














 c0a23051de /  
ProjExD\_Group14



<> Code  Pull requests  Actions  Projects  Wiki 

  C0A23051/time 

ProjExD\_Group14 / dodge\_bomb.py 

 Go to file  t 



kinoshita Time機能の途中

89bf1ee · 11 minutes ago



281 lines (237 loc) · 8.57 KB

```
1  import os
2  import pygame as pg
3  import random
4  import sys
5  import time
6
7
8  WIDTH, HEIGHT = 1100, 650
9  start_time = None
10 time_limit = 10
11 ▼ DELTA = {pg.K_UP: ( 0,-5),
12             pg.K_DOWN: ( 0,+5),
13             pg.K_LEFT: (-5, 0),
14             pg.K_RIGHT:(+5, 0),
15             }
16
17 os.chdir(os.path.dirname(os.path.abspath(__file__)))
18
19 ▼ def check_bound(obj_rct: pg.rect) -> tuple[bool, bool]:
20     """
21     引数 : こうかとん または 爆弾のRect
22     戻り値: 真理値タプル (横判定結果、縦判定結果)
23     画面内ならTrue 画面外ならFalse
24     """
25     yoko,tate = True,True
26     if obj_rct.left < WIDTH*2/5 or WIDTH*3/5 < obj_rct.right:
27         yoko = False
```

 C0A23051/time 

ProjExD\_Group14 / dodge\_bomb.py

↑ Top

Code

Blame

Raw



```
19  def check_bound(obj_rct: pg.rect) -> tuple[bool, bool]:
33 ▼ class Bird:
34     """
35     ゲームキャラクター (こうかとん) に関するクラス
36     """
37 ▼     delta = { # 押下キーと移動量の辞書
38                 pg.K_UP: (0, -5),
39                 pg.K_DOWN: (0, +5),
40                 pg.K_LEFT: (-5, 0),
41                 pg.K_RIGHT: (+5, 0),
```

```

42     }
43     img0 = pg.transform.rototozoom(pg.image.load("fig/3.png"), 0, 0.9)
44     img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん (右向き)
45     imgs = { # 0度から反時計回りに定義
46         (+5, 0): img, # 右
47         (+5, -5): pg.transform.rototozoom(img, 45, 0.9), # 右上
48         (0, -5): pg.transform.rototozoom(img, 90, 0.9), # 上
49         (-5, -5): pg.transform.rototozoom(img0, -45, 0.9), # 左上
50         (-5, 0): img0, # 左
51         (-5, +5): pg.transform.rototozoom(img0, 45, 0.9), # 左下
52         (0, +5): pg.transform.rototozoom(img, -90, 0.9), # 下
53         (+5, +5): pg.transform.rototozoom(img, -45, 0.9), # 右下
54     }
55
56     def __init__(self, xy: tuple[int, int]):
57         """
58         こうかとん画像Surfaceを生成する
59         引数 xy: こうかとん画像の初期位置座標タプル
60         """
61         self.img = __class__.imgs[(+5, 0)]
62         self.rct: pg.Rect = self.img.get_rect()
63         self.rct.center = xy
64
65     def change_img(self, num: int, screen: pg.Surface):
66         """
67         こうかとん画像を切り替え、画面に転送する
68         引数1 num: こうかとん画像ファイル名の番号
69         引数2 screen: 画面Surface
70         """
71         self.img = pg.transform.rototozoom(pg.image.load(f"fig/{num}.png"), 0, 0.9)
72         screen.blit(self.img, self.rct)
73
74     def update(self, key_lst: list[bool], screen: pg.Surface):
75         """
76         押下キーに応じてこうかとんを移動させる
77         引数1 key_lst: 押下キーの真理値リスト
78         引数2 screen: 画面Surface
79         """
80         sum_mv = [0, 0]
81         for k, mv in __class__.delta.items():
82             if key_lst[k]:
83                 sum_mv[0] += mv[0]
84                 sum_mv[1] += mv[1]
85         self.rct.move_ip(sum_mv)
86         if check_bound(self.rct) != (True, True):
87             self.rct.move_ip(-sum_mv[0], -sum_mv[1])
88         if not (sum_mv[0] == 0 and sum_mv[1] == 0):
89             self.img = __class__.imgs[tuple(sum_mv)]
90         screen.blit(self.img, self.rct)
91
92     def gameover(screen):
93         """
94         ゲームオーバー時の処理
95         引数: screen
96         戻り値: 無し
97         背景をブラックアウト、GameOverの文字を中心に、その上に泣いているこうかとの画像を配置
98         """
99         black= pg.Surface((WIDTH, HEIGHT))

```

```
100 pg.draw.rect(black,(0,0,0),(0,0,WIDTH,HEIGHT))
101 black.set_alpha(100)
102 screen.blit(black,(0,0))
103 cry_img = pg.transform.rotozoom(pg.image.load("fig/8.png"),0,1.2)
104 cry_rct = cry_img.get_rect(center=(WIDTH//2,HEIGHT//2-50))
105 screen.blit(cry_img,cry_rct)
106 fonto = pg.font.Font(None,80)
107 txt = fonto.render("GameOver",True,
108                    (0,150,255))
109 txt_rct = txt.get_rect()
110 txt_rct.center = WIDTH//2,HEIGHT//2
111 screen.blit(txt,txt_rct)
112 pg.display.update()
113 time.sleep(5)
114
115 ✓ def enemy(num,screen: pg.surface):
116     if num == 1:
117         en_img1 = pg.transform.rotozoom(pg.image.load("fig/DQM12.webp"),0,0.6)
118         en_rct1 = en_img1.get_rect()
119         en_rct1.centerx = 150
120         en_rct1.centery = HEIGHT/2
121         screen.blit(en_img1,en_rct1)
122         stage1()
123
124     elif num == 2:
125         en_img2 = pg.transform.rotozoom(pg.image.load("fig/en1.png"),0,0.4)
126         en_rct2 = en_img2.get_rect()
127         en_rct2.centerx = WIDTH-150
128         en_rct2.centery = HEIGHT/2
129         screen.blit(en_img2,en_rct2)
130         stage2()
131
132     elif num == 3:
133         en_img3 = pg.transform.rotozoom(pg.image.load("fig/en5.png"),0,1)
134         en_img4 = pg.transform.rotozoom(pg.image.load("fig/en6.png"),0,1)
135         en_rct3 = en_img3.get_rect()
136         en_rct3.centerx = 150
137         en_rct3.centery = HEIGHT/2
138         en_rct4 = en_img4.get_rect()
139         en_rct4.centerx = WIDTH-150
140         en_rct4.centery = HEIGHT/2
141         screen.blit(en_img3,en_rct3)
142         screen.blit(en_img4,en_rct4)
143         stage3()
144
145     elif num == 4:
146         en_img5 = pg.transform.rotozoom(pg.image.load("fig/en7.png"),0,0.6)
147         en_rct5 = en_img5.get_rect()
148         en_rct5.centerx = 150
149         en_rct5.centery = HEIGHT/2
150         screen.blit(en_img5,en_rct5)
151         stageEX()
152
153
154 def stage1():
155     return 0
156
157 def stage2():
158     return 0
159
160 def stage3():
161     return 0
162
163 def stageEX():
164     return 0
165
166 def main():
167     screen = pg.display.set_mode((WIDTH,HEIGHT))
168     pg.display.set_caption("Dodge Bomb")
169     clock = pg.time.Clock()
170     num = 1
171     while True:
172         enemy(num,screen)
173         num += 1
174         if num == 5:
175             num = 1
176         clock.tick(60)
177     pg.quit()
178
```

```
158         return 0
159
160     def stage3():
161         return 0
162
163     def stageEX():
164         return 0
165
166     def timescore(screen, stage):
167         global start_time
168         if start_time is None:
169             start_time = time.time() # タイマー開始
170
171         spent_time = time.time() - start_time
172         end_time = max(0, time_limit - spent_time)
173
174         font = pg.font.Font(None, 36)
175         time_text = font.render(f"Time: {int(end_time)}s", True, (255, 255, 255))
176         screen.blit(time_text, (10, 10))
177
178         # EXステージでのクリア表示
179         if stage == 4 and end_time <= 0:
180             black_scr = pg.Surface((WIDTH, HEIGHT))
181             pg.draw.rect(black_scr, (0, 0, WIDTH, HEIGHT))
182             black_scr.set_alpha(180)
183             screen.blit(black_scr, (0, 0))
184
185             kk_img = pg.transform.rotozoom(pg.image.load("fig/8.png"), 0, 0.9)
186             kk_rct = kk_img.get_rect()
187             kk_rct.center = WIDTH / 2 + 180, HEIGHT / 2
188             screen.blit(kk_img, kk_rct)
189             kk_rct.center = WIDTH / 2 - 180, HEIGHT / 2
190             screen.blit(kk_img, kk_rct)
191
192             clear_font = pg.font.Font(None, 80)
193             clear_text = clear_font.render("クリア", True, (255, 255, 255))
194             clear_text_rect = clear_text.get_rect()
195             clear_text_rect.center = WIDTH / 2, HEIGHT / 2
196             screen.blit(clear_text, clear_text_rect)
197             pg.display.update()
198
199             # 'n' キー待機
200             waiting = True
201             while waiting:
202                 for event in pg.event.get():
203                     if event.type == pg.QUIT:
204                         pg.quit()
205                         sys.exit()
206                     elif event.type == pg.KEYDOWN and event.key == pg.K_n:
207                         waiting = False
208                         start_time = None
209
210             return stage + 1 # ここでステージ更新を停止または調整
211
212         elif end_time <= 0:
213             # 次のステージへの移行とタイマーリセット
214             black_scr = pg.Surface((WIDTH, HEIGHT))
215             black_scr.fill((0, 0, 0))
216             black_scr.set_alpha(180)
```

```
216     black_scr.set_alpha(100),
217     screen.blit(black_scr, (0, 0))
218
219     msg = font.render("Please Press 'N'", True, (255, 255, 255))
220     screen.blit(msg, (WIDTH // 2 - msg.get_width() // 2, HEIGHT // 2))
221     pg.display.update()
222
223     # 'n' キー待機
224     waiting = True
225     while waiting:
226         for event in pg.event.get():
227             if event.type == pg.QUIT:
228                 pg.quit()
229                 sys.exit()
230             elif event.type == pg.KEYDOWN and event.key == pg.K_n:
231                 waiting = False
232                 start_time = None
233
234         return stage + 1
235
236     return stage
237
238
239     def skill():
240         return 0
241
242     ✓ def main():
243         pg.display.set_caption("避ける！こうかとん")
244         screen = pg.display.set_mode((WIDTH, HEIGHT))
245         bg_img = pg.transform.rotozoom(pg.image.load("fig/bg.png"), 0, 1.9)
246         bird = Bird([WIDTH/2, HEIGHT/2])
247         stage = 1
248
249         clock = pg.time.Clock()
250         tmr = 0
251         while True:
252             for event in pg.event.get():
253                 if event.type == pg.QUIT:
254                     return
255             screen.blit(bg_img, [0, 0])
256
257
258             key_lst = pg.key.get_pressed()
259             sum_mv = [0, 0]
260
261
262             for key, tpl in DELTA.items():
263                 if key_lst[key]:
264                     sum_mv[0] += tpl[0] #横方向
265                     sum_mv[1] += tpl[1] #縦方向
266
267
268             bird.update(key_lst, screen)
269             enemy(stage, screen)
270             stage = timescore(screen, stage)
271             skill()
272             pg.display.update()
273             tmr += 1
274             clock.tick(50)
```

```
275
276
277     if __name__ == "__main__":
278         pg.init()
279         main()
280         pg.quit()
281         sys.exit()
```