c0a21023 / **ProjExD** Public

<> **Code**   Issues 19   Pull requests   Actions   Projects   Security   ···

main ▾   ···

ProjExD / ex06 / **test.py** / <> Jump to ▾

c0a21023 基本機能＋追加機能実装   **History**

1 contributor

184 lines (158 sloc) | 5.88 KB   ···

```python
# -*- coding:utf-8 -*-
import pygame
from pygame.locals import *
import sys

# ボールの動きを計算
def calc_ball(ball_x, ball_y, ball_vx, ball_vy, bar1_x, bar1_y, bar2_x, bar2_y,wall_x,wall_y,wall_1,ball_1):
        global flag
        #プレイヤーのバーにあたった時
        # wall_rct=wall_1.get_rect()
        # ball_rct=ball_1.get_rect()
        if ball_x <= bar1_x + 10.:
            if ball_y >= bar1_y - 7.5 and ball_y <= bar1_y + 42.5:
                ball_x = 20.
                ball_vx = -ball_vx
        #AIのバーにあたった時
        if ball_x >= bar2_x - 15.:
            if ball_y >= bar2_y - 7.5 and ball_y <= bar2_y + 42.5:
                ball_x = 605.
                ball_vx = -ball_vx
        #障害物にあたった時
        #右側にあたった時
        if wall_1.colliderect(ball_1) and not flag:
            ball_vx=-ball_vx
            flag = True
        if not wall_1.colliderect(ball_1):
            flag = False


        #画面外に出たとき
        if ball_x < 5.:
            ball_x, ball_y = 320., 232.5
        elif ball_x > 620.:
            ball_x, ball_y = 307.5, 232.5
        if ball_y <= 10.:
            ball_vy = -ball_vy
            ball_y = 10.
        elif ball_y >= 457.5:
            ball_vy = -ball_vy
            ball_y = 457.5

        return ball_x, ball_y, ball_vx, ball_vy

# AIの動きを計算
def calc_ai(ball_x, ball_y, bar2_x, bar2_y):
    dy = ball_y - bar2_y
    if dy > 80: bar2_y += 20
    elif dy > 50: bar2_y += 15
    elif dy > 30: bar2_y += 12
    elif dy > 10: bar2_y += 8
    elif dy < -80: bar2_y -= 20
    elif dy < -50: bar2_y -= 15
    elif dy < -30: bar2_y -= 12
    elif dy < -10: bar2_y -= 8

    if bar2_y >= 420.: bar2_y = 420.
    elif bar2_y <= 10.: bar2_y = 10.
    return bar2_y

# プレイヤーの動き
def calc_player(bar1_y, bar1_dy):
    bar1_y += bar1_dy
    if bar1_y >= 420.: bar1_y = 420.
    elif bar1_y <= 10. : bar1_y = 10.
    return bar1_y

#障害物の動き
def wall_mov(wall_x,wall_y,wall_vx,wall_vy):
    if wall_y <= 15.:
        wall_vy = -wall_vy
        wall_y = 15.
    elif wall_y >= 400.5:
        wall_vy = -wall_vy
        wall_y = 400.5
    return wall_x, wall_y, wall_vx, wall_vy

# 得点の計算
def calc_score(ball_x, score1, score2):
    if ball_x < 5.:
        score2 += 1
    if ball_x > 620.:
        score1 += 1
    return score1, score2

# イベント処理
def event(bar1_dy):
    for event in pygame.event.get():
        if event.type == QUIT:          # 閉じるボタンが押されたら終了
            pygame.quit()
            sys.exit()
        if event.type == KEYDOWN:       # キーを押したら
            if event.key == K_UP:
                bar1_dy = -10
            elif event.key == K_DOWN:
                bar1_dy = 10
        elif event.type == KEYUP:       # キーを押し終わったら
            if event.key == K_UP:
                bar1_dy = 0.
            elif event.key == K_DOWN:
                bar1_dy = 0.
    return bar1_dy

def main():
    global flag
    # 各パラメータ
    bar1_x, bar1_y = 10. , 215.
    bar2_x, bar2_y = 620., 215.
    wall_x,wall_y=325,215.
    ball_x, ball_y = 307.5, 232.5
    bar1_dy, bar2_dy = 0. , 0.
    ball_vx, ball_vy = 250., 250.
    wall_vx, wall_vy = 250., 250.
    score1, score2 = 0,0
    ball_r = 7
    flag = False

    # pygameの設定
    pygame.init()                                       # Pygameの初期化
    screen = pygame.display.set_mode((640,480),0,32)    # 画面の大きさ
    pygame.display.set_caption("PONG")                  # 画面タイトル
    clock = pygame.time.Clock()
    font = pygame.font.SysFont(None,40)                 # 画面文字の設定

    # 背景の設定
    back = pygame.Surface((640,480))
    background = back.convert()

    # ボールを打つバーの設定
    bar = pygame.Surface((10,50))
    bar1 = bar.convert()
    bar1.fill((255,255,255))
    bar2 = bar.convert()
    bar2.fill((255,255,255))

    # ボールの設定
    circ_sur = pygame.Surface((20,20))
    pygame.draw.circle(circ_sur,(255,255,255),(ball_r, ball_r), ball_r)
    ball = circ_sur
    ball.set_colorkey((0,0,0))

    #障害物の設定
    wall_s = pygame.Surface((10,90))
    wall = wall_s.convert()
    wall.fill((255,255,255))

    while (1):
        # 各オブジェクトの描画
        scr=screen.blit(background,(0,0))
        screen.fill((75,0,125))
        pygame.draw.aaline(screen,(255,255,255),(330,5),(330,475))  # 中央線の描画
        bar_1=screen.blit(bar1,(bar1_x,bar1_y))                           # プレイヤー側バーの描画
        bar_2=screen.blit(bar2,(bar2_x,bar2_y))                           # CPU側バーの描画
        wall_1=screen.blit(wall,(wall_x,wall_y))                                 #
        ball_1=screen.blit(ball,(ball_x, ball_y))                          # ボールの描画
        screen.blit(font.render(str(score1), True,(255,255,255)),(250.,10.))
        screen.blit(font.render(str(score2), True,(255,255,255)),(400.,10.))



        # プレイヤー側バーの位置
        bar1_dy = event(bar1_dy)
        bar1_y = calc_player(bar1_y,bar1_dy)

        # ボールの移動
        time_passed = clock.tick(30)
        time_sec = time_passed / 1000.0
        ball_x += ball_vx * time_sec
        ball_y += ball_vy * time_sec
        wall_y += wall_vy * time_sec

        # 得点の計算
        score1, score2 = calc_score(ball_x, score1, score2)

        # CPUのバー速度を計算
        bar2_y = calc_ai(ball_x, ball_y, bar2_x, bar2_y)

        # ボールの速度・位置を計算
        ball_x, ball_y, ball_vx, ball_vy = calc_ball(ball_x, ball_y, ball_vx, ball_vy, bar1_x, bar1_y, bar2_x, bar2_y,wall_x,wall_y,wall_1,ball_1)
        wall_x, wall_y, wall_vx, wall_vy = wall_mov(wall_x,wall_y,wall_vx,wall_vy)
        pygame.display.update()                                     # 画面を更新


if __name__ == "__main__":
    main()
```