```
# -*- 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, 640, 400)    # 画面サイズの設定値（横６４０ピクセルｘ縦４００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）
REST_TIME = 60                          # ゲームの制限時間（秒で指定）
LANE_LEFT = -100                        # 車道の左端座標
LANE_RIGHT = 740                        # 車道の右端座標
LANE_Y = [320,240,160,80]               # 車線のY座標（固定値）
LANE_SPEED_LOW = [-80,-40,40,100]       # 車線ごとの最低速度（マイナスは左移動）
LANE_SPEED_HIGH= [-150,-80,100,200]     # 車線ごとの最高速度（マイナスは左移動）
MAN_X = 320                             # 人のX座標（固定値）
MAN_START_Y = 370                       # 人のスタートY座標
MAN_END_Y = 10                          # 人のエンドY座標
MAN_SPEED_MAX = 150                     # 人の最大速度（秒速ピクセル　Pixel/sec）
LOOP_HEAD = 0.6                         # 頭アニメーションループ時間（秒で指定）
CAR_SPR_FILE = ["taxi.png", "ktrack.png", "track.png", "sports.png"]    # 車画像ファイル、プログラムと同じ場所に置く事

def ManStartPos():                      # 人をスタート位置にセットする関数
    global manY, manSpeed                                                   # グローバル変数へアクセス宣言
    manY = MAN_START_Y                                                      # 人の初期座標
    manSpeed = 0                                                            # 人の現在の速度

def CarRestart(i):                      # 車をスタート位置にセットする関数
    if LANE_SPEED_LOW[i] < 0:                                               # 左移動かつ
        if carX[i] <= LANE_LEFT:                                            #  左の端を過ぎたら
            carX[i] = LANE_RIGHT                                            #   右の出現位置に移動
            carSpeed[i] = random.uniform(LANE_SPEED_LOW[i],LANE_SPEED_HIGH[i])  #   移動速度も再計算
    else:                                                                   # 右移動かつ
        if carX[i] >= LANE_RIGHT:                                           #  右の端を過ぎたら
            carX[i] = LANE_LEFT                                             #   左の出現位置に移動
            carSpeed[i] = random.uniform(LANE_SPEED_LOW[i],LANE_SPEED_HIGH[i])  #   移動速度も再計算
```

```
35
36 # 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)
37 pygame.init()                                                    # Pygameの初期化（pygameを使う前に一度実行する）
38 screen = pygame.display.set_mode(SCREEN.size)                    # 設定したサイズでウィンドウを作成
39 pygame.display.set_caption("AcrossRoadway")
40 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
41 sysfont = pygame.font.SysFont(None, 40)                         # 標準フォント指定(Noneは標準)
42 success = failed = 0                                            # 横断数と失敗数の初期化
43 timeHead = anmHead = anmBody = 0                                # アニメーション情報の初期化
44 restTime = 0                                                    # 残り時間の初期化
45 seRun = pygame.mixer.Sound("run.mp3")                          # 足音音声の読み込み
46 seSuccess = pygame.mixer.Sound("success.mp3")                  # 横断成功音声の読み込み
47 seFalse = pygame.mixer.Sound("false.mp3")                      # 横断失敗音声の読み込み
48 carSpeed = [0, 0, 0, 0]                                        # 車ごとの速度領域を確保
49 carX = [LANE_LEFT, LANE_LEFT, LANE_RIGHT, LANE_RIGHT]          # 車ごとの最初のX座標を設定
50 carSpr = []                                                     # 車の画像イメージ格納場所の初期化
51 carRect = []                                                    # 車画像の矩形情報格納場所の初期化
52 for i in range(0,len(LANE_SPEED_LOW)):                         # 車線の分だけ初期値を作成する
53     carSpr.append(pygame.image.load(CAR_SPR_FILE[i]).convert_alpha())   # 車画像ファイルの読み込み
54     carRect.append(carSpr[-1].get_rect())                       # 車画像の矩形情報取得
55     CarRestart(i)                                               # 車の位置と速度をセット
56 ManStartPos()                                                   # 人を初期位置にセットする
57 manSprHead = [pygame.image.load("manHead.png").convert_alpha()]     # 人の頭画像ファイルの読み込み
58 manSprHead.append(pygame.transform.flip(manSprHead[-1],True,False)) # 人の頭の左右反転イメージの作成
59 manSprBody = [pygame.image.load("manBody.png").convert_alpha()]     # 人の体画像ファイルの読み込み
60 manSprBody.append(pygame.transform.flip(manSprBody[-1],True,False)) # 人の体の左右反転イメージの作成
61 manRect = manSprBody[0].get_rect()                               # 人画像の矩形切り出し
62 manRect.center = (MAN_X, manY)                                   # 人画像の初期位置
63 manHitRect = pygame.Rect(MAN_X - manRect.width/2,manY,manRect.width,manRect.height/2) # 人の当たり判定（下半身）
64
65 while (True):                                                   # リアルタイム処理の無限ループ
66   screen.fill("gray50")                                         # 画面を灰色に塗り潰す
67     pygame.draw.rect(screen,"gray32",pygame.Rect(0,40,640,320))   # 車道のアスファルト
68     pygame.draw.rect(screen,"white",pygame.Rect(0,44,640,4))      # 上の歩道境界(白の実線)
```

```
    pygame.draw.rect(screen,"white",pygame.Rect(0,352,640,4))            # 下の歩道境界(白の実線)
    pygame.draw.rect(screen,"orangered2",pygame.Rect(0,198,640,4))       # 中央分離帯(オレンジ実線)
    for x in range(0, 7):                                                # 破線の描画
        pygame.draw.rect(screen,"white",pygame.Rect(100 * x,118,50,4))   #  車線境界(白の破線)
        pygame.draw.rect(screen,"white",pygame.Rect(100 * x,278,50,4))   #  車線境界(白の破線)

    for i in range(0,len(LANE_SPEED_LOW)):                              # 車線の分だけ車の処理をする
       carX[i] = carX[i] + carSpeed[i] / FPS                             # 速度に応じた座標移動
       carRect[i].center = (carX[i], LANE_Y[i])                          # 車の描画位置の変更
        screen.blit(carSpr[i], carRect[i])                               # 車の描画
       CarRestart(i)                                                     # 車の再出現の判定と再出現処理
        if pygame.Rect.colliderect(manHitRect, carRect[i]):              # 人との接触事故判定
            ManStartPos()                                                #  人を初期位置に
            failed = failed + 1                                          #  横断失敗数を加算
            seFalse.play()                                               #  横断失敗音声を再生

    manY = manY - manSpeed / FPS                                         # 速度と制御間隔に応じて人のY座標を変える
    manSpeed = max(0, manSpeed - manSpeed / FPS * 4)                     # 人の減速(最低が0)
    if manY <= MAN_END_Y:                                                # 上の歩道に到着したら
        ManStartPos()                                                    #  人を初期位置に
        success = success + 1                                            #  横断成功数を加算
        seSuccess.play()                                                 #  横断成功音声を再生
    manRect.center = (MAN_X, manY)                                       # 人の描画位置変更
    screen.blit(manSprBody[anmBody], manRect)                            # 人の体画像を描画
    screen.blit(manSprHead[anmHead], manRect)                            # 人の頭画像を描画
    manHitRect.center = (MAN_X, manY + manRect.height / 4)               # 人の当たり判定位置変更

    screen.blit(sysfont.render("SUCCESS="+str(success), False, (0,0,0)), (10,0))        # 横断成功数の描画
    screen.blit(sysfont.render("TIME:"+str(int(restTime)), False, (0,0,0)), (260,0))    # 残り時間の描画
    screen.blit(sysfont.render("FAILED="+str(failed), False, (0,0,0)), (460,0))         # 横断失敗数の描画

    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（残り時間やアニメーションのため）
    restTime = max(0, restTime - deltaTime) # 残り時間をフレーム経過時間分だけ減らす（最小は０）
```

```
103     if restTime == 0:                       # 残り時間が０（ゲームオーバー）か
104         ManStartPos()                       # 人をスタート位置に戻す
105         screen.blit(sysfont.render("Push Enter to Start", False, (0,255,0)), (180,200)) # ゲーム開始を促す文章表示
106     timeHead = timeHead + deltaTime         # 人の頭のアニメーションタイマーを加算
107     if timeHead >= LOOP_HEAD:               # アニメーションループの設定時間を超えたら
108         timeHead = timeHead - LOOP_HEAD     #  タイマーを戻す
109         anmHead = -anmHead + 1              #  アニメーションパターンの変更
110     pygame.display.update()                 # ゲーム画面の更新を行う
111
112     for event in pygame.event.get():                            # イベントを全て取得するループ
113         if event.type == pygame.KEYDOWN:                            # キーが押されてて
114             if event.key == pygame.K_SPACE and restTime > 0:        #  スペースバーで残り時間ある時
115                 manSpeed = MAN_SPEED_MAX                            #   人の移動開始
116                 seRun.play()                                        #   足音の再生
117                 anmBody = -anmBody + 1                              #   歩行アニメーション
118             elif event.key == pygame.K_RETURN and restTime == 0:    #  リターンキーでゲームオーバーの時
119                 restTime = REST_TIME                                #   残り時間をセットする
120             elif event.key == pygame.K_ESCAPE:                      #  ESCキーが押された時
121                 pygame.quit(); sys.exit()                           #   終了処理
122         elif event.type == pygame.QUIT:                             #  終了イベントを受けた時
123             pygame.quit(); sys.exit()                               #   終了処理
```