




c0a23126f2 /
ProjExD_Group16



[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

ProjExD_Group16 / README.md 

...



c0a23126f2 read更新2

293bae4 · yesterday



30 lines (23 loc) · 1.04 KB

Preview Code Blame

Raw     

Stella Trip

実行環境の必要条件

- 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

 c0b23128fb / ProjExD_Group16



[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

 [C0B23128/Enemy](#) [ProjExD_Group16](#) / flying_kokaton.py 

t ...



c0b23128fb 完成

a25521b · now



164 lines (136 loc) · 4.91 KB

[Code](#) [Blame](#)

[Raw](#)     

```
1  import os
2  import random
3  import sys
4  import time
5  import pygame as pg
6
7
8  WIDTH = 1000 # ゲームウィンドウの幅
9  HEIGHT = 600 # ゲームウィンドウの高さ
10 NUM_OF_BOMBS = 0
11 os.chdir(os.path.dirname(os.path.abspath(__file__)))
12
13
14 def check_bound(obj_rct: pg.Rect) -> tuple[bool]:
15     """
16     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
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     """
20     tate = 0
21     if obj_rct.bottom < 0 :
22         tate = 1
23     if HEIGHT < obj_rct.top:
24         tate = 2
25     return tate
26
27
28 class Bird:
29     """
30     ゲームキャラクター（こうかとん）に関するクラス
31     """
32
33     def __init__(self, xy: tuple[int, int]):
34         """
35         こうかとん画像Surfaceを生成する
36         引数 xy：こうかとん画像の初期位置座標タプル
37         """
38         self.img = pg.transform.flip(pg.image.load("fig/3.png"), True, False)
39         self.rct: pg.Rect = self.img.get_rect()
40         self.rct.center = xy
41         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         """
50         こうかとん画像を切り替え、画面に転送する
51         引数1 num : こうかとん画像ファイル名の番号
52         引数2 screen : 画面Surface
53         """
54         screen.blit(self.img, self.rct)
55
56     def update(self, screen: pg.Surface):
57         """
58         押下キーに応じてこうかとんを移動させる
59         引数2 screen : 画面Surface
60         """
61         tate = check_bound(self.rct)
62         if tate == 1:
63             d = 600
64             self.rct.move_ip((0,d))
65         if tate == 2:
66             d = -600
67             self.rct.move_ip((0,d))
68
69
70     class Enemy:
71         """
72         敵機に関するクラス
73         """
74     def __init__(self):
75         self.img = pg.image.load("fig/pg_fall.png")
76         self.rct: pg.Rect = self.img.get_rect()
77         self.rct.centerx = WIDTH
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         """
83         敵機を速度ベクトルself.vyに基づき移動（降下）させる
84         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
85         引数 screen : 画面Surface
86         """
87         self.rct.move_ip(self.vx, self.vy)
88         screen.blit(self.img, self.rct)
89
90
91     class Beam:
92     def __init__(self, bird: Bird):
93         self.img = pg.transform.rotozoom(pg.image.load("fig/beam.png"), 0, 2.0)
94         self.rct: pg.Rect = self.img.get_rect() #Rect
95         self.rct.left = bird.rct.right
96         self.rct.centery = bird.rct.centery
97         self.vx, self.vy = +5, 0
98
99     def update(self, screen: pg.Surface):
```

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

```
158
159
160     if __name__ == "__main__":
161         pg.init()
162         main()
163         pg.quit()
164         sys.exit()
```