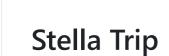


30 lines (23 loc) · 1.04 KB

Preview



実行環境の必要条件

Code Blame

- python >= 3.10
- pygame >= 2.1

ゲームの概要

ジャンル

• スコア系のrunゲーム

内容

- GameOver要素で敵キャラにあたることと、落とし穴に落ちること
- 走った距離でスコアが大きくなる

•

ゲームの実装

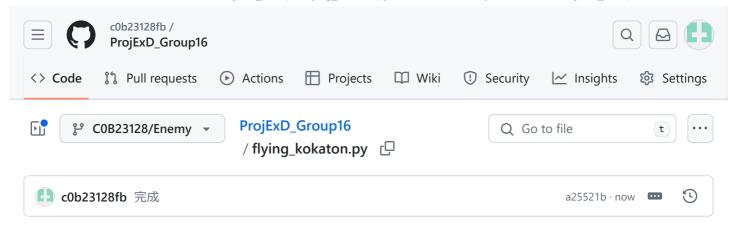
共通基本機能

- 背景画像と主人公キャラクターの描画
- 敵キャラクターの描画
- ビーム実装
- fight_kokaton.pyを基盤として作成

担当追加機能

• ワープエフェクト(担当:渡邊奏):主人公が移動する際のエフェクトを追加する

- 移動距離(score)(担当:彦坂 飛和):時間で加算されるようにする
- アイテム機能(ビーム)(担当:袖山睦生):敵を破壊する、ビームを実装
- 障害物機能(担当:丸山 航平):落とし穴、破壊できる敵、破壊できない敵
- リザルト、GameOver画面(担当:伊藤一真):移動距離、GameOver画面の実装



164 lines (136 loc) · 4.91 KB

```
Raw ☐ ± 0 -
Code
        Blame
   1
          import os
    2
          import random
    3
          import sys
   4
          import time
   5
         import pygame as pg
   6
         WIDTH = 1000 # ゲームウィンドウの幅
   8
         HEIGHT = 600 # ゲームウィンドウの高さ
   9
         NUM_OF_BOMBS = 0
   10
         os.chdir(os.path.dirname(os.path.abspath(__file__)))
   11
   12
   13
         def check_bound(obj_rct: pg.Rect) -> tuple[bool]:
   14
   15
             オブジェクトが画面内or画面外を判定し、真理値タプルを返す関数
   16
   17
             引数:こうかとんRect, または、爆弾Rect
   18
             戻り値:縦方向のはみ出し判定結果(画面上:1/画面下:2)
   19
             tate = 0
   20
             if obj_rct.bottom < 0 :</pre>
   21
   22
                 tate = 1
   23
             if HEIGHT < obj_rct.top:</pre>
   24
                 tate = 2
   25
             return tate
   26
   27
   28
         class Bird:
   29
             ゲームキャラクター (こうかとん) に関するクラス
   30
   31
   32
   33
             def __init__(self, xy: tuple[int, int]):
   34
                 こうかとん画像Surfaceを生成する
   35
                 引数 xy:こうかとん画像の初期位置座標タプル
   36
   37
                 self.img = pg.transform.flip(pg.image.load("fig/3.png"), True, False)
   38
   39
                 self.rct: pg.Rect = self.img.get_rect()
                 self.rct.center = xy
   40
                 self.d = 0
```

```
42
              self.tm = 0
43
              self.bg_img = pg.image.load("fig/pg_space.jpg")
44
              self.bg_img2 = pg.transform.flip(self.bg_img, True, False)
45
46
47
48
          def change_img(self, num: int, screen: pg.Surface):
49
              こうかとん画像を切り替え、画面に転送する
50
              引数1 num:こうかとん画像ファイル名の番号
51
              引数2 screen:画面Surface
52
53
              screen.blit(self.img, self.rct)
54
55
           def update(self, screen: pg.Surface):
56
57
              押下キーに応じてこうかとんを移動させる
58
              引数2 screen: 画面Surface
60
              tate = check bound(self.rct)
61
              if tate == 1:
62
                  d = 600
63
64
                  self.rct.move ip((0,d))
              if tate == 2:
65
                  d = -600
66
                  self.rct.move_ip((0,d))
67
68
69
70

✓ class Enemy:

71
           敵機に関するクラス
72
           0.00
73
74
          def __init__(self):
              self.img = pg.image.load("fig/pg fall.png")
75
76
              self.rct: pg.Rect = self.img.get_rect()
              self.rct.centerx = WIDTH
77
78
              self.rct.centery = random.choice([0, 60, 200, 260, 320, 380, 440, 500])
79
              self.vx, self.vy = -20, 0
80
81
          def update(self, screen: pg.Surface):
82
              敵機を速度ベクトルself.vyに基づき移動(降下)させる
83
              ランダムに決めた停止位置 boundまで降下したら、 stateを停止状態に変更する
84
              引数 screen: 画面Surface
85
86
              self.rct.move_ip(self.vx, self.vy)
87
              screen.blit(self.img, self.rct)
88
89
90
      class Beam:
91
92
          def __init__(self, bird: Bird):
              self.img = pg.transform.rotozoom(pg.image.load("fig/beam.png"), 0, 2.0)
93
              self.rct: pg.Rect = self.img.get_rect() #Rect
94
              self.rct.left = bird.rct.right
95
              self.rct.centery = bird.rct.centery
96
97
              self.vx, self.vy = +5, 0
98
           def update(self, screen: pg.Surface):
```

```
100
                爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
101
102
                引数 screen:画面Surface
103
104
                if check_bound(self.rct) == (True, True):
                     self.rct.move_ip(self.vx, self.vy)
105
106
                    screen.blit(self.img, self.rct)
107
108
109 ∨ def main():
            pg.display.set_caption("たたかえ!こうかとん")
110
            screen = pg.display.set_mode((WIDTH, HEIGHT))
111
            bg img = pg.image.load("fig/pg space.jpg")
112
113
            bird = Bird((100, 300))
            beam = None
114
            clock = pg.time.Clock()
115
            tmr = 0
116
117
            emys = []
118
            key_lst = pg.key.get_pressed()
119
120
            while True:
121
122
                for event in pg.event.get():
123
                     if event.type == pg.QUIT:
124
                         return
                    if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
125
                         beam = Beam(bird)
126
127
                    if event.type == pg.KEYDOWN and event.key == pg.K_UP:
                         d = -100
128
129
                         bird.rct.move_ip((0,d))
                    if event.type == pg.KEYDOWN and event.key == pg.K_DOWN:
130
                         d = 100
131
132
                         bird.rct.move_ip((0,d))
133
                screen.blit(bg_img, [0, 0])
134
135
                x = bird.tm \% 2400
                screen.blit(bird.bg_img, [-x, 0])
136
                screen.blit(bird.bg_img2,[-x+1200,0])
137
                screen.blit(bird.bg_img, [-x+2400, 0])
138
                screen.blit(bird.img, bird.rct)
139
140
                if tmr % 20 == 0:
141
                     emys.append(Enemy())
142
                for emy in emys:
143
                     emy.update(screen)
144
                pg.display.update()
145
                bird.tm += 1
146
147
148
149
                     # if emys.rct.colliderect(bird.rct):
150
151
                #key_lst = pg.key.get_pressed()
                bird.update(screen)
152
                if beam is not None:
153
154
                    beam.update(screen)
155
                pg.display.update()
156
                tmr += 1
157
                clock.tick(50)
```