






c0a2407017 /  
ProjExD\_4



<> Code Issues Pull requests Actions Projects Security

 main **ProjExD\_4 / musou\_kokaton.py** 

Go to file t

 c0a24017 Merge branch 'main' of [https://github.com/c0a2407017/ProjExD\\_4](https://github.com/c0a2407017/ProjExD_4) a5df568 · 2 days ago 

501 lines (427 loc) · 18.3 KB

Code Blame



```
1  import math
2  import os
3  import random
4  import sys
5  import time
6  import pygame as pg
7
8
9  WIDTH = 1100 # ゲームウィンドウの幅
10 HEIGHT = 650 # ゲームウィンドウの高さ
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: こうかとかんや爆弾、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 def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
29     """
30     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
31     引数1 org: 爆弾SurfaceのRect
32     引数2 dst: こうかとかんSurfaceのRect
33     戻り値: orgから見たdstの方向ベクトルを表すタプル
34     """
35     x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
36     norm = math.sqrt(x_diff**2+y_diff**2)
37     return x_diff/norm, y_diff/norm
38
39
40 class Bird(pg.sprite.Sprite):
41     """
42     ゲームキャラクター（こうかとかん）に関するクラス
```

```

43     """
44     delta = { # 押下キーと移動量の辞書
45         pg.K_UP: (0, -1),
46         pg.K_DOWN: (0, +1),
47         pg.K_LEFT: (-1, 0),
48         pg.K_RIGHT: (+1, 0),
49     }
50
51
52     def __init__(self, num: int, xy: tuple[int, int]):
53         """
54         こうかтон画像Surfaceを生成する
55         引数1 num: こうかтон画像ファイル名の番号
56         引数2 xy: こうかтон画像の位置座標タプル
57         """
58         super().__init__()
59         img0 = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 0.9)
60         img = pg.transform.flip(img0, True, False) # デフォルトのこうかтон
61         self.imgs = {
62             (+1, 0): img, # 右
63             (+1, -1): pg.transform.rotozoom(img, 45, 0.9), # 右上
64             (0, -1): pg.transform.rotozoom(img, 90, 0.9), # 上
65             (-1, -1): pg.transform.rotozoom(img0, -45, 0.9), # 左上
66             (-1, 0): img0, # 左
67             (-1, +1): pg.transform.rotozoom(img0, 45, 0.9), # 左下
68             (0, +1): pg.transform.rotozoom(img, -90, 0.9), # 下
69             (+1, +1): pg.transform.rotozoom(img, -45, 0.9), # 右下
70         }
71         self.dire = (+1, 0)
72         self.image = self.imgs[self.dire]
73         self.rect = self.image.get_rect()
74         self.rect.center = xy
75         self.speed = 10
76         self.state = "normal"
77         self.hyper_life = 500
78
79
80     def change_img(self, num: int, screen: pg.Surface):
81         """
82         こうかтон画像を切り替え、画面に転送する
83         引数1 num: こうかтон画像ファイル名の番号
84         引数2 screen: 画面Surface
85         """
86         self.image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 0.9)
87         screen.blit(self.image, self.rect)
88
89     def update(self, key_lst: list[bool], screen: pg.Surface):
90         """
91         押下キーに応じてこうかтонを移動させる
92         引数1 key_lst: 押下キーの真理値リスト
93         引数2 screen: 画面Surface
94         """
95         sum_mv = [0, 0]
96         speed = 20 if key_lst[pg.K_LSHIFT] else 10 # 左Shiftキーで加速 李
97         for k, mv in __class__.delta.items():
98             if key_lst[k]:
99                 sum_mv[0] += mv[0]
100                 sum_mv[1] += mv[1]

```

```

101         self.rect.move_ip(self.speed*sum_mv[0], self.speed*sum_mv[1])
102         if check_bound(self.rect) != (True, True):
103             self.rect.move_ip(-self.speed*sum_mv[0], -self.speed*sum_mv[1])
104         if not (sum_mv[0] == 0 and sum_mv[1] == 0):
105             self.dire = tuple(sum_mv)
106             self.image = self.imgs[self.dire]
107
108         if self.state == "hyper":
109             if self.hyper_life<0:
110                 self.state="normal"
111                 self.hyper_life-=1
112                 self.image = pg.transform.laplacian(self.image)
113
114         screen.blit(self.image, self.rect)
115
116         self.speed = 14 if key_lst[pg.K_LSHIFT] else 7
117
118
119     class Bomb(pg.sprite.Sprite):
120         """
121         爆弾に関するクラス
122         """
123         colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
124
125     def __init__(self, emy: "Enemy", bird: Bird):
126         """
127         爆弾円Surfaceを生成する
128         引数1 emy: 爆弾を投下する敵機
129         引数2 bird: 攻撃対象のこうかとん
130         """
131         super().__init__()
132         self.active = True #后加的添加状态标记
133         #后加的↓
134     def update(self):
135         if self.active:
136             self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
137             if check_bound(self.rect) != (True, True):
138                 self.kill()
139         #后加的↑
140
141
142         rad = random.randint(10, 50) # 爆弾円の半径: 10以上50以下の乱数
143         self.image = pg.Surface((2*rad, 2*rad))
144         color = random.choice(__class__.colors) # 爆弾円の色: クラス変数からランダム選択
145         pg.draw.circle(self.image, color, (rad, rad), rad)
146         self.image.set_colorkey((0, 0, 0))
147         self.rect = self.image.get_rect()
148         # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
149         self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
150         self.rect.centerx = emy.rect.centerx
151         self.rect.centery = emy.rect.centery+emy.rect.height//2
152         self.speed = 6
153
154     def update(self):
155         """
156         爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
157         引数 screen: 画面Surface
158         """

```

```
159         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
160         if check_bound(self.rect) != (True, True):
161             self.kill()
162
163
164     ✓ class Beam(pg.sprite.Sprite):
165         """
166         ビームに関するクラス
167         """
168     ✓ def __init__(self, bird: Bird, angle_offset: int = 0):
169         """
170         ビーム画像Surfaceを生成する
171         引数 bird: ビームを放つこうかとん
172         引数 angle_offset: ビームの角度オフセット
173         """
174         super().__init__()
175         self.vx, self.vy = bird.dire
176         angle = math.degrees(math.atan2(-self.vy, self.vx))
177         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 1.0)
178         base_angle = math.degrees(math.atan2(-self.vy, self.vx))
179         angle = base_angle + angle_offset # 角度オフセットを追加
180         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 1.0)
181         self.vx = math.cos(math.radians(angle))
182         self.vy = -math.sin(math.radians(angle))
183         self.rect = self.image.get_rect()
184         self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
185         self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
186         self.rect.centery = bird.rect.centery + bird.rect.height * self.vy
187         self.rect.centerx = bird.rect.centerx + bird.rect.width * self.vx
188         self.speed = 10
189
190     ✓ def update(self):
191         """
192         ビームを速度ベクトルself.vx, self.vyに基づき移動させる
193         引数 screen: 画面Surface
194         """
195         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
196         if check_bound(self.rect) != (True, True):
197             self.kill()
198
199
200     ✓ class NeoBeam(pg.sprite.Sprite):
201         """
202         ネオビームに関するクラス
203         """
204         def __init__(self, bird: Bird, num : int):
205             super().__init__()
206             self.bird = bird
207             self.num = num
208
209     ✓ def gen_beams(self):
210         beam_ls = []
211         for i in range(-50, +51, int (100 / (self.num - 1))) :
212             beam_ls += [i]
213         return beam_ls
214
215
216     ✓ class Explosion(pg.sprite.Sprite):
```

```

217     """
218     爆発に関するクラス
219     """
220     def __init__(self, obj: "Bomb|Enemy", life: int):
221         """
222         爆弾が爆発するエフェクトを生成する
223         引数1 obj: 爆発するBombまたは敵機インスタンス
224         引数2 life: 爆発時間
225         """
226         super().__init__()
227         img = pg.image.load(f"fig/explosion.gif")
228         self.imgs = [img, pg.transform.flip(img, 1, 1)]
229         self.image = self.imgs[0]
230         self.rect = self.image.get_rect(center=obj.rect.center)
231         self.life = life
232
233     def update(self):
234         """
235         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
236         爆発エフェクトを表現する
237         """
238         self.life -= 1
239         self.image = self.imgs[self.life//10%2]
240         if self.life < 0:
241             self.kill()
242
243     class Gravity(pg.sprite.Sprite):
244         """
245         重力に関するクラス
246         """
247     def __init__(self, life: int):
248         super().__init__()
249         self.image = pg.Surface((WIDTH, HEIGHT))
250         self.rect = self.image.get_rect()
251         self.rect.center = WIDTH//2, HEIGHT//2
252         pg.draw.rect(self.image, (0, 0, 0), (0, 0, WIDTH, HEIGHT))
253         self.image.set_alpha(128)
254         self.life = life
255
256     def update(self):
257         self.life -= 1
258         if self.life < 0:
259             self.kill()
260
261
262
263     # 追加機能3: 電磁パルス (EMP) 李
264     class EMP(pg.sprite.Sprite):
265     def __init__(self, emys: pg.sprite.Group, bombs: pg.sprite.Group, screen : pg.Surface):
266         super().__init__()
267         self.emys = emys
268         self.bombs = bombs
269         self.screen = screen
270         self.life = 30 #0.5秒
271
272         #创建黄色半透明矩形
273         self.image = pg.Surface((WIDTH, HEIGHT))
274         self.image.fill((255, 255, 0))#黄色

```

```

275         self.image.set_alpha(128)#半透明
276         self.rect = self.image.get_rect()
277         # 立即调用无效化方法
278         self.deactivate_enemies()
279         self.deactivate_bombs()
280
281
282     def deactivate_enemies(self):
283         # Enemyインスタンスを無効化する, Bombインスタンスを無効化する
284         # 敌机无效化, 禁止投弹
285         for emy in self.emys:
286             emy.interval = float("inf") #停止投弹
287             emy.image = pg.transform.laplacian(emy.image)#滤镜效果
288             emy.disable = True #敌机状态为无效化
289             emy.speed = 0 #敌机速度为0
290             emy.state = "disable" #敌机状态为无效化
291     def deactivate_bombs(self):
292         #炸弹无效化
293         for bomb in self.bombs:
294             bomb.speed = 0 #完全停止
295             bomb.active = False #炸弹状态为无效化
296
297     def update(self):
298         #EMP持续时间
299         self.life -= 1
300         if self.life < 0:
301             self.kill() #持续时间结合后删除EMP效果
302
303     #以上是EMP的实现
304
305
306
307     class Enemy(pg.sprite.Sprite):
308         """
309         敵機に関するクラス
310         """
311         imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
312
313     def __init__(self):
314         super().__init__()
315         self.disable = False #后加的
316         self.image = pg.transform.rotozoom(random.choice(__class__.imgs), 0, 0.8)
317         self.rect = self.image.get_rect()
318         self.rect.center = random.randint(0, WIDTH), 0
319         self.vx, self.vy = 0, +6
320         self.bound = random.randint(50, HEIGHT//2) # 停止位置
321         self.state = "down" # 降下状态or停止状态
322         self.interval = random.randint(50, 300) # 爆弹投下インターバル
323
324     def update(self):
325         """
326         敵機を速度ベクトルself.vyに基づき移動(降下)させる
327         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
328         引数 screen: 画面Surface
329         """
330         if not self.disable: #后加的
331             if self.rect.centery > self.bound:
332                 self.vy = 0

```

```
333         self.state = "stop"
334         self.rect.move_ip(self.vx, self.vy)
335
336
337     class Score:
338         """
339         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
340         爆弾: 1点
341         敵機: 10点
342         """
343     def __init__(self):
344         self.font = pg.font.Font(None, 50)
345         self.color = (0, 0, 255)
346         self.value = 0
347         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
348         self.rect = self.image.get_rect()
349         self.rect.center = 100, HEIGHT-50
350
351     def update(self, screen: pg.Surface):
352         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
353         screen.blit(self.image, self.rect)
354
355
356     class Shield(pg.sprite.Sprite):
357         """
358         こうかとの防御シールド
359         """
360     def __init__(self, bird: Bird, life: int):
361         super().__init__()
362         self.bird = bird
363         self.life = life
364
365         height = self.bird.rect.height * 2
366         self.original_image = pg.Surface((20, height), flags=pg.SRCALPHA)
367         pg.draw.rect(self.original_image, (0, 0, 255), (0, 0, 20, height))
368
369         self.image = self.original_image
370         self.rect = self.image.get_rect(center=self.bird.rect.center)
371
372
373     def update(self):
374         self.life -= 1
375         if self.life < 0:
376             self.kill()
377
378         vx, vy = self.bird.dire
379         angle = math.degrees(math.atan2(-vy, vx))
380
381         self.image = pg.transform.rotozoom(self.original_image, angle, 0.9)
382         self.rect = self.image.get_rect(center=self.bird.rect.center)
383
384         self.rect.centerx += vx * self.bird.rect.width
385         self.rect.centery += vy * self.bird.rect.height
386
387
388
389     def main():
390         pg.display.set_caption("真! こうかтон無双")
```

```

391     screen = pg.display.set_mode((WIDTH, HEIGHT))
392     bg_img = pg.image.load(f"fig/pg_bg.jpg")
393     score = Score()
394     emps = pg.sprite.Group() # EMP后加的
395     bird = Bird(3, (900, 400))
396     bombs = pg.sprite.Group()
397     beams = pg.sprite.Group()
398     exps = pg.sprite.Group()
399     emys = pg.sprite.Group()
400     shield = pg.sprite.Group()
401     gravity = pg.sprite.Group()
402
403     tmr = 0
404     clock = pg.time.Clock()
405     while True:
406         key_lst = pg.key.get_pressed()
407         for event in pg.event.get():
408             if event.type == pg.QUIT:
409                 return 0
410             if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
411                 beams.add(Beam(bird))
412             if event.type == pg.KEYDOWN and event.key == pg.K_s and score.value >= 50 and len(shield):
413                 shield.add(Shield(bird, 400)) # 防御壁を発動(400フレーム)
414                 score.value -= 50 # スコアを50消費
415             if event.type == pg.KEYDOWN and event.key == pg.K_RSHIFT:
416                 if score.value > 100:
417                     score.value -= 100
418                     bird.state = "hyper"
419                     bird.hyper_life = 500
420
421             if event.type == pg.KEYDOWN and event.key == pg.K_RETURN:
422                 if score.value >= 200:
423                     score.value -= 200
424                     gravity.add(Gravity(400))
425             if event.type == pg.KEYDOWN and key_lst[pg.K_LSHIFT] and key_lst[pg.K_SPACE]:
426                 neobeam = NeoBeam(bird, 5)
427                 neobeam_ls = neobeam.gen_beams()
428                 for i in neobeam_ls:
429                     beams.add(Beam(bird, i))
430
431             elif event.type == pg.KEYDOWN and event.key == pg.K_e and score.value >= 20: # EMP判断是否可以
432                 emps.add(EMP(emys, bombs, screen))
433                 score.value -= 20 # 消耗分数
434             screen.blit(bg_img, [0, 0])
435
436             if tmr % 200 == 0: # 200フレームに1回, 敵機を出現させる
437                 emys.add(Enemy())
438
439             for emy in emys:
440                 if emy.state == "stop" and tmr % emy.interval == 0:
441                     # 敵機が停止状態に入ったら, intervalに応じて爆弾投下
442                     bombs.add(Bomb(emy, bird))
443
444             for emy in pg.sprite.groupcollide(emys, beams, True, True).keys(): # ビームと衝突した敵機リスト
445                 exps.add(Explosion(emy, 100)) # 爆発エフェクト
446                 score.value += 10 # 10点アップ
447                 bird.change_img(6, screen) # こうかたん喜びエフェクト
448

```



```
449     for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys(): # ビームと衝突した爆弾リス
450         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
451         score.value += 1 # 1点アップ
452
453     for emy in pg.sprite.groupcollide(emy, gravity, True, False).keys(): # 重力と衝突した敵機リス
454         exps.add(Explosion(emy, 100)) # 爆発エフェクト
455         score.value += 10 # 10点アップ
456         bird.change_img(6, screen) # こうかтон喜びエフェクト
457     for bomb in pg.sprite.groupcollide(bombs, gravity, True, False).keys(): # 重力と衝突した爆弾リス
458         exps.add(Explosion(bomb, 50)) # 爆発エフェクト score.value += 1 # 1点アップ
459         bird.change_img(6, screen) # こうかтон喜びエフェクト
460
461     for bomb in pg.sprite.spritecollide(bird, bombs, True): # こうかтонと衝突した爆弾リスト
462         if bird.state == "hyper":
463             score.value += 1
464             break
465         bird.change_img(8, screen) # こうかтон悲しみエフェクト
466         score.update(screen)
467         pg.display.update()
468         time.sleep(2)
469         return
470
471     for shields in shield:
472         for bomb in pg.sprite.spritecollide(shields, bombs, True): # 防御壁と衝突した爆弾を削除
473             exps.add(Explosion(bomb, 50)) # 爆発エフェクト
474             score.value += 1 # スコアを1加算
475
476     bird.update(key_lst, screen)
477     beams.update()
478     beams.draw(screen)
479     emys.update()
480     emys.draw(screen)
481     emps.update() # EMP后加的
482     emps.draw(screen) # 绘制EMP效果 ,后加的
483     bombs.update()
484     bombs.draw(screen)
485     exps.update()
486     exps.draw(screen)
487     shield.update()
488     shield.draw(screen)
489     gravity.update()
490     gravity.draw(screen)
491     score.update(screen)
492     pg.display.update()
493     tmr += 1
494     clock.tick(50)
495
496
497 if __name__ == "__main__":
498     pg.init()
499     main()
500     pg.quit()
501     sys.exit()
```