Michito-Kawachi 敵の攻撃:未完成 7 minutes ago

123 lines (104 loc) · 3.35 KB

Code Blame Raw

```python
import math
import random
import time
import sys
import pygame as pg


WIDTH = 800
HEIGHT = 500


class Enemy(pg.sprite.Sprite):
    """
    敵の攻撃に関するクラス
    """
    def __init__(self, pos_x: int, pos_y: int):
        """
        弾を生成する
        引数1 pos_x: 弾が出るx座標
        引数2 pos_y: 弾が出るy座標
        """
        super().__init__()

        rad = 5
        self.image = pg.Surface((2*rad, 2*rad))
        pg.draw.circle(self.image, (255, 255, 255), (rad, rad), rad)
        self.image.set_colorkey((0, 0, 0))
        self.rect = self.image.get_rect()
        self.speed = 1
        #self.vx, self.vy = __class__.calc_orientation(self.rect, pos_x-10, pos_y-10)
        self.vx = 2
        self.vy = 3
        self.rect.centerx = pos_x
        self.rect.centery = pos_y

    def check_out(obj: pg.Rect):
        """
```

```
38              弾が画面外に出たかを判定する関数
39              引数 obj: 弾のRect
40              横方向 縦方向のはみ出し判定結果
41              （画面内: True/画面外: False）
42              """
43              yoko, tate = True, True
44              if obj.right < 0 or obj.left > WIDTH:
45                  yoko = False
46              if obj.bottom < 0 or obj.bottom > 400:
47                  tate = False
48              return yoko, tate
49
50          def calc_orientation(org: pg.Rect, pos_x, pos_y) -> tuple[float, float]:
51              """
52              orgから見て，pos_x，pos_yがどこにあるかを計算し，方向ベクトルをタプルで返す
53              引数1 org: 爆弾SurfaceのRect
54              引数2 pos_x: 目標のx座標
55              引数3 pos_y: 目標のy座標
56              戻り値: orgから見た目標の方向ベクトルを表すタプル
57              """
58              x_diff, y_diff = pos_x-org.centerx, pos_y-org.centery
59              norm = math.sqrt(x_diff**2+y_diff**2)
60              return -x_diff/norm, -y_diff/norm
61
62          def flower(self, pos_x, pos_y):
63              """
64              拡散弾を作る関数
65              引数 pos_x: x座標，pos_y: y座標
66              """
67              self.lst = []
68              origin_x = random.randint(50, 750)
69              #origin_x =
70
71          def beam(self, pos_x, pos_y):
72              """
73              直線の3連弾を作る関数
74              引数1 pos_x: 原点x座標
75              引数2 pos_y: 原点y座標
76              """
77              pass
78
79          def update(self):
80              """
81              爆弾を速度ベクトルself.vx，self.vyに基づき移動させる
82              """
83              self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
84              if __class__.check_out(self.rect) != (True, True):
85                  self.kill()
86
87
88      def main():
89          pg.display.set_caption("Under tale")
90          screen = pg.display.set_mode((WIDTH, HEIGHT))
91          sikaku1 = pg.Surface((400, 200))
92          bg = pg.Surface((800, 400))
```

```python
        pg.draw.rect(sikaku1, (255, 255, 255), (0, 0, 400, 200))
        pg.draw.rect(sikaku1, (0, 0, 0), (5, 5, 390, 190))
        pg.draw.rect(bg, (0, 0, 0), (0, 0, WIDTH, 400))


        enemies= pg.sprite.Group()
        tmr = 0
        clock = pg.time.Clock()

        while True:
            screen.blit(bg, (0, 0))
            screen.blit(sikaku1, (200, 200))
            for event in pg.event.get():
                if event.type == pg.QUIT:
                    return 0
            tmr += 1
            if 3<tmr%60<15:
                enemies.add(Enemy(20, 20))

            enemies.update()
            enemies.draw(screen)

            pg.display.update()
            clock.tick(50)


    if __name__ == "__main__":
        pg.init()
        main()
        pg.quit()
        sys.exit()
```