```
# -*- coding: utf-8 -*-
import pygame                           # ゲーム作成用の pygame ライブラリ
from pygame.locals import *             # マウスやキーボードの定義など
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
import math                             # 数学関係のライブラリ

(WIDTH, HEIGHT) = (640, 400)            # 画面の大きさ（横 640 ピクセル x 縦 400 ピクセル）
(CENTERX, CENTERY) = (WIDTH / 2, HEIGHT / 2)# 中心座標
SCREEN = pygame.Rect(0, 0, WIDTH, HEIGHT)   # 画面サイズの設定値
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）
BENEFICIAL = 2                          # 益虫の開始番号
BUGTYPE = 4                             # 虫の種類
SUICIDE = 10                            # 虫の生存可能時間
LIFETIME = 60                           # ゲームの制限時間
BUGLENMIN = 450                         # 出現距離最短
BUGLENMAX = 650                         # 出現距離最長
BUGSPEEDMIN = 100                       # 虫速度最低
BUGSPEEDMAX = 200                       # 虫速度最高
BUGRADCENTER = 1.0                      # 虫の初期方向の変動量
SPEEDMODUSPEED = 100                    # 速度変調用幅
RADMODURADIUS = 1                       # 角度変調用幅
SPEEDMODURAD = 12                       # 速度変調用角速度
RADMODURAD = 12                         # 角度変調用角速度
BUGDROPTIME = 1                         # 虫出現間隔
```

```
class Bug():                                                    # 虫のクラス
    def __init__(self):                                         # 作られた時の初期化関数
        self.kill = False                                       # 死亡用要求の申請
        self.tim = 0                                            # 経過時間
        self.type = random.randrange(BUGTYPE)                   # 虫の種類(0,1=害虫)
        self.len = random.uniform(BUGLENMIN, BUGLENMAX)         # 出現距離
        self.speed = random.uniform(BUGSPEEDMIN, BUGSPEEDMAX)   # 移動速度
        self.rad = random.uniform(0, math.pi * 2.0)             # 基本の進行方向
        self.dir = 0                                            # 画像の向き
        self.x = - math.cos(self.rad) * self.len + CENTERX      # 初期 X 座標
        self.y = - math.sin(self.rad) * self.len + CENTERY      # 初期 Y 座標
        self.rad += random.uniform(-BUGRADCENTER, BUGRADCENTER) # 進行方向を揺らがす
        self.speedModuRad = random.uniform(0, SPEEDMODURAD)     # 速度変調用角速度
        self.speedModuSpeed = random.uniform(0, SPEEDMODUSPEED) # 速度変調用速度幅
        self.radModuRad = random.uniform(0, RADMODURAD)         # 角度変調用角速度
        self.radModuRadius = random.uniform(0, RADMODURADIUS)   # 角度変調用角度幅
    def update(self, tim):                                      # ループの度の処理関数
        self.tim += tim                                         # 生存限界時間経過処理
        tm = self.tim                                           # 経過時間の一時変数
        if tm >= SUICIDE:                                       # 長時間放置されたら
            self.kill = True                                    # 自分で死亡（消去）する
        rd = self.rad + (self.radModuRadius * math.sin(tm * self.radModuRad))   # 現在の向きを算出
        sp = self.speed * ((100 - self.speedModuSpeed * math.sin(tm * self.speedModuRad)) / 100)   # 速度算出
        self.x += math.cos(rd) * sp * tim                       # 角度と速度から X 移動
```

```
        self.y += math.sin(rd) * sp * tim                    # 角度と速度から Y 移動
        self.dir = -90 - rd * 180 / math.pi                  # 画像の向きを算出

# 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("Mosquite")                       # ウィンドウのタイトルを設定(蚊の英名)
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 は標準フォントです)
deltaTime = elapsedTime = lifeTime = hit = miss = 0          # 数値変数の初期化
attack = playing = False                                     # 論理変数の初期化
(px, py) = (0, 0)                                            # 虫叩き(プレイヤー)の座標初期値
bug = []                                                     # 虫のクラス管理場所の初期化
seHit = pygame.mixer.Sound("hit.mp3")                        # 害虫駆除音の読み込み
seMiss = pygame.mixer.Sound("miss.mp3")                      # 失敗（益虫駆除）音の読み込み
swatterSpr = pygame.image.load("smasher.png").convert_alpha()# 虫叩き画像の読み込み
swatterRect = swatterSpr.get_rect()                          # 虫叩き画像の描画範囲矩形切り出し
bugSpr = [pygame.image.load("bug01.png").convert_alpha()]    # 虫 1(害虫)蚊画像の読み込み
bugSpr.append(pygame.image.load("bug02.png").convert_alpha())# 虫 2(害虫)ゴキブリ画像の読み込み
bugSpr.append(pygame.image.load("bug03.png").convert_alpha())# 虫 3(益虫)蜘蛛画像の読み込み
bugSpr.append(pygame.image.load("bug04.png").convert_alpha())# 虫 4(益虫)蜻蛉画像の読み込み
while (True):                                                # リアルタイム処理の無限ループ
    screen.fill("gray")                                      # 画面を灰色で塗り潰す
    if playing:                                              # ゲーム中なら
        elapsedTime += deltaTime                             # 虫発生管理時間の更新
```

```
    if (elapsedTime >= BUGDROPTIME):                                    # 一定時間おきに発生処理
        elapsedTime -= BUGDROPTIME                                      # 虫発生管理時間を巻き戻し
        bug.append(Bug())                                               # 虫の発生
    lifeTime -= deltaTime                                               # ゲーム残り時間を減らす
    if lifeTime <= 0:                                                   # 時間切れか？
        lifetime = 0                                                    # 時間がマイナスにならないように
        playing = False                                                 # ゲームプレイの終了
        bug = []                                                        # 虫を強制的に初期化
else:                                                                   # ゲームオーバーなら
    screen.blit(sysfont.render("Push R-Click to Start", False, (0,255,0)), (190,200)) # ゲーム開始を促す文章表示
screen.blit(sysfont.render("TIME:"+str(int(lifeTime)), False, "blue"), (20,0))   # 残り時間の描画
screen.blit(sysfont.render("HIT:"+str(int(hit)), False, "green"), (260,0))       # 虫駆除数の描画
screen.blit(sysfont.render("MISS:"+str(int(miss)), False, "red"), (460,0))       # 間違い回数の描画
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
for b in reversed(bug):                                                 # 全ての虫の処理ループ
    if b.kill:                                                          # 虫からの削除要請の確認
        bug.remove(b)                                                   # 虫の駆除(削除)処理
        bug.append(Bug())                                               # 虫の発生
    else:                                                               # 虫からの削除要請が無ければ通常処理
        b.update(deltaTime)                                             # 虫の通常動作の更新処理
        rotSpr = pygame.transform.rotate(bugSpr[b.type], b.dir) # 移動方向に回転した画像生成
        rotRect = rotSpr.get_rect()                                     # 画像矩形作成
        rotRect.center = (b.x, b.y)                                     # 画像矩形の位置を設定
        screen.blit(rotSpr, rotRect)                                    # 虫の描画
```

```
        if attack & pygame.Rect.colliderect(swatterRect, rotRect): # 虫叩きに接触したなら
            attack = False                                          # 攻撃の解除(纏めて駆除したいならコメントアウトする)
            if b.type < BENEFICIAL:                                  # 害虫なら
                hit+=1                                               # 害虫駆除数を更新
                seHit.play()                                        # 駆除成功音声の再生
            else:                                                   # 益虫なら
                miss+=1                                              # 間違い回数を更新
                seMiss.play()                                       # 駆除失敗音声の再生
            bug.remove(b)# del(b)                                                # 虫の駆除(削除)処理
            bug.append(Bug())                                        # 虫の発生
attack = False                                                      # 攻撃の解除
swatterRect.center = (px, py)                                      # 虫叩きの描画位置変更
screen.blit(swatterSpr, swatterRect)                               # 虫叩きの描画
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 == KEYDOWN and event.key == K_ESCAPE) or event.type == QUIT:      # 終了イベントなら
        pygame.quit(); sys.exit()                                  # 終了処理
    elif event.type == MOUSEMOTION:                                        # マウスの移動イベントなら
        px, py = event.pos                                         # マウスの位置を取得
    if event.type == MOUSEBUTTONDOWN:                                          # クリックイベントなら
        if playing and event.button == 1:                          # ゲーム中で左ボタンなら
            attack = True                                           # 攻撃する
        elif playing == False and event.button == 3:               # ゲームオーバーで右ボタンなら
            playing = True                                          # ゲーム開始する
```

```
lifeTime = LIFETIME                    # 制限時間をセット
elapsedTime = hit = miss = 0           # 虫発生間隔と駆除数の初期化
bug.append(Bug())                      # 最初の一匹を発生
```