










 c0a242882d / ProjExD_3



 **Code**  Issues **6**  Pull requests  Discussions  Actions  Projects  Wiki  Sec

 **main** **ProjExD_3 / fight_kokaton.py** 

Go to file

t

...



c0a242882d 演習3:爆発表示

5864b97 · 34 minutes ago



263 lines (232 loc) · 9.11 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         こうかとおん画像Surfaceを生成する
54         引数 xy: こうかとお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         こうかとおん画像を切り替え、画面に転送する
63         引数1 num: こうかとお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おんを移動させる
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         こうかとお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おん (Birdインスタンス)
96         """
97         self.img = pg.image.load("fig/beam.png")
98         self.rct = self.img.get_rect()
99         self.rct.centery = bird.rct.centery
100        self.rct.left = bird.rct.right # ビームの左座標=こうかとおんの右座標

```

```
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     class Score:
144     def __init__(self):
145         self.fonto = pg.font.SysFont("hgp創英角林° ヴァ° 体", 30)
146         self.color = (0, 0, 255)
147         self.value = 0
148         self.img = self.fonto.render(f"スコア: {self.value}", 0, self.color)
149         self.rect = self.img.get_rect()
150         self.rect.center = 100, HEIGHT-50
151     def update(self, screen: pg.Surface):
152         self.img = self.fonto.render(f"スコア: {self.value}", 0, self.color)
153         screen.blit(self.img, self.rect)
154
155     class Explosion:
156         """
157         爆発エフェクトに関するクラス
158         """
```

```

159  ✓ def __init__(self, center: tuple[int, int]):
160      """
161      爆発画像Surfaceを生成する
162      引数 center: 爆発の中心座標
163      """
164      ex_img = pg.image.load("fig/explosion.gif") #爆発gifを読み込む
165      self.imgs = [
166          ex_img,
167          pg.transform.flip(ex_img, True, False),
168          pg.transform.flip(ex_img, False, True),
169      ]
170      self.life = 30 # 爆発の表示時間
171      self.rct = self.imgs[0].get_rect()
172      self.rct.center = center
173
174  ✓ def update(self, screen: pg.Surface):
175      """
176      爆発画像を交互に反転表示して爆発演出
177      引数 screen: 画面Surface
178      """
179      self.life -= 1 #爆発経過時間を1減算
180      img = self.imgs[self.life % len(self.imgs)]
181      screen.blit(img, self.rct)
182
183
184
185  ✓ def main():
186      pg.display.set_caption("たたかえ！こうかとん")
187      screen = pg.display.set_mode((WIDTH, HEIGHT))
188      bg_img = pg.image.load("fig/pg_bg.jpg")
189      bird = Bird((300, 200))
190      bomb = Bomb((255, 0, 0), 10)
191      score = Score()
192      bombs=[] # 爆弾用の空のリスト
193      beams=[] # ビーム用のリスト
194      explosions = [] # Explosionインスタンス用の空のリスト
195      for _ in range(NUM_OF_BOMBS):
196          bombs.append(Bomb((255,0,0),10))
197      # 内包表記
198      beam = None # ゲーム初期化時にはビームは存在しない
199      clock = pg.time.Clock()
200      tmr = 0
201      while True:
202          for event in pg.event.get():
203              if event.type == pg.QUIT:
204                  return
205              if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
206                  # スペースキー押下でBeamクラスのインスタンス生成
207                  beams.append(Beam(bird))
208          screen.blit(bg_img, [0, 0])
209
210          for n2,bomb2 in enumerate(bombs):
211              for n,beam2 in enumerate(beams):
212                  if bomb2.rct.colliderect(beam2.rct):
213                      explosions.append(Explosion(bomb.rct.center)) #爆発追加
214                      bombs[n2]=None
215                      beams[n]=None
216                      bird.change_img(6, screen)

```

```
217         pg.display.update()
218         score.value += 1 # スコアを1点加算
219         beams = [beam for beam in beams if beam is not None]
220         bombs=[bomb for bomb in bombs if bomb is not None]
221
222         new_explosions = []
223         for ex in explosions:
224             if ex.life > 0: # 爆発時間が0以上なら表示
225                 ex.update(screen)
226                 new_explosions.append(ex)
227         explosions = new_explosions # 新しい画像に置き換える
228
229         for bomb in bombs:
230             if bird.rct.colliderect(bomb.rct):
231                 # 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
232                 bird.change_img(8, screen)
233                 fonto = pg.font.Font(None, 80)
234                 txt = fonto.render("Game Over", True, (255, 0, 0))
235                 screen.blit(txt, [WIDTH//2-150, HEIGHT//2])
236
237                 score.update(screen) # スコアの描画
238
239                 pg.display.update()
240                 time.sleep(1)
241                 return
242
243
244
245         key_lst = pg.key.get_pressed()
246         bird.update(key_lst, screen)
247         for i,beam in enumerate(beams):
248             if check_bound(beam.rct)==(False,False): # ビームが存在する時
249                 del beams[i]
250             else:
251                 beam.update(screen)
252         for bomb in bombs:
253             bomb.update(screen)
254         pg.display.update()
255         tmr += 1
256         clock.tick(50)
257
258
259 if __name__ == "__main__":
260     pg.init()
261     main()
262     pg.quit()
263     sys.exit()
```