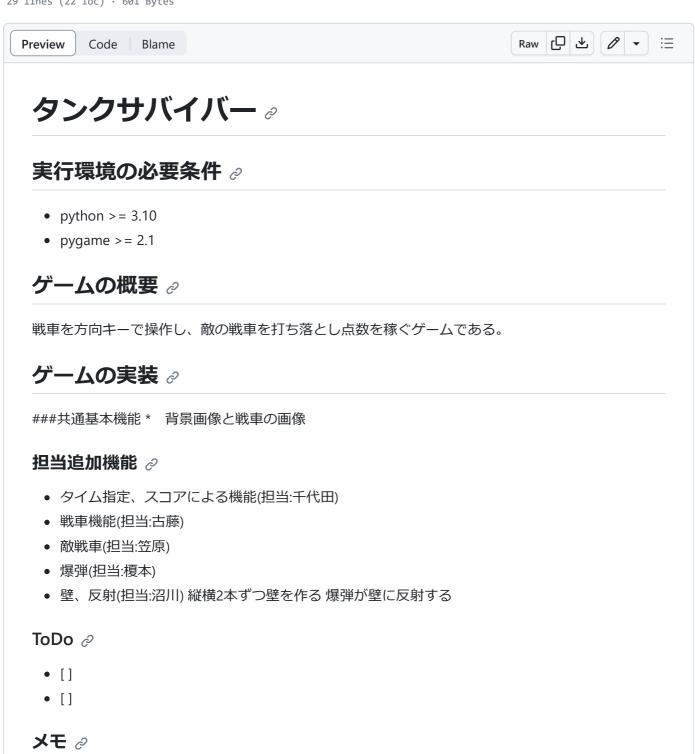
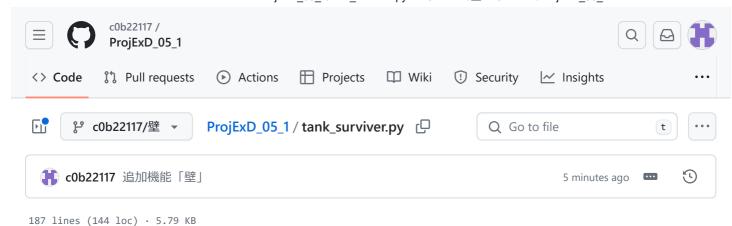


29 lines (22 loc) · 601 Bytes





```
1
      import math
2
      import random
3
      import sys
4
      import time
5
6
      import pygame as pg
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
      class Wall(pg.sprite.Sprite):
12
13
          def __init__(self, x, y, width, height, color):
14
             super().__init__()
             self.image = pg.Surface((width, height))
15
             self.image.fill(color)
16
17
             self.rect = self.image.get_rect()
             self.rect.topleft = (x, y)
18
19
20
21
     def check_bound(obj: pg.Rect) -> tuple[bool, bool]:
22
23
          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
24
          引数 obj:オブジェクト(爆弾, こうかとん, ビーム) SurfaceのRect
25
          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)
26
27
28
          yoko, tate = True, True
          if obj.left < 0 or WIDTH < obj.right: # 横方向のはみ出し判定
29
30
             yoko = False
          if obj.top < 0 or HEIGHT < obj.bottom: # 縦方向のはみ出し判定
31
32
             tate = False
33
          return yoko, tate
34
35
      def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
36
37
          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
38
39
          引数1 org: 爆弾SurfaceのRect
          引数2 dst:こうかとんSurfaceのRect
40
          戻り値:orgから見たdstの方向ベクトルを表すタプル
41
42
43
          x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
          norm = math sart(x diff**2+v diff**2)
```

```
_ · y _ u _ · ·
   45
              return x_diff/norm, y_diff/norm
   46
   47
   48
          class Bird(pg.sprite.Sprite):
   49
              ゲームキャラクター (こうかとん) に関するクラス
   50
              ....
   51
   52
              delta = { # 押下キーと移動量の辞書
   53
                  pg.K_UP: (0, -1),
                  pg.K_DOWN: (0, +1),
   54
   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
                   こうかとん画像Surfaceを生成する
   61
                  引数1 num:こうかとん画像ファイル名の番号
   62
   63
                  引数2 xy:こうかとん画像の位置座標タプル
   64
                  super().__init__()
   66
                  img0 = pg.transform.rotozoom(pg.image.load(f"fig/my_tank.png"), 0, 2.0)
   67
                  img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
   68
   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
                      (0, +1): pg.transform.rotozoom(img, -90, 1.0), #下
   76
   77
                      (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
   78
                  }
   79
                  self.dire = (+1, 0)
   80
                  self.image = self.imgs[self.dire]
   81
                  self.rect = self.image.get_rect()
   82
                  self.rect.center = xy
   83
                  self.speed = 10
   85
   86
                  self.state = "normal" # 初期状態は通常状態
   87
   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
    48
           class Bird(pg.sprite.Sprite):
    91
               def change_img(self, num: int, screen: pg.Surface):
                  Selt.image = pg.transtorm.rotozoom(pg.image.ioau(t exos/tig/{num}.png ), v, z.v)
   99
                  screen.blit(self.image, self.rect)
  100
  101
  102
              def change_state(self,state,hyper_life):
```

```
104
                こうかとんの状態を切り替えるメゾット
105
                引数1 state: 状態を表す
106
107
                引数2 hyper_life: 発動時間
108
109
                self.state = state
                self.hyper_life = hyper_life
110
111
112
113
114 🗸
            def update(self, key_lst: list[bool], screen: pg.Surface):
115
                押下キーに応じてこうかとんを移動させる
116
117
                引数1 key_lst:押下キーの真理値リスト
                引数2 screen:画面Surface
118
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]
                for k, mv in __class__.delta.items():
128
129
                    if key lst[k]:
130
                        self.rect.move_ip(+self.speed*mv[0], +self.speed*mv[1])
                        sum_mv[0] += mv[0]
131
132
                        sum_mv[1] += mv[1]
133
                if check_bound(self.rect) != (True, True):
                    for k, mv in __class__.delta.items():
134
                        if key_lst[k]:
135
136
                            self.rect.move_ip(-self.speed*mv[0], -self.speed*mv[1])
                if not (sum_mv[0] == 0 \text{ and } sum_mv[1] == 0):
137
                    self.dire = tuple(sum mv)
138
                    self.image = self.imgs[self.dire]
139
140
141
                if self.state == "hyper":
142
                    self.hyper_life -= 1
143
144
                    self.image = pg.transform.laplacian(self.image)
145
                    if self.hyper life < 0:</pre>
146
                        self.change_state("normal", -1)
147
148
149
                screen.blit(self.image, self.rect)
150
151
            def get_direction(self) -> tuple[int, int]:
                return self.dire
152
153
154
155 ∨ def main():
            pg.display.set_caption("タンクサバイバー")
156
            screen = pg.display.set_mode((WIDTH, HEIGHT))
157
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))
            tate_bar2 = Wall(1400, 300, 30, 300, (0, 0, 255))
```

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