


 c0a2305216 / ProjExD_Group09



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

  hp

ProjExD_Group09 / README.md 

t

...



c0a2303274 こうかといん狩獵

e463754 · 4 hours ago



37 lines (29 loc) · 917 Bytes

こうかといん狩獵DX

実行環境の必要条件

- python >= 3.10
- pygame >= 2.1

ゲームの概要

- (モンスターハンターDX)
- こうかといん狩獵DX

ゲームの遊び方

- 某ゲーム会社Cの代表作MHのオマージュアクション。
- 通常攻撃やスキル、回避などを駆使しながら敵をハントする。

ゲームの実装

共通基本機能

- 背景画像
- 主人公キャラクターの描画
- 主人公のモーション作成
- エネミーの描画
- エネミーのモーション作成
- アイテム作成
- HP系,スタミナ,ダメージ系

担当追加機能

- 背景画像 ます
- 主人公系 おくだ



hp



ProjExD_Group09 / README.md

↑ Top

Preview

Code

Blame



Raw






ToDo

- ☐ セーブ機能
- ☐ エネミーの種類を増やす

メモ

 c0a2305216 / ProjExD_Group09



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



hp ▾

ProjExD_Group09 / koukaton_hunter.py



Go to file

t



Tsuboderaaaa_c0a230... SPクラス実装+その他微調整

20bdfdc · 8 minutes ago



549 lines (484 loc) · 19.9 KB

```
1
2 import math
3 import os
4 import random
5 import sys
6 import time
7 import pygame as pg
8 # from pygame.sprite import _Group
9
10
11 WIDTH = 1100 # ゲームウィンドウの幅
12 HEIGHT = 650 # ゲームウィンドウの高さ
13 os.chdir(os.path.dirname(os.path.abspath(__file__)))
14
15
16 ✓ def check_bound(obj_rct: pg.Rect) -> tuple[bool, bool]:
17     """
18     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
19     引数：こうかとんや爆弾、ビームなどのRect
20     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）
21     """
22     yoko, tate = True, True
23     if obj_rct.left < 0 or WIDTH < obj_rct.right:
24         yoko = False
25     if obj_rct.top < 0 or HEIGHT < obj_rct.bottom:
26         tate = False
27     return yoko, tate
28
29
30 ✓ def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
31     """
32     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
33     引数1 org：爆弾SurfaceのRect
34     引数2 dst：こうかとんSurfaceのRect
35     戻り値：orgから見たdstの方向ベクトルを表すタプル
36     """
37     x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
38     norm = math.sqrt(x_diff**2+y_diff**2)
39     return x_diff/norm, y_diff/norm
40
41
42 class Bird(pg.sprite.Sprite):
```

```

43  """
44  ゲームキャラクター（こうかとん）に関するクラス
45  """
46  delta = { # 押下キーと移動量の辞書
47      pg.K_UP: (0, -1),
48      pg.K_DOWN: (0, +1),
49      pg.K_LEFT: (-1, 0),
50      pg.K_RIGHT: (+1, 0),
51  }
52
53  def __init__(self, num: int, xy: tuple[int, int]):
54      """
55      こうかとん画像Surfaceを生成する
56      引数1 num : こうかとん画像ファイル名の番号
57      引数2 xy : こうかとん画像の位置座標タプル
58      """
59      super().__init__()
60      img0 = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
61      img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
62      self.imgs = {
63          (+1, 0): img, # 右
64          (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
65          (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
66          (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
67          (-1, 0): img0, # 左
68          (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
69          (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
70          (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
71      }
72      self.dire = (+1, 0)
73      self.image = self.imgs[self.dire]
74      self.rect = self.image.get_rect()
75      self.rect.center = xy
76      self.speed = 10
77      # self.state = "normal"
78      # self.hyper_life = 0
79      # self.value = 0
80
81  def change_img(self, num: int, screen: pg.Surface):
82      """
83      こうかとん画像を切り替え、画面に転送する
84      引数1 num : こうかとん画像ファイル名の番号
85      引数2 screen : 画面Surface
86      """
87      self.image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
88      screen.blit(self.image, self.rect)
89
90  def update(self, key_lst: list[bool], screen: pg.Surface):
91      """
92      押下キーに応じてこうかとんを移動させる
93      引数1 key_lst : 押下キーの真理値リスト
94      引数2 screen : 画面Surface
95      無敵の発動条件、発動時間、消費スコアの設定
96      """
97      sum_mv = [0, 0]
98      for k, mv in __class__.delta.items():
99          if key_lst[k]:
100              sum_mv[0] += mv[0]

```

```

101         sum_mv[1] += mv[1]
102     self.rect.move_ip(self.speed * sum_mv[0], self.speed * sum_mv[1])
103     if check_bound(self.rect) != (True, True):
104         self.rect.move_ip(-self.speed * sum_mv[0], -self.speed * sum_mv[1])
105     if not (sum_mv[0] == 0 and sum_mv[1] == 0):
106         self.dire = tuple(sum_mv)
107         self.image = self.imgs[self.dire]
108     screen.blit(self.image, self.rect)
109
110
111     class Bomb(pg.sprite.Sprite):
112         """
113         爆弾に関するクラス
114         """
115         colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
116
117     def __init__(self, emy: "Enemy", bird: Bird):
118         """
119         爆弾円Surfaceを生成する
120         引数1 emy : 爆弾を投下する敵機
121         引数2 bird : 攻撃対象のこうかとん
122         """
123         super().__init__()
124         rad = random.randint(10, 50) # 爆弾円の半径 : 10以上50以下の乱数
125         self.rad = rad
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
137     def update(self):
138         """
139         爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
140         引数 screen : 画面Surface
141         """
142         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
143         if check_bound(self.rect) != (True, True):
144             self.kill()
145
146
147     class Beam(pg.sprite.Sprite):
148         """
149         ビームに関するクラス
150         """
151     def __init__(self, bird: Bird, angle0: int = 0):
152         """
153         ビーム画像Surfaceを生成する
154         引数 bird : ビームを放つこうかとん
155         """
156         super().__init__()
157         self.vx, self.vy = bird.dire
158         angle = math.degrees(math.atan2(-self.vy, self.vx)) + angle0

```

```
159         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 2.0)
160         self.vx = math.cos(math.radians(angle))
161         self.vy = -math.sin(math.radians(angle))
162         self.rect = self.image.get_rect()
163         self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
164         self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
165         self.speed = 10
166
167     def update(self):
168         """
169         ビームを速度ベクトルself.vx, self.vyに基づき移動させる
170         引数 screen : 画面Surface
171         """
172         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
173         if check_bound(self.rect) != (True, True):
174             self.kill()
175
176     class NeoBeam(Beam):
177     def __init__(self, bird: Bird, num: int):
178         super().__init__(bird)
179         self.bird = bird
180         self.num = num
181         # NeoBeam.gen_beams(self)
182
183     def gen_beams(self):
184         bls = []
185         bnum = range(-50, +51, 100 // (self.num - 1))
186         for i in bnum:
187             bls.append(Beam(self.bird, i))
188         return bls
189
190
191     class Explosion(pg.sprite.Sprite):
192         """
193         爆発に関するクラス
194         """
195     def __init__(self, obj: "Bomb|Enemy", life: int):
196         """
197         爆弾が爆発するエフェクトを生成する
198         引数1 obj : 爆発するBombまたは敵機インスタンス
199         引数2 life : 爆発時間
200         """
201         super().__init__()
202         img = pg.image.load(f"fig/explosion.gif")
203         self.imgs = [img, pg.transform.flip(img, 1, 1)]
204         self.image = self.imgs[0]
205         self.rect = self.image.get_rect(center=obj.rect.center)
206         self.life = life
207
208     def update(self):
209         """
210         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
211         爆発エフェクトを表現する
212         """
213         self.life -= 1
214         self.image = self.imgs[self.life//10%2]
215         if self.life < 0:
216             self.kill()
```

```
217
218
219 ✓ class Gravity(pg.sprite.Sprite):
220 ✓     def __init__(self, life: int):
221         super().__init__()
222         self.image = pg.Surface((WIDTH, HEIGHT))
223         self.rect = self.image.get_rect()
```



hp



ProjExD_Group09 / koukaton_hunter.py

↑ Top

Code

Blame

Raw



```
228         self.life -= 1
229         if self.life < 0:
230             self.kill()
231         # screen.blit(self.img, self.rect)
232
233
234 ✓ class Enemy(pg.sprite.Sprite):
235     """
236     敵機に関するクラス
237     """
238     imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
239
240 ✓     def __init__(self):
241         super().__init__()
242         self.image = random.choice(__class__.imgs)
243         self.rect = self.image.get_rect()
244         self.rect.center = random.randint(0, WIDTH), 0
245         self.vx, self.vy = 0, +6
246         self.bound = random.randint(50, HEIGHT//2) # 停止位置
247         self.state = "down" # 降下状態or停止状態
248         self.interval = random.randint(50, 300) # 爆弾投下インターバル
249
250 ✓     def update(self):
251         """
252         敵機を速度ベクトルself.vyに基づき移動（降下）させる
253         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
254         引数 screen : 画面Surface
255         """
256         if self.rect.centery > self.bound:
257             self.vy = 0
258             self.state = "stop"
259             self.rect.move_ip(self.vx, self.vy)
260
261
262 ✓ class Score:
263     """
264     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
265     爆弾 : 1点
266     敵機 : 10点
267     """
268 ✓     def __init__(self):
269         self.font = pg.font.Font(None, 50)
270         self.color = (0, 0, 255)
271         self.value = 400
272         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
273         self.rect = self.image.get_rect()
274         self.rect.center = 100, HEIGHT-50
```

```

275
276     def update(self, screen: pg.Surface):
277         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
278         screen.blit(self.image, self.rect)
279
280
281     class Shield(pg.sprite.Sprite):
282         """
283         こうかとの前に防御壁を出現させ、着弾を防ぐクラス
284         """
285     def __init__(self, bird: Bird, life: int):
286         super().__init__()
287         self.size = (20, bird.rect.height*2) # 大きさのタプル
288         self.image = pg.Surface(self.size) # 空のSurfaceを作成
289         self.life = life # 発動時間の設定
290         self.color = (0, 0, 255) # 矩形色を青色に指定
291         pg.draw.rect(self.image, self.color, (0, 0, 20, bird.rect.height*2))
292         self.vx, self.vy = bird.dire
293         angle = math.degrees(math.atan2(-self.vy, self.vx))
294         self.image = pg.transform.rotozoom(self.image, angle, 1.0)
295         self.image.set_colorkey((0, 0, 0))
296         self.rect = self.image.get_rect()
297         self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
298         self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
299
300     def update(self):
301         self.life -= 1
302         if self.life < 0:
303             self.kill()
304
305
306     class Emp:
307         """
308         enmを発動
309         引数 bombs: Bombインスタンスグループ emys: Enemyインスタンスグループ
310             screen: 画面Surface
311         """
312     def __init__(self, bombs: pg.sprite.Group, emys: pg.sprite.Group, screen: pg.Surface):
313         for emy in emys:
314             emy.interval = math.inf
315             emy.image = pg.transform.laplacian(emy.image)
316             emy.image.set_colorkey((0,0,0))
317         for bomb in bombs:
318             bomb.speed /= 2
319         self.image = pg.Surface((WIDTH, HEIGHT))
320         pg.draw.rect = (self.image, (255, 255, 0), (0, 0, 1600, 900))
321         self.image.set_alpha(100)
322         screen.blit(self.image, [0, 0])
323         time.sleep(0.05)
324         pg.display.update()
325
326
327     class HP:
328         """
329         HPを管理するクラス
330         引数 hp: HPの設定値 int name: HPバーに表示する名前 str
331             xy: HPバーを表示する左端中央の座標(int, int) sz: HPバーのサイズ int
332         """

```



```

333 ✓ def __init__(self, hp: int, name: str, xy: tuple, sz: int):
334     self.font = pg.font.SysFont("hgp創英角ﾎｯﾌﾟ体", sz)
335     self.color = (0, 200, 0)
336     self.max_hp = hp
337     self.hp = hp
338     self.name = name
339     self.x = xy[0]
340     self.y = xy[1]
341     self.size = sz*30, sz*2
342     self.image1 = pg.Surface(self.size)
343     pg.draw.rect(self.image1, (0, 0, 0), (0, 0, self.size[0], self.size[1]))
344     pg.draw.rect(self.image1, self.color, (0, 0, self.size[0]*self.hp/self.max_hp, self.size[1]))
345     self.image2 = self.font.render(f"{self.name} : {self.hp}/{self.max_hp}", 0, (255, 255, 255))
346     self.rect1 = self.image1.get_rect()
347     self.rect2 = self.image2.get_rect()
348     self.rect1.midleft = self.x, self.y
349     self.rect2.midleft = self.x, self.y
350
351 ✓ def damage(self, damage: int): # ダメージを受けた時のメソッド
352     self.hp -= damage
353     if self.hp > self.max_hp:
354         self.hp = self.max_hp
355     elif self.hp < 0:
356         self.hp = 0
357
358 ✓ def update(self, screen: pg.Surface): # HPゲージの更新
359     if self.hp <= (self.max_hp*0.5):
360         self.color = (200, 200, 0)
361     if self.hp <= (self.max_hp*0.2):
362         self.color = (200, 0, 0)
363     pg.draw.rect(self.image1, (0, 0, 0), (0, 0, self.size[0], self.size[1]))
364     pg.draw.rect(self.image1, self.color, (0, 0, self.size[0]*self.hp/self.max_hp, self.size[1]))
365     self.image2 = self.font.render(f"{self.name} : {self.hp}/{self.max_hp}", 0, (255, 255, 255))
366     screen.blit(self.image1, self.rect1)
367     screen.blit(self.image2, self.rect2)
368
369
370 ✓ class SP:
371     """
372     スタミナを管理するクラス
373     """
374 ✓ def __init__(self, sp: int, xy: tuple, sz: int):
375     self.font = pg.font.SysFont("hgp創英角ﾎｯﾌﾟ体", sz)
376     self.color = (240, 120, 0)
377     self.max_sp = sp
378     self.sp = sp
379     self.nsp = 0.25
380     self.x = xy[0]
381     self.y = xy[1]
382     self.size = sz*25, sz
383     self.image1 = pg.Surface(self.size)
384     pg.draw.rect(self.image1, (0, 0, 0), (0, 0, self.size[0], self.size[1]))
385     pg.draw.rect(self.image1, self.color, (0, 0, self.size[0]*self.sp/self.max_sp, self.size[1]))
386     self.image2 = self.font.render(f"SP: {self.sp}/{self.max_sp}", 0, (255, 255, 255))
387     self.rect1 = self.image1.get_rect()
388     self.rect2 = self.image2.get_rect()
389     self.rect1.midleft = self.x, self.y+sz*2
390     self.rect2.midleft = self.x, self.y+sz*2

```

```
391
392 ✓ def pay_sp(self, damage: int): # ダメージを受けた時のメソッド
393     if self.sp > self.max_sp:
394         self.sp = self.max_sp
395     elif self.sp-damage < 0:
396         self.sp = 0
397     else:
398         self.sp -= damage
399
400 ✓ def update(self, screen: pg.Surface): # SPゲージの更新
401     if self.sp == self.max_sp:
402         self.nsp = 0.25
403     elif self.sp <= 5:
404         self.nsp = 0.125
405     self.sp += self.nsp
406     if self.sp > self.max_sp:
407         self.sp = self.max_sp
408     pg.draw.rect(self.image1, (0, 0, 0), (0, 0, self.size[0], self.size[1]))
409     pg.draw.rect(self.image1, self.color, (0, 0, self.size[0]*self.sp/self.max_sp, self.size[1]))
410     self.image2 = self.font.render(f"SP: {int(self.sp)}/{self.max_sp}", 0, (255, 255, 255))
411     screen.blit(self.image1, self.rect1)
412     screen.blit(self.image2, self.rect2)
413
414
415 ✓ def main():
416     pg.display.set_caption("こうかとん狩猟DX")
417     screen = pg.display.set_mode((WIDTH, HEIGHT))
418     bg_img = pg.image.load(f"fig/pg_bg.jpg")
419     score = Score()
420
421     bird = Bird(3, (900, 400))
422     bombs = pg.sprite.Group()
423     beams = pg.sprite.Group()
424     exps = pg.sprite.Group()
425     emys = pg.sprite.Group()
426     shields = pg.sprite.Group() # インスタンスをShieldグループに追加
427     gravity = pg.sprite.Group()
428     k_hp = HP(100, "幻想魔獣こうかとん", (30, 100), 12)
429     k_sp = SP(100, (30, 100), 12)
430     e_hp = HP(300, "未確認飛行物体", (240, 40), 20)
431
432     tmr = 0
433     clock = pg.time.Clock()
434     state = "normal"
435     hyper_life = 0
436     while True:
437         key_lst = pg.key.get_pressed()
438         for event in pg.event.get():
439             if event.type == pg.QUIT:
440                 return 0
441             if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
442                 if event.mod == 1:
443                     if k_sp.sp >= 50:
444                         k_sp.sp -= 50
445                         nbeam = NeoBeam(bird, 5)
446                         beams.add(nbeam.gen_beams())
447                         # beams.add(i for i in NeoBeam(bird, 5))
448                 else:
```

```
449         if k_sp.sp >= 15:
450             k_sp.sp -= 15
451             beams.add(Beam(bird))
452     if event.type == pg.KEYDOWN and event.key == pg.K_c:
453         if score.value >= 50 and len(shields) == 0:
454             shields.add(Shield(bird, 400))
455             score.value -= 50
456     if event.type == pg.KEYDOWN and event.key == pg.K_e:
457         if score.value >= 20: #電磁パルス
458             Emp(bombs, emys, screen)
459             score.value -= 20
460
461     if event.type == pg.KEYDOWN and event.key == pg.K_RETURN:
462         if score.value >= 200:
463             gravity.add(Gravity(400))
464             score.value -= 200
465     #無敵発動する方法と条件
466     if key_lst[pg.K_RSHIFT] and score.value >= 100 and state == "normal":
467         state = "hyper"
468         hyper_life = 500
469         score.value -= 100
470     #無敵発動中の状態
471     if state == "hyper":
472         bird.image = pg.transform.laplacian(bird.image)
473         hyper_life -= 1
474         if hyper_life < 0:
475             state = "normal"
476             # image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0) # 元の画像
477
478     # screen.blit(image, pg.rect)
479     screen.blit(bg_img, [0, 0])
480
481     if tmr%200 == 0: # 200フレームに1回, 敵機を出現させる
482         emys.add(Enemy())
483
484     for emy in emys:
485         if emy.state == "stop" and tmr%emy.interval == 0:
486             # 敵機が停止状態に入ったら, intervalに応じて爆弾投下
487             bombs.add(Bomb(emy, bird))
488
489     for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
490         exps.add(Explosion(emy, 100)) # 爆発エフェクト
491         score.value += 10 # 10点アップ
492         e_hp.damage(10) # 敵に10ダメージ与える
493         bird.change_img(6, screen) # こうかとい喜びエフェクト
494
495     for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
496         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
497         score.value += 1 # 1点アップ
498
499     for shield in pg.sprite.groupcollide(bombs, shields, True, True).keys():
500         exps.add(Explosion(shield, 50)) # 爆発エフェクト
501         score.value += 1 # 1点アップ
502
503     for bomb in pg.sprite.groupcollide(bombs, gravity, True, False).keys():
504         exps.add(Explosion(bomb, 50))
505
506     for emy in pg.sprite.groupcollide(emys, gravity, True, False).keys():
```

```
507         exps.add(Explosion(emy, 50))
508
509     for bomb in pg.sprite.spritecollide(bird, bombs, True):
510         if state == "hyper":
511             exps.add(Explosion(bomb, 50))
512         if state == "normal":
513             k_hp.damage(bomb.rad//2) # こうかとんが10ダメージ受ける
514             if k_hp.hp == 0: # こうかとんのHPが0になったとき
515                 bird.change_img(8, screen) # こうかとん悲しみエフェクト
516                 score.update(screen)
517                 pg.display.update()
518                 time.sleep(2)
519             return
520
521
522
523     bird.update(key_lst, screen)
524     beams.update()
525     beams.draw(screen)
526     emys.update()
527     emys.draw(screen)
528     bombs.update()
529     bombs.draw(screen)
530     gravity.update()
531     gravity.draw(screen)
532     exps.update()
533     exps.draw(screen)
534     score.update(screen)
535     shields.update()
536     shields.draw(screen)
537     k_hp.update(screen)
538     k_sp.update(screen)
539     e_hp.update(screen)
540     pg.display.update()
541     tmr += 1
542     clock.tick(50)
543
544
545 if __name__ == "__main__":
546     pg.init()
547     main()
548     pg.quit()
549     sys.exit()
```