import os  
import sys  
import math  
import time  
import pygame  
current\_path = os.getcwd()  
import pymunk as pm  
from characters import Bird  
from level import Level  
  
pygame.init()  
screen = pygame.display.set\_mode((1200, 650))  
redbird = pygame.image.load(  
 "../resources/images/red-bird3.png").convert\_alpha()  
background2 = pygame.image.load(  
 "../resources/images/background3.png").convert\_alpha()  
sling\_image = pygame.image.load(  
 "../resources/images/sling-3.png").convert\_alpha()  
full\_sprite = pygame.image.load(  
 "../resources/images/full-sprite.png").convert\_alpha()  
rect = pygame.Rect(181, 1050, 50, 50)  
cropped = full\_sprite.subsurface(rect).copy()  
pig\_image = pygame.transform.scale(cropped, (30, 30))  
buttons = pygame.image.load(  
 "../resources/images/selected-buttons.png").convert\_alpha()  
pig\_happy = pygame.image.load(  
 "../resources/images/pig\_failed.png").convert\_alpha()  
stars = pygame.image.load(  
 "../resources/images/stars-edited.png").convert\_alpha()  
rect = pygame.Rect(0, 0, 200, 200)  
star1 = stars.subsurface(rect).copy()  
rect = pygame.Rect(204, 0, 200, 200)  
star2 = stars.subsurface(rect).copy()  
rect = pygame.Rect(426, 0, 200, 200)  
star3 = stars.subsurface(rect).copy()  
rect = pygame.Rect(164, 10, 60, 60)  
pause\_button = buttons.subsurface(rect).copy()  
rect = pygame.Rect(24, 4, 100, 100)  
replay\_button = buttons.subsurface(rect).copy()  
rect = pygame.Rect(142, 365, 130, 100)  
next\_button = buttons.subsurface(rect).copy()  
clock = pygame.time.Clock()  
rect = pygame.Rect(18, 212, 100, 100)  
play\_button = buttons.subsurface(rect).copy()  
clock = pygame.time.Clock()  
running = True  
# the base of the physics  
space = pm.Space()  
space.gravity = (0.0, -700.0)  
pigs = []  
birds = []  
balls = []  
polys = []  
beams = []  
columns = []  
poly\_points = []  
ball\_number = 0  
polys\_dict = {}  
mouse\_distance = 0  
rope\_lenght = 90  
angle = 0  
x\_mouse = 0  
y\_mouse = 0  
count = 0  
mouse\_pressed = False  
t1 = 0  
tick\_to\_next\_circle = 10  
RED = (255, 0, 0)  
BLUE = (0, 0, 255)  
BLACK = (0, 0, 0)  
WHITE = (255, 255, 255)  
sling\_x, sling\_y = 135, 450  
sling2\_x, sling2\_y = 160, 450  
score = 0  
game\_state = 0  
bird\_path = []  
counter = 0  
restart\_counter = False  
bonus\_score\_once = True  
bold\_font = pygame.font.SysFont("arial", 30, bold=True)  
bold\_font2 = pygame.font.SysFont("arial", 40, bold=True)  
bold\_font3 = pygame.font.SysFont("arial", 50, bold=True)  
wall = False  
  
# Static floor  
static\_body = pm.Body(body\_type=pm.Body.STATIC)  
static\_lines = [pm.Segment(static\_body, (0.0, 060.0), (1200.0, 060.0), 0.0)]  
static\_lines1 = [pm.Segment(static\_body, (1200.0, 060.0), (1200.0, 800.0), 0.0)]  
for line in static\_lines:  
 line.elasticity = 0.95  
 line.friction = 1  
 line.collision\_type = 3  
for line in static\_lines1:  
 line.elasticity = 0.95  
 line.friction = 1  
 line.collision\_type = 3  
space.add(static\_lines)  
  
  
def to\_pygame(p):  
 *"""Convert pymunk to pygame coordinates"""* return int(p.x), int(-p.y+600)  
  
  
def vector(p0, p1):  
 *"""Return the vector of the points  
 p0 = (xo,yo), p1 = (x1,y1)"""* a = p1[0] - p0[0]  
 b = p1[1] - p0[1]  
 return (a, b)  
  
  
def unit\_vector(v):  
 *"""Return the unit vector of the points  
 v = (a,b)"""* h = ((v[0]\*\*2)+(v[1]\*\*2))\*\*0.5  
 if h == 0:  
 h = 0.000000000000001  
 ua = v[0] / h  
 ub = v[1] / h  
 return (ua, ub)  
  
  
def distance(xo, yo, x, y):  
 *"""distance between points"""* dx = x - xo  
 dy = y - yo  
 d = ((dx \*\* 2) + (dy \*\* 2)) \*\* 0.5  
 return d  
  
  
def load\_music():  
 *"""Load the music"""* song1 = '../resources/sounds/angry-birds.ogg'  
 pygame.mixer.music.load(song1)  
 pygame.mixer.music.play(-1)  
  
  
def sling\_action():  
 *"""Set up sling behavior"""* global mouse\_distance  
 global rope\_lenght  
 global angle  
 global x\_mouse  
 global y\_mouse  
 # Fixing bird to the sling rope  
 v = vector((sling\_x, sling\_y), (x\_mouse, y\_mouse))  
 uv = unit\_vector(v)  
 uv1 = uv[0]  
 uv2 = uv[1]  
 mouse\_distance = distance(sling\_x, sling\_y, x\_mouse, y\_mouse)  
 pu = (uv1\*rope\_lenght+sling\_x, uv2\*rope\_lenght+sling\_y)  
 bigger\_rope = 102  
 x\_redbird = x\_mouse - 20  
 y\_redbird = y\_mouse - 20  
 if mouse\_distance > rope\_lenght:  
 pux, puy = pu  
 pux -= 20  
 puy -= 20  
 pul = pux, puy  
 screen.blit(redbird, pul)  
 pu2 = (uv1\*bigger\_rope+sling\_x, uv2\*bigger\_rope+sling\_y)  
 pygame.draw.line(screen, (0, 0, 0), (sling2\_x, sling2\_y), pu2, 5)  
 screen.blit(redbird, pul)  
 pygame.draw.line(screen, (0, 0, 0), (sling\_x, sling\_y), pu2, 5)  
 else:  
 mouse\_distance += 10  
 pu3 = (uv1\*mouse\_distance+sling\_x, uv2\*mouse\_distance+sling\_y)  
 pygame.draw.line(screen, (0, 0, 0), (sling2\_x, sling2\_y), pu3, 5)  
 screen.blit(redbird, (x\_redbird, y\_redbird))  
 pygame.draw.line(screen, (0, 0, 0), (sling\_x, sling\_y), pu3, 5)  
 # Angle of impulse  
 dy = y\_mouse - sling\_y  
 dx = x\_mouse - sling\_x  
 if dx == 0:  
 dx = 0.00000000000001  
 angle = math.atan((float(dy))/dx)  
  
  
def draw\_level\_cleared():  
 *"""Draw level cleared"""* global game\_state  
 global bonus\_score\_once  
 global score  
 level\_cleared = bold\_font3.render("Level Cleared!", 1, WHITE)  
 score\_level\_cleared = bold\_font2.render(str(score), 1, WHITE)  
 if level.number\_of\_birds >= 0 and len(pigs) == 0:  
 if bonus\_score\_once:  
 score += (level.number\_of\_birds-1) \* 10000  
 bonus\_score\_once = False  
 game\_state = 4  
 rect = pygame.Rect(300, 0, 600, 800)  
 pygame.draw.rect(screen, BLACK, rect)  
 screen.blit(level\_cleared, (450, 90))  
 if score >= level.one\_star and score <= level.two\_star:  
 screen.blit(star1, (310, 190))  
 if score >= level.two\_star and score <= level.three\_star:  
 screen.blit(star1, (310, 190))  
 screen.blit(star2, (500, 170))  
 if score >= level.three\_star:  
 screen.blit(star1, (310, 190))  
 screen.blit(star2, (500, 170))  
 screen.blit(star3, (700, 200))  
 screen.blit(score\_level\_cleared, (550, 400))  
 screen.blit(replay\_button, (510, 480))  
 screen.blit(next\_button, (620, 480))  
  
  
def draw\_level\_failed():  
 *"""Draw level failed"""* global game\_state  
 failed = bold\_font3.render("Level Failed", 1, WHITE)  
 if level.number\_of\_birds <= 0 and time.time() - t2 > 5 and len(pigs) > 0:  
 game\_state = 3  
 rect = pygame.Rect(300, 0, 600, 800)  
 pygame.draw.rect(screen, BLACK, rect)  
 screen.blit(failed, (450, 90))  
 screen.blit(pig\_happy, (380, 120))  
 screen.blit(replay\_button, (520, 460))  
  
  
def restart():  
 *"""Delete all objects of the level"""* pigs\_to\_remove = []  
 birds\_to\_remove = []  
 columns\_to\_remove = []  
 beams\_to\_remove = []  
 for pig in pigs:  
 pigs\_to\_remove.append(pig)  
 for pig in pigs\_to\_remove:  
 space.remove(pig.shape, pig.shape.body)  
 pigs.remove(pig)  
 for bird in birds:  
 birds\_to\_remove.append(bird)  
 for bird in birds\_to\_remove:  
 space.remove(bird.shape, bird.shape.body)  
 birds.remove(bird)  
 for column in columns:  
 columns\_to\_remove.append(column)  
 for column in columns\_to\_remove:  
 space.remove(column.shape, column.shape.body)  
 columns.remove(column)  
 for beam in beams:  
 beams\_to\_remove.append(beam)  
 for beam in beams\_to\_remove:  
 space.remove(beam.shape, beam.shape.body)  
 beams.remove(beam)  
  
  
def post\_solve\_bird\_pig(arbiter, space, \_):  
 *"""Collision between bird and pig"""* surface=screen  
 a, b = arbiter.shapes  
 bird\_body = a.body  
 pig\_body = b.body  
 p = to\_pygame(bird\_body.position)  
 p2 = to\_pygame(pig\_body.position)  
 r = 30  
 pygame.draw.circle(surface, BLACK, p, r, 4)  
 pygame.draw.circle(surface, RED, p2, r, 4)  
 pigs\_to\_remove = []  
 for pig in pigs:  
 if pig\_body == pig.body:  
 pig.life -= 20  
 pigs\_to\_remove.append(pig)  
 global score  
 score += 10000  
 for pig in pigs\_to\_remove:  
 space.remove(pig.shape, pig.shape.body)  
 pigs.remove(pig)  
  
  
def post\_solve\_bird\_wood(arbiter, space, \_):  
 *"""Collision between bird and wood"""* poly\_to\_remove = []  
 if arbiter.total\_impulse.length > 1100:  
 a, b = arbiter.shapes  
 for column in columns:  
 if b == column.shape:  
 poly\_to\_remove.append(column)  
 for beam in beams:  
 if b == beam.shape:  
 poly\_to\_remove.append(beam)  
 for poly in poly\_to\_remove:  
 if poly in columns:  
 columns.remove(poly)  
 if poly in beams:  
 beams.remove(poly)  
 space.remove(b, b.body)  
 global score  
 score += 5000  
  
  
def post\_solve\_pig\_wood(arbiter, space, \_):  
 *"""Collision between pig and wood"""* pigs\_to\_remove = []  
 if arbiter.total\_impulse.length > 700:  
 pig\_shape, wood\_shape = arbiter.shapes  
 for pig in pigs:  
 if pig\_shape == pig.shape:  
 pig.life -= 20  
 global score  
 score += 10000  
 if pig.life <= 0:  
 pigs\_to\_remove.append(pig)  
 for pig in pigs\_to\_remove:  
 space.remove(pig.shape, pig.shape.body)  
 pigs.remove(pig)  
  
  
# bird and pigs  
space.add\_collision\_handler(0, 1).post\_solve=post\_solve\_bird\_pig  
# bird and wood  
space.add\_collision\_handler(0, 2).post\_solve=post\_solve\_bird\_wood  
# pig and wood  
space.add\_collision\_handler(1, 2).post\_solve=post\_solve\_pig\_wood  
load\_music()  
level = Level(pigs, columns, beams, space)  
level.number = 0  
level.load\_level()  
  
while running:  
 # Input handling  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 running = False  
 elif event.type == pygame.KEYDOWN and event.key == pygame.K\_ESCAPE:  
 running = False  
 elif event.type == pygame.KEYDOWN and event.key == pygame.K\_w:  
 # Toggle wall  
 if wall:  
 space.remove(static\_lines1)  
 wall = False  
 else:  
 space.add(static\_lines1)  
 wall = True  
  
 elif event.type == pygame.KEYDOWN and event.key == pygame.K\_s:  
 space.gravity = (0.0, -10.0)  
 level.bool\_space = True  
 elif event.type == pygame.KEYDOWN and event.key == pygame.K\_n:  
 space.gravity = (0.0, -700.0)  
 level.bool\_space = False  
 if (pygame.mouse.get\_pressed()[0] and x\_mouse > 100 and  
 x\_mouse < 250 and y\_mouse > 370 and y\_mouse < 550):  
 mouse\_pressed = True  
 if (event.type == pygame.MOUSEBUTTONUP and  
 event.button == 1 and mouse\_pressed):  
 # Release new bird  
 mouse\_pressed = False  
 if level.number\_of\_birds > 0:  
 level.number\_of\_birds -= 1  
 t1 = time.time()\*1000  
 xo = 154  
 yo = 156  
 if mouse\_distance > rope\_lenght:  
 mouse\_distance = rope\_lenght  
 if x\_mouse < sling\_x+5:  
 bird = Bird(mouse\_distance, angle, xo, yo, space)  
 birds.append(bird)  
 else:  
 bird = Bird(-mouse\_distance, angle, xo, yo, space)  
 birds.append(bird)  
 if level.number\_of\_birds == 0:  
 t2 = time.time()  
 if event.type == pygame.MOUSEBUTTONUP and event.button == 1:  
 if (x\_mouse < 60 and y\_mouse < 155 and y\_mouse > 90):  
 game\_state = 1  
 if game\_state == 1:  
 if x\_mouse > 500 and y\_mouse > 200 and y\_mouse < 300:  
 # Resume in the paused screen  
 game\_state = 0  
 if x\_mouse > 500 and y\_mouse > 300:  
 # Restart in the paused screen  
 restart()  
 level.load\_level()  
 game\_state = 0  
 bird\_path = []  
 if game\_state == 3:  
 # Restart in the failed level screen  
 if x\_mouse > 500 and x\_mouse < 620 and y\_mouse > 450:  
 restart()  
 level.load\_level()  
 game\_state = 0  
 bird\_path = []  
 score = 0  
 if game\_state == 4:  
 # Build next level  
 if x\_mouse > 610 and y\_mouse > 450:  
 restart()  
 level.number += 1  
 game\_state = 0  
 level.load\_level()  
 score = 0  
 bird\_path = []  
 bonus\_score\_once = True  
 if x\_mouse < 610 and x\_mouse > 500 and y\_mouse > 450:  
 # Restart in the level cleared screen  
 restart()  
 level.load\_level()  
 game\_state = 0  
 bird\_path = []  
 score = 0  
 x\_mouse, y\_mouse = pygame.mouse.get\_pos()  
 # Draw background  
 screen.fill((130, 200, 100))  
 screen.blit(background2, (0, -50))  
 # Draw first part of the sling  
 rect = pygame.Rect(50, 0, 70, 220)  
 screen.blit(sling\_image, (138, 420), rect)  
 # Draw the trail left behind  
 for point in bird\_path:  
 pygame.draw.circle(screen, WHITE, point, 5, 0)  
 # Draw the birds in the wait line  
 if level.number\_of\_birds > 0:  
 for i in range(level.number\_of\_birds-1):  
 x = 100 - (i\*35)  
 screen.blit(redbird, (x, 508))  
 # Draw sling behavior  
 if mouse\_pressed and level.number\_of\_birds > 0:  
 sling\_action()  
 else:  
 if time.time()\*1000 - t1 > 300 and level.number\_of\_birds > 0:  
 screen.blit(redbird, (130, 426))  
 else:  
 pygame.draw.line(screen, (0, 0, 0), (sling\_x, sling\_y-8),  
 (sling2\_x, sling2\_y-7), 5)  
 birds\_to\_remove = []  
 pigs\_to\_remove = []  
 counter += 1  
 # Draw birds  
 for bird in birds:  
 if bird.shape.body.position.y < 0:  
 birds\_to\_remove.append(bird)  
 p = to\_pygame(bird.shape.body.position)  
 x, y = p  
 x -= 22  
 y -= 20  
 screen.blit(redbird, (x, y))  
 pygame.draw.circle(screen, BLUE,  
 p, int(bird.shape.radius), 2)  
 if counter >= 3 and time.time() - t1 < 5:  
 bird\_path.append(p)  
 restart\_counter = True  
 if restart\_counter:  
 counter = 0  
 restart\_counter = False  
 # Remove birds and pigs  
 for bird in birds\_to\_remove:  
 space.remove(bird.shape, bird.shape.body)  
 birds.remove(bird)  
 for pig in pigs\_to\_remove:  
 space.remove(pig.shape, pig.shape.body)  
 pigs.remove(pig)  
 # Draw static lines  
 for line in static\_lines:  
 body = line.body  
 pv1 = body.position + line.a.rotated(body.angle)  
 pv2 = body.position + line.b.rotated(body.angle)  
 p1 = to\_pygame(pv1)  
 p2 = to\_pygame(pv2)  
 pygame.draw.lines(screen, (150, 150, 150), False, [p1, p2])  
 i = 0  
 # Draw pigs  
 for pig in pigs:  
 i += 1  
 # print (i,pig.life)  
 pig = pig.shape  
 if pig.body.position.y < 0:  
 pigs\_to\_remove.append(pig)  
  
 p = to\_pygame(pig.body.position)  
 x, y = p  
  
 angle\_degrees = math.degrees(pig.body.angle)  
 img = pygame.transform.rotate(pig\_image, angle\_degrees)  
 w,h = img.get\_size()  
 x -= w\*0.5  
 y -= h\*0.5  
 screen.blit(img, (x, y))  
 pygame.draw.circle(screen, BLUE, p, int(pig.radius), 2)  
 # Draw columns and Beams  
 for column in columns:  
 column.draw\_poly('columns', screen)  
 for beam in beams:  
 beam.draw\_poly('beams', screen)  
 # Update physics  
 dt = 1.0/50.0/2.  
 for x in range(2):  
 space.step(dt) # make two updates per frame for better stability  
 # Drawing second part of the sling  
 rect = pygame.Rect(0, 0, 60, 200)  
 screen.blit(sling\_image, (120, 420), rect)  
 # Draw score  
 score\_font = bold\_font.render("SCORE", 1, WHITE)  
 number\_font = bold\_font.render(str(score), 1, WHITE)  
 screen.blit(score\_font, (1060, 90))  
 if score == 0:  
 screen.blit(number\_font, (1100, 130))  
 else:  
 screen.blit(number\_font, (1060, 130))  
 screen.blit(pause\_button, (10, 90))  
 # Pause option  
 if game\_state == 1:  
 screen.blit(play\_button, (500, 200))  
 screen.blit(replay\_button, (500, 300))  
 draw\_level\_cleared()  
 draw\_level\_failed()  
 pygame.display.flip()  
 clock.tick(50)  
 pygame.display.set\_caption("fps: " + str(clock.get\_fps()))