GIT : https://github.com/alexxandre80/Projet_Python

Boucle du jeu lors de l'écran principale et du choix des noms des joueurs :

Lors du clique sur le bouton « Jouer » cela nous permet de changer d'écran pour aller sur l'écran « Player 1 Name » qui est l'écran pour faire le choix du nom du joueur 1

Lors du clique sur le bouton « Quitter » on quitte le jeu

```
# Ecran du menu:
if game_screen == "Main Screen":
    # Boucle des événements du jeu:
    for event in pygame.event.get():
        # Si l'événement est QUIT fermer la fenêtre
        if event.type == QUIT:
            # Définit l'état de lecture sur false,
            # quittant ainsi la boucle principale:
            running = False

        if event.type == pygame.MOUSEBUTTONDOWN:
            if start_button.buttonClick():
                game_screen = "Player 1 Name"

            if exit_button.buttonClick():
                running = False

    # Dessine le background:
    screen.fill(GRAY)
    button_group.draw(screen)

    text.displayTextMainMenu("Worms Python", WHITE, screen, screen_size, "top_center2")

    # Mises à jour stuff:
    pygame.display.update()

# Écran du nom du joueur 1:
elif game_screen == "Player 1 Name":
```

Sur l'écran du choix du nom du joueur 1 on peut quitter le jeu en cliquant sur la croix de la fenêtre,

Avec l'événement pygame.KEYDOWN on récupère les touches qui on était presser par l'utilisateur, si le joueur appuie sur la touche « ENTER » cela valide son nom et le garde en mémoire et on passe à l'écran du joueur 2

```
# Écran du nom du joueur 1:
elif game_screen == "Player 1 Name":

    # Obtenir des événements du jeux:
    events = pygame.event.get()
    # Boucle à travers les événements du jeux:
    for event in events:
        # Si l'événement est QUIT
        if event.type == QUIT:
            # Définit l'état de lecture sur false, quittant ainsi la boucle principale:
            running = False

        # Vérifie s'il y a eu une pression sur une touche:
        if event.type == pygame.KEYDOWN:
            # Vérifie si la touche appuyée était la touche ENTER:
            if event.key == pygame.K_RETURN:
                # Ne se lance que si le joueur a appuyer sur la touche:
                if len(player_1.name) != 0:
                    # Changment d'écran:
                    game_screen = "Player 2 Name"

    # Dessine le background:
    screen.fill(GRAY)

    # Affichage du texte:
    text.displayTextNameScreen("Veuillez saisir le nom du joueur 1 (appuyez sur ENTER lorsque vous êtes prêt):",
                               WHITE, screen, screen_size, "center_top3")

    # Afficher et obtenir le nom du joueur:
    player_1.name = ti.textInputBox(player_1.name, WHITE,
                                    screen, screen_size, events, text.font_36)

    pygame.display.update()
```

L'écran de sélection du nom du joueur 2 est sensiblement le même que le joueur 1 mis à part qu'une fois que le joueur 2 valide son nom on passe à l'écran « Playing » qui est l'écran de jeu

```
# Écran du nom du joueur 2:
elif game_screen == "Player 2 Name":

    # Obtenir des événements du jeux:
    events = pygame.event.get()
    # Boucle à travers les événements du jeux:
    for event in events:
        # Si l'événement est QUIT
        if event.type == QUIT:
            # Définit l'état de lecture sur false, quittant ainsi la boucle principale:
            running = False

        # Vérifie s'il y a eu une pression sur une touche:
        if event.type == pygame.KEYDOWN:
            # Vérifie si la touche appuyée était la touche ENTER:
            if event.key == pygame.K_RETURN:
                # Ne se lance que si le joueur a appuyer sur la touche:
                if len(player_2.name) != 0:
                    # Changment d'écran:
                    game_screen = "Playing"

    # Dessine le background:
    screen.fill(GRAY)

    # Afficher et obtenir le nom du joueur:
    text.displayTextNameScreen("Veuillez saisir le nom du joueur 2 (appuyez sur ENTER lorsque vous êtes prêt):",
                               WHITE, screen, screen_size, "center_top3")

    # Displaying and getting player's name:
    player_2.name = ti.textInputBox(player_2.name, WHITE,
                                    screen, screen_size, events, text.font_36)

    pygame.display.update()
```

Lors de l'écran de jeu on regarde si le joueur 1 est en collision avec le sol idem pour le joueur 2. Lors d'un tir de projectile on regarde s'il est en collision avec le terrain si il rencontre un obstacle alors on passe « bombhit » à True afin de dire que l'on touche quelque chose et on arrête le mouvement du projectile

```
# Ecran de jeu:
elif game_screen == "Playing":

    # Vérification des collisions avec le terrain:
    if terrainCollision == True:
        if pygame.sprite.spritecollide(player_1, terrain.terrain_group, False):
            player_1.rect.y -= 4
            player_1.standing = True

        else:
            player_1.standing = False

        if pygame.sprite.spritecollide(player_2, terrain.terrain_group, False):
            player_2.rect.y -= 4
            player_2.standing = True

        else:
            player_2.standing = False

        if pygame.sprite.spritecollide(bomb, terrain.terrain_group, True):
            bomb.rect.y += 10
            if pygame.sprite.spritecollide(bomb, terrain.terrain_group, True):
                lastBombPosition = bomb.rect.center
                bombHit = True
                bomb.stop_movement()

    # Obtient la position de la souris:
    mousePosition = pygame.mouse.get_pos()

    # Vérification des touches enfoncées:
    pressed_keys = pygame.key.get_pressed()
```

On réalise une boucle de jeu pour voir si un joueur clique sur la croix pour fermer le jeu. On vérifie aussi si le par rapport au tour du joueur s'il a cliqué quelque part dans la fenêtre avec la souris alors on récupère les statistiques du projectile qui sera lancé. Si le joueur appuye sur 1 ou 2 il change d'armes.

```
# Boucle à travers les événements de jeu
for event in pygame.event.get():
    # Si l'événement est QUIT
    if event.type == QUIT:
        # Définit l'état de lecture sur false, quittant ainsi la boucle principale:
        running = False

    # Vérifie s'il y a eu un appui sur le bouton de la souris:
    if event.type == pygame.MOUSEBUTTONDOWN:
        if bomb.moving == False:
            if playerTurn == "1":
                # Obtient les statistiques de la bombe et la lance:
                getBombStats(bomb, player_1, mousePosition)
                done = False

            elif playerTurn == "2":
                # Obtient les statistiques de la bombe et la lance:
                getBombStats(bomb, player_2, mousePosition)
                done = False

    # Vérifie s'il y a eu une pression sur une touche:
    if event.type == pygame.KEYDOWN:
        # Vérifie si la touche appuyée était le numéro 1:
        if event.key == pygame.K_1:
            if bomb.moving == False:
                bomb = roquette
                bomb_group.empty()
                bomb_group.add(roquette)

        # Vérifie si la touche appuyée était le numéro 2:
        if event.key == pygame.K_2:
            if bomb.moving == False:
                bomb = bomb_neutron
                bomb_group.empty()
                bomb_group.add(bomb_neutron)
```

Si c'est le tour du joueur 1, s'il appuye sur la touche « D » et si un projectile n'est pas en mouvement il se déplace à droite et pareil pour la touche « A » mais il se déplace à gauche. On trace ensuite la ligne pour avoir une prévisualisation de la trajectoire du projectile. Si un projectile entre en collision avec quelque chose on arrête son mouvement si c'est le joueur 2 on lui inflige des dégâts s'il arrive au 0 point de vie on passe à l'écran de victoire sinon on passe au tour du joueur 2

```
# Tour des joueurs:
if playerTurn == "1":
    # Si la touche appuyée est D:
    if pressed_keys[K_d] and not bomb.moving:
        player_1.move("right")
    # Si la touche appuyée est A:
    if pressed_keys[K_a] and not bomb.moving:
        player_1.move("left")

    # Tracer la ligne entre le joueur et la position de la souris:
    get_distance(player_1.rect.center, mousePosition)
    pygame.draw.line(screen, INFINANCE, player_1.rect.center, mousePosition, 5)

    # Contrôles de collision, s'il y en a,on arrête le mouvement de la bombe et
    # on fait les dégats à l'ennemi:
    if pygame.sprite.collide_rect(bomb, player_2):
        lastBombPosition = bomb.rect.center
        bombHit = True
        bomb.stop_movement()
        player_2.health -= bomb.damage
        # Vérifie s'il y a un gagnant
        if player_2.health <= 0:
            player_2.health = 0
            winner = player_1.name
            loser = player_2.name
            game_screen = "Winner Screen"

    if done == False:
        if bomb.moving == False:
            bomb.reset_stats()
            bomb_group, bomb = reset_bomb(bomb_group, roquette)
            playerTurn = "2"
            done = True
            player_1.movements = 0
```

Si c’est le tour du joueur 2 le code est le meme mise a part qu’il se deplace avec les fleches directionnel du clavier.

```
elif playerTurn == "2":
    # Si la touche enfoncée, est flèche vers la droite:
    if pressed_keys[K_RIGHT] and not bomb.moving:
        player_2.move("right")
    # Si la touche enfoncée, est flèche vers la gauche:
    if pressed_keys[K_LEFT] and not bomb.moving:
        player_2.move("left")

    # Tracer la ligne entre le joueur et la position de la souris:
    get_distance(player_1.rect.center, mousePosition)
    pygame.draw.line(screen, INFINANCE, player_2.rect.center, mousePosition, 5)

    # Contrôles de collision, s'il y en a,on arrête le mouvement de la bombe et
    # on fait les dégats à l'ennemi:
    if pygame.sprite.collide_rect(bomb, player_1):
        lastBombPosition = bomb.rect.center
        bombHit = True
        bomb.stop_movement()
        player_1.health -= bomb.damage
        # Vérifie s'il y a un gagnant
        if player_1.health <= 0:
            player_1.health = 0
            winner = player_2.name
            loser = player_1.name
            game_screen = "Winner Screen"

    if done == False:
        if bomb.moving == False:
            bomb.reset_stats()
            bomb_group, bomb = reset_bomb(bomb_group, roquette)
            playerTurn = "1"
            done = True
            player_2.movements = 0
```

Sur l’écran de Victoire on verifie les evenement si le joueur qui le jeu ou si il appuye sur « ENTER » pour revenir au menu principal alors on remet a 0 les statistique des joueurs et on retourne au menu principal

```
    # Ecran de victoire:
    elif game_screen == "Winner Screen":
        # Boucle à travers les événements de jeu
        for event in pygame.event.get():
                # Si l'événement est QUIT
            if event.type == QUIT:
                    # Définit l'état de lecture sur false, quittant ainsi la boucle principale:
                running = False

            # Vérifie s'il y a eu une pression sur une touche:
            if event.type == pygame.KEYDOWN:
                # Vérifie si la touche appuyée était la touche ENTER:
                if event.key == pygame.K_RETURN:
                    # Réinitialise les points de vie des joueurs:
                    player_1.resetPlayer()
                    player_2.resetPlayer()

                    # Changement d'écran:
                    game_screen = "Main Screen"

        # Dessine le background:
        screen.fill(GRAY)

        # Affichage du texte:
        text.displayTextWinnerScreen("Toutes nos félicitations!",
                                     WHITE, screen, screen_size, "center_top3")
        # Afficher qui a gagné:
        text.displayWhoWon(GREEN, winner, loser, screen, screen_size,
                           "center_top")

        pygame.display.update()


# Quitte le jeu:
pygame.display.quit()
```

Cette classe nous permet de créer une maps avec des blocs ou les joueurs se déplace de taille aléatoire cela nous permet d'avoir une maps aléatoire

```
def generateHeightsRandom(self, minHeight, maxHeight):
    # Prend la position X pour chaque "bloc":
    widths = [width
              for width in range(0, round(self.screen_width/self.original_width))]
    print("Largeurs: ", len(widths))

    # Crée des hauteurs aléatoires pour chaque "bloc":
    heights = [random.randint(minHeight, maxHeight)
               for counter in widths]
    print("Hauteurs: ", len(heights))
    print(heights)

    for i in range(self.smooth_factor):
        for n in range(2, len(heights) - 3):
            heights[n] = sum(heights[n-2: n+3])/5

    for i in range(self.smooth_factor):
        heights[0] = sum(heights[:5])/5
        heights[1] = sum(heights[:5])/5
        heights[len(heights) - 3] = sum(heights[len(heights)-5:])/5
        heights[len(heights) - 2] = sum(heights[len(heights)-5:])/5
        heights[len(heights) - 1] = sum(heights[len(heights)-5:])/5

    heights = [int(i) for i in heights]

    counter = 0
    while counter < len(widths):
        if heights[counter] == 1:
            image = Block(self.sprite, self.scale)
            image.rect.x = counter * self.original_width
            image.rect.y = self.screen_height - self.original_height

            self.terrain_group.add(image)

        else:
            for i in range(1, heights[counter] + 1):
                image = Block(self.sprite, self.scale)
                image.rect.x = counter * self.original_width
                image.rect.y = self.screen_height - self.original_height * i

                self.terrain_group.add(image)

        counter += 1
```

Dans note page « sprites.py » si la génération n'est pas aléatoire ou utilise une génération prédéfinie

```
# Terrain:
terrain = terrain.Terrain("Sprites/DirtBlock_70x70.png", blockScale, screen_size, smooth_factor)
if terrainGenRandom == True:
    terrain.generateHeightsRandom(minTerrainHeight, maxTerrainHeight)
else:
    terrain.generateHeightsPreset(terrainPreset)
```

```python
def generateHeightsPreset(self, preset):
    # Prend la position X pour chaque "bloc":
    widths = [width
              for width in range(0, round(self.screen_width/self.original_width))]
    print("Largeurs: ", len(widths))

    # Crée des hauteurs aléatoires pour chaque "bloc":
    if preset == 1:
        heights = [2, 2, 2, 3, 4, 5, 5, 6, 6, 6, 4, 3, 2, 1, 1, 1, 2, 2, 4, 4, 4, 5, 5, 6, 7, 5, 4, 4, 3, 3, 3, 3, 3, 2, 2, 2]

    print("Hauteurs: ", len(heights))
    print(heights)
    counter = 0
    while counter < len(widths):
        if heights[counter] == 1:
            image = Block(self.sprite, self.scale)
            image.rect.x = counter * self.original_width
            image.rect.y = self.screen_height - self.original_height

            self.terrain_group.add(image)

        else:
            for i in range(1, heights[counter] + 1):
                image = Block(self.sprite, self.scale)
                image.rect.x = counter * self.original_width
                image.rect.y = self.screen_height - self.original_height * i

                self.terrain_group.add(image)

        counter += 1
```

Reset_bomb : Nous permet de définir la bombe qui sera choisi par défaut lors du chargement du jeu ou au moment du changement du joueur

```python
def reset_bomb(group, default_bomb):
    # Définit la "bombe" sur la bombe par défaut:
    bomb = default_bomb
    # Efface le groupe de la bombe:
    group.empty()
    # Ajoute la bombe par défaut au groupe:
    group.add(default_bomb)

    return group, bomb
```