

Michito-Kawachi /  
ProjExD\_05

&lt;&gt; Code

Pull requests

Actions

Projects

Wiki

Security

Insights



enemy\_attack



ProjExD\_05 / README.md



Michito-Kawachi 敵の攻撃:未完成

3 minutes ago



24 lines (22 loc) · 748 Bytes

Preview

Code

Blame

Raw



# ゲームのタイトル

## 実行環境の必要条件

- python >= 3.10
- pygame >= 2.1

## ゲームの概要

主人公キャラクターを十字キーなどで行動決定し、戦闘を行う

## ゲームの実装

### 共通基本機能

- プレイヤーの枠

### 担当追加機能

- 敵の攻撃（自分の実装箇所）
- プレイヤーの動き
- プレイヤーの選択と行動(戦う、道具)
- プレイヤーの選択と行動(話す、見逃す)
- ライフ管理(UIの表示)
- 

### ToDo

- □ 花火のように放射状に拡散するような弾幕を追加する
- □ ランダムな場所から弾を発射する

## メモ [🔗](#)

- 背景画像として黒のSurfaceを追加した
- tmrとclock変数をmainに追加した

Michito-Kawachi /  
ProjExD\_05

&lt;&gt; Code

Pull requests

Actions

Projects

Wiki

Security

Insights

...



enemy\_attack ▾

...

ProjExD\_05 / under\_tale.py



Michito-Kawachi 敵の攻撃:未完成

7 minutes ago



123 lines (104 loc) · 3.35 KB

Code

Blame

Raw



```
1  import math
2  import random
3  import time
4  import sys
5  import pygame as pg
6
7
8  WIDTH = 800
9  HEIGHT = 500
10
11
12  class Enemy(pg.sprite.Sprite):
13      """
14      敵の攻撃に関するクラス
15      """
16      def __init__(self, pos_x: int, pos_y: int):
17          """
18          弾を生成する
19          引数1 pos_x: 弾が出るx座標
20          引数2 pos_y: 弾が出るy座標
21          """
22          super().__init__()
23
24          rad = 5
25          self.image = pg.Surface((2*rad, 2*rad))
26          pg.draw.circle(self.image, (255, 255, 255), (rad, rad), rad)
27          self.image.set_colorkey((0, 0, 0))
28          self.rect = self.image.get_rect()
29          self.speed = 1
30          #self.vx, self.vy = __class__.calc_orientation(self.rect, pos_x-10, pos_y-10)
31          self.vx = 2
32          self.vy = 3
33          self.rect.centerx = pos_x
34          self.rect.centery = pos_y
35
36      def check_out(obj: pg.Rect):
37          """
```

```

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))
93     #bg = pg.Surface((800, 400))

```

```
93 pg.draw.rect(sikaku1, (255, 255, 255), (0, 0, 400, 200))
94 pg.draw.rect(sikaku1, (0, 0, 0), (5, 5, 390, 190))
95 pg.draw.rect(bg, (0, 0, 0), (0, 0, WIDTH, 400))
96
97
98 enemies= pg.sprite.Group()
99 tmr = 0
100 clock = pg.time.Clock()
101
102 while True:
103     screen.blit(bg, (0, 0))
104     screen.blit(sikaku1, (200, 200))
105     for event in pg.event.get():
106         if event.type == pg.QUIT:
107             return 0
108     tmr += 1
109     if 3<tmr%60<15:
110         enemies.add(Enemy(20, 20))
111
112     enemies.update()
113     enemies.draw(screen)
114
115     pg.display.update()
116     clock.tick(50)
117
118
119 if __name__ == "__main__":
120     pg.init()
121     main()
122     pg.quit()
123     sys.exit()
```