


 orgilmgl0301 / ProjExD_Group05



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

ProjExD_Group05 / musou_kokaton.py



orgilmgl0301 敵のパワーアップ

7 minutes ago



429 lines (373 loc) · 15.4 KB

CodeBlame

RawCopyDownloadEditDropdownCompare

```
1  import math
2  import os
3  import random
4  import sys
5  import time
6  import pygame as pg
7
8
9  WIDTH, HEIGHT = 1000, 600 # ゲームウィンドウの幅, 高さ
10 os.chdir(os.path.dirname(os.path.abspath(__file__)))
11
12
13  def check_bound(obj_rct:pg.Rect) -> tuple[bool, bool]:
14      """
15      Rectの画面内外判定用の関数
16      引数: こうかとんRect, または, 爆弾Rect, またはビームRect
17      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)
18      """
19      yoko, tate = True, True
20      if obj_rct.left < 0 or WIDTH < obj_rct.right: # 横方向のはみ出し判定
21          yoko = False
22      if obj_rct.top < 0 or HEIGHT < obj_rct.bottom: # 縦方向のはみ出し判定
23          tate = False
24      return yoko, tate
25
26
27  def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
28      """
29      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
30      引数1 org: 爆弾SurfaceのRect
31      引数2 dst: こうかとんSurfaceのRect
32      戻り値: orgから見たdstの方向ベクトルを表すタプル
33      """
34      x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
35      norm = math.sqrt(x_diff**2+y_diff**2)
36      return x_diff/norm, y_diff/norm
37
38
```

```
class Bird(pg.sprite.Sprite):
    """
    ゲームキャラクター（こうかとん）に関するクラス
    """
    delta = { # 押下キーと移動量の辞書
        pg.K_UP: (0, -1),
        pg.K_DOWN: (0, +1),
        pg.K_LEFT: (-1, 0),
        pg.K_RIGHT: (+1, 0),
    }

    def __init__(self, num: int, xy: tuple[int, int]):
        """
        こうかとん画像Surfaceを生成する
        引数1 num: こうかとん画像ファイル名の番号
        引数2 xy: こうかとん画像の位置座標タプル
        """
        super().__init__()
        img0 = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
        img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
        self.imgs = {
            (+1, 0): img, # 右
            (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
            (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
            (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
            (-1, 0): img0, # 左
            (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
            (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
            (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
        }
        self.dire = (+1, 0)
        self.image = self.imgs[self.dire]
        self.rect = self.image.get_rect()
        self.rect.center = xy
        self.speed = 10
        self.high_speed = 20 #feature1
        self.high = False

    def change_img(self, num: int, screen: pg.Surface):
        """
        こうかとん画像を切り替え、画面に転送する
        引数1 num: こうかとん画像ファイル名の番号
        引数2 screen: 画面Surface
        """
        self.image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
        screen.blit(self.image, self.rect)

    def update(self, key_lst: list[bool], screen: pg.Surface):
        """
        押下キーに応じてこうかとんを移動させる
        引数1 key_lst: 押下キーの真値リスト
        引数2 screen: 画面Surface
        """
        sum_mv = [0, 0]
        for k, mv in __class__.delta.items():
```

```

94         if key_lst[k]:
95             sum_mv[0] += mv[0]
96             sum_mv[1] += mv[1]
97     if pg.key.get_mods() & pg.KMOD_LSHIFT: # 左Shiftキーが押されているか確認
98         self.speed = self.high_speed # 高速化
99         self.high = True
100     else:
101         self.speed = 10
102         self.high = False
103     self.rect.move_ip(self.speed*sum_mv[0], self.speed*sum_mv[1])
104     if check_bound(self.rect) != (True, True):
105         self.rect.move_ip(-self.speed*sum_mv[0], -self.speed*sum_mv[1])
106     if not (sum_mv[0] == 0 and sum_mv[1] == 0):
107         self.dire = tuple(sum_mv)
108         self.image = self.imgs[self.dire]
109     screen.blit(self.image, self.rect)
110
111
112     class Bomb(pg.sprite.Sprite):
113         """
114         爆弾に関するクラス
115         """
116         colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
117
118     def __init__(self, emy: "Enemy", bird: Bird):
119         """
120         爆弾円Surfaceを生成する
121         引数1 emy: 爆弾を投下する敵機
122         引数2 bird: 攻撃対象のこうかとん
123         """
124         super().__init__()
125         rad = random.randint(10, 50) # 爆弾円の半径: 10以上50以下の乱数
126         self.image = pg.Surface((2*rad, 2*rad))
127         color = random.choice(__class__.colors) # 爆弾円の色: クラス変数からランダム選択
128         pg.draw.circle(self.image, color, (rad, rad), rad)
129         self.image.set_colorkey((0, 0, 0))
130         self.rect = self.image.get_rect()
131         # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
132         self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
133         self.rect.centerx = emy.rect.centerx
134         self.rect.centery = emy.rect.centery+emy.rect.height/2
135         self.speed = 6
136         self.state = 'active'
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 Bomb(pg.sprite.Sprite):

```

```
140 class Beam(pg.sprite.Sprite):
141     """
142     ビームに関するクラス
143     """
144     def __init__(self, bird: Bird, angle0 = 0):
145         """
146         ビーム画像Surfaceを生成する
147         引数 bird: ビームを放つこうかとん
148         """
149         super().__init__()
150         self.vx, self.vy = bird.dire
151         angle = math.degrees(math.atan2(-self.vy, self.vx)) + angle0
152         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 2.0)
153         self.vx = math.cos(math.radians(angle))
154         self.vy = -math.sin(math.radians(angle))
155         self.rect = self.image.get_rect()
156         self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
157         self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
158         self.speed = 10
159
160     def update(self):
161         """
162         ビームを速度ベクトルself.vx, self.vyに基づき移動させる
163         引数 screen: 画面Surface
164         """
165         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
166         if check_bound(self.rect) != (True, True):
167             self.kill()
168
169 class NeoBeam :
170     """
171     ビーム複数発射
172     """
173     def __init__(self, bird: Bird, num: int):
174         self.bird = bird
175         self.num = num
176
177     def gen_beams(self) -> list[Beam]:
178         return [Beam(self.bird, angle) for angle in range(-50, +51, int(100/(self.num-1)))]
179
180 class Explosion(pg.sprite.Sprite):
181     """
182     爆発に関するクラス
183     """
184     def __init__(self, obj: "Bomb|Enemy", life: int):
185         """
186         爆弾が爆発するエフェクトを生成する
187         引数1 obj: 爆発するBombまたは敵機インスタンス
188         引数2 life: 爆発時間
189         """
190         super().__init__()
191         img = pg.image.load(f"fig/explosion.gif")
192         self.imgs = [img, pg.transform.flip(img, 1, 1)]
```

```

203         self.image = self.imgs[0]
204         self.rect = self.image.get_rect(center=obj.rect.center)
205         self.life = life
206
207     def update(self):
208         """
209         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
210         爆発エフェクトを表現する
211         """
212         self.life -= 1
213         self.image = self.imgs[self.life//10%2]
214         if self.life < 0:
215             self.kill()
216
217
218     class Enemy(pg.sprite.Sprite):
219         """
220         敵機に関するクラス
221         """
222         imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
223
224     def __init__(self):
225         super().__init__()
226         self.image = random.choice(__class__.imgs)
227         self.rect = self.image.get_rect()
228         self.rect.center = random.randint(0, WIDTH), 0
229         self.vy = +6
230         self.bound = random.randint(50, HEIGHT/2) # 停止位置
231         self.state = "down" # 降下状態or停止状態
232         self.interval = random.randint(50, 300) # 爆弾投下インターバル
233
234     def update(self):
235         """
236         敵機を速度ベクトルself.vyに基づき移動（降下）させる
237         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
238         引数 screen: 画面Surface
239         """
240         if self.rect.centery > self.bound:
241             self.vy = 0
242             self.state = "stop"
243             self.rect.centery += self.vy
244
245
246     class Score:
247         """
248         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
249         爆弾: 1点
250         敵機: 10点
251         """
252     def __init__(self):
253         self.font = pg.font.Font(None, 50)
254         self.color = (0, 0, 255)
255         self.value = 0
256         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
257         self.rect = self.image.get_rect()

```

```

257         self.rect = self.image.get_rect()
258         self.rect.center = 100, HEIGHT-50
259
260     def update(self, screen: pg.Surface):
261         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
262         screen.blit(self.image, self.rect)
263
264     class Gravity(pg.sprite.Sprite):
265         """
266         画面全体を覆う重力場を発生させるクラス
267         """
268     def __init__(self, life):
269         super().__init__()
270         self.image = pg.Surface((WIDTH, HEIGHT))
271         pg.draw.rect(self.image, (0, 0, 0), (0, 0, WIDTH, HEIGHT))
272         self.image.set_alpha(255)
273         self.life = life
274         self.rect = self.image.get_rect()
275
276     def update(self):
277         self.life -= 1
278         if self.life < 0:
279             self.kill()
280
281     class EMP(pg.sprite.Sprite):
282         """
283         電磁パルス
284         """
285     def __init__(self, enemies: pg.sprite.Group, bombs: pg.sprite.Group, screen: pg.Surface):
286         super().__init__()
287         self.enemies = enemies
288         self.bombs = bombs
289         self.screen = screen
290         self.image = pg.Surface((WIDTH, HEIGHT), pg.SRCALPHA)
291         self.rect = self.image.get_rect()
292         self.active = False
293         self.counter = 0
294
295     def activate(self):
296         self.active = True
297         self.counter = 3
298
299         for enemy in self.enemies:
300             enemy.interval = float('inf')
301             enemy.image = pg.transform.laplacian(enemy.image)
302
303         for bomb in self.bombs:
304             bomb.speed /= 2
305             bomb.state = 'inactive'
306
307     def update(self):
308         if self.active:
309             self.image.fill((255, 255, 0, 128))
310             self.screen.blit(self.image, self.rect)
311             self.counter -= 1

```

```

312         if self.counter <= 0:
313             self.active = False
314             self.image.fill((0, 0, 0, 0))
315
316
317     def main():
318         pg.display.set_caption("真! こうかтон無双")
319         screen = pg.display.set_mode((WIDTH, HEIGHT))
320         bg_img = pg.image.load(f"fig/pg_bg.jpg")
321         score = Score()
322
323         bird = Bird(3, (900, 400))
324         bombs = pg.sprite.Group()
325         beams = pg.sprite.Group()
326         neobeams = pg.sprite.Group()
327         exps = pg.sprite.Group()
328         emys = pg.sprite.Group()
329
330         emp = EMP(emys, bombs, screen)
331
332         gravities = pg.sprite.Group() # feature2
333
334
335         tmr = 0
336         frame = 200 #オルギル 敵の出現度を上げるためフレームを初期化
337         clock = pg.time.Clock()
338         while True:
339             key_lst = pg.key.get_pressed()
340             for event in pg.event.get():
341                 if event.type == pg.QUIT:
342                     return 0
343                 if event.type == pg.KEYDOWN:
344                     if key_lst[pg.K_LSHIFT] and event.key == pg.K_SPACE and (score.value >= 100): #オルギルー
345                         # print("a")
346                         beams.add(NeoBeam(bird, 5).gen_beams())
347                     elif event.key == pg.K_SPACE:
348                         beam = Beam(bird)
349                         beams.add(Beam(bird))
350                 if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
351                     beams.add(Beam(bird))
352                 if event.type == pg.KEYDOWN and event.key == pg.K_e:
353                     if score.value >= 20:
354                         score.value -= 20
355                         emp.activate()
356                 if event.type == pg.KEYDOWN and event.key == pg.K_RSHIFT: #オルギル 前回追加技能をちょっと直した
357                     if score.value > 200: #消費スコアが200より大きい
358                         #K_LSHIFT から K_RSHIFTに変更
359                         score.value -= 200
360                         gravities.add(Gravity(100)) #オルギル 400が長い過ぎるので100に変更
361             screen.blit(bg_img, [0, 0])
362
363             if tmr%frame == 0: #オルギルー200フレームに1回、敵機を出現させる
364                 emys.add(Enemy())
365
366         for emy in emys:

```

```

300     for emy in emys:
301         if emy.state == "stop" and tmr%emy.interval == 0:
302             # 敵機が停止状態に入ったら, intervalに応じて爆弾投下
303             bombs.add(Bomb(emy, bird))
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420

```

for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
 exps.add(Explosion(emy, 100)) # 爆発エフェクト
 score.value += 10 # 10点アップ
 bird.change_img(6, screen) # こうかたん喜びエフェクト

for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
 exps.add(Explosion(bomb, 50)) # 爆発エフェクト
 #score.value += 1 # 1点アップ

for bomb in pg.sprite.spritecollide(bird, bombs, True):
 if bomb.state == "active":
 bird.change_img(8, screen) # こうかたん悲しみエフェクト
 score.update(screen)
 pg.display.update()
 time.sleep(2)
 return

return

for enemy in emys:
 for gravity in gravities:
 if pg.sprite.collide_rect(enemy, gravity):
 exps.add(Explosion(enemy, 100))
 enemy.kill()

gravities.update()
 gravities.draw(screen)
 bird.update(key_lst, screen)
 beams.update()
 beams.draw(screen)
 emys.update()
 emys.draw(screen)
 bombs.update()
 bombs.draw(screen)
 exps.update()
 exps.draw(screen)
 emp.update()
 score.update(screen)
 pg.display.update()
 tmr += 1
 clock.tick(50)

"""
 オルギル

スコアがある時点を超えると敵の出現度が上がった行く
 """

if score.value >= 100: #オルギル スコアが100超えるとフレームが80になる
 frame = 80

if score.value >= 500: #オルギル スコアが500超えるとフレームが80になる
 frame = 50


```
421         if score.value >= 800: #オルギル スコアが800超えるとフレームが80になる
422             frame = 20
423
424
425     if __name__ == "__main__":
426         pg.init()
427         main()
428         pg.quit()
429         sys.exit()
```