

ShogoAkai / ProjExD\_04

🔍

📁

👤

<> Code

🕒 Issues

🔗 Pull requests 2

🔄 Actions

📁 Projects

📖 Wiki

🛡 Security

⋮

- Overview
- Yours
- Active
- Stale
- All
- New branch

🔍 Search branches...

Default branch

main

🔄 Updated 2 minutes ago by ShogoAkai

Your branches

C0A22043/feature5

🔄 Updated 1 hour ago by c0a22043

Active branches

C0A22035/speed

🔄 Updated 44 minutes ago by ShogoAkai

C0A22047/feature3

🔄 Updated 46 minutes ago by ShogoAkai

C0A22003/feature2

🔄 Updated 50 minutes ago by ShogoAkai

C0A22075/feature6

🔄 Updated 1 hour ago by c0a2207576

C0A22043/feature5

🔄 Updated 1 hour ago by c0a22043

View more active branches >

```

1 import math
2 import random
3 import sys
4 import time
5
6 import pygame as pg
7
8
9 WIDTH = 1600 # ゲームウィンドウの幅
10 HEIGHT = 900 # ゲームウィンドウの高さ
11
12
13 def check_bound(obj: pg.Rect) -> tuple[bool, bool]:
14     """
15     オブジェクトが画面内か画面外かを判定し、真値タプルを返す
16     引数 obj : オブジェクト（爆弾，こうかとん，ビーム）SurfaceのRect
17     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）
18     """
19     yoko, tate = True, True
20     if obj.left < 0 or WIDTH < obj.right: # 横方向のはみ出し判定
21         yoko = False
22     if obj.top < 0 or HEIGHT < obj.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
39 class Bird(pg.sprite.Sprite):
40     """
41     どうぶつ屋のカキ（こうかとん）に関するクラス

```

```
41     ゲームキャラクター（こうかとん）に関するツプス
42     """
43     ✓ delta = { # 押下キーと移動量の辞書
44         pg.K_UP: (0, -1),
45         pg.K_DOWN: (0, +1),
46         pg.K_LEFT: (-1, 0),
47         pg.K_RIGHT: (+1, 0),
48     }
49
50     ✓ def __init__(self, num: int, xy: tuple[int, int]):
51         """
52         こうかとん画像Surfaceを生成する
53         引数1 num: こうかとん画像ファイル名の番号
54         引数2 xy: こうかとん画像の位置座標タプル
55         """
56         super().__init__()
57         img0 = pg.transform.rotozoom(pg.image.load(f"ex04/fig/{num}.png"), 0, 2.0)
58         img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
59         self.imgs = {
60             (+1, 0): img, # 右
61             (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
62             (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
63             (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
64             (-1, 0): img0, # 左
65             (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
66             (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
67             (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
68         }
69         self.dire = (+1, 0)
70         self.image = self.imgs[self.dire]
71         self.rect = self.image.get_rect()
72         self.rect.center = xy
73         self.speed = 10
74
75
76
77
78         self.state = "normal" #追加機能4
79         self.hyper_life = -1
80
81
82
83
84     ✓ def change_img(self, num: int, screen: pg.Surface):
85         """
86         こうかとん画像を切り替え、画面に転送する
87         引数1 num: こうかとん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"ex04/fig/{num}.png"), 0, 2.0)
91         screen.blit(self.image, self.rect)
92
93
94
95
96
97     ✓ def change_state(self, state: str, hyper_life: int):
```

```

98         """
99         こうかとんの状態を変更する
100     Args:
101         state (str): 新しい状態 ("normal" または "hyper")
102         hyper_life (int): "hyper" 状態の発動時間 (フレーム数)
103     """
104     self.state = state
105     self.hyper_life = hyper_life
106
107
108
109
110 ✓ def update(self, key_lst: list[bool], screen: pg.Surface):
111     """
112     押下キーに応じてこうかとんを移動させる
113     引数1 key_lst: 押下キーの真理値リスト
114     引数2 screen: 画面Surface
115     """
116     sum_mv = [0, 0]
117     for k, mv in __class__.delta.items():
118         if key_lst[k]:
119             self.rect.move_ip(+self.speed*mv[0], +self.speed*mv[1])
120             sum_mv[0] += mv[0]
121             sum_mv[1] += mv[1]
122     if check_bound(self.rect) != (True, True):
123         for k, mv in __class__.delta.items():
124             if key_lst[k]:
125                 self.rect.move_ip(-self.speed*mv[0], -self.speed*mv[1])
126     if not (sum_mv[0] == 0 and sum_mv[1] == 0):
127         self.dire = tuple(sum_mv)
128         self.image = self.imgs[self.dire]
129     screen.blit(self.image, self.rect)
130
131
132
133     if self.state == "hyper": #追加機能4
134         self.hyper_life -= 1
135         if self.hyper_life < 0:
136             self.change_state("normal", -1)
137         else:
138             # 画像を変換したものに切り替える
139             self.image = pg.transform.laplacian(self.image)
140
141
142
143
144
145     def get_direction(self) -> tuple[int, int]:
146         return self.dire
147
148
149 ✓ class Bomb(pg.sprite.Sprite):
150     """
151     爆弾に関するクラス
152     """
153     colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]

```

```

154
155 ✓ def __init__(self, emy: "Enemy", bird: Bird):
156     """
157     爆弾円Surfaceを生成する
158     引数1 emy : 爆弾を投下する敵機
159     引数2 bird : 攻撃対象のこうかとん
160     """
161     super().__init__()
162     rad = random.randint(10, 50) # 爆弾円の半径 : 10以上50以下の乱数
163     color = random.choice(__class__.colors) # 爆弾円の色 : クラス変数からランダム選択
164     self.image = pg.Surface((2*rad, 2*rad))
165     pg.draw.circle(self.image, color, (rad, rad), rad)
166     self.image.set_colorkey((0, 0, 0))
167     self.rect = self.image.get_rect()
168     # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
169     self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
170     self.rect.centerx = emy.rect.centerx
171     self.rect.centery = emy.rect.centery+emy.rect.height/2
172     self.speed = 6
173
174 ✓ def update(self):
175     """
176     爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
177     引数 screen : 画面Surface
178     """
179     self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
180     if check_bound(self.rect) != (True, True):
181         self.kill()
182
183
184 ✓ class Beam(pg.sprite.Sprite):
185     """
186     ビームに関するクラス
187     """
188 ✓ def __init__(self, bird: Bird):
189     """
190     ビーム画像Surfaceを生成する
191     引数 bird : ビームを放つこうかとん
192     """
193     super().__init__()
194     self.vx, self.vy = bird.get_direction()
195     angle = math.degrees(math.atan2(-self.vy, self.vx))
196     self.image = pg.transform.rotozoom(pg.image.load(f'ex04/fig/beam.png'), angle, 2.0)
197     self.vx = math.cos(math.radians(angle))
198     self.vy = -math.sin(math.radians(angle))
199     self.rect = self.image.get_rect()
200     self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
201     self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
202     self.speed = 10
203
204 ✓ def update(self):
205     """
206     ビームを速度ベクトルself.vx, self.vyに基づき移動させる
207     引数 screen : 画面Surface
208     """
209     self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
210     if check_bound(self.rect) != (True, True):

```

```

211         self.kill()
212
213
214     class Explosion(pg.sprite.Sprite):
215         """
216         爆発に関するクラス
217         """
218     def __init__(self, obj: "Bomb|Enemy", life: int):
219         """
220         爆弾が爆発するエフェクトを生成する
221         引数1 obj: 爆発するBombまたは敵機インスタンス
222         引数2 life: 爆発時間
223         """
224         super().__init__()
225         img = pg.image.load("ex04/fig/explosion.gif")
226         self.imgs = [img, pg.transform.flip(img, 1, 1)]
227         self.image = self.imgs[0]
228         self.rect = self.image.get_rect(center=obj.rect.center)
229         self.life = life
230
231     def update(self):
232         """
233         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
234         爆発エフェクトを表現する
235         """
236         self.life -= 1
237         self.image = self.imgs[self.life//10%2]
238         if self.life < 0:
239             self.kill()
240
241
242     class Enemy(pg.sprite.Sprite):
243         """
244         敵機に関するクラス
245         """
246         imgs = [pg.image.load(f"ex04/fig/alien{i}.png") for i in range(1, 4)]
247
248     def __init__(self):
249         super().__init__()
250         self.image = random.choice(__class__.imgs)
251         self.rect = self.image.get_rect()
252         self.rect.center = random.randint(0, WIDTH), 0
253         self.vy = +6
254         self.bound = random.randint(50, HEIGHT/2) # 停止位置
255         self.state = "down" # 降下状態or停止状態
256         self.interval = random.randint(50, 300) # 爆弾投下インターバル
257
258     def update(self):
259         """
260         敵機を速度ベクトルself.vyに基づき移動（降下）させる
261         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
262         引数 screen: 画面Surface
263         """
264         if self.rect.centery > self.bound:
265             self.vy = 0
266             self.state = "stop"
267

```

```
267         self.rect.centery += self.vy
268
269
270 ✓ class Score:
271     """
272     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
273     爆弾: 1点
274     敵機: 10点
275     """
276 ✓ def __init__(self):
277     self.font = pg.font.Font(None, 50)
278     self.color = (0, 0, 255)
279     self.score = 0
280     self.image = self.font.render(f"Score: {self.score}", 0, self.color)
281     self.rect = self.image.get_rect()
282     self.rect.center = 100, HEIGHT-50
283
284     def score_up(self, add):
285         self.score += add
286
287
288
289
290     def score_down(self, add):
291         self.score -= add
292
293
294
295
296     def update(self, screen: pg.Surface):
297         self.image = self.font.render(f"Score: {self.score}", 0, self.color)
298         screen.blit(self.image, self.rect)
299
300
301
302 ✓ class Gravity(pg.sprite.Sprite):
303 ✓     def __init__(self, bird, size, life):
304         super().__init__()
305         self.image = pg.Surface((2 * size, 2 * size))
306         pg.draw.circle(self.image, (10, 10, 10), (size, size), size)
307         self.image.set_alpha(200)
308         self.image.set_colorkey((0, 0, 0))
309         self.rect = self.image.get_rect()
310         self.rect.center = bird.rect.center
311         self.life = life
312
313     def update(self):
314         self.life -= 1
315         if self.life < 0:
316             self.kill()
317
318
319
320
321 ✓ class NeoGravity(pg.sprite.Sprite):
322     """
323     重力場に関するクラス
```



main ▾

ProjExD\_04 / musou\_kokaton.py

↑ Top

Code

Blame

Raw



```
321     class NeoGravity(pg.sprite.Sprite):
322     def __init__(self, life):
323         self.image.set_colorkey((0, 0, 0))
331         pg.draw.rect(self.image,(10, 10, 10), pg.Rect(0, 0, WIDTH, HEIGHT))
332         self.rect = self.image.get_rect()
333         self.rect.center = WIDTH/2, HEIGHT/2
334
335     def update(self):
336         """
337         発動時間を1限算し、発動時間中は重力場を無効にする
338         """
339
340         self.life -= 1
341         if self.life < 0:
342             self.kill()
343
344
345     def main():
346         pg.display.set_caption("真！こうかとん無双")
347         screen = pg.display.set_mode((WIDTH, HEIGHT))
348         bg_img = pg.image.load("ex04/fig/pg_bg.jpg")
349         score = Score()
350
351         bird = Bird(3, (900, 400))
352         bombs = pg.sprite.Group()
353         beams = pg.sprite.Group()
354         exps = pg.sprite.Group()
355         emys = pg.sprite.Group()
356
357
358         gras = pg.sprite.Group()
359
360
361         neos = pg.sprite.Group()
362
363
364
365
366         tmr = 0
367         clock = pg.time.Clock()
368         while True:
369             key_lst = pg.key.get_pressed()
370             for event in pg.event.get():
371                 if event.type == pg.QUIT:
372                     return 0
373                 if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
374                     beams.add(Beam(bird))
375
376                 if event.type == pg.KEYDOWN and event.key == pg.K_LSHIFT:
377                     bird.speed = 20
378                 if event.type == pg.KEYUP and event.key == pg.K_LSHIFT:
379                     bird.speed = 10
380
```



```
380
381
382     if event.type == pg.KEYDOWN and event.key == pg.K_TAB:
383         if score.score >= 50: #あとで50に変わる
384             gras.add(Gravity(bird, 200, 500))
385             score.score_up(-50)
386
387
388     if event.type == pg.KEYDOWN and event.key == pg.K_RETURN:
389         if score.score > -100: # :スコアを200に変更
390             neos.add(NeoGravity(400))
391             score.score -= 200
392
393
394     # 右Shiftキーが押され、スコアが10より大の場合に "hyper" 状態にする
395     if key_lst[pg.K_RSHIFT] and score.score >= 10:
396         bird.change_state("hyper", 500)
397         score.score_down(10)
398
399
400
401     screen.blit(bg_img, [0, 0])
402
403     if tmr%200 == 0: # 200フレームに1回, 敵機を出現させる
404         emys.add(Enemy())
405
406     for emy in emys:
407         if emy.state == "stop" and tmr%emy.interval == 0:
408             # 敵機が停止状態に入ったら, intervalに応じて爆弾投下
409             bombs.add(Bomb(emy, bird))
410
411     for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
412         exps.add(Explosion(emy, 100)) # 爆発エフェクト
413
414         score.score_up(10) # 10点アップ
415
416
417         score.score_up(10) # 10点アップ
418
419
420         score.score_up(100) # 10点アップ
421
422         score.score_up(10) # 10点アップ
423
424
425
426         bird.change_img(6, screen) # こうかとい喜びエフェクト
427
428     for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
429         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
430
431         score.score_up(1) # 1点アップ
432
433
434
435         score.score_up(1) # 1点アップ
436
```

```
437     for gra in pg.sprite.groupcollide(bombs, gras, True, False).keys():
438         exps.add(Explosion(gra, 50)) # 爆発エフェクト
439
440
441         score.score_up(100) # 1点アップ
442     for bomb in pg.sprite.groupcollide(emics, neos, True, False).keys():
443         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
444         score.score_up(1) # 1点アップ
445
446     for emy in pg.sprite.groupcollide(emics, neos, True, False).keys():
447         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
448         score.score_up(1) # 1点アップ
449
450
451         score.score_up(1) # 1点アップ
452
453     if bird.state == "hyper":
454         for bomb in pg.sprite.spritecollide(bird, bombs, True):
455             exps.add(Explosion(bomb, 50)) # 爆発エフェクト
456             score.score_up(1) # 1点アップ
457
458
459
460     if len(pg.sprite.spritecollide(bird, bombs, True)) != 0:
461         bird.change_img(8, screen) # こうかとん悲しみエフェクト
462         score.update(screen)
463         pg.display.update()
464         time.sleep(2)
465         return
466
467     bird.update(key_lst, screen)
468     beams.update()
469     beams.draw(screen)
470     emys.update()
471     emys.draw(screen)
472     bombs.update()
473     bombs.draw(screen)
474     exps.update()
475     exps.draw(screen)
476
477     gras.update()
478     gras.draw(screen)
479
480
481     neos.update()
482     neos.draw(screen)
483
484
485
486     score.update(screen)
487     pg.display.update()
488     tmr += 1
489     clock.tick(50)
490
491
492     if __name__ == "__main__":
```

```
493     pg.init()  
494     main()  
495     pg.quit()  
496     sys.exit()
```