


 c0b23112b1 / ProjExD_Group09



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

 C0B23112/item ▾

ProjExD_Group09

Go to file t ...

/ group09_kokaton.py 

 c0b23112b1 アイテム追加

4e46dfe · 1 minute ago ... 

395 lines (344 loc) · 14.4 KB

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

[Raw](#)     

```
1  import math
2  import os
3  import random
4  import sys
5  import time
6  import pygame as pg
7
8  WIDTH, HEIGHT = 1600, 900 # ゲームウィンドウの幅, 高さ
9  os.chdir(os.path.dirname(os.path.abspath(__file__)))
10
11  def check_bound(obj_rct:pg.Rect) -> tuple[bool, bool]:
12      """
13      Rectの画面内外判定用の関数
14      引数: こうかとんRect, または, 爆弾Rect, またはビームRect
15      戻り値: 横方向判定結果, 縦方向判定結果 (True: 画面内/False: 画面外)
16      """
17      yoko, tate = True, True
18      if obj_rct.left < 0 or WIDTH < obj_rct.right: # 横方向のはみ出し判定
19          yoko = False
20      if obj_rct.top < 0 or HEIGHT < obj_rct.bottom: # 縦方向のはみ出し判定
21          tate = False
22      return yoko, tate
23
24
25  def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
26      """
27      orgから見て, dstがどこにあるかを計算し, 方向ベクトルをタプルで返す
28      引数1 org: 爆弾SurfaceのRect
29      引数2 dst: こうかとんSurfaceのRect
30      戻り値: orgから見たdstの方向ベクトルを表すタプル
31      """
32      x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
33      norm = math.sqrt(x_diff**2+y_diff**2)
34      return x_diff/norm, y_diff/norm
35
36
37  class Bird(pg.sprite.Sprite):
38      """
39      ゲームキャラクター (こうかとん) に関するクラス
40      """
41      delta = { # 押下キーと移動量の辞書
```

```
42     pg.K_UP: (0, -1),
43     pg.K_DOWN: (0, +1),
44     pg.K_LEFT: (-1, 0),
45     pg.K_RIGHT: (+1, 0),
46 }
47
48 ✓ def __init__(self, num: int, xy: tuple[int, int]):
49     """
50     こうかтон画像Surfaceを生成する
51     引数1 num : こうかтон画像ファイル名の番号
52     引数2 xy : こうかтон画像の位置座標タプル
53     """
54     super().__init__()
55     img0 = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
56     img = pg.transform.flip(img0, True, False) # デフォルトのこうかтон
57     self.imgs = {
58         (+1, 0): img, # 右
59         (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
60         (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
61         (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
62         (-1, 0): img0, # 左
63         (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
64         (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
65         (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
66     }
67     self.dire = (+1, 0)
68     self.image = self.imgs[self.dire]
69     self.rect = self.image.get_rect()
70     self.rect.center = xy
71     self.speed = 10
72
73 ✓ def bird_check(self):
74     """
75     シフトを押している間速さを変える関数
76     """
77     key_lst = pg.key.get_pressed()
78     if key_lst[pg.K_LSHIFT]:
79         self.speed = 20
80     else:
81         self.speed = 10
82
83
84 ✓ def change_img(self, num: int, screen: pg.Surface):
85     """
86     こうかтон画像を切り替え、画面に転送する
87     引数1 num : こうかтон画像ファイル名の番号
88     引数2 screen : 画面Surface
89     """
90     self.image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
91     screen.blit(self.image, self.rect)
92
93 ✓ def update(self, key_lst: list[bool], screen: pg.Surface):
94     """
95     押下キーに応じてこうかтонを移動させる
96     引数1 key_lst : 押下キーの真理値リスト
97     引数2 screen : 画面Surface
98     """
99     sum_mv = [0, 0]
```

```

100     for k, mv in __class__.delta.items():
101         if key_lst[k]:
102             sum_mv[0] += mv[0]
103             sum_mv[1] += mv[1]
104     self.rect.move_ip(self.speed*sum_mv[0], self.speed*sum_mv[1])
105     if check_bound(self.rect) != (True, True):
106         self.rect.move_ip(-self.speed*sum_mv[0], -self.speed*sum_mv[1])
107     if not (sum_mv[0] == 0 and sum_mv[1] == 0):
108         self.dire = tuple(sum_mv)
109         self.image = self.imgs[self.dire]
110     screen.blit(self.image, self.rect)
111
112
113     class Bomb(pg.sprite.Sprite):
114         """
115         爆弾に関するクラス
116         """
117         colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
118
119     def __init__(self, emy: "Enemy", bird: Bird):
120         """
121         爆弾円Surfaceを生成する
122         引数1 emy : 爆弾を投下する敵機
123         引数2 bird : 攻撃対象のこうかとん
124         """
125         super().__init__()
126         rad = random.randint(10, 50) # 爆弾円の半径 : 10以上50以下の乱数
127         self.image = pg.Surface((2*rad, 2*rad))
128         color = random.choice(__class__.colors) # 爆弾円の色 : クラス変数からランダム選択
129         pg.draw.circle(self.image, color, (rad, rad), rad)
130         self.image.set_colorkey((0, 0, 0))
131         self.rect = self.image.get_rect()
132         # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
133         self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
134         self.rect.centerx = emy.rect.centerx
135         self.rect.centery = emy.rect.centery+emy.rect.height/2
136         self.speed = 6
137
138     def update(self):
139         """
140         爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
141         引数 screen : 画面Surface
142         """
143         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
144         if check_bound(self.rect) != (True, True):
145             self.kill()
146
147
148     class Beam(pg.sprite.Sprite):
149         """
150         ビームに関するクラス
151         """
152     def __init__(self, bird: Bird):
153         """
154         ビーム画像Surfaceを生成する
155         引数 bird : ビームを放つこうかとん
156         """
157         super().__init__()

```

```

158         self.vx, self.vy = bird.dire
159         angle = math.degrees(math.atan2(-self.vy, self.vx))
160         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 2.0)
161         self.vx = math.cos(math.radians(angle))
162         self.vy = -math.sin(math.radians(angle))
163         self.rect = self.image.get_rect()
164         self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
165         self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
166         self.speed = 10
167
168     def update(self):
169         """
170         ビームを速度ベクトルself.vx, self.vyに基づき移動させる
171         引数 screen : 画面Surface
172         """
173         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
174         if check_bound(self.rect) != (True, True):
175             self.kill()
176
177
178     class Explosion(pg.sprite.Sprite):
179         """
180         爆発に関するクラス
181         """
182     def __init__(self, obj: "Bomb|Enemy", life: int):
183         """
184         爆弾が爆発するエフェクトを生成する
185         引数1 obj : 爆発するBombまたは敵機インスタンス
186         引数2 life : 爆発時間
187         """
188         super().__init__()
189         img = pg.image.load(f"fig/explosion.gif")
190         self.imgs = [img, pg.transform.flip(img, 1, 1)]
191         self.image = self.imgs[0]
192         self.rect = self.image.get_rect(center=obj.rect.center)
193         self.life = life
194
195     def update(self):
196         """
197         爆発時間を1減算した爆発経過時間_lifeに応じて爆発画像を切り替えることで
198         爆発エフェクトを表現する
199         """
200         self.life -= 1
201         self.image = self.imgs[self.life//10%2]
202         if self.life < 0:
203             self.kill()
204
205
206     class Enemy(pg.sprite.Sprite):
207         """
208         敵機に関するクラス
209         """
210         imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
211
212     def __init__(self):
213         super().__init__()
214         self.image = random.choice(__class__.imgs)
215         self.rect = self.image.get_rect()
216         self.rect.center = random.randint(0, WIDTH), 0

```

```

210 self.rect.center = random.randrange(0, WIDTH), 0
211
212 self.vy = +6
213
214 self.bound = random.randrange(50, HEIGHT/2) # 停止位置
215
216 self.state = "down" # 降下状態or停止状態
217
218 self.interval = random.randrange(50, 300) # 爆弾投下インターバル
219
220
221
222 ✓ def update(self):
223     """
224     敵機を速度ベクトルself.vyに基づき移動（降下）させる
225     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
226     引数 screen : 画面Surface
227     """
228     if self.rect.centery > self.bound:
229         self.vy = 0
230         self.state = "stop"
231     self.rect.centery += self.vy
232
233
234 ✓ class Score:
235     """
236     打ち落とした爆弾, 敵機の数スコアとして表示するクラス
237     爆弾 : 1点
238     敵機 : 10点
239     """
240 ✓ def __init__(self):
241     self.font = pg.font.Font(None, 50)
242     self.color = (0, 0, 255)
243     self.value = 0
244     self.image = self.font.render(f"Score: {self.value}", 0, self.color)
245     self.rect = self.image.get_rect()
246     self.rect.center = 100, HEIGHT-50
247
248     def update(self, screen: pg.Surface):
249         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
250         screen.blit(self.image, self.rect)
251
252 ✓ class item(pg.sprite.Sprite):
253     """
254     アイテムを表示させるクラス
255     """
256     imx = 200 #画像の横幅
257     imy = 150 #画像の縦幅
258
259 ✓ def __init__(self):
260     super().__init__()
261     self.yoko :bool = True #横にはみだしているかどうか
262     self.img1 = pg.image.load("fig/r_item1.png") #画像ロード
263     self.img2 = pg.image.load("fig/ramen.jpg")
264     self.img3 = pg.image.load("fig/gyoza.jpg")
265     self.lis = [pg.transform.scale(self.img1,(item.imx,item.imy)),
266                 pg.transform.scale(self.img2,(item.imx,item.imy)),
267                 pg.transform.scale(self.img3,(item.imx,item.imy))] #画像サイズ変更
268     self.image = random.choice(self.lis) #画像リストから出すアイテムをランダムに選ぶ
269     self.y = random.randrange(100,HEIGHT-100) #アイテムのy軸は全体が見える位置に出す
270     self.rect = self.image.get_rect()
271     self.rect.center = WIDTH,self.y #初期位置は右端
272
273 ✓ def update(self):
274     """

```

```
275     アイテムを右から左に流れるようにする処理
276     アイテムが左に見えなくなるまで流れたらグループから削除する処理
277     """
278     #self.rect.move_ip((-10,0))
279     self.rect.move_ip(-10,0)
280     if self.rect.right < 0: # 横方向のはみ出し判定
281         self.yoko :bool = False
282     if not self.yoko:
283         self.kill() #はみ出た画像削除
284
285
286     ▼ def main():
287         pg.display.set_caption("こうかтон疾風伝")
288         screen = pg.display.set_mode((WIDTH, HEIGHT))
289         bg_img = pg.image.load(f"fig/pg_bg.jpg")
290         bg_img2 = pg.image.load(f"fig/pg_bg.jpg")
291         bg_img2 = pg.transform.flip(bg_img2,True,False)
292         score = Score()
293         bird = Bird(3, (900, 400))
294         bombs = pg.sprite.Group()
295         beams = pg.sprite.Group()
296         exps = pg.sprite.Group()
297         emys = pg.sprite.Group()
298         items = pg.sprite.Group() #アイテム
299
300         muteki :bool = False #無敵かどうか
301         muteki_time :int = 0 #無敵時間カウント
302         muteki_rt :int = 300 #無敵継続時間
303
304         tmr = 0
305
306         font = pg.font.Font(None, 150) #フォント指定
307
308
309         clock = pg.time.Clock()
310         while True:
311             if tmr%500 == 0:
312                 items.add(item()) #アイテム追加
313
314             bird.bird_check()
315             key_lst = pg.key.get_pressed()
316             for event in pg.event.get():
317                 if event.type == pg.QUIT:
318                     return
319                 if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
320                     beams.add(Beam(bird))
321
322             x = tmr%3200
323
324             screen.blit(bg_img, [-x, 0])
325             screen.blit(bg_img2, [-x+1600, 0])
326             screen.blit(bg_img, [-x+3200, 0])
327             screen.blit(bg_img2, [-x+4800, 0])
328
329             if tmr%2000 == 0: # 200フレームに1回, 敵機を出現させる
330                 emys.add(Enemy())
331
332             for emy in emys:
```

```
333         if emy.state == "stop" and tmr%emy.interval == 0:
334             # 敵機が停止状態に入ったら, intervalに応じて爆弾投下
335             bombs.add(Bomb(emy, bird))
336
337     for emy in pg.sprite.groupcollide(emics, beams, True, True).keys():
338         exps.add(Explosion(emy, 100)) # 爆発エフェクト
339         score.value += 10 # 10点アップ
340         bird.change_img(6, screen) # こうかとおん喜びエフェクト
341     for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
342         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
343         score.value += 1 # 1点アップ
344
345     if not muteki:
346         if len(pg.sprite.spritecollide(bird, bombs, True)) != 0:
347             bird.change_img(8, screen) # こうかとおん悲しみエフェクト
348             score.update(screen)
349             pg.display.update()
350             time.sleep(2)
351             return
352
353     if pg.sprite.spritecollide(bird, items, True): #アイテム取ったら
354         bird.change_img(6, screen) #喜ぶ
355         if muteki: #既に無敵なら
356             muteki_rt += 30 #無敵時間が30秒増える
357         else:
358             muteki = True #無敵になる
359     if muteki: #無敵ならば
360         muteki_time += 1 #無敵時間計測
361         #score.value += 1 #スコア上がり
362         bird.speed = 50 #スピードも一時的に上がる
363         image = font.render(f"INVINCIBLE TIME:{muteki_rt-muteki_time}", 0, (random.randint(0,255),
364         screen.blit(image, (WIDTH/10,HEIGHT/2))
365         bird.change_img(9,screen)
366
367     if muteki_time > muteki_rt: #無敵時間なければ
368         muteki = False
369         muteki_time = 0
370         bird.speed = 10
371         muteki_rt = 300
372
373
374     bird.update(key_lst, screen)
375     beams.update()
376     beams.draw(screen)
377     emys.update()
378     emys.draw(screen)
379     bombs.update()
380     bombs.draw(screen)
381     exps.update()
382     exps.draw(screen)
383     score.update(screen)
384     items.update()
385     items.draw(screen)
386     pg.display.update()
387     tmr += 10
388     clock.tick(50) #50
389
390
```

```
391     if __name__ == "__main__":  
392         pg.init()  
393         main()  
394         pg.quit()  
395         sys.exit()
```