








 c0b22117 / ProjExD\_05\_1



<> Code  Pull requests  Actions  Projects  Wiki  Security  Insights ...

  c0b22117/壁 ▾ ProjExD\_05\_1 / tank\_surviver.py 

 Go to file  t ...



c0b22117 追加機能「壁」

5 minutes ago



187 lines (144 loc) · 5.79 KB

```
1  import math
2  import random
3  import sys
4  import time
5
6  import pygame as pg
7
8
9  WIDTH = 1600 # ゲームウィンドウの幅
10 HEIGHT = 900 # ゲームウィンドウの高さ
11
12 class Wall(pg.sprite.Sprite):
13     def __init__(self, x, y, width, height, color):
14         super().__init__()
15         self.image = pg.Surface((width, height))
16         self.image.fill(color)
17         self.rect = self.image.get_rect()
18         self.rect.topleft = (x, y)
19
20
21
22 def check_bound(obj: pg.Rect) -> tuple[bool, bool]:
23     """
24     オブジェクトが画面内か画面外かを判定し、真理値タプルを返す
25     引数 obj: オブジェクト (爆弾, こうかとん, ビーム) SurfaceのRect
26     戻り値: 横方向, 縦方向のはみ出し判定結果 (画面内: True / 画面外: False)
27     """
28     yoko, tate = True, True
29     if obj.left < 0 or WIDTH < obj.right: # 横方向のはみ出し判定
30         yoko = False
31     if obj.top < 0 or HEIGHT < obj.bottom: # 縦方向のはみ出し判定
32         tate = False
33     return yoko, tate
34
35
36 def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
37     """
38     orgから見て, dstがどこにあるかを計算し, 方向ベクトルをタプルで返す
39     引数1 org: 爆弾SurfaceのRect
40     引数2 dst: こうかとんSurfaceのRect
41     戻り値: orgから見たdstの方向ベクトルを表すタプル
42     """
43     x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
44     norm = math.sqrt(x_diff**2+y_diff**2)
```

```

45         return x_diff/norm, y_diff/norm
46
47
48     class Bird(pg.sprite.Sprite):
49         """
50         ゲームキャラクター（こうかとん）に関するクラス
51         """
52     delta = { # 押下キーと移動量の辞書
53         pg.K_UP: (0, -1),
54         pg.K_DOWN: (0, +1),
55         pg.K_LEFT: (-1, 0),
56         pg.K_RIGHT: (+1, 0),
57     }
58
59     def __init__(self, num: int, xy: tuple[int, int]):
60         """
61         こうかとん画像Surfaceを生成する
62         引数1 num: こうかとん画像ファイル名の番号
63         引数2 xy: こうかとん画像の位置座標タプル
64         """
65         super().__init__()
66
67         img0 = pg.transform.rotozoom(pg.image.load(f"fig/my_tank.png"), 0, 2.0)
68         img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
69         self.imgs = {
70             (+1, 0): img, # 右
71             (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
72             (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
73             (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
74             (-1, 0): img0, # 左
75             (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
76             (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
77             (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
78         }
79
80         self.dire = (+1, 0)
81         self.image = self.imgs[self.dire]
82         self.rect = self.image.get_rect()
83         self.rect.center = xy
84         self.speed = 10
85
86
87         self.state = "normal" # 初期状態は通常状態
88         self.hyper_life = -1
89
90
91     def change_img(self, num: int, screen: pg.Surface):

```



c0b22117/壁

ProjExD\_05\_1 / tank\_surviver.py

↑ Top

Code

Blame

Raw



```

48     class Bird(pg.sprite.Sprite):
91         def change_img(self, num: int, screen: pg.Surface):
98             self.image = pg.transform.rotozoom(pg.image.load(f"ex05/fig/{num}.png"), 0, 2.0)
99             screen.blit(self.image, self.rect)
100
101
102
103     def change_state(self, state, hyper_life):

```

```
104     """
105     こうかとの状態を切り替えるメソッド
106     引数1 state: 状態を表す
107     引数2 hyper_life: 発動時間
108     """
109     self.state = state
110     self.hyper_life = hyper_life
111
112
113
114     def update(self, key_lst: list[bool], screen: pg.Surface):
115         """
116         押下キーに応じてこうかとを移動させる
117         引数1 key_lst: 押下キーの真理値リスト
118         引数2 screen: 画面Surface
119         """
120
121
122         if key_lst[pg.K_LSHIFT]:
123             self.speed = 2
124         else:
125             self.speed = 1
126
127         sum_mv = [0, 0]
128         for k, mv in __class__.delta.items():
129             if key_lst[k]:
130                 self.rect.move_ip(+self.speed*mv[0], +self.speed*mv[1])
131                 sum_mv[0] += mv[0]
132                 sum_mv[1] += mv[1]
133             if check_bound(self.rect) != (True, True):
134                 for k, mv in __class__.delta.items():
135                     if key_lst[k]:
136                         self.rect.move_ip(-self.speed*mv[0], -self.speed*mv[1])
137             if not (sum_mv[0] == 0 and sum_mv[1] == 0):
138                 self.dire = tuple(sum_mv)
139                 self.image = self.imgs[self.dire]
140
141
142         if self.state == "hyper":
143             self.hyper_life -= 1
144             self.image = pg.transform.laplacian(self.image)
145             if self.hyper_life < 0:
146                 self.change_state("normal", -1)
147
148
149         screen.blit(self.image, self.rect)
150
151     def get_direction(self) -> tuple[int, int]:
152         return self.dire
153
154
155     def main():
156         pg.display.set_caption("タンクサバイバー")
157         screen = pg.display.set_mode((WIDTH, HEIGHT))
158         bg_img = pg.image.load("fig/pg_bg.jpg")
159         pg.display.set_caption("kabe")
160
161         tate_bar1 = Wall(200, 300, 30, 300, (0, 0, 255))
162         yoko_bar1 = Wall(300, 200, 1000, 30, (0, 0, 255))
163         tate_bar2 = Wall(1400, 300, 30, 300, (0, 0, 255))
```

```
164     yoko_bar2 = Wall(300, 700, 1000, 30, (0, 0, 255))
165
166     all_sprites = pg.sprite.Group(tate_bar1, yoko_bar1, tate_bar2, yoko_bar2)
167
168
169     bird = Bird(3, (900, 400))
170     while True:
171         key_lst = pg.key.get_pressed()
172         for event in pg.event.get():
173             if event.type == pg.QUIT:
174                 return 0
175
176         screen.blit(bg_img, [0, 0])
177         all_sprites.draw(screen)
178
179         bird.update(key_lst, screen)
180         pg.display.update()
181
182
183 if __name__ == "__main__":
184     pg.init()
185     main()
186     pg.quit()
187     sys.exit()
```