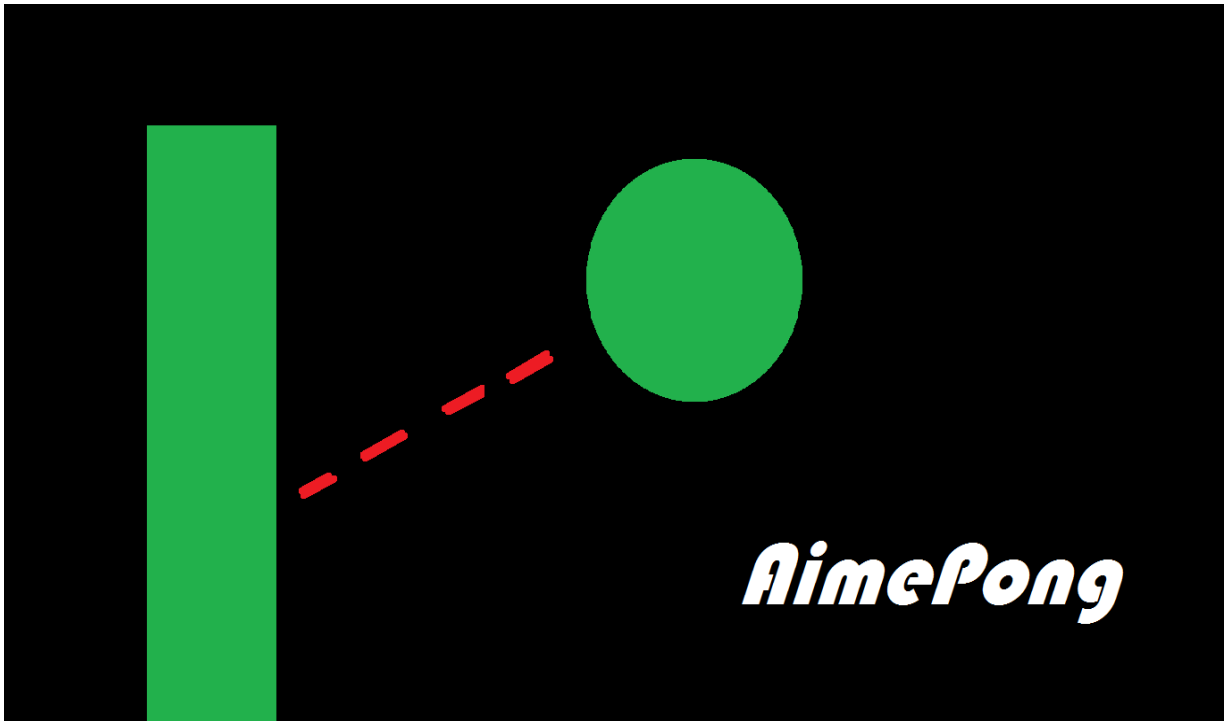
Aim Pong

Project ISN

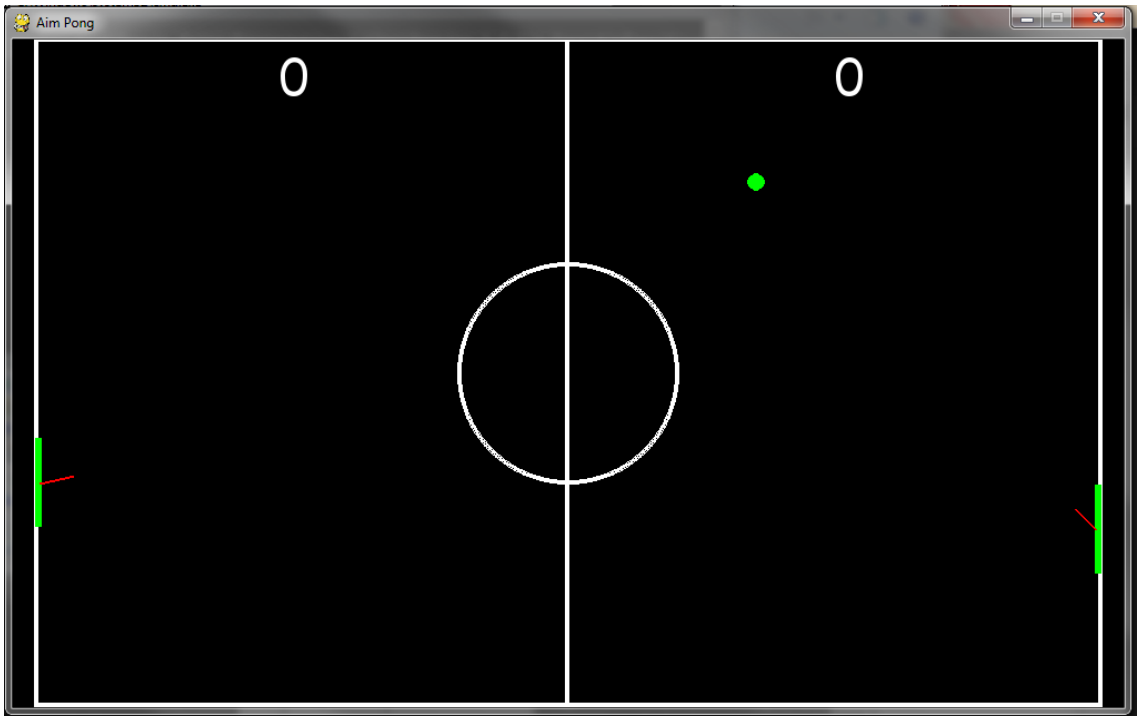


Développé par : João Farinha

I. Règles De jeu :

C’est un jeu de Pong ou le joueur peut choisir l’orientation de la balle grâce à un réticule dans la raquette. Le joueur 1 ou en mode Solo on déplace la raquette grâce aux touches Haut et Bas et on fait ce déplacer le réticule avec les touches Droite et Gauche. Le joueur 2 en mode deux joueurs déplace sa raquette grâce aux touches Z et S afin de bouger la raquette de haut en bas et peut déplacer le réticule grâce aux touches Q et D. Lorsque la balle frappe la raquette elle part aussi tôt dans la direction du réticule au moment de l’impact.

Capture d’écran du menu principal

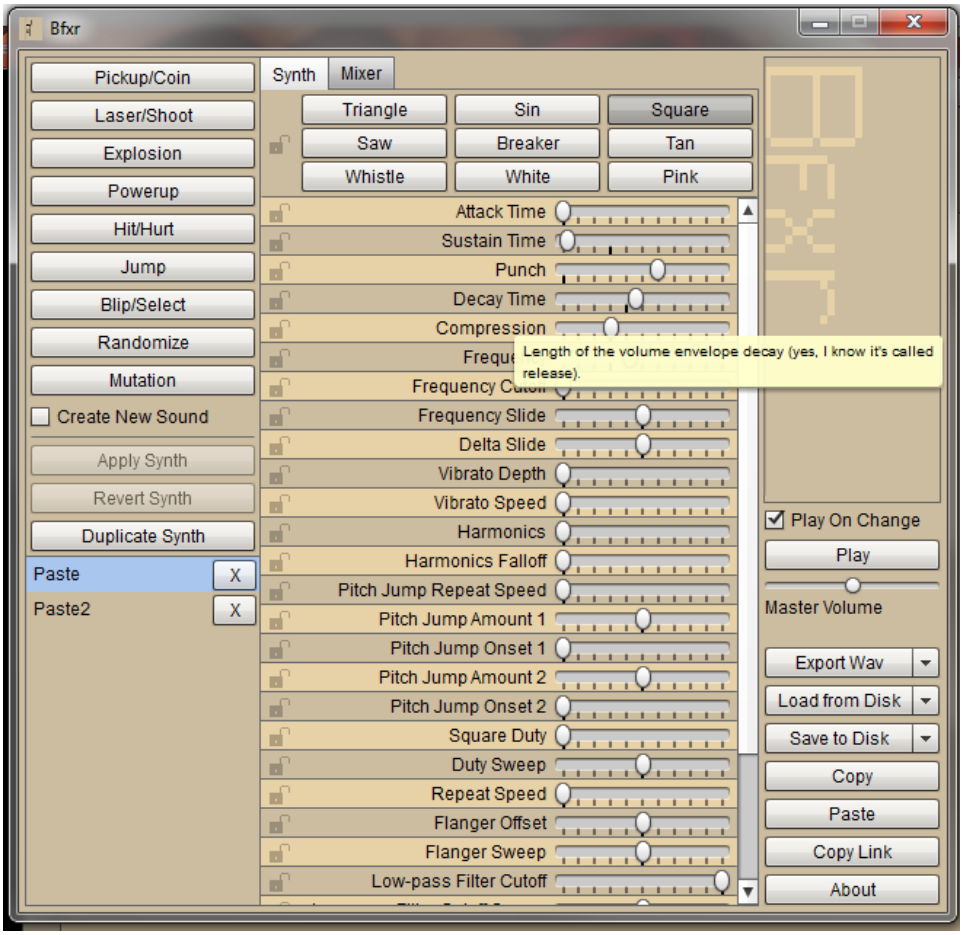


Capture d’écran d’une partie en cours contre l’AI

II. Le Project :

L'objective de ce projet est de créer un Pong un peu original, c'est base du Pong original mais en plus dynamique avec un mode solo intégré et la capacité de choisir la trajectoire de la balle.

Ce jeu a été développé en python 3.2 avec Pygame, on a utilisé un logiciel appelle Bfxr afin de générer les sons style 8Bits du jeu, Paint et la fonction Print Screen afin d'obtenir les images de jeu. On a utilisé la documentation Pygame (http://www.pygame.org/docs/) afin de de connaitre les la bibliothèque Pygame, La documentation python (https://docs.python.org/3.2/) pour les spécificités du langage python, ce site (http://www.letoileauxsecrets.fr/couleurs/couleurs-bleu.html) pour les codes couleur en RGB.



Capture d'écran du logiciel Bfxr

L'intégralité de ce Project a été développée par moi, j'ai commencé par développer le menu principal en suite le mode deux joueurs et j'ai fini par ajouter l'AI qui complète le mode Solo.

III. Listing de fonctions:

NOM	Paramètres	Description
mainLoop	Pas de paramètres	Fonction principal qui prend en charge le menu et le la boucle de jeu.
menu	mouseX : position X de la souris mouseY : position Y de la souris mouseclick : Booléen correspondant à l'action de click gauche de la souris	Dessin les menu à l'écran et gère l'input de l'utilisateur et les entités appelées bouton
button	x : position X de l'entité bouton y : position Y de l'entité bouton buttonOrder : l'ordre dans laquelle ce bouton apparait dans le menu c'est-à-dire premier, second, etc…	Crée des entités appelées boutons afin d'être utilisées dans le menu, renvoie entité button
gameLoop	bot : Booléen qui décide entre le mode Solo et le mode 2 joueurs	Boucle principal du jeu ou on prend les commandes du joueur, met à jour la position des entités, et on dessine les entités à l'écran.
drawnGame	entities : dictionnaire ou sont toutes les entités du jeu score : c'est une tuple ou sont les points du jouer 1 et 2	Fonction qui dessine le terrain, les entités c'est-à-dire les raquettes, et la balle et les points de chaque joueur
eventManager	events : dictionnaire de booléens ou sont enregistrées les events bot : Booléen qui décide entre le mode Solo et le mode 2 joueurs	Fonction qui prend en charge et enregistre l'entrée de commandes du joueur, renvoie events
gameUpdate	entities : dictionnaire ou sont toutes les entités du jeu keyStatus : dictionnaire de booléens ou sont enregistrées les events	Fonction qui met à jour les entités en fonction des commandes entrées par le jouer, et la physique du jeu
gameLogic	entities : dictionnaire ou sont toutes les entités du jeu score : c'est une tuple ou sont les points du jouer 1 et 2	Fonction qui gère les collisions et que met en place les règles du jeu, détecte aussi les points et renvoie le score
gameReset	entities : dictionnaire ou sont toutes les entités du jeu	Fonction qui reset le jeu après chaque point
AI	entities : dictionnaire ou sont toutes les entités du jeu timer : entier qui correspond au ticks de jeu passe lors de la dernière mise à jour	Fonction qui permet de émuler le joueur 2 dans le mode Solo

IV. Bilan:

On peut améliorer ce jeu en utilisant des spirites ajouter une musique de fond et on peut faire pas mal d'améliorations au niveau de l'intelligence artificiel du mode Solo, surtout dans son system de visé. On peut aussi mètre 3 balles dans sur le terrain et lorsque un joueur arrêt une balle il la garde pour la tirer l'lorsque qu'il souhaite, du coup le joueur pour garder les 3 balles du terrain et les tirer les une après les autres dans déférentes directions afin de déstabiliser le joueur en face ou l'AI.

Une des plus grandes difficultés de ce projet a été l'instabilité de python, c'est-à-dire mon inexpérience de programmation en python plus les limites du langage pour créer des applications en temps réel tel des jeux, qui avec la date d'échéance qui s'approchait ma empêche de créer le Aim Pong a 3 balles tel que j'avais planifie au début et m'a obligé à simplifier le jeu tel qu'il est maintenant. J'aurais peut prévenir sa en développant ce jeu en C++ avec SFML (Simple Fast Media Library), car mes connaissances en C++ sont beaucoup plus élevées.

Ce fut une très bonne expérience et sa m'a permis de pas mal élargir mes horizons sur le langage python que je trouve très bon pour créer une démo rapide d'une idée ou alors des petits scripts utiles mais pas pour une application en temps réel tel un jeu.

Si je devais diffuser ce projet sur Internet je le ferais en Licence libre afin d'en faire a tous profiter de ce chef d'œuvre qui a déboute l'ère du jeu vidéo mais que personne ne ce rappel plus qui existe.

V. Annexes :

Listing du code

```
import pygame, sys, math, random

pygame.init()
WINDOWSIZE = WINDOWWIDTH, WINDOWHEIGHT = 1000, 600 #Constants de la fenetre
SCREEN = pygame.display.set_mode(WINDOWSIZE) #Fenetre
pygame.display.set_caption("Aim Pong")
comicSansMs = pygame.font.SysFont("comicsansms", 45) #Font Utilise

#Constantes de colleur
BLACK = (0, 0, 0)
BLUE = (0, 0, 255)
WHITE = (255, 255, 255)
SKYBLUE = (119, 181, 254)
GREEN = (0, 255, 0)
RED = (255, 0, 0)

#Function principal du program
def mainLoop():
  global SCREEN
  mouseX = 0 #position X de la souris
  mouseY = 0 #position Y de la souris
  mouseClick = False #Click sourris
  mainRunning = True #Si False le jeu ce ferme

  while mainRunning == True:
    mouseClick = False

    #prend en charge Les events dans le menu
    for event in pygame.event.get():
      if event.type == pygame.QUIT:
        mainRunning = False
      elif event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE:
        mainRunning = False
      elif event.type == pygame.MOUSEMOTION:
        mouseX, mouseY = event.pos
      elif event.type == pygame.MOUSEBUTTONDOWN:
        mouseX, mouseY = event.pos
        mouseClick = True

    menuChoice = menu(mouseX, mouseY, mouseClick)

    #Decide quoi faire en fonction du choi de l'utilisateur dans le menu
    if menuChoice == 3:
      mainRunning = False
    elif menuChoice == 2:
      gameLoop(False)
    elif menuChoice == 1:
      gameLoop(True)

#Fonction qui gère le menu Principal
def menu(mouseX, mouseY,mouseClick):
  global BLACK, BLUE, DARKBLUE, SKYBLUE, comicSansMs
```

```
    soloButonColor = BLUE
    doubleButtonColor = BLUE
    exitButtonColor = BLUE

    #Creation des boutons
    soloPlayer = button(4, 8, 1)
    doublePlayer = button(4, 8, 2)
    progExit = button(4, 8, 3)

    #Ecriture dans les boutons
    rendSoloPlayer = comicSansMs.render("Solo Player", True, BLACK)
    rendDoublePlayer = comicSansMs.render("2 Players", True, BLACK)
    rendProgExit = comicSansMs.render("Quit Game", True, BLACK)

     #Gestion des boutons
    if soloPlayer.collidepoint(mouseX, mouseY):
        soloButonColor = SKYBLUE
        if mouseClick == True:
            return 1
    elif doublePlayer.collidepoint(mouseX, mouseY):
        doubleButtonColor = SKYBLUE
        if mouseClick == True:
            return 2
    elif progExit.collidepoint(mouseX, mouseY):
        exitButtonColor = SKYBLUE
        if mouseClick == True:
            return 3

    #Dessin des boutons
    SCREEN.fill(BLACK)

    pygame.draw.rect(SCREEN, soloButonColor, soloPlayer)
    pygame.draw.rect(SCREEN, doubleButtonColor, doublePlayer)
    pygame.draw.rect(SCREEN, exitButtonColor, progExit)

    SCREEN.blit(rendSoloPlayer, (soloPlayer.x, soloPlayer.y - 5))
    SCREEN.blit(rendDoublePlayer, (doublePlayer.x, doublePlayer.y - 5))
    SCREEN.blit(rendProgExit, (progExit.x, progExit.y - 5))

    pygame.display.flip()

 #Function   qui crée des entites bouton
def button(x, y, buttonOrder):
    global WINDOWWIDTH, WINDOWHEIGHT

    position = posX, posY = (WINDOWWIDTH  - (WINDOWWIDTH / x)) / 2, (WINDOWHEIGHT / y) *
buttonOrder + ((WINDOWHEIGHT / y) * (buttonOrder - 1)) + ((WINDOWHEIGHT / y) / 2)
    buttonSize = buttonW, buttonH = WINDOWWIDTH / x, (WINDOWHEIGHT / y)
    button = pygame.rect.Rect(posX, posY, buttonW, buttonH)

    return button

#Boucle de jeu
def gameLoop(bot):
    global WINDOWWIDTH, WINDOWHEIGHT

    gameRunning = True #Si False le programe return au menu principal
    clock = pygame.time.Clock()
```

```
    racketSize = racketWidth, racketHeight = 6, 80 #Taille de la raquette
    ballSize = ballWidth, ballHeight = 16, 16 #TSaille de la balle

     #creation du dictionnaire d'edentités
     entities = {'rightRacket': pygame.rect.Rect((WINDOWWIDTH - 20) - racketWidth,
(WINDOWHEIGHT / 2) - (racketHeight / 2), racketWidth, racketHeight),
                 'leftRacket': pygame.rect.Rect(20, (WINDOWHEIGHT / 2) - (racketHeight / 2),
racketWidth, racketHeight),
                 'ball': pygame.rect.Rect((WINDOWWIDTH / 2) - (ballWidth / 2), (WINDOWHEIGHT / 2) -
(ballHeight / 2), ballWidth, ballHeight),
                 'rightAim': (-18, 0),
                 'leftAim': (18, 0),
                 'ballMoment': (18, 0)}

    #Creations du dictionaire events
    events = {'quit': False,
              'escape': False,
              'w': False,
              's': False,
              'a': False,
              'd': False,
              'up': False,
              'down': False,
              'right': False,
              'left': False}

    score = (0, 0)
    gameReset(entities) #reset le jeu avant de le lancer

    while gameRunning == True:
        clock.tick(60)
        events = eventManager(events, bot)   #Enregistre les touches pressées

        #si le jour apuie sur Echape le jeu return ao menu principal
        if events is not None:
            if events['quit'] == True or events['escape'] == True:
                gameRunning = False

        if bot == True: #si mode sole active
            AI(entities, time)

        gameUpdate(entities, events) #met à jour les entités
        score = gameLogic(entities, score) #Gere les collisions et le score
        drawGame(entities, score) #Dessine le jeu

#Fonction qui dessine le jeu
def drawGame(entities, score):
    global WINDOWWIDTH, WINDOWHEIGHT, SCREEN, GREEN, BLACK
    rightAimX, rightAimY = entities['rightAim']
    leftAimX, leftAimY = entities['leftAim']
    tempScoreL, tempScoreR = score

    SCREEN.fill(BLACK)

    #Dessin du Terrain
    pygame.draw.line(SCREEN, WHITE, (20, 0), (20, WINDOWHEIGHT), 4)
    pygame.draw.line(SCREEN, WHITE, (WINDOWWIDTH - 22, 0), (WINDOWWIDTH - 22,
WINDOWHEIGHT), 4)
```

```
    pygame.draw.line(SCREEN, WHITE, ((WINDOWWIDTH / 2) - 2, 0), ((WINDOWWIDTH / 2) - 2,
WINDOWHEIGHT), 4)
    pygame.draw.line(SCREEN, WHITE, (20, 0), (WINDOWWIDTH - 22, 0), 3)
    pygame.draw.line(SCREEN, WHITE, (20, WINDOWHEIGHT - 3), (WINDOWWIDTH - 22,
WINDOWHEIGHT - 3), 4)
    pygame.draw.circle(SCREEN, WHITE, (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2)), 100,
4)

    #Dessin des Entitiés
    pygame.draw.rect(SCREEN, GREEN, entities['rightRacket'])
    pygame.draw.rect(SCREEN, GREEN, entities['leftRacket'])
    pygame.draw.line(SCREEN, RED, (entities['rightRacket'].x, entities['rightRacket'].y +
(entities['rightRacket'].height / 2)), ((entities['rightRacket'].x + (rightAimX * 2)),
entities['rightRacket'].y + (entities['rightRacket'].height / 2) + (rightAimY * 2)), 2)
    pygame.draw.line(SCREEN, RED, (entities['leftRacket'].x + entities['leftRacket'].width - 2,
entities['leftRacket'].y + (entities['leftRacket'].height / 2)),((entities['leftRacket'].x +
entities['leftRacket'].width - 2) + (leftAimX * 2), entities['leftRacket'].y +
(entities['leftRacket'].height / 2) + (leftAimY * 2)), 2)
    pygame.draw.ellipse(SCREEN, GREEN, entities['ball'])



    #Dessin Tableau des Scores
    rendPlayer1 = comicSansMs.render(str(tempScoreL), True, WHITE)
    rendPlayer2 = comicSansMs.render(str(tempScoreR), True, WHITE)

    SCREEN.blit(rendPlayer1, (WINDOWWIDTH / 4 - 10, 0))
    SCREEN.blit(rendPlayer2, ((WINDOWWIDTH / 4) * 3 - 10, 0))

    pygame.display.flip()

#Fonction qui prend en charge les entre de commandes
def eventManager(events, bot):
    if events is not None:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                events['quit'] = True
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    events['escape'] = True
                if bot != True:
                    if event.key == pygame.K_w:
                        events['w'] = True
                    elif event.key == pygame.K_s:
                        events['s'] = True
                    elif event.key == pygame.K_a:
                        events['a'] = True
                    elif event.key == pygame.K_d:
                        events['d'] = True
                if event.key == pygame.K_UP:
                    events['up'] = True
                elif event.key == pygame.K_DOWN:
                    events['down'] = True
                elif event.key == pygame.K_RIGHT:
                    events['right'] = True
                elif event.key == pygame.K_LEFT:
                    events['left'] = True
            elif event.type == pygame.KEYUP:
                if bot != True:
```

```
            if event.key == pygame.K_ESCAPE:
                events['escape'] = False
            elif event.key == pygame.K_w:
                events['w'] = False
            elif event.key == pygame.K_s:
                events['s'] = False
            elif event.key == pygame.K_a:
                events['a'] = False
            elif event.key == pygame.K_d:
                events['d'] = False
        if event.key == pygame.K_UP:
            events['up'] = False
        elif event.key == pygame.K_DOWN:
            events['down'] = False
        elif event.key == pygame.K_RIGHT:
            events['right'] = False
        elif event.key == pygame.K_LEFT:
            events['left'] = False

    return events #Renvoi les events

#fonction visant à mettre à jour les entités
def gameUpdate(entities, keyStatus):
    global WINDOWSWIDHT, WINDOWHEIGHT

    #deplacement des Raquettes
    if keyStatus is not None:
        if entities['rightRacket'].y >= 0:
            entities['rightRacket'].y = entities['rightRacket'].y - (6 * keyStatus['up'])
        if entities['rightRacket'].y <= WINDOWHEIGHT - entities['rightRacket'].height:
            entities['rightRacket'].y = entities['rightRacket'].y + (6 * keyStatus['down'])
        if entities['leftRacket'].y >= 0:
            entities['leftRacket'].y = entities['leftRacket'].y - (6 * keyStatus['w'])
        if entities['leftRacket'].y <= WINDOWHEIGHT - entities['leftRacket'].height:
            entities['leftRacket'].y = entities['leftRacket'].y + (6 * keyStatus['s'])

        #Deplacement du reticule droit
        rightAimX, rightAimY = entities['rightAim']
        if rightAimY >= -10:
            rightAimY = rightAimY - keyStatus['right']
        if rightAimY <= 10:
            rightAimY = rightAimY + keyStatus['left']
        rightAimX = 18 - math.sqrt(pow(rightAimY, 2))
        if rightAimX >= 0:
            rightAimX = rightAimX * (-1)
        entities['rightAim'] = rightAimX, rightAimY

        #Deplacement du reticule gauche
        leftAimX, leftAimY = entities['leftAim']
        if leftAimY >= -10:
            leftAimY = leftAimY - (0.15 * keyStatus['d'])
        if leftAimY <= 10:
            leftAimY = leftAimY + (0.15 * keyStatus['a'])
        leftAimX = 18 - math.sqrt(pow(leftAimY, 2))
        if leftAimX <= 0:
            leftAimX = leftAimX * (-1)
        entities['leftAim'] = leftAimX, leftAimY
```

```
        #Deplacement de la balle chaque frame
        ball1MomentX, ball1MomentY = entities['ballMoment']
        if (ball1MomentX / 2.5) < 1 and (ball1MomentX / 2.5) > - 1:
            entities['ball'].x = entities['ball'].x + 1
        else:
            entities['ball'].x = entities['ball'].x + (ball1MomentX / 2)

        entities['ball'].y = entities['ball'].y + (ball1MomentY / 2)


def gameLogic(entities, score):
    tempScoreL, tempScoreR = score
    hitSound = pygame.mixer.Sound("hitSound.wav")
    pointSound = pygame.mixer.Sound("pointSound.wav")
    hitSound.set_volume(0.5)
    pointSound.set_volume(0.5)

    #si la bale touche le haut de la fenêtre
    if entities['ball'].y <= 0:
        ballMomentX, ballMomentY = entities['ballMoment']
        ballMomentY = math.sqrt(pow(ballMomentY,2))
        entities['ballMoment'] = ballMomentX, ballMomentY
    #si la bale touche le bas de la fenêtre
    if  entities['ball'].y >= WINDOWHEIGHT - entities['ball'].height:
        ballMomentX, ballMomentY = entities['ballMoment']
        ballMomentY = 0 - math.sqrt(pow(ballMomentY,2))
        entities['ballMoment'] = ballMomentX, ballMomentY
    #lorsque la bale touche la raquette droite
    if entities['ball'].colliderect(entities['rightRacket']):
        hitSound.play()
        entities['ballMoment'] = entities['rightAim']
    #lorsque la bale touche la raquette gauche
    if entities['ball'].colliderect(entities['leftRacket']):
        hitSound.play()
        entities['ballMoment'] = entities['leftAim']
    #si la bale rentre dans les buts du joueur 1
    if entities['ball'].x >= WINDOWWIDTH:
        pointSound.play()
        tempScoreL = tempScoreL + 1
        gameReset(entities)
    #si la baller entre dans les buts du joueur 2
    if entities['ball'].x <= 0:
        pointSound.play()
        tempScoreR = tempScoreR + 1
        gameReset(entities)

    score = tempScoreL, tempScoreR
    return score #renvoi le score

def gameReset(entities):
    global WINDOWWIDTH, WINDOWHEIGHT

    tempMomentX, tempMomentY = entities['ballMoment']

    #reset les raquettes et le reticule
    entities['rightAim'] = (-18, 0)
    entities['leftAim'] =  (18, 0)
    entities['ball'].x = (WINDOWWIDTH / 2) - (entities['ball'].width / 2)
```

```
    entities['ball'].y = (WINDOWHEIGHT / 2) - (entities['ball'].height / 2)
    entities['rightRacket'].y = (WINDOWHEIGHT / 2) - (entities['ball'].height / 2)
    entities['leftRacket'].y = (WINDOWHEIGHT / 2) - (entities['ball'].height / 2)

    #lance la bale dans une direction aleatoir du terrain pour demarrer le second point
    tempMomentY = (random.randrange(34) - 17)
    if random.randrange(2) == 0:
        tempMomentX = 18 - tempMomentY
    else:
        tempMomentX = (18 - tempMomentY) * -1

    entities['ballMoment']  = (tempMomentX / 2), (tempMomentY / 2)

#Inteligence artificial qui est utilize dans le mode Solo
def AI(entities, timer):
    ballMomentumX, ballMomentumY = entities['ballMoment']
    bMX, bMY = 0, 0
    bHitX, bHitY = 0, 0
    offset = 0
    if timer < pygame.time.get_ticks() - 500:
        if ballMomentumX < 0: #si la balle va en direction du bot
            #on calcule le point ou la bale est en range de la raquette, on choisit une vise aléatoire
            temp = (entities['ball'].x - bHitX) // ballMomentumX
            bHitY = math.sqrt(pow((ballMomentumY * temp) - entities['ball'].y + (entities['ball'].height
/ 2), 2))
            bMX = ballMomentumX
            bMY = ballMomentumY
            offset = random.randrange(90) – 45 #on Crée un offset random afin faire le bot rater
            bHitY = offset + bHitY  #on ajute l'offset ao poit de collision avec la raquette
            tempAimX, tempAimY = entities['leftAim']
            tempAimX = random.randrange(1, 35) - 10
            tempAimY = tempAimX - 18
            entities['leftAim'] = tempAimX, tempAimY
            timer = pygame.time.get_ticks()

        #on bouge la raquette vers le poit de collision + l'offeset determine avant
        if entities['leftRacket'].y + (entities['leftRacket'].height / 2) > bHitY - 9 and
entities['leftRacket'].y + (entities['leftRacket'].height / 2) > bHitY + 9:
            if entities['leftRacket'].y >= 0:
                entities['leftRacket'].y = entities['leftRacket'].y - 6
        elif entities['leftRacket'].y + (entities['leftRacket'].height / 2) < bHitY - 9 and
entities['leftRacket'].y + (entities['leftRacket'].height / 2) < bHitY + 9:
            if entities['leftRacket'].y <= WINDOWHEIGHT - entities['leftRacket'].height:
                entities['leftRacket'].y = entities['leftRacket'].y + 6




mainLoop() #on exécute la boucle principal
pygame.quit() #on ferme pygame une fois que la boucle principal a fini d'être exécute
```