



c0b23078ca /  
ProjExD\_Group03



<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights



ProjExD\_Group03 / README.md



c0b23078ca 追加機能の説明文追加

2863c9b · 1 minute ago



37 lines (31 loc) · 901 Bytes

Preview

Code

Blame

Raw



# こうかとんシューティング(仮)

## 実行環境の必要条件

- python >= 3.10
- pygame >= 2.1

## ゲームの概要

縦か横スクロールゲーム

## ゲームの遊び方

- 十字キーで移動
- 特定キーでビームを放つ
- 敵を倒してスコアを稼ぐ
- 残機が0になったらゲームオーバー
- スコアを消費して特殊機能が発動可能
- 一定スコアでボスが出現

## ゲームの実装

### 共通基本機能

- 背景画像とキャラクターの描画
- スコア表示
- 弾数表示
- 横スクロール

### 担当追加機能

- 残機(くどう)

- 障害物(みき)
- アイテム(ひらなべ)
- 敵のレベルアップ(しのはら):スコアの合計値によって敵機から投下される爆弾の大きさと速度が変化する。
- クリア画面,ラスボス作成(きざき)

## ToDo

- []
- []

## メモ

- 
-

c0b23078ca /  
ProjExD\_Group03[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [ProjExD\\_Group03](#) / [musou\\_kokaton.py](#) 

c0b23078ca スコアの合計値による爆弾の変化

5334d42 · 5 minutes ago



521 lines (459 loc) · 18.8 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
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
12 os.chdir(os.path.dirname(os.path.abspath(__file__)))
13
14
15 def check_bound(obj_rct: pg.Rect) -> tuple[bool, bool]:
16     """
17     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
18     引数：こうかとんや爆弾、ビームなどのRect
19     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）
20     """
21     yoko, tate = True, True
22     if obj_rct.left < 0 or WIDTH < obj_rct.right:
23         yoko = False
24     if obj_rct.top < 0 or HEIGHT < obj_rct.bottom:
25         tate = False
26     return yoko, tate
27
28
29 def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
30     """
31     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
32     引数1 org：爆弾SurfaceのRect
33     引数2 dst：こうかとんSurfaceのRect
34     戻り値：orgから見たdstの方向ベクトルを表すタプル
35     """
36     x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
37     norm = math.sqrt(x_diff**2+y_diff**2)
38     return x_diff/norm, y_diff/norm
39
40
41 class Bird(pg.sprite.Sprite):
42     """
43     ゲームキャラクター（こうかとん）に関するクラス
44     """
```

```
45  ✓ delta = { # 押下キーと移動量の辞書
46          pg.K_UP: (0, -1),
47          pg.K_DOWN: (0, +1),
48          pg.K_LEFT: (-1, 0),
49          pg.K_RIGHT: (+1, 0),
50      }
51
52  ✓ def __init__(self, num: int, xy: tuple[int, int]):
53      """
54      こうかどん画像Surfaceを生成する
55      引数1 num: こうかどん画像ファイル名の番号
56      引数2 xy: こうかどん画像の位置座標タプル
57      """
58      super().__init__()
59      img0 = pg.transform.rotozoom(pg.image.load(f"fig/3.png"), 0, 1.5)
60      img = pg.transform.flip(img0, True, False) # デフォルトのこうかどん
61      simg0 = pg.transform.rotozoom(pg.image.load(f"fig/0.png"), 0, 1.5)
62      simg = pg.transform.flip(simg0, True, False)
63      self.imgs = {
64          (+1, 0): img, # 右
65          (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
66          (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
67          (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
68          (-1, 0): img0, # 左
69          (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
70          (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
71          (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
72      }
73      self.simgs = {
74          (+1, 0): simg, # 右
75          (+1, -1): simg, # 右上
76          (0, -1): simg, # 上
77          (-1, -1): simg0, # 左上
78          (-1, 0): simg0, # 左
79          (-1, +1): simg0, # 左下
80          (0, +1): simg0, # 下
81          (+1, +1): simg, # 右下
82      }
83      self.dire = (+1, 0)
84      self.image = self.imgs[self.dire]
85      self.rect = self.image.get_rect()
86      self.rect.center = xy
87      self.simage = self.simgs[self.dire]
88      self.srect = self.simage.get_rect()
89      self.srect.center = self.rect.center
90      self.speed = 10
91      self.mode = 0
92
93  ✓ def change_img(self, num: int, screen: pg.Surface):
94      """
95      こうかどん画像を切り替え、画面に転送する
96      引数1 num: こうかどん画像ファイル名の番号
97      引数2 screen: 画面Surface
98      """
99      self.image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 1.5)
100      screen.blit(self.image, self.rect)
101
102  ✓ def update(self, key_lst: list[bool], screen: pg.Surface):
103      """
104      押下キーに応じてこうかどんを移動させる
```

```

105 引数1 key_lst : 押下キーの真理値リスト
106 引数2 screen : 画面Surface
107 """
108 sum_mv = [0, 0]
109 for k, mv in __class__.delta.items():
110     if key_lst[k]:
111         sum_mv[0] += mv[0]
112         sum_mv[1] += mv[1]
113 self.rect.move_ip(self.speed*sum_mv[0], self.speed*sum_mv[1])
114 if check_bound(self.rect) != (True, True):
115     self.rect.move_ip(-self.speed*sum_mv[0], -self.speed*sum_mv[1])
116 if not (sum_mv[0] == 0 and sum_mv[1] == 0):
117     self.dire = tuple(sum_mv)
118     self.image = self.imgs[self.dire]
119 screen.blit(self.image, self.rect)
120 if self.mode == 1:
121     self.srect.center = self.rect.center
122     if not (sum_mv[0] == 0 and sum_mv[1] == 0):
123         self.simage = self.simgs[self.dire]
124         screen.blit(self.simage, self.srect)
125
126
127  class Bomb(pg.sprite.Sprite):
128     """
129     爆弾に関するクラス
130     """
131     colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
132
133  def __init__(self, emy: "Enemy", bird: Bird, tscore):
134     """
135     爆弾円Surfaceを生成する
136     引数1 emy : 爆弾を投下する敵機
137     引数2 bird : 攻撃対象のこうかとん
138     """
139     super().__init__()
140     if tscore <= 50 : # スコアの合計が50点以下
141         rad = random.randint(10, 10) # 爆弾円の半径 : 10の乱数
142         self.speed = 2 # 爆弾の投下速度2
143     elif 51 <= tscore <= 100 : # スコアの合計が51点以上100点以下
144         rad = random.randint(10, 20) # 爆弾円の半径 : 10以上20以下の乱数
145         self.speed = 3 # 爆弾の投下速度3
146     elif 101 <= tscore <= 150 : # スコアの合計が101点以上150点以下
147         rad = random.randint(10, 30) # 爆弾円の半径 : 10以上30以下の乱数
148         self.speed = 4 # 爆弾の投下速度4
149     elif 151 <= tscore <= 200 : # スコアの合計が151点以上200点以下
150         rad = random.randint(10, 40) # 爆弾円の半径 : 10以上40以下の乱数
151         self.speed = 5 # 爆弾の投下速度5
152     elif 201 <= tscore <= 250 : # スコアの合計が201点以上250点以下
153         rad = random.randint(10, 50) # 爆弾円の半径 : 10以上50以下の乱数
154         self.speed = 6 # 爆弾の投下速度6
155     self.image = pg.Surface((2*rad, 2*rad))
156     color = random.choice(__class__.colors) # 爆弾円の色 : クラス変数からランダム選択
157     pg.draw.circle(self.image, color, (rad, rad), rad)
158     self.image.set_colorkey((0, 0, 0))
159     self.rect = self.image.get_rect()
160     # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
161     self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
162     self.rect.centerx = emy.rect.centerx
163     self.rect.centery = emy.rect.centery+emy.rect.height//2
164

```

```
165 ✓ def update(self):
166     """
167     爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
168     引数 screen : 画面Surface
169     """
170     self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
171     if check_bound(self.rect) != (True, True):
172         self.kill()
173
174
175 ✓ class Beam(pg.sprite.Sprite):
176     """
177     ビームに関するクラス
178     """
179 ✓ def __init__(self, bird: Bird, angle0=0, mode=0):
180     """
181     ビーム画像Surfaceを生成する
182     引数 bird : ビームを放つこうかとん
183     """
184     super().__init__()
185     self.vx, self.vy = bird.dire
186     angle = math.degrees(math.atan2(-self.vy, self.vx))
187     angle+=angle0
188     if mode == 0:
189         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 1.0)
190     elif mode == 1:
191         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 1.0)
192     self.vx = math.cos(math.radians(angle))
193     self.vy = -math.sin(math.radians(angle))
194     self.rect = self.image.get_rect()
195     self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
196     self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
197     self.speed = 15
198     self.count = 0
199
200 ✓ def update(self):
201     """
202     ビームを速度ベクトルself.vx, self.vyに基づき移動させる
203     引数 screen : 画面Surface
204     """
205     self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
206     if check_bound(self.rect) != (True, True):
207         self.count += 1
208         if self.count == 15:
209             self.kill()
210
211
212 ✓ class Explosion(pg.sprite.Sprite):
213     """
214     爆発に関するクラス
215     """
216 ✓ def __init__(self, obj: "Bomb|Enemy", life: int):
217     """
218     爆弾が爆発するエフェクトを生成する
219     引数1 obj : 爆発するBombまたは敵機インスタンス
220     引数2 life : 爆発時間
221     """
222     super().__init__()
223     img = pg.image.load(f"fig/explosion.gif")
224     self.imgs = [img, pg.transform.flip(img, 1, 1)]
```

```
225     self.image = self.imgs[0]
226     self.rect = self.image.get_rect(center=obj.rect.center)
227     self.life = life
228
229     def update(self):
230         """
231         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
232         爆発エフェクトを表現する
233         """
234         self.life -= 1
235         self.image = self.imgs[self.life//10%2]
236         if self.life < 0:
237             self.kill()
238
239
240     class Enemy(pg.sprite.Sprite):
241         """
242         敵機に関するクラス
243         """
244         imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
245
246         def __init__(self):
247             super().__init__()
248             self.image = random.choice(__class__.imgs)
249             self.rect = self.image.get_rect()
250             self.rect.center = random.randint(0, WIDTH), 0
251             self.vx, self.vy = 0, +6
252             self.bound = random.randint(50, HEIGHT//2) # 停止位置
253             self.state = "down" # 降下状態or停止状態
254             self.interval = random.randint(50, 300) # 爆弾投下インターバル
255
256         def update(self):
257             """
258             敵機を速度ベクトルself.vyに基づき移動（降下）させる
259             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
260             引数 screen : 画面Surface
261             """
262             if self.rect.centery > self.bound:
263                 self.vy = 0
264                 self.state = "stop"
265             self.rect.move_ip(self.vx, self.vy)
266
267
268     class NeoBeam(pg.sprite.Sprite):
269
270         def __init__(self, bird: Bird, num):
271             super().__init__()
272             self.bird = bird
273             self.num = num
274
275         def gen_beams(self):
276             return [Beam(self.bird, angle0, 1) for angle0 in range(-50, +51, 100//(self.num-1))]
277
278
279     class Score:
280         """
281         打ち落とす爆弾、敵機の数スコアとして表示するクラス
282         爆弾 : 1点
283         敵機 : 10点
284         """
```

```
285  ✓ def __init__(self):
286         self.font = pg.font.Font(None, 50)
287         self.color = (250, 20, 20)
288         self.value = 0
289         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
290         self.rect = self.image.get_rect()
291         self.rect.center = 100, HEIGHT-50
292
293     def update(self, screen: pg.Surface):
294         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
295         screen.blit(self.image, self.rect)
296
297  ✓ class Bullet(pg.sprite.Sprite):
298     """
299     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
300     爆弾: 1点
301     敵機: 10点
302     """
303  ✓ def __init__(self, bullet :pg.Surface, step=1, value=1, mvalue=1, mct=0, ns=0):
304         self.font = pg.font.Font(None, 60)
305         self.color = (10, 10, 10)
306         self.mct = mct
307         self.ct = 0
308         self.step = step
309         self.ns = ns
310         self.mode = 0
311
312         self.value = value
313         self.mvalue = mvalue
314         self.bimg = bullet
315         self.back = pg.Surface((240,60))
316         pg.draw.rect(self.back, (192,192,192), (0,0,140,60))
317         pg.draw.rect(self.back, (100,255,100), (140,0,240,60))
318         self.back.set_alpha(128)
319         self.image = self.font.render(f"{self.value}", 0, self.color)
320         self.brct = self.bimg.get_rect()
321         self.bact = self.back.get_rect()
322         self.rect = self.image.get_rect()
323         self.brct.center = WIDTH-174, HEIGHT-(60*step) + 30
324         self.bact.center = WIDTH-120, HEIGHT-(60*step) + 30
325         self.rect.center = WIDTH-75, HEIGHT-(60*step) + 30
326
327  ✓ def update(self, screen: pg.Surface, score):
328         if self.value == 0:
329             self.color = (255, 10, 10)
330             if self.ct > 0:
331                 self.ct -= 1
332         else:
333             self.color = (10, 10, 10)
334             if self.ct == 0 and self.ns <= score:
335                 self.value = self.mvalue
336             if self.mode == 1:
337                 self.value -= 1
338                 if self.value == 0:
339                     self.mode = 0
340             self.image = self.font.render(f"{self.value}", 0, self.color)
341             screen.blit(self.back, self.bact)
342             screen.blit(self.bimg, self.brct)
343             screen.blit(self.image, self.rect)
344
```



```
345
346 ✓ class Shield(pg.sprite.Sprite):
347     """
348     敵の攻撃を防ぐシールドを展開する
349     """
350 ✓ def __init__(self, bird: Bird, life:int):
351     """
352     シールドの基本情報
353     """
354     super().__init__()
355     bird.mode = 1
356     self.vx, self.vy = bird.dire
357     img = pg.Surface((bird.rect.height*1.8, bird.rect.height*1.8))
358     pg.draw.circle(img, (50, 255, 50), (bird.rect.height*0.9, bird.rect.height*0.9), bird.rect.height*0.9)
359     self.image = img
360     self.image.set_alpha(100)
361     self.image.set_colorkey((0, 0, 0))
362     self.rect = self.image.get_rect()
363     self.rect.centery = bird.rect.centery
364     self.rect.centerx = bird.rect.centerx
365     self.life = life
366
367 ✓ def update(self, bird: Bird, key_lst: list[bool]):
368     """
369     shieldをこうかとの位置に基づき移動させる
370     引数 screen : 画面Surface
371     """
372     self.rect.center = bird.rect.center
373     self.life -= 1
374     if self.life == 0:
375         bird.mode = 0
376         self.kill()
377
378
379 ✓ class Gravity(pg.sprite.Sprite):
380     """
381     重力場を発生させるクラス
382     """
383 ✓ def __init__(self):
384     super().__init__()
385     self.image = pg.Surface((WIDTH, HEIGHT))
386     self.life = 400
387     pg.draw.rect(self.image, (100, 255, 100), (0, 0, WIDTH, HEIGHT))
388     self.image.set_alpha(128)
389     self.rect = self.image.get_rect()
390     self.rect.center = WIDTH//2, HEIGHT//2
391
392     def update(self):
393         self.life -= 1
394         if self.life < 0:
395             self.kill()
396
397
398
399
400 ✓ def main():
401     pg.display.set_caption("真！こうかとん無双")
402     screen = pg.display.set_mode((WIDTH, HEIGHT))
403     bg_img = pg.image.load(f"fig/pg_bg.jpg")
404     score = Score()
```

```

405     tscore = 0
406
407     bu_beam = Bullet(pg.transform.rotozoom(pg.image.load(f"fig/beam.png"),0,0.5),4,5,5,100,0)
408     bu_unig = Bullet(pg.transform.rotozoom(pg.image.load(f"fig/beam.png"),0,0.5),3,1,1,200,0)
409     bu_psysc = Bullet(pg.transform.rotozoom(pg.image.load(f"fig/0.png"),0,1),2,0,400,800,50)
410     bu_grav = Bullet(pg.transform.rotozoom(pg.image.load(f"fig/6.png"),0,1),1,0,400,1200,200)
411
412     bird = Bird(3, (550, 600))
413     bombs = pg.sprite.Group()
414     beams = pg.sprite.Group()
415     exps = pg.sprite.Group()
416     emys = pg.sprite.Group()
417     shields = pg.sprite.Group()
418     gravitys = pg.sprite.Group()
419     bullets = [bu_beam,bu_unig,bu_psysc,bu_grav]
420
421     tmr = 0
422     clock = pg.time.Clock()
423     while True:
424         key_lst = pg.key.get_pressed()
425         for event in pg.event.get():
426             if event.type == pg.QUIT:
427                 return 0
428             if event.type == pg.KEYDOWN and event.key == pg.K_c and len(shields) == 0 and score.value >= 5:
429                 if bu_psysc.ct <= 0:
430                     shields.add(Shield(bird,400))
431                     bu_psysc.mode = 1
432                     bu_psysc.ct = bu_psysc.mct
433                     score.value -= 50
434             if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
435                 if key_lst[pg.K_LSHIFT]:
436                     if bu_unig.ct <= 0:
437                         m_beam=NeoBeam(bird,3)
438                         beams.add(m_beam.gen_beams())
439                         bu_unig.value -= 1
440                         bu_unig.ct = bu_unig.mct
441                     else:
442                         if -5 < bu_beam.ct <= 0:
443                             beams.add(Beam(bird))
444                             bu_beam.value -= 1
445                             bu_beam.ct -= 1
446                         if bu_beam.ct <= (-5):
447                             bu_beam.ct = bu_beam.mct
448             if event.type == pg.KEYDOWN and event.key == pg.K_RETURN and len(gravitys) == 0 and score.valu
449                 if bu_grav.ct <= 0:
450                     gravitys.add(Gravity())
451                     bird.change_img(6, screen)
452                     bu_grav.mode = 1
453                     bu_grav.ct = bu_grav.mct
454                     score.value -= 200
455             screen.blit(bg_img, [0, 0])
456
457             if tmr%25 == 0: # 200フレームに1回, 敵機を出現させる
458                 emys.add(Enemy())
459
460             for emy in emys:
461                 if emy.state == "stop" and tmr%emy.interval == 0:
462                     # 敵機が停止状態に入ったら, intervalに応じて爆弾投下
463                     bombs.add(Bomb(emy, bird,score.value))
464

```

```
465     for emy in pg.sprite.groupcollide(emys, beams, True, False).keys():
466         exps.add(Explosion(emy, 100)) # 爆発エフェクト
467         score.value += 10 # 10点アップ
468         tscore += 10 # 10点加算
469
470     if bird.mode == 1 and score.value >= 1000:
471         for emy in pg.sprite.groupcollide(emys, shields, True, False).keys():
472             exps.add(Explosion(emy, 100))
473
474     for bomb in pg.sprite.groupcollide(bombs, beams, True, False).keys():
475         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
476         score.value += 1 # 1点アップ
477         tscore += 1 # 1点加算
478
479     for bomb in pg.sprite.groupcollide(bombs, shields, True, False).keys():
480         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
481
482     if len(pg.sprite.spritecollide(bird, bombs, True)) != 0:
483         bird.change_img(8, screen) # こうかとん悲しみエフェクト
484         score.update(screen)
485         pg.display.update()
486         time.sleep(2)
487         return
488
489     for emy in pg.sprite.groupcollide(emys, gravitys, True, False).keys(): # 敵との衝突判定
490         exps.add(Explosion(emy, 50)) # 爆発エフェクト
491
492     for bomb in pg.sprite.groupcollide(bombs, gravitys, True, False).keys(): # 爆弾との衝突判定
493         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
494
495
496     bird.update(key_lst, screen)
497     beams.update()
498     beams.draw(screen)
499     emys.update()
500     emys.draw(screen)
501     bombs.update()
502     bombs.draw(screen)
503     exps.update()
504     exps.draw(screen)
505     score.update(screen)
506     shields.update(bird, key_lst)
507     shields.draw(screen)
508     gravitys.update()
509     gravitys.draw(screen)
510     for i in bullets:
511         i.update(screen, score.value)
512     pg.display.update()
513     tmr += 1
514     clock.tick(50)
515
516
517 if __name__ == "__main__":
518     pg.init()
519     main()
520     pg.quit()
521     sys.exit()
```