


 c0c24020ae / ProjExD_3



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

ProjExD_3 / fight_kokaton.py



c0c24020ae multibeamブランチ

8055167 · 37 minutes ago



197 lines (174 loc) · 7 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 = 1100 # ゲームウィンドウの幅
9  HEIGHT = 650 # ゲームウィンドウの高さ
10 NUM_OF_BOMBS = 5 # 爆弾の個数
11 os.chdir(os.path.dirname(os.path.abspath(__file__)))
12
13
14 def check_bound(obj_rct: pg.Rect) -> tuple[bool, 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：こうかとんや爆弾、ビームなどのRect
18     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）
19     """
20     yoko, tate = True, True
21     if obj_rct.left < 0 or WIDTH < obj_rct.right:
22         yoko = False
23     if obj_rct.top < 0 or HEIGHT < obj_rct.bottom:
24         tate = False
25     return yoko, 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     delta = { # 押下キーと移動量の辞書
33         pg.K_UP: (0, -5),
34         pg.K_DOWN: (0, +5),
35         pg.K_LEFT: (-5, 0),
36         pg.K_RIGHT: (+5, 0),
37     }
38     img0 = pg.transform.rotozoom(pg.image.load("fig/3.png"), 0, 0.9)
39     img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん（右向き）
40     imgs = { # 0度から反時計回りに定義
41         (+5, 0): img, # 右
42         (+5, -5): pg.transform.rotozoom(img, 45, 0.9), # 右上
```

```

43         (0, -5): pg.transform.rotozoom(img, 90, 0.9), # 上
44         (-5, -5): pg.transform.rotozoom(img0, -45, 0.9), # 左上
45         (-5, 0): img0, # 左
46         (-5, +5): pg.transform.rotozoom(img0, 45, 0.9), # 左下
47         (0, +5): pg.transform.rotozoom(img, -90, 0.9), # 下
48         (+5, +5): pg.transform.rotozoom(img, -45, 0.9), # 右下
49     }
50
51     def __init__(self, xy: tuple[int, int]):
52         """
53         こうかとん画像Surfaceを生成する
54         引数 xy: こうかとん画像の初期位置座標タプル
55         """
56         self.img = __class__.imgs[(+5, 0)]
57         self.rct: pg.Rect = self.img.get_rect()
58         self.rct.center = xy
59
60     def change_img(self, num: int, screen: pg.Surface):
61         """
62         こうかとん画像を切り替え、画面に転送する
63         引数1 num: こうかとん画像ファイル名の番号
64         引数2 screen: 画面Surface
65         """
66         self.img = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 0.9)
67         screen.blit(self.img, self.rct)
68
69     def update(self, key_lst: list[bool], screen: pg.Surface):
70         """
71         押下キーに応じてこうかとんを移動させる
72         引数1 key_lst: 押下キーの真理値リスト
73         引数2 screen: 画面Surface
74         """
75         sum_mv = [0, 0]
76         for k, mv in __class__.delta.items():
77             if key_lst[k]:
78                 sum_mv[0] += mv[0]
79                 sum_mv[1] += mv[1]
80         self.rct.move_ip(sum_mv)
81         if check_bound(self.rct) != (True, True):
82             self.rct.move_ip(-sum_mv[0], -sum_mv[1])
83         if not (sum_mv[0] == 0 and sum_mv[1] == 0):
84             self.img = __class__.imgs[tuple(sum_mv)]
85         screen.blit(self.img, self.rct)
86
87
88     class Beam:
89         """
90         こうかとんが放つビームに関するクラス
91         """
92     def __init__(self, bird: "Bird"):
93         """
94         ビーム画像Surfaceを生成する
95         引数 bird: ビームを放つこうかとん (Birdインスタンス)
96         """
97         self.img = pg.image.load(f"fig/beam.png") # こうかとんSurface
98         self.rct = self.img.get_rect() # こうかとんRect
99         self.rct.centery = bird.rct.centery # こうかとんの中心縦座標
100        self.rct.left = bird.rct.right # ビームの左座標 = こうかとんの右座標

```

```
101         self.vx, self.vy = +5, 0
102
103     def update(self, screen: pg.Surface):
104         """
105         ビームを速度ベクトルself.vx, self.vyに基づき移動させる
106         引数 screen: 画面Surface
107         """
108         if check_bound(self.rct) == (True, True):
109             self.rct.move_ip(self.vx, self.vy)
110             screen.blit(self.img, self.rct)
111
112
113     class Bomb:
114         """
115         爆弾に関するクラス
116         """
117     def __init__(self, color: tuple[int, int, int], rad: int):
118         """
119         引数に基づき爆弾円Surfaceを生成する
120         引数1 color: 爆弾円の色タプル
121         引数2 rad: 爆弾円の半径
122         """
123         self.img = pg.Surface((2*rad, 2*rad))
124         pg.draw.circle(self.img, color, (rad, rad), rad)
125         self.img.set_colorkey((0, 0, 0))
126         self.rct = self.img.get_rect()
127         self.rct.center = random.randint(0, WIDTH), random.randint(0, HEIGHT)
128         self.vx, self.vy = +5, +5
129
130     def update(self, screen: pg.Surface):
131         """
132         爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
133         引数 screen: 画面Surface
134         """
135         yoko, tate = check_bound(self.rct)
136         if not yoko:
137             self.vx *= -1
138         if not tate:
139             self.vy *= -1
140         self.rct.move_ip(self.vx, self.vy)
141         screen.blit(self.img, self.rct)
142
143
144     def main():
145         pg.display.set_caption("たたかえ！こうかとん")
146         screen = pg.display.set_mode((WIDTH, HEIGHT))
147         bg_img = pg.image.load("fig/pg_bg.jpg")
148         bird = Bird((300, 200))
149         beam = None
150         # bomb = Bomb((255, 0, 0), 10)
151         bombs = [Bomb((255, 0, 0), 10) for _ in range(NUM_OF_BOMBS)]
152         clock = pg.time.Clock()
153         tmr = 0
154         while True:
155             for event in pg.event.get():
156                 if event.type == pg.QUIT:
157                     return
158                 if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
```

```
159         # スペースキー押下でBeamクラスのインスタンス生成
160         beam = Beam(bird)
161     screen.blit(bg_img, [0, 0])
162
163
164     # if bomb is not None:
165     for bomb in bombs:
166         if bird.rct.colliderect(bomb.rct):
167             # ゲームオーバー時に、こうかとん画像を切り替え、1秒間表示させる
168             bird.change_img(8, screen)
169             pg.display.update()
170             time.sleep(1)
171             return
172
173     #if beam is not None:
174     for j, bomb in enumerate(bombs):
175         if beam is not None:
176             if beam.rct.colliderect(bomb.rct): # ビームと爆弾の衝突判定
177                 beam = None # ビームを消す
178                 bombs[j] = None # 爆弾を消す
179                 bird.change_img(6, screen) # よろこびエフェクト
180             bombs = [bomb for bomb in bombs if bomb is not None] # 撃ち落とされてない爆弾だけのリストに
181
182     key_lst = pg.key.get_pressed()
183     bird.update(key_lst, screen)
184     if beam is not None:
185         beam.update(screen)
186     for bomb in bombs:
187         bomb.update(screen)
188     pg.display.update()
189     tmr += 1
190     clock.tick(50)
191
192
193 if __name__ == "__main__":
194     pg.init()
195     main()
196     pg.quit()
197     sys.exit()
```