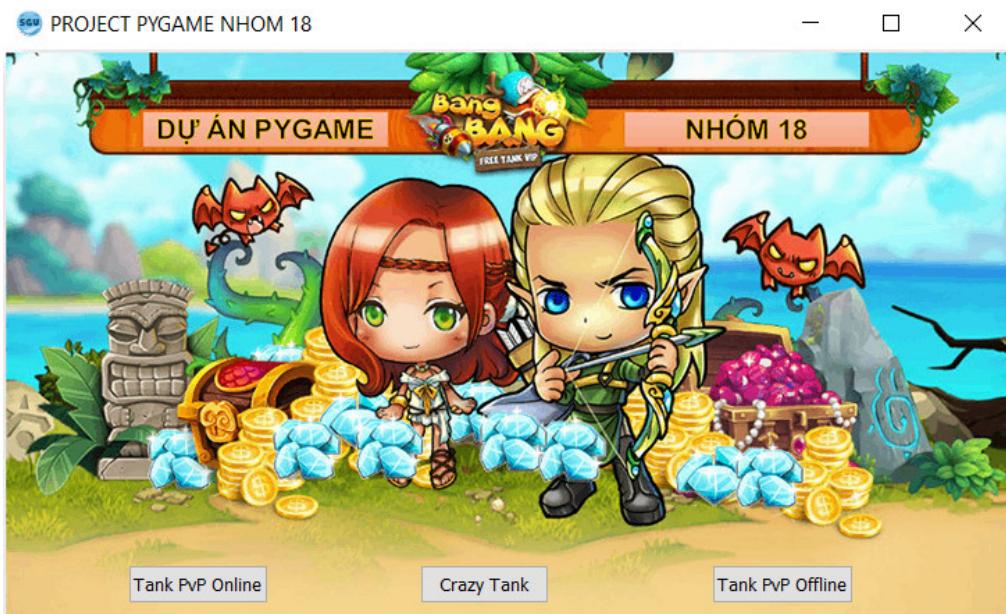
    game_process.wait()
    self.sound.play()
```

Một phương thức được gắn với một slot PyQt, được kích hoạt khi một sự kiện nhất định xảy ra (đừng nhạc nền, khởi động game mà người dùng chọn từ giao diện, khi người dùng thoát game thì phát lại nhạc nền).

Khi ta nhấn nút thoát, chương trình sẽ gọi đến hàm **on_click_exit()** sẽ gọi đến phương thức thoát chương trình.

```
@pyqtSlot()
def on_click_exit(self):
    self.sound.stop()
    QApplication.quit()
```

Giao diện chương trình sau khi hoàn chỉnh:





2 MODE GAME VƯỢT MƯA THIÊN THẠCH

2.1 Giới thiệu

Chào mừng đến với game "Crazy Tank": Bạn đã sẵn sàng tham gia vào một cuộc phiêu lưu ngoài vũ trụ đầy kịch tính và hấp dẫn chưa? Trong trò chơi này, bạn sẽ nhập vai là một chiến binh tinh nhuệ điều khiển chiếc tank mạnh mẽ để chiến đấu chống lại các cơn mưa thiên thạch nguy hiểm.

Nhiệm vụ của bạn là di chuyển tank một cách linh hoạt, né tránh tất cả các thiên thạch trên đường đi để bảo vệ không gian ngoài vũ trụ. Sử dụng kỹ năng và phản xạ nhanh nhạy để vượt qua những thách thức khó khăn và giành chiến thắng.

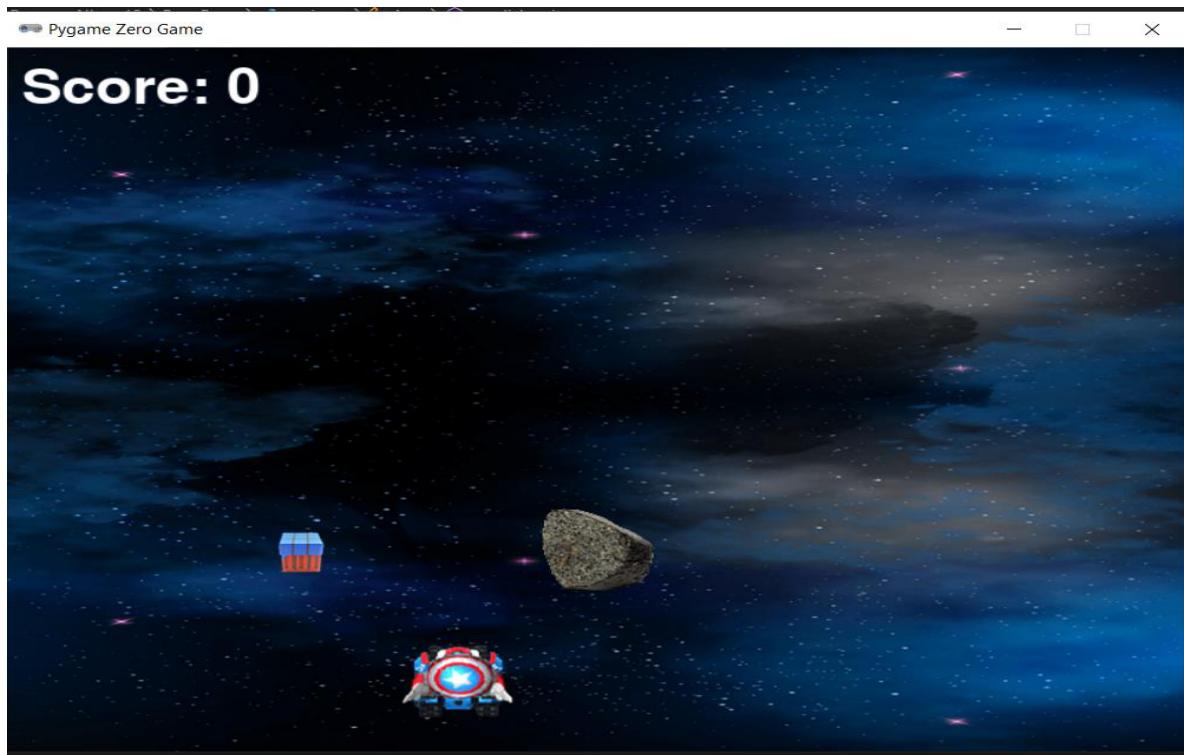
Để tối ưu hóa trò chơi và làm cho cách chơi trở nên dễ dàng hơn, phù hợp với mọi độ tuổi và thiết bị, nhóm em sử dụng cơ chế điều khiển thông qua chuột hoặc cảm ứng trên các điện thoại có màn hình cảm ứng với chỉ một thao tác duy nhất để điều khiển. Điều này giúp người chơi dễ dàng tương tác với trò chơi một cách tự nhiên và thuận tiện, mang lại trải nghiệm chơi game tốt hơn và thú vị hơn. chơi game tốt hơn và thú vị hơn.

2.2 Hướng dẫn cách chơi

Bước 1: Người chơi chỉ cần thao tác điều khiển xe tăng bằng chuột hoặc touchpad trên laptop để di chuyển xe theo hướng mong muốn, tạo cảm giác dễ dàng và linh hoạt khi chơi.



Bước 2: Trong trò chơi "Crazy Tank", người chơi cần nhặt hộp quà để tích điểm, né tránh thiên thạch để sống sót và tích điểm nhanh nhất. Điều này yêu cầu phản xạ nhanh nhạy và kỹ năng điều khiển xe tăng để đạt điểm số cao nhất.

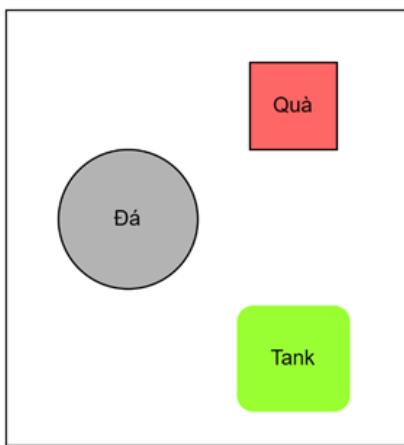


Lưu ý: Khi người chơi thu thập đủ nhiều hộp quà thì tốc độ của trò chơi sẽ tăng lên càng lúc càng nhanh để tăng thêm độ khó cho trò chơi, bên cạnh đó càng ngày theo độ khó của trò chơi âm nhạc sẽ sôi động hơn và nhịp nhanh hơn.

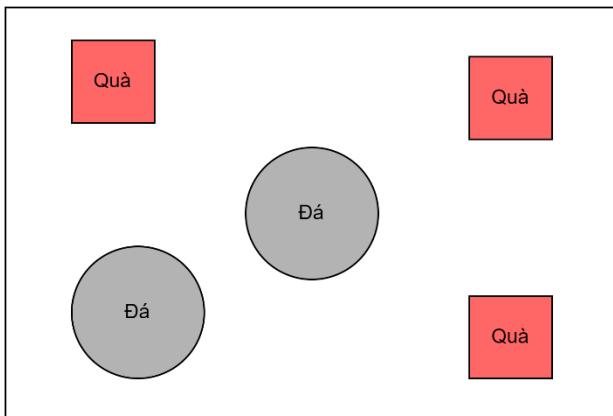
2.3 Ý tưởng

Trong trò chơi, ta có ba đối tượng chính:

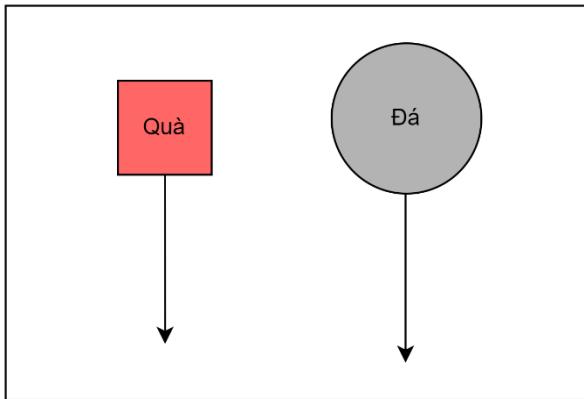
- Xe tăng: Điều khiển bởi người chơi, tự động tiến lên và không thể dừng lại. - Hộp quà: Biểu diễn bằng hộp quà có hình dạng đặc trưng, người chơi cần nhặt để tích điểm.
- Thiên thạch : Được biểu diễn bằng các khối thiên thạch, nguy hiểm và người chơi cần tránh để tránh bị tiêu diệt.



Trong trò chơi, người chơi điều khiển xe tăng để nhặt hộp quà và tránh thiên thạch để tích điểm và duy trì sự sống. Hộp quà và thiên thạch xuất hiện ngẫu nhiên trên bản đồ, tạo sự bất ngờ và thách thức, tăng cường hứng thú và cảm giác thử thách khi chơi.



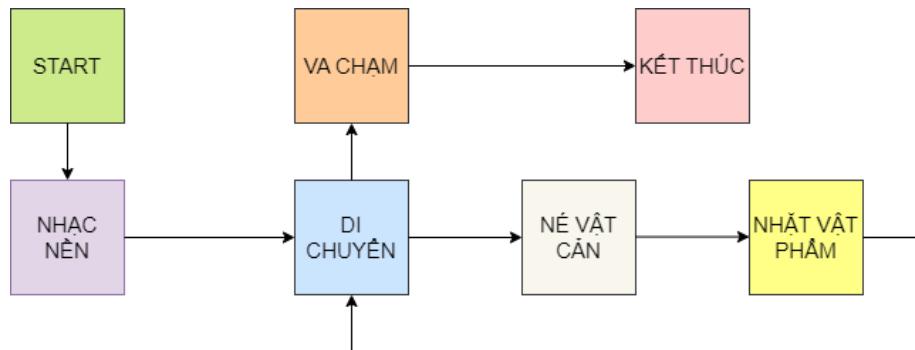
Để tạo cảm giác xe tăng di chuyển tốc độ cao, hộp quà và thiên thạch rơi xuống từ trên xuống, giúp người chơi cảm thấy như đang phải phản xạ nhanh để nhặt hộp quà và tránh thiên thạch. Việc này tạo cảm giác động đất và thú vị khi chơi.



Để hướng người chơi né thiên thạch và nhặt hộp quà, áp dụng các chiến lược như đặt hộp quà gần thiên thạch, tăng tốc độ rơi của hộp quà, và đặt hộp quà ở vị trí chiến lược. Những chiến lược này tạo thách thức và khuyến khích người chơi kết hợp kỹ năng để đạt điểm số cao.

2.4 Mô hình chương trình

Cơ chế quá trình hoạt động



2.5 Quy trình thực hiện

2.5.1 Thư viện cần dùng

```
import pgzrun, random
from random import randint
```

Trong trò chơi sử dụng thư viện **pgzero** của Pygame, việc sử dụng thư viện **random** để xuất hiện các vật thể như hộp quà và thiên thạch một cách ngẫu nhiên giúp tạo ra sự đa dạng và hấp dẫn cho trò chơi. Bằng cách này, người chơi sẽ không gặp phải sự lặp lại và nhảm chán trong trải nghiệm chơi game của họ. Đồng thời, việc sử dụng **pgzero** giúp phát triển trò chơi một cách dễ dàng và hiệu quả cho các trò chơi có quy mô nhỏ và đơn giản.

2.5.2 Cài đặt một số biến cơ bản

```
WIDTH, HEIGHT = 800, 600
WHITE, VIOLET = (255,255,255), (255,0,255)
diem = 0
ket_thuc = False
background.draw()
```

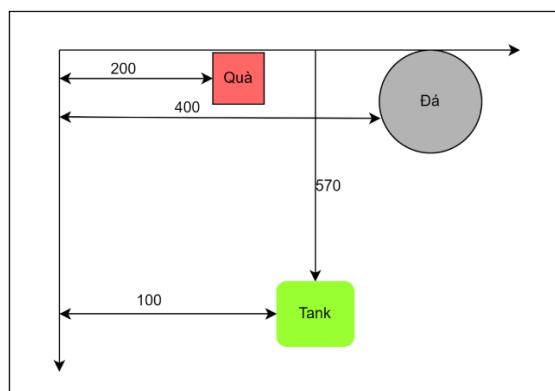
Thiết lập kích thước cửa sổ, màu sắc và tính điểm khi nhặt hộp quà là những điểm quan trọng trong trò chơi, tạo trải nghiệm hấp dẫn cho người chơi.

2.5.3 Định dạng hình ảnh và vị trí

```
# Định dạng hình nền
background = Actor("nengame")

# Định dạng tank, hộp quà, thiên thạch
tank = Actor('captain')
tank.x, tank.y = 100, 550
item = Actor('hopqua')
item.x, item.y = 200, 0
rock = Actor('thienthach')
rock.x, rock.y = 400, 0
```

Ta sẽ gán đường dẫn của các hình ảnh cho các biến để dễ thao tác với chúng, vị trí của chúng được xác định dựa trên trục tọa độ Oxy.



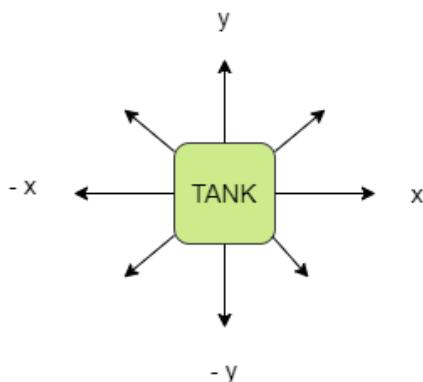
Trong trò chơi, vị trí ban đầu tạm thời của các đối tượng được thiết lập như sau:

- Xe tăng: vị trí ban đầu tại $x = 100$, $y = 550$. Xe tăng sẽ được điều khiển bởi chuột khi người chơi chơi.
- Hộp quà: vị trí ban đầu tại $x = 200$, $y = 0$. Hộp quà sẽ xuất hiện ngẫu nhiên trong trò chơi để người chơi nhặt.
- Thiên thạch: vị trí ban đầu tại $x = 400$, $y = 0$. Thiên thạch cũng sẽ xuất hiện ngẫu nhiên trong trò chơi, tạo thêm thách thức cho người chơi.

2.5.4 Cơ chế điều khiển xe tăng

```
# Set up điều khiển tank bằng chuột
def on_mouse_move(pos,rel,buttons):
    tank.x = pos[0]
    tank.y = pos[1]
```

Trong trò chơi, việc điều khiển xe tăng bằng chuột giúp người chơi linh hoạt di chuyển xe tăng theo vị trí chuột trên màn hình. Các tham số như ‘pos’, ‘rel’, và ‘buttons’ được sử dụng để xác định vị trí và trạng thái của chuột trong quá trình điều khiển.



2.5.5 Tăng tốc độ của trò chơi

```
# Tăng tốc độ rơi của thiên thạch dựa vào điểm mà player đang có
item.y = item.y + 2 + diem / 4
rock.y = rock.y + 2 + diem / 4
```

Trong trò chơi, tốc độ rơi của hộp quà và thiên thạch tăng dần theo điểm số của người chơi. Việc này tạo ra sự thách thức gia tăng khi người chơi cố gắng đạt điểm số cao hơn để đối phó với tốc độ tăng của trò chơi.

2.5.6 Xử lí khi xe tăng nhặt hộp quà

```
# Xe tăng nhặt hộp quà
if item.colliderect(tank):
    item.x = random.randint(20,780)
    item.y = 0
    diem += 1
```



Trong trò chơi, khi xe tăng nhặt hộp quà, vị trí của hộp quà sẽ được thiết lập ngẫu nhiên trên trục x và điểm số của người chơi tăng lên. Điều này tạo ra cơ chế nhặt hộp quà để người chơi tăng điểm khi chơi trò chơi.

```
while rock.colliderect(item):
    rock.x = random.randint(20,780)
    item.x = random.randint(20,780)
```

Ta dùng hàm random để hộp quà xuất hiện ngẫu nhiên theo trục Ox tức là theo chiều ngang, hộp quà sẽ xuất hiện tại vị trí đầu của trục Oy và rơi xuống, xử lý việc khi thiên thạch (rock) va chạm với hộp quà (item) trong trò chơi.

Việc xử lý này giúp tránh tình huống thiên thạch và hộp quà va chạm với nhau, khi đó chúng sẽ được đặt lại vị trí ngẫu nhiên trên màn hình. Điều này giúp tránh tình trạng va chạm liên tục giữa hai đối tượng và tạo ra một trải nghiệm chơi game mượt mà hơn.

Lưu ý: để tránh trường hợp hàm random hoạt động sai, ví dụ chúng random các hộp quà ra khỏi kích thước của bản đồ, ta phải giới hạn vị trí của chúng.

Xử lý giới hạn của màn hình

```
if item.y > 600:
    item.y = 0
```

Xử lý giới hạn màn hình trong một ứng dụng hoặc trò chơi. Nó kiểm tra xem một vật phẩm (item) có vượt quá giới hạn dưới của màn hình không. Nếu tọa độ y của vật phẩm lớn hơn **600** (giả sử **600** là giới hạn dưới của màn hình), nó sẽ đặt lại tọa độ y của vật phẩm thành **0**, làm cho vật phẩm xuất hiện trở lại ở phía trên màn hình. Điều này giúp giữ cho vật phẩm không bị "mất" khỏi màn hình khi nó di chuyển xuống quá nhanh.

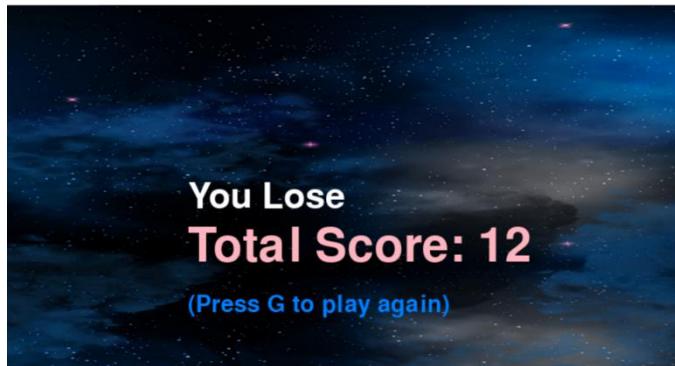
2.5.7 Xử lý khi xe tăng va trúng thiên thạch

```
# Thiên thạch rơi trúng xe tăng
if rock.colliderect(tank):
    ket_thuc = True
    sounds.bum.play()
    sounds.nhacnen.stop()
```

Khi một thiên thạch va chạm với xe tăng, trò chơi sẽ kết thúc. Âm thanh "bum" sẽ phát ra và nhạc nền sẽ được dừng để tạo hiệu ứng cảm xúc và tập trung vào âm thanh va chạm.

```
if ket_thuc:  
    screen.draw.text('You Lose',(210,200), color = WHITE, fontsize=60)  
    screen.draw.text('Total Score: '+ str(diem),(210,250), color = VIOLET, fontsize=80)  
    screen.draw.text('(Press G to play again)',(210,330), color = BLUE, fontsize=40)  
    if keyboard.G:  
        reset_game()  
        sounds.nhacnen.play()  
        sounds.nhacnen.set_volume(0.5)  
    else:  
        screen.draw.text('Score: '+ str(diem),(10,15), color = WHITE, fontsize=60)  
        tank.draw()  
        item.draw()  
        rock.draw()
```

Nếu trò chơi kết thúc, thông báo "**You Lose**" sẽ được hiển thị cùng với điểm số tổng cộng và hướng dẫn chơi lại. Nếu người chơi nhấn **phím "G"**, trò chơi sẽ được thiết lập lại và nhạc nền sẽ được phát lại. Nếu trò chơi đang diễn ra, điểm số hiện tại sẽ được hiển thị và các đối tượng trò chơi sẽ xuất hiện trên màn hình.

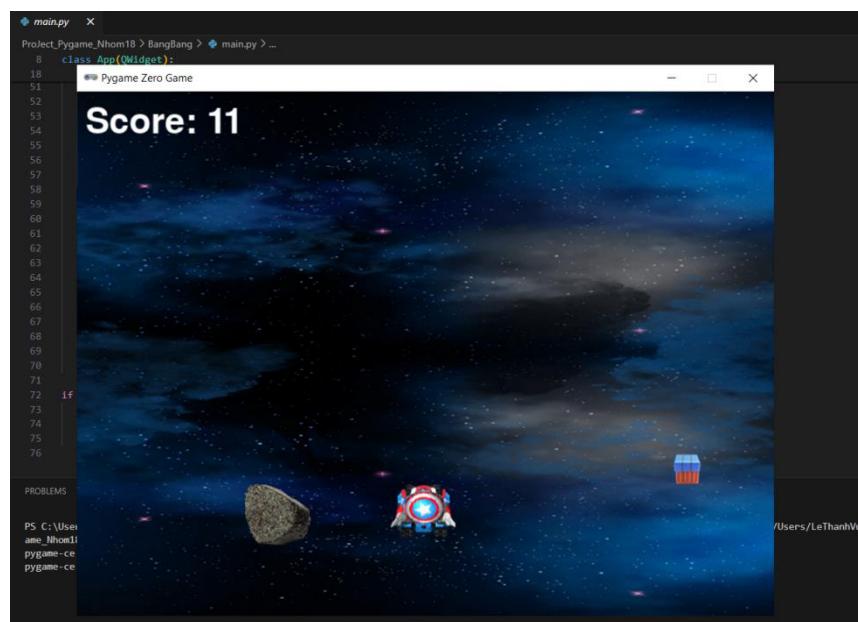


2.5.8 Xử lí cửa sổ game

Trước khi chạy game, set up vị trí cho cửa sổ lúc nó hiện lên sẽ nằm phía trên và ở giữa. Thực hiện thiết lập biến môi trường **SDL_VIDEO_CENTERED** thành 1 và biến môi trường **SDL_VIDEO_WINDOW_POS** thành một chuỗi thẻ hiện vị trí cửa sổ mong muốn.

```
window_position = "{}{}".format(  
    (os.get_terminal_size().columns - WIDTH) // 2,  
    (os.get_terminal_size().lines - HEIGHT) // 2  
)  
  
os.environ['SDL_VIDEO_CENTERED'] = '1'  
os.environ['SDL_VIDEO_WINDOW_POS'] = window_position
```

Sử dụng biến môi trường **SDL_VIDEO_CENTERED** và **SDL_VIDEO_WINDOW_POS** để căn giữa cửa sổ đồ họa trong thiết bị đầu cuối. Điều này đảm bảo cửa sổ được tạo sẽ hiển thị ở vị trí căn giữa trên thiết bị đầu cuối.



2.5.9 Hiệu ứng âm thanh

Nhạc nền: **sounds.nhacnen.play()**

Âm lượng nhạc nền: **sounds.nhacnen.set_volume(0.5)** Va vào thiên thạch: **sounds.bum.play()**
Ngừng nhạc nền: **sounds.nhacnen.stop()**

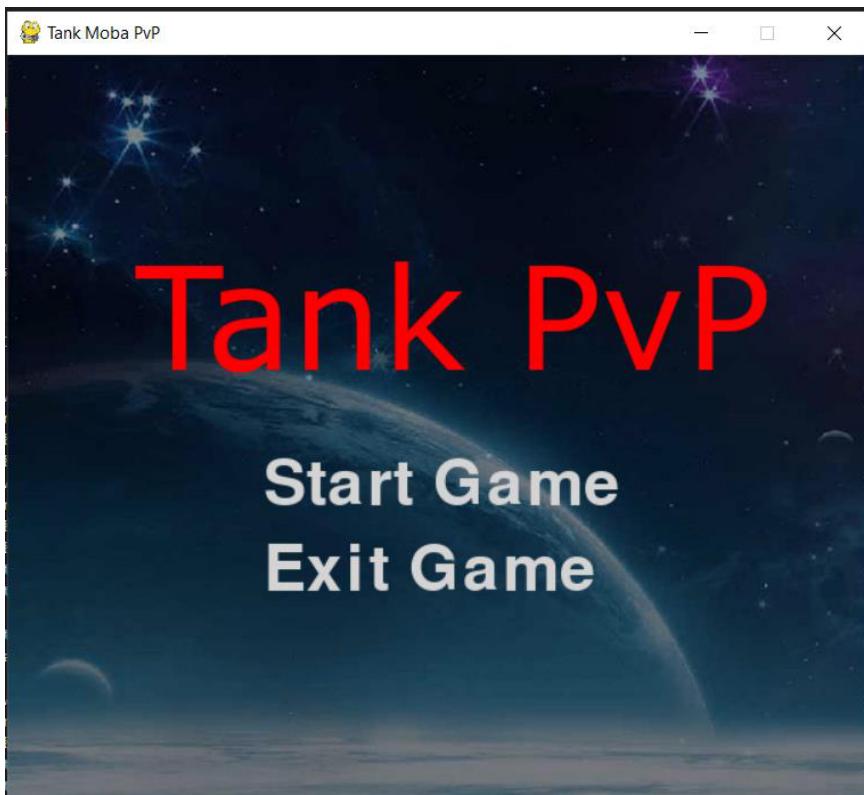


3 MODE TANK PVP OFFLINE

Nhắc đến trò chơi điện tử (hay game), chắc chắn không ai xa lạ với thuật ngữ này, đây là thứ gắn liền với tuổi thơ của biết bao thế hệ, gắn bó với mọi người, chắc chắn ai cũng từng chơi qua một lần. Dạng sơ khai nhất của game chính là các trò chơi như cờ caro được chơi trên giấy, cờ vua, cờ tướng,... Hiện nay trò chơi đã được nâng lên một tầm cao mới, từ khi công nghệ cũng như internet phát triển, các lập trình viên đã mô hình hóa và đưa thành công nó lên các nền tảng thiết bị điện tử. Ngày nay trò chơi điện tử đã không ngừng được cải thiện lối chơi, đồ họa, các thể loại ngày càng hấp dẫn, thu hút đông đảo giới trẻ yêu thích. Khi nói đến game thì nhiều người thường nghĩ rằng nó là một thứ vô bổ, lãng phí thời gian, ảnh hưởng đến health và học tập. Và dĩ nhiên ở chiều ngược lại, các nhà khoa học đã chứng minh rằng, nếu chơi game điều độ, có chừng mực thì sẽ giúp nhóm em xả stress sau những giờ học tập và làm việc căng thẳng, và đối với một số tựa game như Minecraft, Lego còn khiến nhóm em rèn luyện khả năng tư duy sáng tạo cho trẻ.

Trong thế giới hỗn loạn và đầy cạnh tranh này, chào mừng bạn đến với đấu trường PvP hoành tráng, nơi kỹ năng, chiến lược và lòng dũng cảm sẽ được thử thách đến cùng cực. Hãy sẵn sàng để với những đối thủ ngang tài ngang sức, chứng minh bản lĩnh và vươn lên dẫn đầu bảng xếp hạng. Với nhiều chế độ chơi đa dạng, từ đấu đội 5v5 căng thẳng đến đấu đơn 1v1 gay cấn, tựa game tank PvP của nhóm mình mang đến trải nghiệm chơi game đỉnh cao cho mọi game thủ. Tùy chỉnh nhân vật của bạn, trang bị cho họ những vũ khí và áo giáp mạnh mẽ nhất, và bước vào đấu trường để chiến đấu vì vinh quang và phần thưởng. Cho dù bạn là một chiến binh dày dạn kinh nghiệm hay một tân binh háo hức, đấu trường PvP luôn chào đón bạn. Hãy rèn luyện kỹ năng của mình, lập nhóm với bạn bè và chứng minh rằng bạn có những gì cần thiết để trở thành nhà vô địch tối thượng.

Tham gia ngay vào tựa game **PvP** của nhóm mình và trải nghiệm cảm giác hồi hộp khi chiến đấu với những người chơi khác trong thời gian thực. Vinh quang đang chờ đón những người dũng cảm nhất. Hãy sẵn sàng cho trận chiến và chứng minh sức mạnh của bạn!



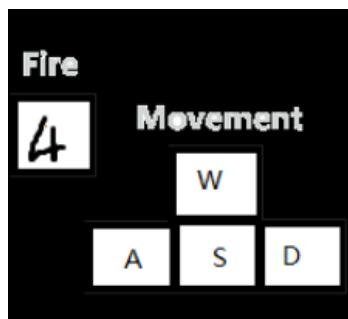
3.1 Hướng dẫn cách chơi

Click **Start Game** để bắt đầu trò chơi.

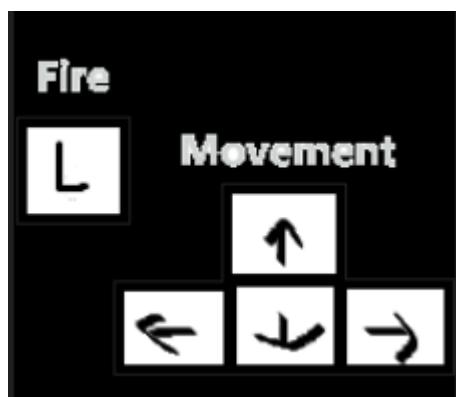


Đến phần chọn tank và phần hướng dẫn dành cho người chơi 1 và người chơi 2.

Đối với người chơi 1 thì sẽ sử dụng các phím chức năng **W A S D** để di chuyển lên, xuống, trái, phải. Phím bắn đạn sẽ là phím số 4.



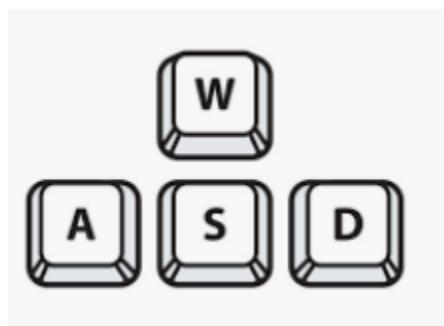
Đối với người chơi 2 thì sẽ sử dụng các phím mũi tên (hãy bật scroll lock nếu bạn không di chuyển được) để di chuyển lên, xuống, trái, phải.



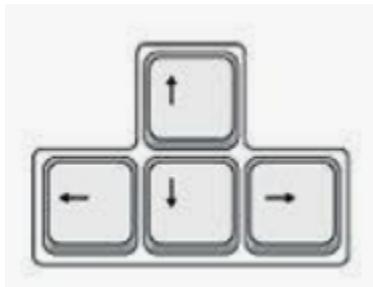
Tiến hành chọn tank:



Đối với người chơi 1 thì dùng phím **A** và **D** để di chuyển qua lại chọn tank yêu thích.

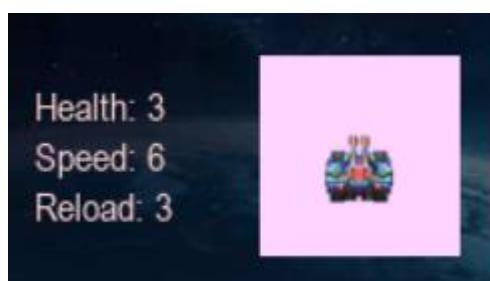


Đối với người chơi 2 thì dùng phím **mũi tên →** để di chuyển qua lại chọn tank yêu thích.

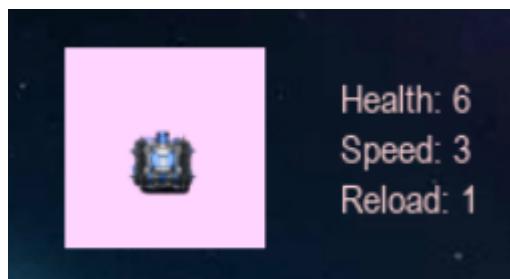


Tùy theo loại tank, mỗi loại đều có chỉ số khác nhau đảm bảo công bằng khi chơi:

-Loại tank ver1 sẽ có ít máu, nhưng tốc độ nhanh, tốc độ nạp đạn chậm



-Loại tank ver2 sẽ có nhiều máu hơn, nhưng tốc độ chậm, tốc độ nạp đạn nhanh.



3.2 Ý tưởng thực hiện

Nhóm quyết định phân chia từng phần của trò chơi thành các hàm riêng biệt do dự án có quy mô khá lớn. Phương pháp này giúp việc giải thích và ghi chú trở nên dễ dàng hơn. Dưới đây là một mô tả tổng quan về pseudocode của trò chơi của chúng tôi. Thông qua việc sử dụng pseudocode và phân chia trò chơi thành các hàm riêng biệt, chúng tôi hy vọng rằng việc hiểu và duy trì mã nguồn của trò chơi sẽ dễ dàng hơn, đồng thời cũng giúp tối ưu hóa quá trình phát triển.

Quá trình chuẩn bị cho trò chơi bắt đầu bằng việc import thư viện pygame, khởi động Pygame, thiết lập Pygame Mixer cho âm thanh, khởi tạo màu sắc và biến font. Sau đó, tạo cửa sổ trò chơi, thiết kế hình nền, nạp âm thanh, hình ảnh bản đồ và xe tăng, đặt điểm xuất hiện xe tăng, tạo nhóm người chơi, nhân vật, thêm xe tăng, quản lý tường và viên đạn, và cuối cùng đặt biến điều khiển cho hàm và vòng lặp trong trò chơi.

Nhóm sẽ ghi điểm vào file điểm khi có, thiết lập số khung hình trên giây để chạy mượt mà, gọi hàm menu để hiển thị giao diện người chơi và xử lý sự kiện chuột. Khi người dùng chọn "chơi trò chơi", hàm menu sẽ kết thúc và dừng phát nhạc nền. Khi chọn "thoát trò chơi", hàm menu

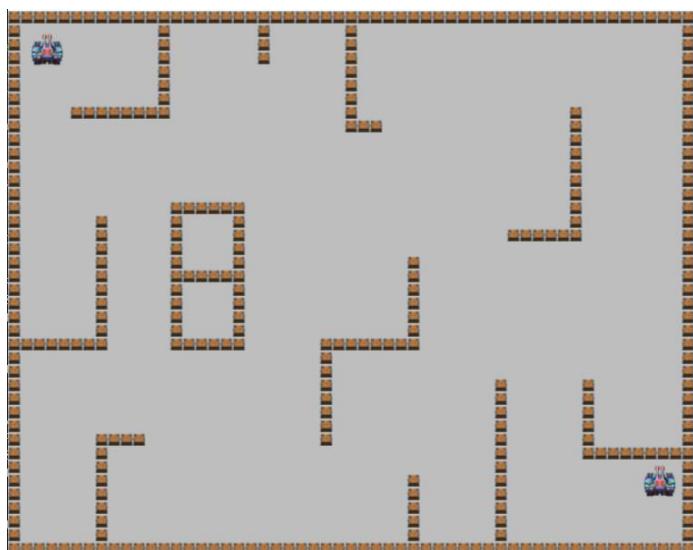
sẽ kết thúc, đặt giá trị false cho biến điều khiển và dừng phát nhạc.

Nhóm tải và phát nhạc nền khi call màn hình lựa chọn xe tăng, hiển thị tên các mục lựa chọn và hình ảnh, yêu cầu người chơi sử dụng phím điều khiển để chuyển đổi xe tăng. Khi người chơi nhấn Enter, xác nhận xe tăng được chọn và cập nhật thông số. Sau đó, chọn bản đồ và kết thúc hàm lựa chọn. Nếu người chơi thoát, hàm sẽ kết thúc và dừng phát nhạc.

Nhóm hẹn giờ cho việc bắn đạn khi màn hình chiến đấu được kích hoạt, sau đó nạp và phát nhạc nền. Trong vòng lặp, họ di chuyển người chơi, xử lý việc bắn đạn và đóng cửa sổ trò chơi. Duyệt qua đạn để xử lý va chạm và hiển thị trạng thái trên màn hình. Khi sức khỏe của người chơi giảm về 0, kết thúc trò chơi và dừng nhạc nền.

Nhóm sẽ tải và phát nhạc nền khi hàm kết thúc trò chơi được gọi, sau đó cập nhật điểm số từ tệp và xác định người chiến thắng. Người chiến thắng sẽ được cộng thêm một điểm và điểm số mới sẽ được ghi lại. Tiêu đề kết quả sẽ được hiển thị trên màn hình và trong vòng lặp, điểm số sẽ được hiển thị và xử lý sự kiện như dừng phát nhạc, thiết lập lại điểm số hoặc kết thúc trò chơi sẽ được thực hiện.

Tạo ra một vùng trống bị giới hạn bởi rìa bản đồ, sau đó thiết lập hai đối thủ (ta và địch), mỗi đối tượng sẽ đặt ở 2 góc bản đồ. Dựa xe tăng của hai người chơi lên bản đồ, player 1 đặt ở góc trái bên trên map, player 2 đặt ở góc dưới bên phải map.



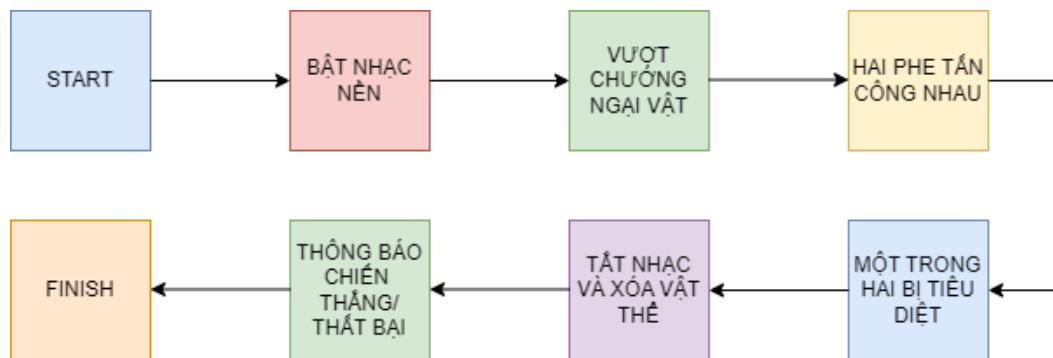
Sau đó 2 chiếc xe tăng sẽ đối đầu lẫn nhau bằng cách bắn đạn nhằm hạ gục đối phương, đồng thời cố gắng di chuyển để né đạn từ địch.

Để tạo ra một trải nghiệm chơi đầy sinh động và hấp dẫn hơn, nhóm em sẽ thiết kế các vách tường gỗ đặc biệt, những bức tường chắn đạn sẽ ngăn cản việc bắn trực tiếp vào nhau. Điều này sẽ tạo ra các tình huống chiến đấu trong game trở nên chân thực hơn, khi người chơi phải tìm cách vượt qua các vách tường này để tiến xa hơn trong trận đấu.



3.3 Mô hình hóa chương trình

3.3.1 Mô tả chi tiết các thành phần cần xử lí



3.3.2 Cơ chế quá trình hoạt động

-Căn chỉnh cửa sổ, kích thước, định dạng xe tăng của người chơi:

- + Căn chỉnh cửa sổ hiển thị để đảm bảo rằng mọi thành phần trên giao diện game được hiển thị một cách rõ ràng và hợp lý.
- + Xử lý việc kích thước của cửa sổ và các thành phần bên trong để đảm bảo sự cân đối và thuận tiện cho người chơi.
- + Định dạng xe tăng của người chơi, bao gồm việc thiết kế và hiển thị xe tăng một cách chân thực và đáng tin cậy trong game.
- + Cơ chế điều khiển sẽ bao gồm việc sử dụng bàn phím để thực hiện các hành động trong trò chơi.
- + Hướng di chuyển của tank sẽ phụ thuộc vào cách người chơi sử dụng các phím điều khiển để di chuyển lên, xuống, trái, phải hay các phím **W A S D**.
- + Vị trí ban đầu của các tank trên bản đồ sẽ được xác định trước khi trò chơi bắt đầu, là ở các vị trí ngẫu nhiên hoặc được đặt sẵn để tạo ra sự cân bằng và chiến lược cho người chơi.

-Thiết lập cơ chế đạn của người chơi:

- + Cơ chế phím bắn của đạn sẽ xử lý việc người chơi sử dụng phím cụ thể để bắn đạn từ xe tăng của mình.
- + Hướng bay của đạn sẽ phụ thuộc vào hướng mà người chơi di chuyển xe tăng và hướng bắn, đảm bảo tính chính xác và logic của việc bắn đạn. + Vị trí theo xe tăng khi bắn đạn sẽ xác định vị trí xuất phát của đạn từ xe tăng, đồng thời cũng ảnh hưởng đến quỹ đạo di chuyển của đạn.
- + Xử lý loại bỏ đạn khi chúng trúng vào tường hay trúng vào đối thủ để đảm bảo tính logic và hiệu quả của trò chơi.
- + Viên đạn sẽ di chuyển theo hướng được xác định bởi thuộc tính direction của nó.
- + Trong mỗi lần cập nhật, viên đạn sẽ di chuyển một khoảng cố định trên màn hình game dựa trên hướng của nó: sang trái, sang phải, lên trên hoặc xuống dưới.



+ Dối với hướng di chuyển sang trái hoặc lên trên, tọa độ của viên đạn sẽ giảm để thực hiện di chuyển.

+ Dối với hướng di chuyển sang phải hoặc xuống dưới, tọa độ của viên đạn sẽ tăng để thực hiện di chuyển.

+ Khoảng cách di chuyển mỗi lần là 6 pixel theo phương ngang hoặc dọc tương ứng với hướng bắn của đạn.

-Định dạng xe tăng của hai người chơi;

+ Để cài đặt cơ chế tự điều khiển, nhóm em cần xử lý việc lập trình để xác định hành vi tự động của các vật thể trong trò chơi, bao gồm việc di chuyển, tấn công và phòng thủ.

+ Cơ chế xoay hướng sẽ giúp tank trong trò chơi tự động nhận biết và phản ứng với các tình huống xung quanh, bao gồm việc xoay hướng để tấn công hoặc tránh né đạn của đối thủ.

+ Vị trí ban đầu trên map và khoảng cách giữa hai người chơi với nhau sẽ ảnh hưởng đến chiến thuật và cách tiếp cận trong trò chơi, cần được xác định một cách cẩn nhắc để tạo ra trải nghiệm chơi game hấp dẫn và thú vị.

-Thiết lập cơ chế bắn đạn của hai người chơi:

+ Xác định các phím điều khiển cho việc bắn đạn của từng người chơi, ví dụ: người chơi 1 sử dụng phím 4 và người chơi 2 sử dụng phím L.

+ Khi người chơi thực hiện hành động bắn, hàm sẽ tạo ra viên đạn. Kích thước của viên đạn được thiết lập là 5x5 pixel.

+ Màu sắc của đạn chủ yếu là đỏ với giá trị RGB là (255,0,0).

+ Vị trí ban đầu của viên đạn được thiết lập tùy theo hướng mà tank của người chơi đang quay mặt: lên trên, xuống dưới, sang trái, hay sang phải.

+ Viên đạn sẽ được đặt tại một vị trí sao cho vừa khít với giữa nòng súng tank khi nó được bắn ra.

+ Sau khi thiết lập, viên đạn sẽ được thêm vào nhóm các đối tượng đạn đã được tạo sẵn trong trò chơi để quản lý.

+ Thiết lập hướng bắn và vận tốc đạn dựa trên hướng mà người chơi đặt xe tăng của mình và hướng bắn từ các phím điều khiển tương ứng.

+ Xử lý va chạm giữa đạn của hai người chơi để xác định việc trúng đích và xuất điểm.

+ Đảm bảo tính cân bằng giữa cơ chế bắn đạn của hai người chơi để tạo ra trải nghiệm chơi game công bằng và hấp dẫn.

-Xuất thông báo ra màn hình khi có một người chơi chiến thắng hoặc thua:

+ Đọc số điểm của hai người chơi từ file BangTinhDiem.txt, lưu trữ dưới dạng dictionary với key là tên người chơi và value là số điểm đã chuyển thành số nguyên.

+ So sánh lượng máu còn lại của hai người chơi để xác định người thắng cuộc (người còn nhiều máu hơn).



+ Hiển thị thông báo "PLAYER 1 WINS" với màu sắc cụ thể nếu người chơi 1 thắng, và tương tự, thông báo "PLAYER 2 WINS" nếu người chơi 2 thắng. Thông báo này được hiển thị với màu vàng.

+ Cập nhật điểm số trong dictionary cho người chơi chiến thắng bằng cách tăng số điểm của họ lên 1.

+ Xuất điểm số mới được cập nhật vào file BangTinhDiem.txt.

+ Hiển thị tiêu đề "END OF THE MATCH" với màu đỏ để cho biết trận đấu đã kết thúc.

+ Hiển thị tiêu đề "Scores" với màu tím hồng, thông báo điểm số hiện tại của cả hai người chơi.

+ Xuất thông báo hướng dẫn "Press R to reset scores!" với màu vàng, hướng dẫn người chơi nếu muốn đặt lại điểm số.

+ Xuất thông báo hướng dẫn "Press ENTER to return to game menu." với màu vàng, hướng dẫn người chơi quay trở lại menu chính của trò chơi.

-Thêm vào các hiệu ứng như âm thanh, giao diện (nếu có):

+ Bật nhạc nền khi vào game và tắt khi kết thúc game:

Khi vào game sử dụng hàm, thư viện âm thanh để phát nhạc nền.

Khi kết thúc game, dừng phát nhạc nền.

+ Phát ra âm thanh khi ta hoặc địch bắn:

Khi người chơi bắn, nhóm em sử dụng hàm, thư viện âm thanh để phát ra âm thanh tương ứng.

+ Phát ra âm thanh khi người chơi bị trúng đạn:

Khi kẻ địch hoặc người chơi bị trúng đạn, nhóm em cũng sử dụng hàm thư viện âm thanh để phát ra âm thanh báo hiệu.

Bằng cách này, việc sử dụng âm thanh trong trò chơi không chỉ tạo ra trải nghiệm âm nhạc hấp dẫn mà còn giúp người chơi dễ dàng nhận biết các tình huống quan trọng trong trò chơi.



3.4 Quy trình thực hiện

3.4.1 Thư viện cần dùng

```
import Math import pygame
from pygame.locals import *
import random
import time
pygame.init()
```

- **Math:** Thư viện toán học chuẩn của Python, cung cấp các hàm toán học như lũy thừa, căn bậc hai, các hàm lượng giác, v.v.
- **pygame:** Thư viện chuyên cho việc phát triển trò chơi, hỗ trợ tạo cửa sổ, xử lý đồ họa, âm thanh, và xử lý sự kiện.
- **from pygame.locals import :** Nhập tất cả các hằng số có sẵn từ module **pygame.locals**, thường được sử dụng để dễ dàng tham chiếu tới các nút bấm, hằng số hệ thống mà không cần phải gọi module **pygame.locals**.
- **random:** Thư viện hỗ trợ tạo ra các số ngẫu nhiên, sử dụng cho việc lựa chọn ngẫu nhiên, xáo trộn dữ liệu, v.v.
- **time:** Thư viện cung cấp các hàm để làm việc với thời gian, bao gồm đo hạn, đợi (sleep), và xuất lại thời gian hệ thống, v.v.

Với dòng lệnh **pygame.init()**, thư viện pygame được khởi tạo, đây là bước cần thiết để bắt đầu làm việc với pygame.

3.4.2 Căn chỉnh cửa sổ

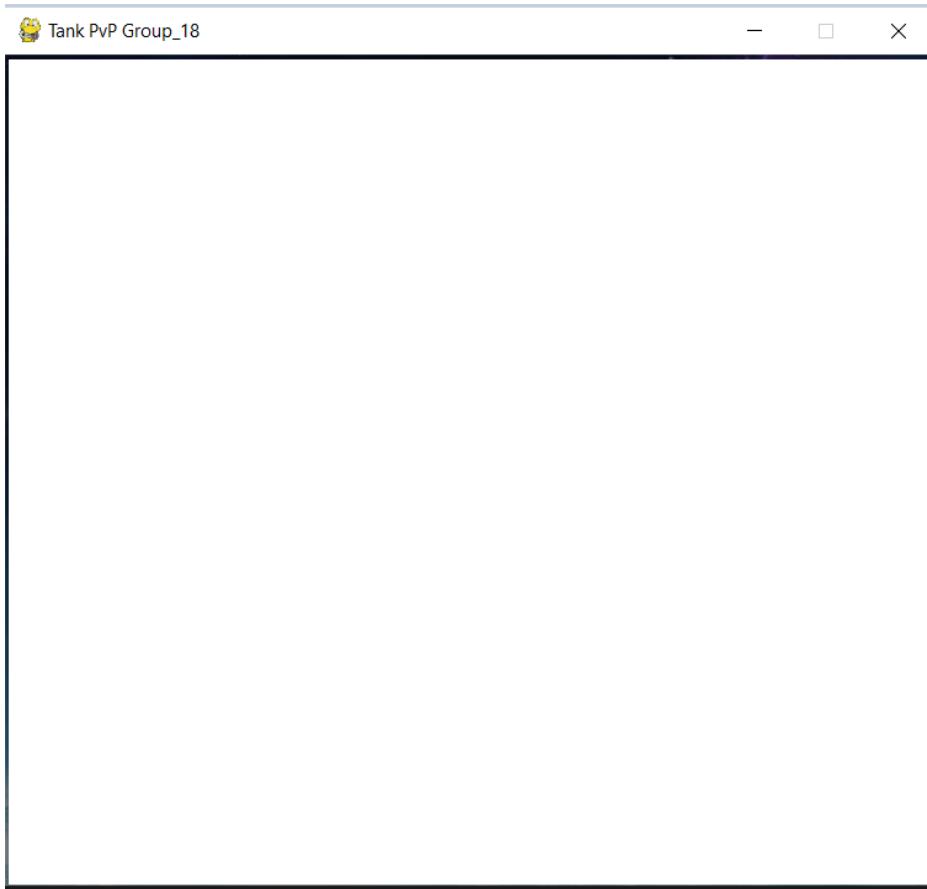
```
size = (605, 540) #kích thước cửa sổ game
screen = pygame.display.set_mode(size)
screenPIC =
pygame.image.load('Proiect_Pygame_Nhom18/BangBang/Tank2P/picture/anhnen.png')
pygame.display.set_caption("Tank PvP Group_18")
background = pygame.Surface(size)
background = background.convert()
background.fill(colours['grey'])
```

- Thiết lập kích thước cửa sổ game là **605x540 pixel**.
- Tạo màn hình hiển thị với kích thước đã thiết lập.
- Tải hình ảnh nền từ đường dẫn '**'ProiectPygameNhom18/BangBang/Tank2P/picture/anhnen.png'** và lưu vào biến **screen-PIC**.
- Đặt tiêu đề cửa sổ game là "**Tank PvP Group_18**".
- Tạo một đối tượng **Surface** mới với kích thước cửa sổ game.
- Chuyển đổi đối tượng **Surface** vừa tạo để tối ưu hiệu suất hiển thị.
- Điều màu nền cho đối tượng **Surface** này bằng giá trị màu 'grey' từ dictionary màu **colours**.

Ở đây ta chọn kích thước của cửa sổ chương trình, trong đó chiều dài và



chiều rộng lần lượt là **605 x540**, đây là kích thước tiêu chuẩn cho game 2D, ngoài ra còn phù hợp với kích thước của hình nền.



3.4.3 Thiết kế giao diện menu game

Xử lý sự kiện cho lựa chọn của người chơi trong menu game thông qua click chuột, tập trung vào tương tác chuột và chức năng thoát. Nó xác định các hàm để tăng cường menu game bằng cách thiết lập nhạc nền, hiển thị tiêu đề game với các lựa chọn "**Start Game**" và "**Exit Game**" được làm nổi bật khi di chuột qua, quản lý sự kiện chuột để bắt đầu hoặc thoát game dựa trên lựa chọn của người chơi. Nó cũng bao gồm việc tải và phát nhạc nền, xuất tiêu đề game "**Tank PvP**" màu đỏ, tạo và đặt các nút "**Start Game**" và "**Exit Game**" trên màn hình. Sử dụng vòng lặp while để theo dõi vị trí chuột, thay đổi hiệu ứng văn bản cho lựa chọn dựa trên vị trí chuột của người chơi, xuất nền và văn bản trò chơi trên màn hình, và cập nhật hiển thị.

Việc thiết lập cơ bản của hàm gameMenu tạo nền tảng cho một màn hình menu thân thiện với người dùng và hấp dẫn về mặt hình ảnh. Bằng cách kết hợp các yếu tố động, cải thiện âm thanh và hình ảnh, và tính năng tương tác, người chơi được cung cấp một chuyển đổi mượt mà vào trải nghiệm chơi game. Thiết kế chu đáo này không chỉ tối ưu hóa điều hướng mà còn tạo ra không khí cho một phiên chơi game sâu sắc.



3.4.3.a Khởi tạo và hiển thị menu game

```
def gameMenu(thisStage):
    global run, selecting, battling, end
    pygame.mixer.music.load(musicList[0])
    pygame.mixer.music.set_volume(0.4)
    pygame.mixer.music.play(-1)

    Title = my_font.render('Tank PvP', True, (255, 0, 0)) # set up
    tiêu đề game màu đỏ

    text_rect = Title.get_rect(center=(screen.get_width() / 2,
    screen.get_height() / 4))

    screen.blit(Title, text_rect)

    Start_label = my_font2.render('Start Game', True, (212,212,212))
    Exit_label = my_font2.render('Exit Game', True, (212,212,212))

    Srect = Rect(210,280,421,339)           #hiệu ứng khi trỏ chuột vào
    Erect = Rect(245,340,380,430)
```

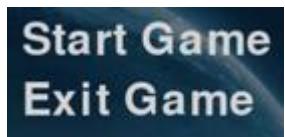
Hàm **gameMenu** trong ứng dụng Pygame đóng vai trò quan trọng trong việc thiết lập màn hình **menu** chính cho trò chơi. Ở trung tâm của hàm **gameMenu** là tham số **thisStage**, cho phép xác định các giai đoạn hoặc trạng thái cụ thể trong giao diện **menu**. Tham số này phục vụ như một yếu tố linh hoạt thích ứng với tương tác của người dùng, hướng dẫn họ qua menu một cách mượt mà.

Việc khai báo các biến toàn cục như **run**, **selecting**, **battling** và **end** cung cấp cho hàm sự linh hoạt để sửa đổi các giá trị này trong quá trình thực thi. Những biến này hoạt động như các cờ kiểm soát, ảnh hưởng đến hành vi của **menu** dựa trên đầu vào của người dùng và sự kiện hệ thống.

Kết hợp âm nhạc vào trải nghiệm chơi game tăng cường không khí tổng thể. Bằng cách tải bản nhạc khởi đầu từ **musicList** và đặt âm lượng của nó là **0.4**, hàm **gameMenu** tạo điều kiện cho một bản nhạc nền hấp dẫn. Bắt đầu phát nhạc trong một vòng lặp đảm bảo một bản nhạc liên tục và cuốn hút suốt quá trình tương tác với **menu**. Các yếu tố hình ảnh của menu đóng vai trò quan trọng trong việc thu hút sự chú ý của người chơi. Tạo một bề mặt cho tiêu đề "**Tank PvP**" với màu đỏ nổi bật không chỉ thu hút mắt mà còn xác định danh tính của trò chơi. Đặt tiêu đề ở giữa màn hình tăng cường sự nổi bật của nó, biến nó thành một điểm trọng tâm cho người chơi.



Việc bổ sung các yếu tố tương tác như nhãn **Start Game** và **Exit Game** thêm chức năng vào menu. Những văn bản này, hiển thị trong **gam màu xám** tinh tế, khuyến khích người chơi bắt đầu trò chơi hoặc thoát ứng dụng. Ngoài ra, xác định các vùng hình chữ nhật (**Srect** và **Erect**) cho phép phát hiện hover và click, hỗ trợ tương tác và phản hồi từ người dùng.





3.4.3.b Xử lý sự kiện thoát khỏi game và chọn tùy chọn trong menu

while thisStage:

```
x, y = pygame.mouse.get_pos()
Opt = 0
if Srect[0] < x < Srect[2] and Srect[1] < y < Srect[3]:
    my_font2.set_underline(True)
    Start_label = my_font2.render('Start Game', True,
(255,255,0))
    screen.blit(Start_label, (183,280))
    pygame.display.flip()

elif Erect[0] < x < Erect[2] and Erect[1] < y < Erect[3]:
    my_font2.set_bold(True)
    Exit_label = my_font2.render('Exit Game', True,
(255,0,0))
    screen.blit(Exit_label, (170, 340))
    pygame.display.flip()

Opt = 2

for ev in pygame.event.get():
    if ev.type == QUIT:
        pygame.mixer.music.stop()
        run = False
        starting = False
        selecting = False
        battling = False
        end = False
        thisStage = False

    elif ev.type == MOUSEBUTTONDOWN:
        if Opt == 1:
            pygame.mixer.music.stop()
            starting = False
            thisStage = False
        elif Opt == 2:
            pygame.mixer.music.stop()
            run = False
            starting = False
            selecting = False
            battling = False
            end = False
            thisStage = False
```



Thiết lập của vòng lặp cho menu trò chơi. Quản lý việc hiển thị và tương tác của người dùng với menu chính, xử lý việc bắt đầu hay thoát trò chơi dựa trên đầu vào từ chuột. Trong vòng lặp while này, các sự kiện chuột và tương tác người dùng được xử lý:

- Hàm `pygame.mouse.get_pos()` lấy vị trí hiện tại của con trỏ chuột.
- **Opt** được khởi tạo với giá trị **0**, có thể được sử dụng để theo dõi lựa chọn của người chơi trong menu.
- if `Srect[0] < x < Srect[2] and Srect[1] < y < Srect[3]`: kiểm tra nếu con trỏ chuột đang ở bên trên nút "**Start Game**". Nếu đúng, thì chữ "**Start Game**" được gán dấu gạch dưới và màu của nó thay đổi thành màu vàng, được thể hiện bằng nhãn **Start_label**.



- elif `Erect[0] < x < Erect[2] and Erect[1] < y < Erect[3]`: thực hiện việc kiểm tra tương tự cho nút "**Exit Game**". Nếu đúng, thì chữ "**Exit Game**" được in đậm và màu đỏ, được thể hiện bằng nhãn **Exit_label**. Biến Opt được thiết lập là 2.



- Các thiết lập nền (`screen.blit(screenPIC, (0,0))`) và tiêu đề (`screen.blit>Title, (90,120))`) cùng với các nhãn "**Start Game**" và "**Exit Game**" được xuất lên màn hình, và màn hình được cập nhật liên tục bằng hàm `pygame.display.flip()`.
- Sau đó, các sự kiện từ `pygame.event.get()`. Nếu sự kiện là **QUIT**, nhạc nền sẽ dừng chơi và tắt cả các flag như **run**, **starting**, **selecting**, **battling**, **end** và **thisStage** đều được thiết lập về **False**, vừa để dừng vòng lặp và vừa để thoát khỏi **menu**.
- Khi sự kiện **MOUSEBUTTONDOWN** (nhấp chuột) xảy ra, tuỳ thuộc vào biến Opt mà hành động tiếp theo sẽ được quyết định:
 - + Nếu **Opt** bằng 1 (có thể bạn không hiển thị đoạn code liên quan đến việc thiết lập Opt thành 1), có thể sẽ bắt đầu trò chơi.
 - + Nếu **Opt** bằng 2, nhạc nền dừng chơi và tắt cả các biến, tương tự như trong sự kiện **QUIT**, được thiết lập về False để thoát kịp thời.



3.4.4 Thiết lập màn hình Select Tank

```
def selectionScreen(thisStage):
    global run, selecting, battling, end

    pygame.mixer.music.load(musicList[1])
    pygame.mixer.music.set_volume(0.4)
    pygame.mixer.music.play(-1)

    Opt1 = 0
    Opt2 = 0

    instructionP1 = pygame.image.load('Project_Pygame_Nhom18/BangBang/Tank2P/picture/moveP2.png')
    instructionP2 = pygame.image.load('Project_Pygame_Nhom18/BangBang/Tank2P/picture/moveP1.png')

    Title = my_font3.render('Select Tank', True, (9,255,242))
    Title_rect = Title.get_rect()
    Title_rect.centerx = screen.get_rect().centerx
    screen.blit(Title, Title_rect)

    p1Title = my_font4.render('P1', True, (255,255,0))
    p2Title = my_font4.render('P2', True, (255,255,0))

    instruction = my_font4.render('Use your left and right controls to switch
between tanks.', True, (255,255,0))
    instruction2 = my_font4.render('Press ENTER to start the battle !!!',
True, (255,255,0))

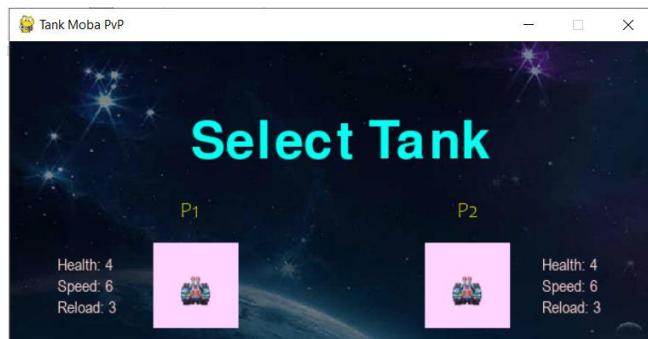
    options = [imageUp_v1,imageUp_v2]
    stats = {0:['4', '6', '3'], 1:['6','3','1']}

    DisplaySurf = pygame.Surface((80,80))
    DisplaySurf.fill((255,212,255))
    DisplaySurf2 = pygame.Surface((80,80))
    DisplaySurf2.fill((255,212,255))

    while thisStage:
        for ev in pygame.event.get():
            if ev.type == QUIT:
                pygame.mixer.music.stop()
                run = False
                selecting = False
                battling = False
                end = False
                thisStage = False
            elif ev.type == KEYDOWN:
```

```
pygame.mixer.music.stop()
readMap()
player_classes(Opt1, player1)
player_classes(Opt2, player2)
selecting = False
thisStage = False
elif ev.key == K_a:
    if Opt1 == 0:
        Opt1 = 1
    else:
        Opt1 -= 1
elif ev.key == K_d:
    if Opt1 == 1:
        Opt1 = 0
    else:
        Opt1 += 1
elif ev.key == K_LEFT:
    if Opt2 == 0:
        Opt2 = 1
    else:
        Opt2 -= 1
elif ev.key == K_RIGHT:
    if Opt2 == 1:
        Opt2 = 0
    else:
        Opt2 += 1
```

Quản lý quá trình chọn xe tăng và cài đặt các thông số trước khi bắt đầu trò chơi. Lựa chọn và thao tác được thực hiện thông qua các phím điều hướng và phím **ENTER**. Hàm này xử lý màn hình lựa chọn xe tăng trước khi bắt đầu trận chiến.



Ta thiết lập biến toàn cục gồm **run**, **selecting**, **battling**, và **end**. Hàm tải và phát nhạc nền từ **musicList[1]** với âm lượng đã được đặt là 0.4 và lặp lại vô hạn.

Opt1 và **Opt2** được khởi tạo để lưu trữ lựa chọn xe tăng của người chơi 1 và người chơi 2. Các hướng dẫn và tiêu đề cho người chơi được tải và tạo từ các file ảnh và đối tượng font có sẵn.

Các biến như **Title**, **p1Title**, **p2Title**, **instruction**, và **instruction2** được tạo để hiển thị tiêu



dè màn hình lựa chọn, hướng dẫn người chơi v.v..., **options** là mảng chứa hình ảnh đầu tiên (hình ảnh ở phía trên) của mỗi loại xe tăng để người chơi lựa chọn. **Stats** là một từ điển chứa thông tin về từng xe tăng, bao gồm "health," "tốc độ đạn," và "tốc độ di chuyển." **DisplaySurf** và **DisplaySurf2** là các đối tượng **Surface** được tạo ra để hiển thị thông tin về xe tăng được chọn; chúng được tô màu để phân biệt.



Hàm chứa vòng lặp **while thisStage** để tiếp tục hiển thị màn hình lựa chọn cho đến khi người chơi hoàn tất việc lựa chọn hoặc thoát khỏi game. Trong vòng lặp, hàm xử lý sự kiện từ **pygame.event.get()**:

- Nếu sự kiện là **QUIT**, nhạc nền sẽ dừng, tất cả các biến trạng thái sẽ được đặt lại và **thisStage** sẽ trở thành **False** để thoát khỏi màn hình lựa chọn.
- Nếu sự kiện là **KEYDOWN**, tuỳ thuộc vào phím được nhấn mà các lựa chọn **Opt1** và **Opt2** (chỉ số lựa chọn xe tăng) sẽ thay đổi:
 - **K_RETURN**: nhạc nền dừng, tải bản đồ game (**readMap()**) không được hiển thị trong đoạn vừa rồi nhưng giả định có liên quan đến cài đặt trận chiến), áp dụng lựa chọn xe tăng vào đối tượng player1 và player2 rồi thoát màn hình lựa chọn.
 - **K_a** và **K_d**: thay đổi lựa chọn của người chơi 1 (P1).
 - **K_LEFT** và **K_RIGHT**: thay đổi lựa chọn của người chơi 2 (P2).





Tạo hiển thị thông tin tank và hướng dẫn:

```
p1statsHealth = my_font5.render('Health: ' + stats[Opt1][0], True,
(255,212,212))
    p1statsSpeed = my_font5.render('Speed: ' + stats[Opt1][1],True,
(255,212,212))
    p1statsReload = my_font5.render('Reload: ' + stats[Opt1][2], True,
(255,212,212))
    p2statsHealth = my_font5.render('Health: ' + stats[Opt2][0], True,
(255,212,212))
    p2statsSpeed = my_font5.render('Speed: ' + stats[Opt2][1],True,
(255,212,212))
    p2statsReload = my_font5.render('Reload: ' + stats[Opt2][2], True,
(255,212,212))

screen.blit(screenPIC, (0,0))
screen.blit>Title, (170, 70))

screen.blit(p1Title, (160, 150))
screen.blit(p2Title, (420, 150))

screen.blit(p1statsHealth, (45,200))
screen.blit(p1statsSpeed, (45,220))
screen.blit(p1statsReload, (45,240))

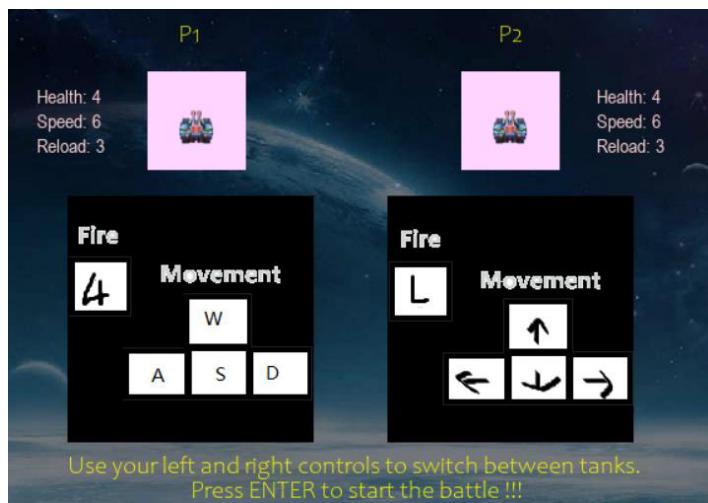
screen.blit(p2statsHealth, (500,200))
screen.blit(p2statsSpeed, (500,220))
screen.blit(p2statsReload, (500,240))

screen.blit(DisplaySurf, (135, 189))
screen.blit(DisplaySurf2, (390, 189))
screen.blit(options[Opt1], (160, 220))
screen.blit(options[Opt2], (415, 220))

screen.blit(instructionP1, (70, 290))
screen.blit(instructionP2, (330, 290))

screen.blit(instruction, (70, 500))
screen.blit(instruction2, (170, 520))
pygame.display.flip()
```

Người dùng thấy được tất cả những lựa chọn và thông tin liên quan trước khi bắt đầu trận chiến, tạo nên giao diện người dùng cuối của màn hình lựa chọn xe tăng trong game.



3.4.5 Định dạng xe tăng

```
players = pygame.sprite.Group()
player1 = pygame.sprite.Sprite()
player1.rect = pygame.Rect((20, 20),(24,24)) player1.direction = 'up'
player1.keys = (K_a, K_d, K_w, K_s,K_4)

player2 = pygame.sprite.Sprite()
player2.rect = pygame.Rect((560, 390), (24,24)) player2.direction = 'up'
player2.keys = (K_LEFT, K_RIGHT, K_UP, K_DOWN,K_l)

players.add(player1)
players.add(player2)

bulletgroup = pygame.sprite.Group()
walls = pygame.sprite.Group()
```

- Khai báo hai danh sách **ver1** và **ver2** chứa các hình ảnh khác nhau cho hai phiên bản người chơi, với các hướng di chuyển khác nhau được biểu diễn qua các hình ảnh lên, xuống, phải và trái.



- Khởi tạo một danh sách **spawnpoints** chứa các vị trí xuất hiện trên màn hình cho các viên đạn, mỗi vị trí được biểu diễn bởi một cặp tuple chỉ định tọa độ x và y.

Sau đó code tiếp tục với việc tạo ra nhóm đối tượng:

- Tạo nhóm players sử dụng **pygame.sprite.Group()** để quản lý các đối tượng người chơi.
- Tạo ra hai đối tượng player1 và player2 bằng cách sử dụng

pygame.sprite.Sprite(). Mỗi đối tượng người chơi được cấp một hình chữ nhật (**rect**) để xác định vị trí và kích thước, một thuộc tính **direction** để biết hướng di chuyển, và một set các **keys** để xác định các phím điều khiển di chuyển và bắn đạn.

- Thêm hai đối tượng **player1** và **player2** vào nhóm **players** bằng phương thức **add**.



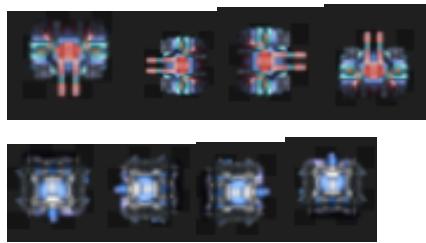
Tiếp theo, code khởi tạo hai nhóm đối tượng khác:

- Nhóm **bulletgroup** sử dụng **pygame.sprite.Group()**, dùng để quản lý các viên đạn trong trò chơi.
- Nhóm **walls** cũng dùng **pygame.sprite.Group()**, dùng để quản lý các bức tường trong game.

Các nhóm này được sử dụng để quản lý và cập nhật trạng thái các đối tượng tương ứng trong trò chơi, giúp dễ dàng thao tác xuất, kiểm tra va chạm và xử lý sự kiện hơn.

3.4.5.a Định dạng và đưa hình ảnh 2 tank lên cửa sổ

```
imageUp_v1 =  
pygame.image.load('ProJect_Pygame_Nhom18/BangBang/Tank2P/picture/tank1  
up.png')  
  
imageDown_v1 =  
pygame.image.load('ProJect_Pygame_Nhom18/BangBang/Tank2P/picture/tank1  
down.png')  
  
imageRight_v1 =  
pygame.image.load('ProJect_Pygame_Nhom18/BangBang/Tank2P/picture/tank1  
right.png')  
imageLeft_v1 =  
pygame.image.load('ProJect_Pygame_Nhom18/BangBang/Tank2P/picture/tank1  
left.png')  
  
imageUp_v2 =  
pygame.image.load('ProJect_Pygame_Nhom18/BangBang/Tank2P/picture/tank2  
up.png')  
imageDown_v2 =  
pygame.image.load('ProJect_Pygame_Nhom18/BangBang/Tank2P/picture/tank2  
down.png')  
imageRight_v2 =  
pygame.image.load('ProJect_Pygame_Nhom18/BangBang/Tank2P/picture/tank2  
right.png')  
imageLeft_v2 =  
pygame.image.load('ProJect_Pygame_Nhom18/BangBang/Tank2P/picture/tank2  
left.png')  
  
ver1 = [imageUp_v1,imageDown_v1,imageRight_v1,imageLeft_v1] ver2 = [imageUp_v2,imageDown_v2]  
spawnpoints = [(30,30),(40,390),(560,130),(560,390)]
```



Tải các hình ảnh cần thiết cho trò chơi, cụ thể là các hình ảnh của xe tăng từ các phiên bản và hướng khác nhau:

- Dùng **pygame.image.load** để tải các hình ảnh xe tăng từ đường dẫn đã cho cho cả hai phiên bản xe tăng (**ver1** và **ver2**). Mỗi phiên bản xe tăng có hình ảnh ứng với hướng di chuyển: lên (up), xuống (down), sang phải (right), sang trái (left).



- Tạo hai danh sách **ver1** và **ver2** để lưu trữ các hình ảnh đã tải. Mỗi danh sách chứa bốn hình ảnh tương ứng với bốn hướng di chuyển của một phiên bản xe tăng.

- Khai báo một danh sách **spawnpoints** gồm các cặp tọa độ, mỗi cặp đại diện cho điểm xuất hiện ban đầu của xe tăng trên màn hình trò chơi.

Danh sách **ver1** và **ver2** sẽ được sử dụng để thay đổi hình ảnh của xe tăng phụ thuộc vào hướng di chuyển hiện tại của nó, giúp xe tăng hiển thị chính xác hình ảnh theo hướng di chuyển.

Tọa độ trong **spawnpoints** sẽ được sử dụng để đặt vị trí bắt đầu của các xe tăng khi bắt đầu game hoặc khi một xe tăng mới được tạo ra sau khi bị hủy.



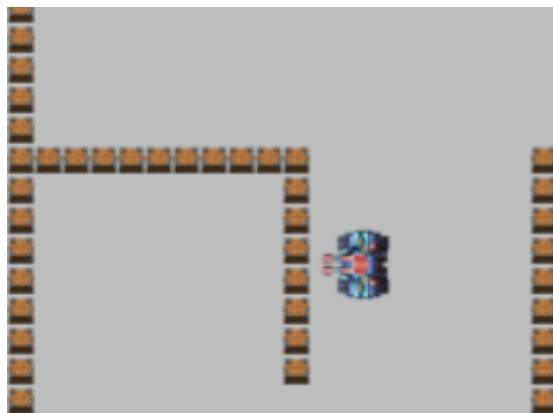
3.4.5.b Cơ chế điều khiển tank của người chơi

```
def movement(player):  
  
    key = pygame.key.get_pressed()  
    if key[player.keys[0]]:  
        player.rect.left -= player.speed  
        player.direction = 'left'  
        player.image = player.pics[3]  
  
    elif key[player.keys[1]]:  
        player.rect.left += player.speed  
        player.direction = 'right'  
        player.image = player.pics[2]  
  
    elif key[player.keys[2]]:  
        player.rect.top -= player.speed  
        player.direction = 'up'  
        player.image = player.pics[0]  
  
    elif key[player.keys[3]]:  
        player.rect.top += player.speed  
        player.direction = 'down'  
        player.image = player.pics[1]  
  
    if len(pygame.sprite.spritecollide(player, players, False)) > 1 or  
    pygame.sprite.spritecollide(player, walls, False):  
  
        if player.direction == 'left':  
            player.rect.left += player.speed  
        if player.direction == 'right':  
            player.rect.left -= player.speed  
        if player.direction == 'up':  
            player.rect.top += player.speed  
        if player.direction == 'down':  
            player.rect.top -= player.speed
```

- Hàm **movement** nhận đối tượng **player** làm đối số và xử lý các sự kiện đầu vào từ bàn phím để di chuyển **player**.
- Hàm sử dụng biến **key** để lấy trạng thái hiện tại của các phím được bấm.
- Mỗi phím kiểm soát một hướng di chuyển cụ thể của **player** (sang trái, sang phải, lên trên, xuống dưới) và cập nhật hình ảnh **player** tương ứng với hướng đó.
- Hàm cũng xử lý va chạm để ngăn **player** không di chuyển xuyên qua các đối tượng khác như người chơi hoặc tường:
 - + Sử dụng **pygame.sprite.spritecollide** để kiểm tra va chạm giữa **player** và nhóm **players** hoặc **walls**.

+ Nếu phát hiện va chạm `len()` của danh sách va chạm lớn hơn 1 hoặc `spritecollide` trả về True, player sẽ phải di chuyển theo hướng ngược lại để tránh tình trạng mắc kẹt.

+ Đối với mỗi hướng di chuyển, nếu phát hiện va chạm, player sẽ được đẩy ngược lại một khoảng bằng `player.speed` để không đè lên player khác hoặc các bức tường trong trò chơi.



3.4.6 Thiết lập chỉ số tank

```
def player_classes(option,player):  
  
    if option == 0:  
  
        player.cooldown = 30 #Giảm cooldown để bắn nhanh hơn  
        player.health = 4  
        player.speed = 5  
        player.pics = ver1  
        player.image = ver1[0]  
  
    elif option == 1:  
        player.cooldown = 45  
        player.health = 6  
        player.speed = 3  
        player.pics = ver2  
        player.image = ver2[0]
```

Hàm `player_classes` cho phép trò chơi phân biệt và điều chỉnh hành vi cụ thể của mỗi loại xe tăng, giúp tăng tính chiến thuật và đa dạng trong gameplay, cấu hình các thuộc tính cho đối tượng `player.sprite` dựa trên lựa chọn của người chơi về loại xe tăng (`player1` hoặc `player2`).

Nhận vào hai tham số là `option` và `player`:

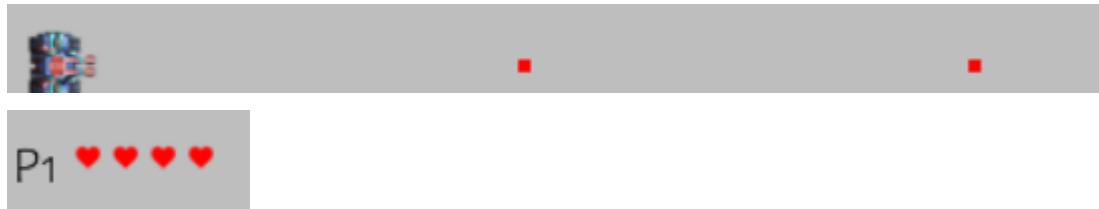
- `option` sẽ quy định loại xe tăng nào đang được cài đặt cho người chơi. - `player` là đối tượng người chơi cần cài đặt.

Căn cứ vào giá trị của `option`:

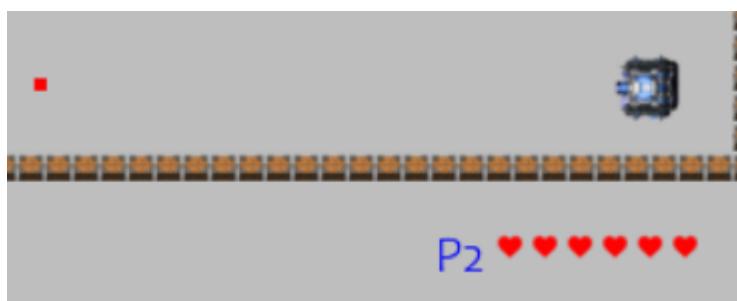
- Nếu `option` là 0, hàm cấu hình player để có các thuộc tính của xe tăng loại 1, nghĩa là có



tốc độ di chuyển và bắn nhanh hơn nhưng có ít máu hơn. Các thuộc tính bao gồm **cooldown**, **health**, **speed**, **pics** và **image** được thiết lập theo giá trị được định sẵn cho loại xe tăng này.



- Nếu **option** là 1, hàm cấu hình **player** để có các thuộc tính của xe tăng loại 2, nghĩa là di chuyển và bắn chậm hơn nhưng có nhiều máu hơn. Các thuộc tính tương tự cũng được thiết lập nhưng với giá trị phù hợp cho loại xe tăng này.



Player.pics được thiết lập để trả tới danh sách các hình ảnh tương ứng với hướng di chuyển của xe tăng theo lựa chọn **ver1** hoặc **ver2**.

Player.image được thiết lập để là hình ảnh xuất hiện đầu tiên của xe tăng, lấy từ hình ảnh đầu tiên ở biến **player.pics**.

3.4.7 Thiết lập Map chiến đấu

```
def readMap():
    Map = open(random.choice(maps), 'r')

    x = 0
    y = 0

    for l in Map:
        builtup = l.split(',')
        builtup[-1] = builtup[-1].strip('\n')
        for D in builtup:
            if D == '.':
                tile = pygame.sprite.Sprite()
                tile.image = tiles
                tile.rect = pygame.Rect(x, y, 12, 12)
                walls.add(tile)
                y += 12
            else:
                y += 12
        x += 11
    y = 0

    Map.close()
```



`readMap` được tạo ra để đọc dữ liệu từ bản đồ của trò chơi và thiết lập màn hình chơi dựa theo nó, sử dụng `sprites` để tạo ra các đối tượng "bức tường", và đặt chúng vào vị trí tương ứng trên màn hình chơi của trò chơi `Pygame`:

Hàm mở một tệp bản đồ được chọn ngẫu nhiên từ danh sách các bản đồ có sẵn trong biến `maps`.

Lệnh **open** mở tệp, và **random.choice** chọn một tệp một cách ngẫu nhiên từ danh sách đó. Hai biến **x** và **y** được khởi tạo để theo dõi vị trí hiện tại trên bản đồ, nơi các đối tượng của trò chơi sẽ được đặt.

Sau đó duyệt qua từng dòng của tệp bản đồ (**for l in Map:**):

- Mỗi dòng được chia (**split**) bởi dấu phẩy để tạo thành một danh sách các ký tự, mỗi ký tự đại diện cho một phần của bản đồ.

- Hàm loại bỏ ký tự xuống dòng n ở cuối mỗi dòng (**builtup[-1] = builtup[-1].strip('n')**). Tiếp theo, hàm duyệt qua từng ký tự trong dòng (**for D in builtup:**)

- Nếu ký tự là dấu chấm (.), nó tương ứng với một "bức tường" trong trò chơi:

- Một đối tượng sprite mới được tạo ra (**tile = pygame.sprite.Sprite()**).

- Ảnh đại diện cho bức tường, biến **tiles**, được thiết lập làm hình ảnh của sprite này (**tile.image = tiles**).

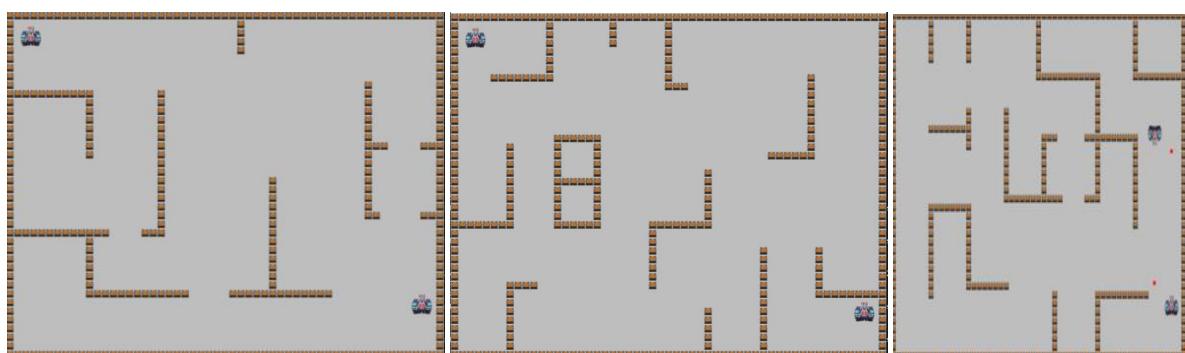
- Đối tượng **sprite** mới đó được gán một hình chữ nhật (**Rect**) với vị trí (**x, y**) và kích thước (**12, 12**).

- Đối tượng **sprite** được thêm vào nhóm **walls**, đã được định nghĩa trước đây như một nhóm của **spritesPygame**, để dễ dàng quản lý các bức tường trong trò chơi.

- **y** được tăng thêm 12 để di chuyển đến vị trí tiếp theo trên cùng một dòng, nơi một **sprite** khác được đặt.

- Nếu ký tự không phải là dấu chấm, **y** vẫn được tăng lên và không có **sprite** nào được tạo ra.

Khi tất cả các ký tự trong dòng đã được xử lý, **x** tăng thêm 11 để bắt đầu dòng tiếp theo, và **y** được đặt lại về 0 để bắt đầu từ đầu dòng mới. Cuối cùng, khi tất cả dữ liệu bản đồ đã được đọc và xử lý, tệp bản đồ (**Map**) được đóng lại.



3.4.8 Thiết lập và xử lí trận đấu

```
def battleScreen(thisStage):
    global run, end
    pygame.mixer.music.load(musicList[2])
    pygame.mixer.music.set_volume(0.4)
```

```
pygame.mixer.music.play(-1)
```

```
timer = 0
```

```
timer2 = 0
```

Thiết lập và quản lý màn hình chiến đấu trong game, bao gồm việc khởi tạo âm nhạc và điều chỉnh âm lượng, bắt đầu vòng lặp, và khởi tạo bộ đếm thời gian cho các sự kiện trong trò chơi.



3.4.8.a Xử lý sự kiện

```
while thisStage:  
    fps = clock.tick(30)  
    timer += 1  
    timer2 += 1  
    for ev in pygame.event.get():  
        if ev.type == QUIT:  
            pygame.mixer.music.stop() run = False  
            battling = False  
            thisStage = False  
            end = False  
  
            movement(player1)  
            movement(player2)  
            bullet_update()  
            key = pygame.key.get_pressed()
```

Vòng lặp đang được thực hiện để xử lý game khi trạng thái "thisStage" đang hoạt động. Nó điều chỉnh tốc độ khung hình, theo dõi thời gian để sử dụng cho các hành động nhất định trong game và xử lý các sự kiện như thoát game, dừng âm nhạc và cập nhật trạng thái chạy của game. Ngoài ra, vòng lặp cũng quản lý việc di chuyển của các người chơi và điều khiển đạn qua các



hàm **movement()** và **bullet_update()**, cũng như lấy thông tin từ các phím được người dùng nhấn.

3.4.8.b Xử lý bắn đạn và thay đổi sprite

```
if key[player1.keys[4]]:
    if timer >= player1.cooldown:
        shoot.play()
        bullet(player1)
        timer = 0

if key[player2.keys[4]]:
    if timer2 >= player2.cooldown:
        shoot.play()
        bullet(player2)
        timer2 = 0
```

Quản lý việc bắn đạn cho hai người chơi trong trò chơi. Khi người chơi nhấn nút bắn đạn (được chỉ định bởi **player1.keys[4]** và **player2.keys[4]**), và nếu thời gian đếm đã đủ lớn so với khoảng thời gian hồi chiêu (**cooldown**), âm thanh bắn đạn sẽ được phát, một viên đạn mới sẽ được tạo từ vị trí của người chơi đó, và bộ đếm thời gian sau đó sẽ được đặt lại về 0 để bắt đầu đếm cho lần bắn tiếp theo.





3.4.8.c Kiểm tra va chạm và tank nổ khi trúng đạn

for bullets in bulletgroup:

```
    if pygame.sprite.collide_rect(bullets,player1):
        bulletgroup.remove(bullets)
        player1.health -= 1
        explode.play() #phát âm thanh vùn
        screen.blit(explosion, player1.rect)
        pygame.display.flip()
        time.sleep(1)

        farthest_point = max(spawnpoints, key=lambda point:
distance(point, player2.rect.center))
        player1.rect.center = farthest_point

    if pygame.sprite.collide_rect(bullets,player2):
        bulletgroup.remove(bullets)
        player2.health -= 1
        explode.play()
        screen.blit(explosion, player2.rect)
        pygame.display.flip()
        time.sleep(1)

        farthest_point = max(spawnpoints, key=lambda point:
distance(point, player1.rect.center))
        player2.rect.center = farthest_point

    if pygame.sprite.spritecollide(bullets,walls,False):
        explode.play()
        bulletgroup.remove(bullets)
        if len(pygame.sprite.spritecollide(bullets,bulletgroup,
False))>1:
            pygame.sprite.spritecollide(bullets,bulletgroup, True)
```

Xử lý va chạm của đạn với các nhân vật trong trò chơi và các đối tượng khác nhau trên màn hình. Khi một viên đạn từ **bulletgroup** va chạm vào người chơi **player1** hoặc **player2**, viên đạn đó sẽ bị loại bỏ khỏi nhóm, người chơi sẽ mất một lượng máu, âm thanh nổ sẽ được phát và một hình ảnh vụ nổ sẽ xuất hiện tại vị trí của người chơi. Người chơi sau đó sẽ được chuyển đến điểm **spawn** xa nhất so với vị trí hiện tại của người chơi kia. Nếu viên đạn va chạm vào tường, nó cũng sẽ bị loại bỏ và phát ra âm thanh nổ.



Nếu có nhiều viên đạn va chạm vào nhau (được kiểm tra bởi `len(pygame.sprite.spritecollide(bullets,bulletgroup, False))>1`), tất cả những viên đạn va chạm sẽ được loại bỏ.

3.4.8.d Cập nhật và hiển thị

```
screen.blit(background, (0,0))
walls.draw(screen)
players.draw(screen)
bulletgroup.draw(screen)
drawPlayerHealth(player1)
drawPlayerHealth(player2)
pygame.display.flip()
pygame.display.update()
```

Đảm bảo rằng người chơi thấy các thay đổi về hình ảnh và trạng thái trong game một cách liền mạch, cập nhật màn hình game và trạng thái mới nhất. Đầu tiên, nền game được xuất lại bằng hàm `blit()`. Sau đó, các đối tượng như tường (`walls`), người chơi (`players`) và nhóm đạn (`bulletgroup`) được xuất lên màn hình. Hai hàm `drawPlayerHealth` dùng để hiển thị thông tin `health` của hai người chơi.

Sau khi tất cả các đối tượng đã được xuất, `pygame.display.flip()` được gọi để cập nhật nội dung toàn bộ cửa sổ hiển thị và `pygame.display.update()` cập nhật một phần của màn hình, nếu cần thiết, để phản ánh các thay đổi. Thông thường, chỉ cần gọi `flip()` hoặc `update()`, nhưng làm cả hai cũng không ảnh hưởng đến hiệu suất nếu `update()` không có tham số.

3.4.8.e Kiểm tra điều kiện kết thúc trận đấu

```
if player1.health == 0 or player2.health == 0:
    pygame.mixer.music.stop()
    battling = False
    thisStage = False
    endScreen(end)
```

Quản lý việc kết thúc trận đấu khi một người chơi đã mất hết `health` và quá trình đơn dẹp liên quan sau trận đấu, xử lý tình huống nếu một trong hai người chơi hết máu (`health` bằng 0). Khi một người chơi hết máu:

Âm nhạc trong trò chơi sẽ dừng lại sử dụng `pygame.mixer.music.stop()`. Biến `battling` sẽ



được đặt thành **False**, đã được sử dụng để duy trì vòng lặp trận đấu chính của trò chơi. Tương tự, biến **thisStage** cũng được đặt thành **False**, để chỉ trạng thái của cấp độ hoặc màn chơi hiện tại. Cuối cùng, hàm **endScreen(end)** được gọi với thông số **end**, sẽ hiển thị màn hình kết thúc của trò chơi hoặc tiến tới một phần khác của trò chơi.

3.4.8.f Thiết lập vị trí máu của player

```
def drawPlayerHealth(player):
    healthimage =
        pygame.image.load('Project_Pygame_Nhom18/BangBang/Tank2P/picture/blood.png')
)

p1healthpos = [60, 500]
p1title = my_font4.render('P1', True, (0,0,0))
p2healthpos = [500,500]
p2title = my_font4.render('P2', True, (0,0,255))

screen.blit(p1title, (35, 500))
screen.blit(p2title, (475, 500))

for c in range(player.health):
    if player == player1:
        screen.blit(healthimage,p1healthpos)
        p1healthpos[0] += 15
    else:
        screen.blit(healthimage, p2healthpos)
        p2healthpos[0] += 14
```



Giúp người chơi theo dõi được lượng máu hiện có trong quá trình chơi, tăng tính tương tác và hồi hộp cho trò chơi. Hàm **drawPlayerHealth()** được định nghĩa để xuất thanh máu cho người chơi dựa trên **health** hiện tại của họ trong game:

- Đầu tiên, hàm tải hình ảnh biểu diễn máu hay **health** từ đường dẫn được cung cấp `pygame.image.load('Project_Pygame_Nhom18/BangBang/Tank2P/picture/blood.png')`.
- Sau đó, hàm đặt vị trí ban đầu cho thanh **health** của người chơi 1 (**p1healthpos**) và người chơi 2 (**p2healthpos**), cùng với việc render tiêu đề 'P1' và 'P2' sử dụng `my_font4.render`, cho biết rõ đây là thanh **health** của người chơi nào.
- `screen.blit(p1title, (35, 500))` và `screen.blit(p2title, (475, 500))` đưa tiêu đề đã render ra màn hình tại các vị trí cụ thể.
- Cuối cùng, vòng lặp for chạy qua từng đơn vị **health** còn lại của người chơi và xuất hình ảnh **health** trên màn hình tại vị trí tương ứng. Mỗi lần một hình ảnh được xuất, vị trí tiếp



theo được tăng lên một khoảng nhất định theo chiều ngang (**p1healthpos[0]** += 15 hoặc **p2healthpos[0]** += 14) để thanh health không bị chồng lên nhau và dễ quan sát.

3.4.9 Thiết lập màn hình cuối, trận chiến kết thúc

3.4.9.a Thiết lập màn hình sau trận đấu

```
def endScreen(thisStage):
    global run, starting, selecting

    pygame.mixer.music.load(musicList[3])
    pygame.mixer.music.set_volume(0.4)
    pygame.mixer.music.play(-1)

    bulletgroup.empty()
    walls.empty()

    scores = open('Project_Pygame_Nhom18/BangBang/Tank2P/BangTinhDiem.txt',
'r')
    for l in scores:
        dataFields = l.split(':')
        dataFields[-1] = dataFields[-1].strip('\n')
        scoredata[dataFields[0]] = int(dataFields[1])
    scores.close()

    winner = max(player1.health,player2.health)
    if winner == player1.health:
        WinnerTitle = my_font2.render('PLAYER 1 WINS', True,
(255,255,0))
        scoredata['Player1'] +=1
    else:
        WinnerTitle = my_font2.render('PLAYER 2 WINS', True,
(255,255,0))
        scoredata['Player2'] +=1

    scores = open('Project_Pygame_Nhom18/BangBang/Tank2P/BangTinhDiem.txt',
'w')
    scores.write('Player1,' + str(scoredata['Player1']) + '\n')
    scores.write('Player2,' + str(scoredata['Player2']) + '\n')
    scores.close()

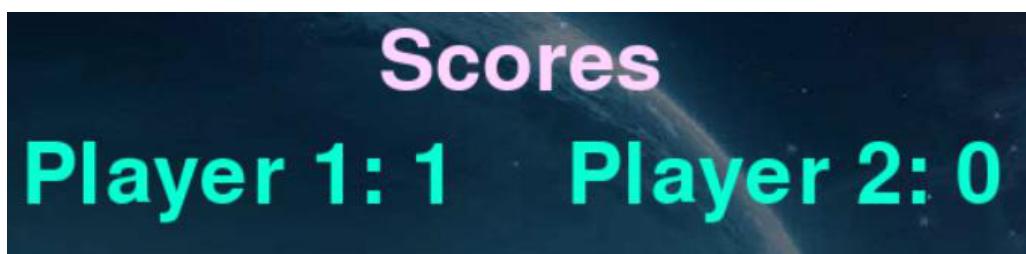
    Title = my_font3.render('END OF THE MATCH', True, (255,0,0))
    scoreTitle = my_font2.render('Scores', True, (255,212,255))
    instruction = my_font4.render('Press R to reset scores!', True,
(255,255,0))
    instruction2 = my_font4.render('Press ENTER to return to game menu.',
True, (255,255,0))
```



Thực hiện hiển thị màn hình kết thúc trận đấu với thông tin về người chiến thắng, điểm số và hướng dẫn cho người chơi. Tải và phát nhạc nền, xóa tất cả các đối tượng đạn và tường trong game. Đọc file điểm số và cập nhật dữ liệu điểm số của mỗi người chơi. Xác định và công bố người chơi chiến thắng dựa trên số lượng máu còn lại, sau đó tăng điểm cho người chơi đó. Xuất điểm mới vào file điểm số. Hiển thị các thông báo cuối trò chơi, bao gồm thông báo kết thúc trận đấu, điểm số, và hướng dẫn cho người chơi biết cách reset điểm số hoặc quay lại menu chính của trò chơi.

Dầu tiên, hàm xuất điểm số của hai người chơi lên màn hình:

- **p1Scores** và **p2Scores** là hai surface chứa text, hiển thị điểm số của người chơi 1 và người chơi 2. Màu sắc của text được đặt là một màu sắc cụ thể (trong trường hợp này là màu cyan với mã màu RGB là (9,255,212)).

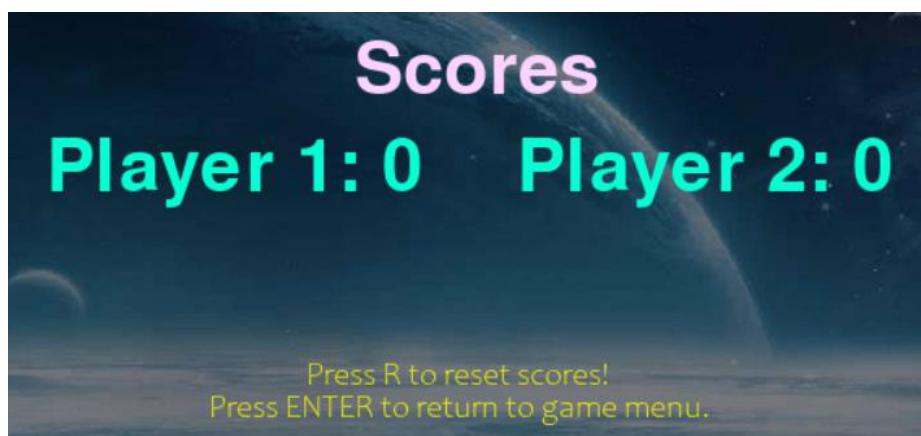


Tiếp theo, hàm xuất các đối tượng đồ họa khác nhau lên màn hình:

- **screen.blit(screenPIC, (0,0))** đặt hình nền cho màn hình.
- **Title, WinnerTitle, scoreTitle, instruction, và instruction2** là các surface chứa các text khác nhau (tiêu đề, người chiến thắng, điểm số, hướng dẫn chơi) được xuất lên màn hình với vị trí tọa độ cụ thể, **pygame.display.flip()** được gọi để cập nhật màn hình hiển thị với các thay đổi đã được xuất lên. Vòng lặp for sau đó kiểm tra các sự kiện nhập từ người chơi:
 - Nếu sự kiện **QUIT** được phát hiện (ví dụ thông qua việc nhấn vào nút 'X' để đóng cửa sổ), hàm sẽ dừng nhạc (**pygame.mixer.music.stop()**) và cập nhật các biến run, end, và thisStage để hủy bỏ vòng lặp và thoát khỏi hàm.
 - Nếu sự kiện **KEYDOWN** được phát hiện (người chơi nhấn phím):
 - Khi phím **Enter (K_RETURN)** được nhấn, đoạn mã cũng dừng phát nhạc nền và cập nhật biến thisStage thành **False**, có thể để thoát ra màn hình trước đó hoặc đóng hàm hiện tại và quay lại menu chính.

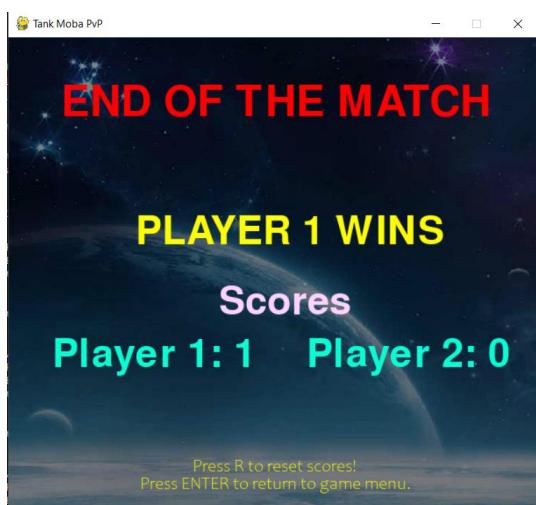


- Khi phím **R** (**K_r**) được nhấn, điểm số cho cả hai người chơi được thiết lập lại thành 0. Điều này cho phép hai người chơi bắt đầu trò chơi mới mà không cần thoát và mở lại ứng dụng.



3.4.9.b Hiển thị kết quả và hướng dẫn

```
while thisStage:  
    p1Scores = my_font2.render('Player 1: ' +str(scoredata['Player1']),  
    True, (9,255,212))  
    p2Scores = my_font2.render('Player 2: ' + str(scoredata['Player2']),  
    True, (9,255,212))  
    #set up tọa độ vị trí căn lề cho các text  
    screen.blit(screenPIC, (0,0))  
    screen.blit(Title, (60,50))  
    screen.blit(WinnerTitle, (145,200))  
    screen.blit(scoreTitle, (240,280))  
    screen.blit(p1Scores, (50,340))  
    screen.blit(p2Scores, (340,340))  
    screen.blit(instruction, (210, 480))  
    screen.blit(instruction2, (150,500))  
    pygame.display.flip()  
  
    for ev in pygame.event.get():  
        if ev.type == QUIT:  
            pygame.mixer.music.stop()  
            run = False  
            end = False  
            thisStage = False  
        elif ev.type == KEYDOWN:  
            if ev.key == K_RETURN:  
                pygame.mixer.music.stop()  
                thisStage = False  
            elif ev.key == K_r:  
                scoredata['Player1'] = 0  
                scoredata['Player2'] = 0
```





- Hàm sử dụng một vòng lặp while để liên tục kiểm tra xem màn hình kết thúc trò chơi đang được hiển thị hay không (**thisStage** là True).
- Trong mỗi lần lặp của vòng lặp này, hàm tạo ra các **surface** chứa text cho điểm số của người chơi 1 và người chơi 2 sử dụng font đã chỉ định trước và màu sắc cụ thể.
- Các surface này sau đó được xuất (**blit**) lên màn hình tại vị trí xác định. Cận cảnh của màn hình kết thúc trận đấu, tiêu đề chiến thắng, điểm số, và hướng dẫn cũng được xuất lên màn hình, **pygame.display.flip()** được gọi để cập nhật toàn bộ màn hình hiển thị với những thay đổi vừa được xuất lên.
- Hàm cũng xử lý sự kiện từ người dùng: Nếu phát hiện sự kiện thoát (**QUIT**), hàm sẽ dừng phát nhạc nền, cập nhật trạng thái của các biến điều khiển trò chơi (**run**, **starting**, **selecting**), và thoát khỏi vòng lặp while, dẫn đến kết thúc hàm. Nếu phát hiện sự kiện nhấn phím (**KEYDOWN**), hàm kiểm tra xem phím được nhấn là phím nào:
- Nếu phím Enter được nhấn, hàm sẽ dừng phát nhạc nền và thoát khỏi vòng lặp **while**, có thể là để trở về menu chính của trò chơi & phím R được nhấn, hàm sẽ reset điểm số của người chơi 1 và người chơi 2 về 0.

3.4.9.c Cập nhật điểm số và quay lại menu chính

```
scores = open('ProJect_Pygame_Nhom18/BangBang/Tank2P/BangTinhDiem.txt',  
'w')  
scores.write('Player1: ' + str(scoredata['Player1']) + '\n')  
scores.write('Player2: ' + str(scoredata['Player2']) + '\n') scores.close()  
  
starting = True  
selecting = True
```

Quản lý việc lưu kết quả của game vào một tệp tin cụ thể rồi sau đó cài đặt lại trạng thái của trò chơi. Hành động cuối cùng trước khi trở lại màn hình chính của trò chơi sau khi một trận đấu kết thúc.

Xuất điểm số của hai người chơi vào một tệp tin: **BangTinhDiem.txt**, 'w'. Đoạn này mở (hoặc tạo) tệp tin BangTinhDiem.txt để xuất thông tin, với chế độ 'w' (write) có nghĩa là nó sẽ xuất đè lên nội dung hiện tại nếu tệp tin đó đã tồn tại.

Sau đó, các điểm số được xuất vào tệp tin:

scores.write('Player1: ' + str(scoredata['Player1']) + 'n') xuất điểm số của người chơi 1 vào tệp tin, xuống dòng để phân cách với dòng tiếp theo.

Đóng tệp tin, hoàn thành quá trình xuất và đảm bảo rằng mọi dữ liệu được lưu một cách đúng đắn: **scores.close()**

Cuối cùng, biến **starting** và **selecting** được thiết lập thành True: Việc cài đặt này có thể là để reset trạng thái của trò chơi, cho phép người chơi bắt đầu một ván đấu mới hoặc trở về màn hình chính/menu chính để lựa chọn các tùy chọn khác hoặc bắt đầu chơi lại.



Project_Pygame_Nhom18 > BangBang > Tank2P > BangTinhDiem.txt

```
1 Player1,1
2 Player2,0
3
```

3.4.9.d Khởi tạo các biến và font chữ

```
pygame.init()
pygame.mixer.init(44100, -16, 2, 2048)
clock = pygame.time.Clock(),...
colours = pygame.color.THECOLORS
48
my_font = pygame.font.SysFont('Verdana', 100)
```

Tank PvP

```
my_font2 = pygame.font.SysFont('moolboran', 66)
```

Start Game
Exit Game

PLAYER 1 WINS

```
my_font3 = pygame.font.SysFont('andalus', 72)
```

END OF THE MATCH

Select Tank

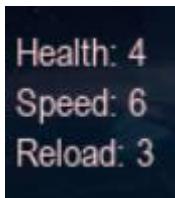
```
my_font4 = pygame.font.SysFont('candara', 20)
```

Press R to reset scores!
Press ENTER to return to game menu.

P2 P1



```
my_font5 = pygame.font.SysFont('arial', 16)
```



Việc khởi tạo các thiết lập ban đầu cho **pygame**, cấu hình những thiết lập cơ bản nhất của trò chơi, từ âm thanh cho đến kiểu chữ, nhằm chuẩn bị sẵn mọi thứ cho việc chạy trò chơi suôn sẻ:

- Khởi động tất cả các module cần thiết để Pygame có thể hoạt động.

- **pygame.mixer.init(44100, -16, 2, 2048)** cấu hình mixer âm thanh để dùng cho việc phát nhạc nền và hiệu ứng âm thanh, với các tham số đặc trưng cho tần số mẫu, độ phân giải âm thanh, và kích thước bộ nhớ đệm.

- Tạo một đối tượng **Clock** được dùng để quản lý thời gian/ tốc độ **refresh** của trò chơi(**pygame.time.Clock()**)

- Khởi code khai báo màu sắc **pygame.color.THECOLORS** cung cấp một bộ sưu tập các màu sắc mà bạn có thể sử dụng trong trò chơi.Các dòng **pygame.font.SysFont(...)** thiết lập nhiều kiểu font khác nhau với kích thước chữ cụ thể, mà sau này có thể dùng để hiển thị văn bản lên màn hình trong trò chơi.

3.4.10 Thêm tiếng nhạc, bắn đạn, nổ

Nhạc nền menu:**musicList = ['Tank2P/music/menuMusic.mp3']**

Âm lượng nhạc nền:

```
pygame.mixer.music.load(musicList[6])
```

```
pygame.mixer.music.set_volume(0.4)
```

Nhạc trong trận đấu: **mbattle.mp3**

Nhạc kết thúc game: **menuMusic.mp3**

Âm thanh vụ nổ: **bum.mp3**

Xe tăng bắn đạn: **fireMusic.mp3**

Nhạc phần chọn tank: **selectSong.mp3**

Để tạo ra một trải nghiệm âm thanh đầy đủ và sống động trong trò chơi hoặc ứng dụng của mình, việc sử dụng các file âm thanh chất lượng cao là vô cùng quan trọng. Bằng cách chuẩn bị trước các file âm thanh phù hợp cho mỗi tình huống như tiếng súng bắn, tiếng bắn đạn, nhạc nền, nhóm tạo ra một môi trường âm thanh đa chiều và phong phú. Khi người chơi tương tác với trò chơi , việc kích hoạt các file âm thanh này vào thời điểm thích hợp sẽ giúp tăng cường trải nghiệm của họ. Ví dụ, khi nhân vật bắn đạn, việc phát ra tiếng súng rất quan trọng để



tạo cảm giác hồi hộp và căng thẳng cho người chơi. Khi đạn trúng người chơi, âm thanh vụ nổ sẽ giúp ta cảm nhận được sức mạnh và hậu quả của hành động đó.

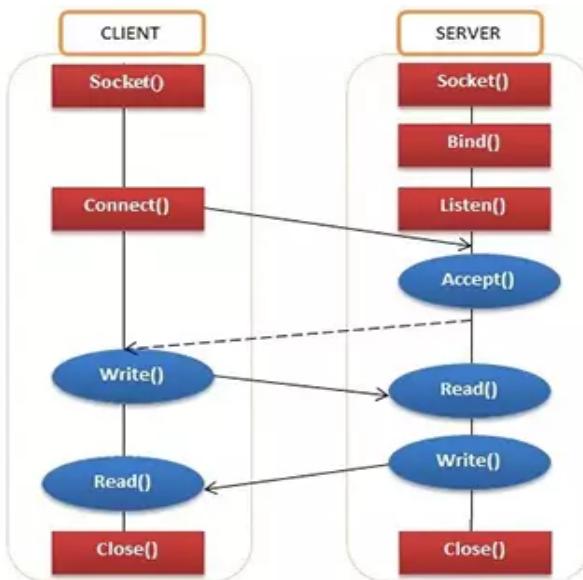
Nhờ việc sử dụng các file âm thanh chuẩn bị từ trước, nhóm tạo ra một không gian âm thanh đồng nhất và chuyên nghiệp cho sản phẩm của mình. Điều này giúp tạo ra sự hòa quyện và tương tác tốt giữa hình ảnh và âm thanh, tạo nên một trải nghiệm trò chơi hoàn chỉnh và cuốn hút người chơi hơn.

4 MODE TANK PVP ONLINE

TANK PVP ONLINE là một trò chơi đối kháng đa người chơi trực tuyến, nơi bạn có thể tham gia vào các trận đấu đầy kịch tính giữa các xe tăng và cạnh tranh với các người chơi khác

Để người chơi có thể kết nối thiết bị với nhau để chơi trực tuyến, game sử dụng Socket trong việc kết nối và giao tiếp, tương tác, chúng ta sẽ dựa vào mô hình client/server:

- Server: ứng dụng có khả năng phục vụ, cung cấp cho client thông tin.
- Client: game gửi yêu cầu đến server.



Mô tả mô hình:

1. Trước tiên chúng ta sẽ tạo ra một máy chủ bằng cách mở một socket - Socket()
2. Sau đó chúng ta sẽ liên kết nó với một host hoặc một máy và một port - Bind()
3. Tiếp theo server sẽ bắt đầu lắng nghe trên port đó - Listen()
4. Yêu cầu kết nối từ client được gửi tới server - Connect()
5. Server sẽ accept yêu cầu từ client và sau đó kết nối được thiết lập - Accept()
6. Bây giờ cả hai đều có thể gửi và nhận tin tại thời điểm đó - Read() / Write()
7. Và cuối cùng khi hoàn thành chúng có thể đóng kết nối - Close()

4.1 Server cho game PVP

Python cung cấp module socket giúp chúng ta dễ dàng thực hiện kết nối client server để giao tiếp với nhau.

Để có thể sử dụng được trước tiên ta phải import module socket vào chương trình

```
import socket
```

Tạo một socket



```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

với

- tham số đầu tiên là Address Family: kiểu thiết lập kết nối. Python hỗ trợ 3 dạng:
 - AF_INET: Ipv4
 - AF_INET6: Ipv6
 - AF_UNIX
- tham số thứ hai là Socket Type: cách thiết lập giao thức
 - SOCK_STREAM: TCP
 - SOCK_DGRAM: UDP

Khai báo port và host:

```
host = "127.0.0.1"
```

```
port = 9999
```

Dăng ký tên cho socket, ràng buộc địa chỉ vào socket:

```
s.bind((host, port))
```

Cho socket đang lắng nghe tối đa 2 kết nối

```
s.listen(2)
```

Hàm khởi tạo server

```
def start_server():
```

```
try:
```

```
    s.bind((host, port))
```

```
    print("PVP tank server started \nBinding to port", port)
```

```
    s.listen(2)
```

```
    accept_players()
```

```
except socket.error as e:
```

```
    print("Server binding error:", e)
```

Xử lý khi client kết nối đến server: Tạo 1 thread để xử lý với kết nối đối với mỗi client

```
def accept_players():
```

```
try:
```

```
    for i in range(2):
```

```
        conn, addr = s.accept()
```

```
        msg = "<< You are player {} >>".format(i+1)
```

```
        conn.send(msg.encode())
```

```
        playerConn.append(conn)
```



```
playerAddr.append(addr)
print("Player {} - [{}:{}].format(i+1, addr[0], str(addr[1])))
threading.Thread(target=handle_client, args=(conn,)).start()
s.close()

except socket.error as e:
    print("Player connection error", e)
except KeyboardInterrupt:
    print("\nKeyboard Interrupt")
    exit()
except Exception as e:
    print("Error occurred:", e)

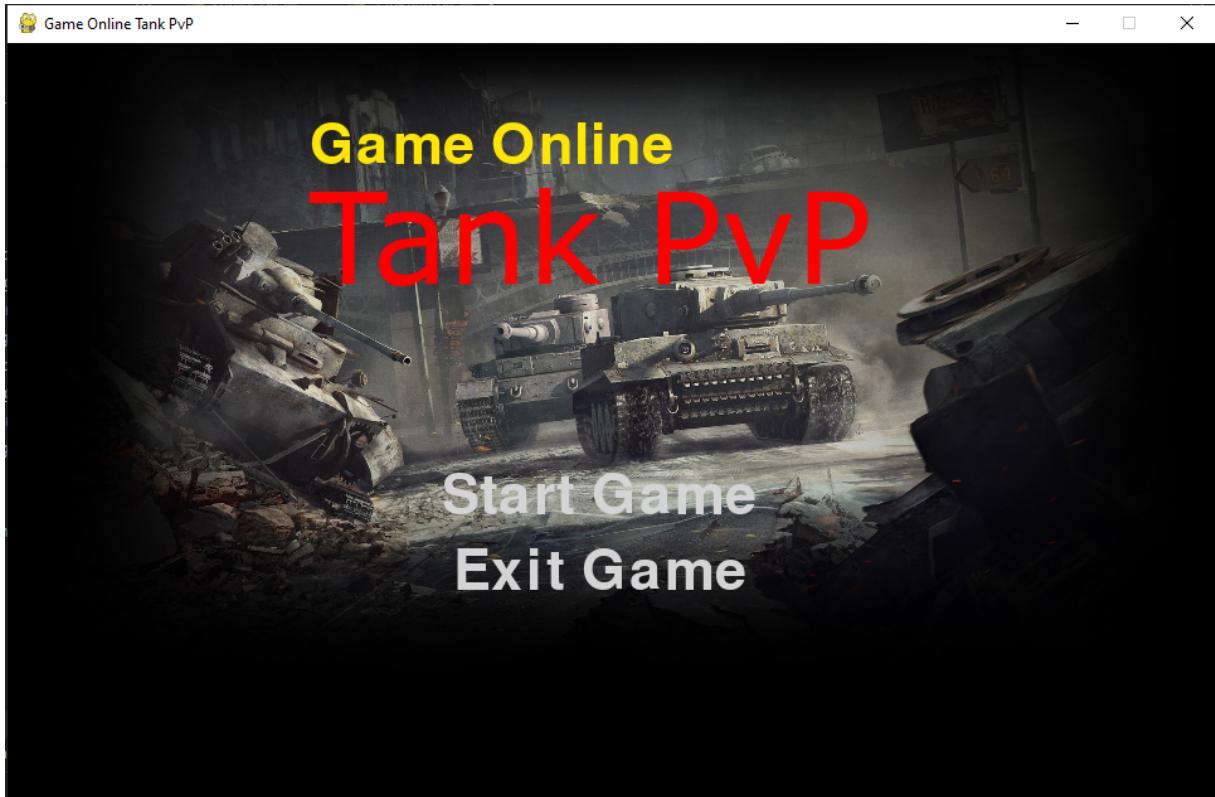
Hàm xử lý dữ liệu được gửi tới từ client và xử lý dữ liệu để gửi đến client khác

def handle_client(conn):
    while True:
        try:
            data = conn.recv(1024).decode()
            print("Received from client:", data)
            # Xử lý dữ liệu từ client tại đây
            if data == "":
                break
            other_conn = [c for c in playerConn if c != conn]
            if other_conn:
                other_conn[0].sendall(data.encode())

        except socket.error as e:
            print("Error receiving data from client:", e)
            break
```

4.2 Game client

Phía client game online sẽ có giao diện tương tự với phiên bản offline, sẽ khác nhau ở phần xử lý sự kiện:



Xử lý khi người dùng chọn start game ở màn hình bắt đầu game: Thực hiện gửi yêu cầu kết nối đến server và nhận dữ liệu trả về, nếu dữ liệu trả về thành công sẽ gán id cho client dựa theo dữ liệu trả về, sau đó tạo thread để gửi và nhận dữ liệu liên tục từ server

try:

```
s.connect((host, port))
print("Connected to :", host, ":", port)
recvData = s.recv(2048 * 10)
print("Receive from server: ", recvData.decode())
bottomMsg = recvData.decode()
if "1" in bottomMsg:
    currentPlayer = 1
    print( "You are player 1")
    Title2 = my_font_title.render('You are player 1', True, (255,255,0))
    screen.blit(Title2, (170, 120))
else:
    currentPlayer = 2
```

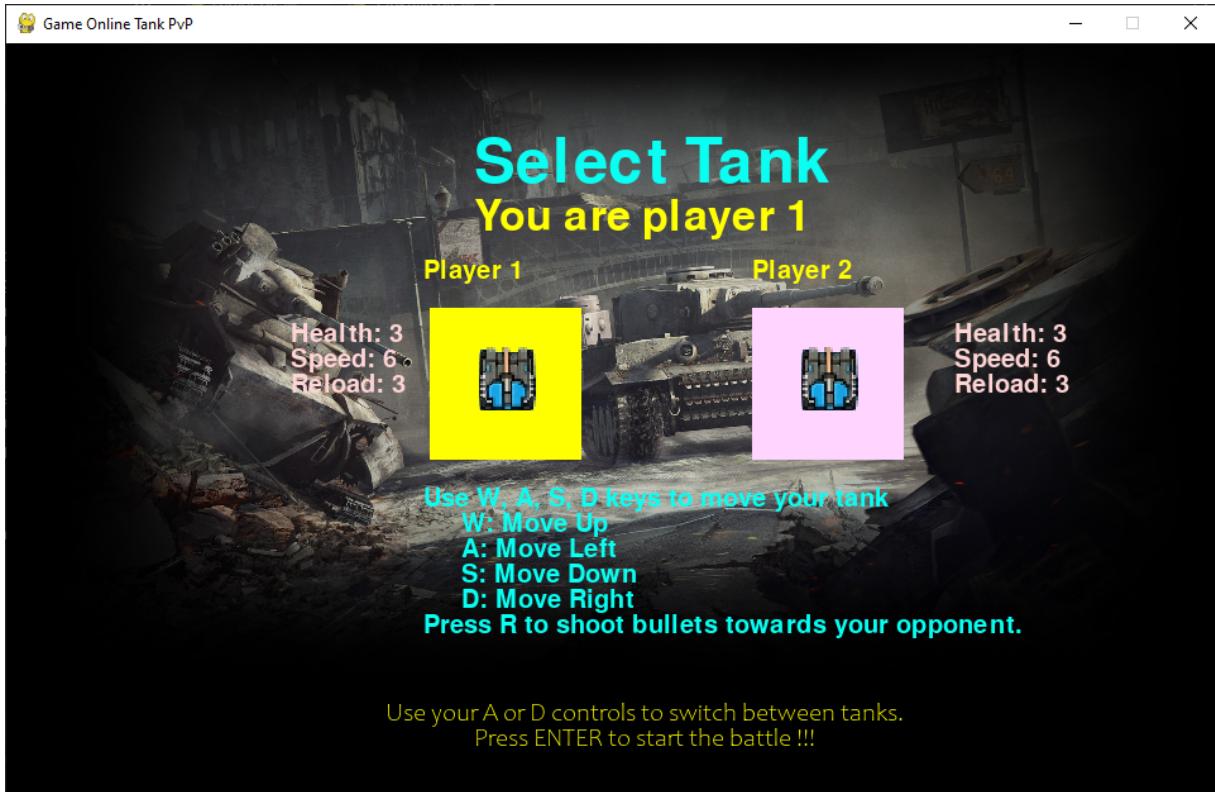


```
print( "You are player 2")
Title2 = my_font_title.render('You are player 2', True, (255,255,0))
screen.blit(Title2, (170, 120))

except socket.error as e:
    Title2 = my_font_title.render('server error', True, (255,255,0))
    screen.blit(Title2, (170, 120))

create_thread(receive_msg)

def receive_msg():
    global msg
    while True:
        try:
            recvData = s.recv(2048 * 10)
            msg = recvData.decode()
            print("Receive form "+str(currentPlayer)+":", msg)
        except socket.error as e:
            print("Socket connection error:", e)
            break
```



Xử lý khi chọn xe tăng khi nhấn nút A hoặc D sẽ gửi dữ liệu tới server để xử lý và gửi đến client để cập nhật

if currentPlayer == 1:

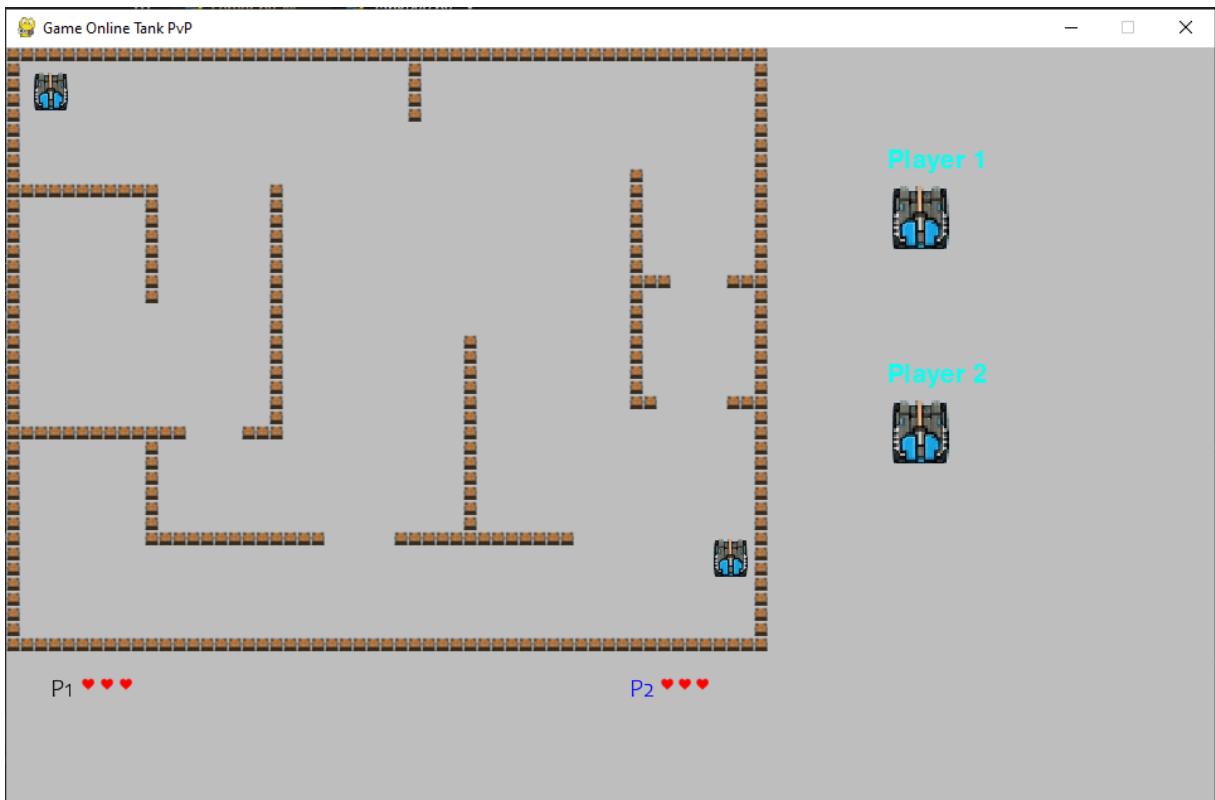
```
    if ev.key == K_a:  
        data = "A"+str(currentPlayer)  
        print("A")  
    if Opt1 == 0:  
        Opt1 =1  
    else:  
        Opt1 -= 1  
    s.sendall(data.encode('utf-8'))  
elif ev.key == K_d:  
    print("D")  
  
    data = "D"+str(currentPlayer)  
    if Opt1 == 1:
```



```
Opt1 = 0
else:
    Opt1 += 1
    s.sendall(data.encode('utf-8'))
else :
    if ev.key == K_a:
        print("A")
        data = "A"+str(currentPlayer)

    if Opt2 == 0:
        Opt2 =1
    else:
        Opt2 -= 1
        s.sendall(data.encode('utf-8'))
    elif ev.key == K_d:
        print("D")
        data = "D"+str(currentPlayer)
    if Opt2 == 1:
        Opt2 = 0
    else:
        Opt2 += 1
        s.sendall(data.encode('utf-8'))
```

Sau khi nhấn enter bắt đầu game, xử lý sự kiện khi nhấn phím di chuyển và phím bắn



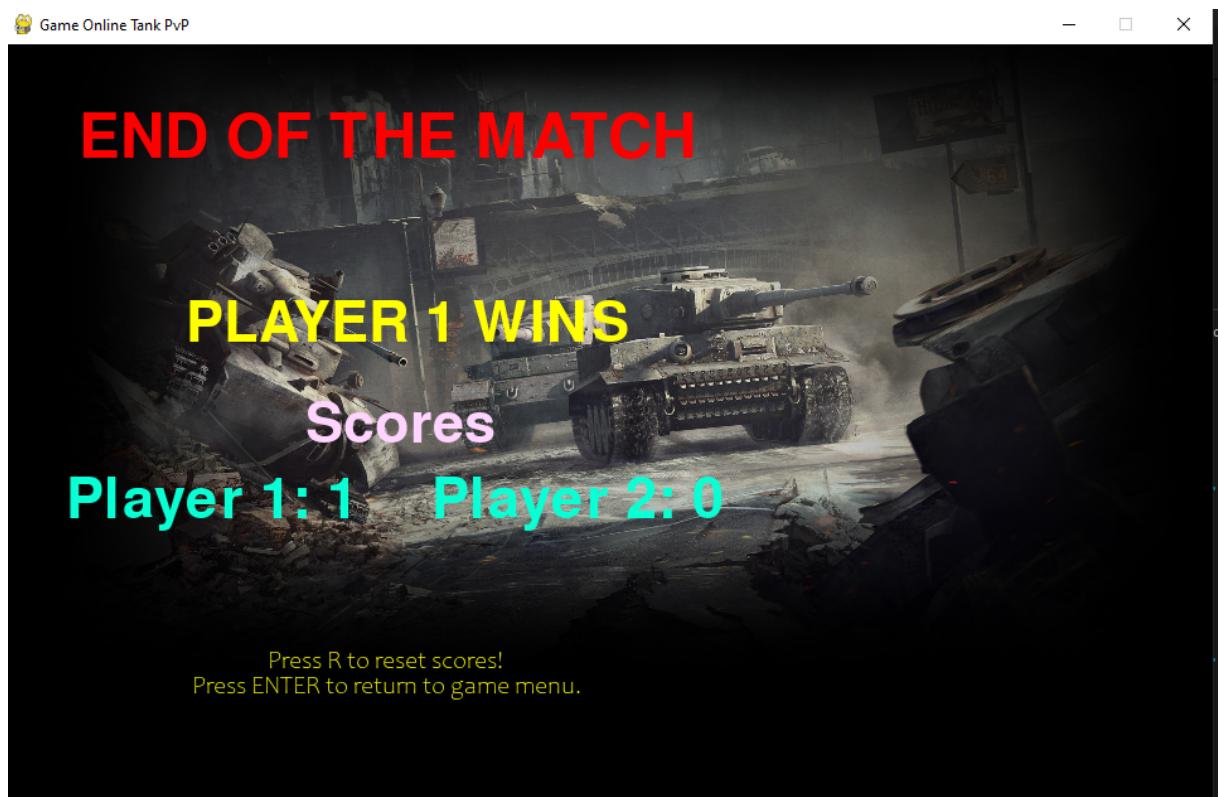
```
if key[player.keys[0]]:  
    player.rect.left -= player.speed # Di chuyển sprite sang trái theo tốc độ  
đã định  
    player.direction = 'left' # Cập nhật hướng di chuyển của sprite  
    player.image = player.pics[3] # Cập nhật hình ảnh của sprite khi người  
chơi hướng di chuyển sang trái  
    data = "LEFT" + str(player.rect.left)  
    s.sendall(data.encode('utf-8'))  
  
elif key[player.keys[1]]:  
    player.rect.left += player.speed  
    player.direction = 'right'  
    player.image = player.pics[2]  
    data = "RIGHT" + str(player.rect.left)  
    s.sendall(data.encode('utf-8'))  
elif key[player.keys[2]]:
```



```
player.rect.top -= player.speed
player.direction = 'up'
player.image = player.pics[0]
data = "UP" + str(player.rect.top)
s.sendall(data.encode('utf-8'))

elif key[player.keys[3]]:
    player.rect.top += player.speed
    player.direction = 'down'
    player.image = player.pics[1]
    data = "DOWN" + str(player.rect.top)
    s.sendall(data.encode('utf-8'))
```

Màn hình kết thúc game:





5 Web demo và tải game

Link : https://thanhvu2702.github.io/web_intro_ProJ_Pygame