```
# -*- coding: utf-8 -*-
import pygame                           # ゲーム作成用のpygameライブラリ
import sys                              # システム関連のライブラリ
import random                           # 乱数関連のライブラリ

SCREEN = pygame.Rect(0, 0, 400, 640)    # 画面サイズの設定値（横４００ピクセルｘ縦６４０ピクセル）
FPS = 60                                # FPS(1秒間の画面更新頻度)の設定値（コンピュータゲームの標準値）
CROW_X_POS = [500, 800]                 # カラスの出現X座標
CROW_Y_RANGE = [50, 600]                # カラスの出現Y座標範囲
CROW_SPEED_RANGE = [-100, 100]          # カラスの移動速度範囲
CROW_TURN_MAX = 10                      # カラスの反転時間最大
DIFFICULT_MAX = 3                       # 最大難易度設定
SPARROW_X = 120                         # スズメのX座標（固定値）
SPARROW_START_Y = 240                   # スズメのスタートY座標
SPARROW_UP_END_Y = 0                    # スズメの限界高度
SPARROW_DOWN_END_Y = 640                # スズメの墜落高度
CLIMB_SPEED_MAX = 300                   # スズメの最大上昇速度（秒速ピクセル　Pixel/sec)
GRAVITY = 850                           # 重力加速度

def SparrowStartPos():                                          # スズメをスタート位置にセットする関数
    global sparrowY, climbSpeed                                 # グローバル変数へアクセス宣言
    sparrowY = SPARROW_START_Y                                  # スズメの初期座標
    climbSpeed = 0                                              # スズメの現在の速度を０に

def CrowResetXY(i):                                             # カラスを右に移動して座標を再設定する関数
        crowX[i] = CROW_X_POS[i]                                # Xは固定位置
        crowY[i] = random.uniform(CROW_Y_RANGE[0],CROW_Y_RANGE[1])  # Yの高さは範囲内で

def CrowStartPos():                                             # 全てのカラスの初期位置を設定する関数
    for i in range(0,len(CROW_X_POS)):                          # カラスの個体数だけ処理
        CrowResetXY(i)                                          # 座標を再設定する共通関数呼出

# ゲーム処理はここから開始します(main関数のような扱いです)
pygame.init()                                                   # Pygameの初期化（pygameを使う前に一度実行する）
screen = pygame.display.set_mode(SCREEN.size)                   # 設定したサイズでウィンドウを作成
pygame.display.set_caption("Sparrow")                           # ウィンドウのタイトルを設定(スズメの英名)
clock = pygame.time.Clock()                                     # 設定したタイミングでのリアルタイム処理設定
sysfont = pygame.font.SysFont(None, 40)                         # 標準フォント指定(Noneは標準)
```

```
elapsedTime = 0                                                 # 飛翔時間を初期化
difficult = 0                                                   # 難易度情報の初期化
deltaTime = 0.0                                                 # フレーム更新時間の初期化
playing = False                                                 # ゲーム状態をゲームオーバーに設定
seFlap = pygame.mixer.Sound("flap.mp3")                         # 飛翔音の読み込み
seHunted = pygame.mixer.Sound("hunted.mp3")                     # 狩られ音の読み込み

crowX = [0, 0]                                                  # カラスのX座標
crowY = [0, 0]                                                  # カラスのY座標
crowSpeed = [0, 0]                                              # カラスの移動速度
crowTurn = [0, 0]                                               # カラスの反転時間
CrowStartPos()                                                  # カラスの開始位置の設定
SparrowStartPos()                                               # スズメを初期位置にセットする
crowSpr = pygame.image.load("crow.png").convert_alpha()         # カラスの画像ファイルの読み込み
crowRect = crowSpr.get_rect()                                   # カラスの描画範囲矩形の設定
sparrowSpr = [pygame.image.load("sparrowd.png").convert_alpha()]    # スズメの滑空画像の読み込み
sparrowSpr.append(pygame.image.load("sparrowu.png").convert_alpha())# スズメの羽ばたき画像の読み込み
sparrowRect = sparrowSpr[0].get_rect()                          # スズメ画像の描画範囲矩形切り出し
sparrowRect.center = (SPARROW_X, sparrowY)                      # スズメ画像の描画位置を設定

bldgSpr = pygame.image.load("building.png").convert_alpha()     # ビル画像の読み込み
bldgRect = bldgSpr.get_rect()                                   # ビル画像の描画範囲矩形切り出し

while (True):                                                   # リアルタイム処理の無限ループ
    screen.fill("skyblue")                                      # 画面をスカイブルーで塗り潰す
    pygame.draw.rect(screen,"gray92",pygame.Rect(0,600,480,40))     # コンクリート地面

    for i in range(2):  # 移動するビルを複数描画する
        bldgRect.center = (200 - ((elapsedTime + i * 4) % 8 - 4) * 80, 540) # 飛翔時間からビルの位置を作る
        screen.blit(bldgSpr, bldgRect)                                      # ビルの描画

    if playing:     # ゲーム中なら
        difficult = min(difficult + 0.2 * deltaTime, DIFFICULT_MAX) # 難易度を上昇させる
        for i in range(0,len(CROW_X_POS)):      # 全てのカラスの処理を繰り返す
            crowX[i] -= 100.0 * deltaTime       # カラスを左に移動
            if difficult >= DIFFICULT_MAX:      # 最高難易度なら突然の反転が出来るようになる
                if crowTurn[i] > 0:             # 反転した事が無い
                    crowTurn[i] -= deltaTime    # 反転待ちタイマーを減らす
```

```
            if crowTurn[i] < 0:          # タイマーが切れたなら
                crowSpeed[i] *= -1       # 反転

        if crowX[i] < -50:               # 画面の左に消えたなら
            crowX[i] += 600              # 画面の右に移動
            crowY[i] = random.uniform(CROW_Y_RANGE[0],CROW_Y_RANGE[1]) # 範囲内で高さを設定する
            crowSpeed[i] = random.uniform(CROW_SPEED_RANGE[0],CROW_SPEED_RANGE[1]) * difficult # 範囲内で速度指定
            crowTurn[i] = random.uniform(2, CROW_TURN_MAX)              # 範囲内で反転時間を設定する

        crowY[i] += crowSpeed[i] * deltaTime    # 上下移動
        if crowY[i] <= 0 or crowY[i] >= 640:    # 移動範囲を超えたなら
            crowSpeed[i] *= -1                  # 移動の向きを反転
        crowRect.center = (crowX[i], crowY[i])  # 描画位置を変更
        screen.blit(crowSpr, crowRect)          # カラスの描画

        if pygame.Rect.colliderect(sparrowRect, crowRect):  # スズメに接触したなら
            SparrowStartPos()                               #  スズメを初期位置に
            seHunted.play()                                 #  捕食音声を再生
            playing = False                                 #  ゲーム状態をゲームオーバーに変更

    elapsedTime += deltaTime                                # 飛行時間の更新
    sparrowY = sparrowY + climbSpeed / FPS                  # 放物線運動に応じたスズメのY座標を求める
    climbSpeed += GRAVITY / FPS                             # 重力加速度を速度に加える
    if sparrowY <= SPARROW_UP_END_Y or sparrowY >= SPARROW_DOWN_END_Y:  # 飛行限界高度か墜落かしたら
        playing = False                                     # ゲーム状態をゲームオーバーに変更
else:   # ゲームオーバーなら
    SparrowStartPos()                                       # スズメをスタート位置に戻す
    screen.blit(sysfont.render("Push Enter to Start", False, (0,255,0)), (80,200)) # ゲーム開始を促す文章表示

clock.tick(FPS)                                             # フレームレート(60fps)
deltaTime = clock.get_time() / 1000.0                       # 前フレームの更新からの経過時間を抽出する
screen.blit(sysfont.render("TIME:"+str(int(elapsedTime)), False, (0,0,0)), (160,0))    # 飛翔時間の描画

sparrowRect.center = (SPARROW_X, sparrowY)                  # スズメの描画位置変更
anmBody = 0 if climbSpeed >= 0 else 1                       # 上昇中は２番目の絵、加工中は１番目の絵
screen.blit(sparrowSpr[anmBody], sparrowRect)               # スズメの体画像を描画
pygame.display.update()                                     # ゲーム画面の更新を行う
```

```
for event in pygame.event.get():                                    # イベントを全て取得するループ
    if event.type == pygame.KEYDOWN:                                # キーが押されてて
        if event.key == pygame.K_SPACE and playing:                 #  スペースバーで残り時間ある時
            climbSpeed = - CLIMB_SPEED_MAX                          #   上昇開始
            seFlap.play()                                           #   羽ばたき音の再生
        elif event.key == pygame.K_RETURN and playing == False:     #  リターンキーでゲームオーバーの時
            elapsedTime = 0                                         #   経過時間の初期化
            difficult = 0                                           #   難易度を最低に
            CrowStartPos()                                          #   カラスを初期位置に
            playing = True                                          #   プレイ中の宣言
        elif event.key == pygame.K_ESCAPE:                          #  ESCキーが押された時
            pygame.quit(); sys.exit()                               #   終了処理
    elif event.type == pygame.QUIT:                                 # 終了イベントを受けた時
        pygame.quit(); sys.exit()                                   #  終了処理
```