













 gatimokey / ProjExD\_05



 Code  Pull requests  Actions  Projects  Wiki  Security  Insights  Settings

 C0A22033/Crow ▾

ProjExD\_05 / README.md 

 Go to file t 




gatimokey 追加機能カラス

4 minutes ago



31 lines (27 loc) · 1.14 KB

# ゲームのタイトル

 C0A22033/Crow ▾

ProjExD\_05 / README.md

↑ Top

Preview

Code

Blame

Raw



• pygame &gt;= 2.1

## ゲームの概要

簡易版マリオとRPG要素を足したゲーム ・ 敵を倒すと「コイン」と「スコア」が出る。種類によって固定 ・ ステージの作成（障害物の地形など） ・ 「コイン」を消費して、「レベル」や「能力」を獲得できる。 ・ 最終スコアを高いことを目指す。（残り秒数+「スコア」 ・ ステージがスクリーン見たく動く

## ゲームの実装

### ###共通基本機能

- 主人公キャラクターに関するクラス
- 画面推移機能に関する機能
- フィールドに関するクラス

### 担当追加機能

- コインに関する機能
- 能力に関する機能
- スコアに関する機能



敵キャラを作る。（カラス）




## ToDo









- ☐ 画像切り替え機能
- ☐ 座標を決定する。


## メモ

- クラス内の変数は、すべて、「get\_変数名」という名前のメソッドを介してアクセスするように設計してある
- すべてのクラスに関係する関数は、クラスの外で定義してある


 gatimokey /  
ProjExD\_05




 Code  Pull requests  Actions  Projects  Wiki  Security  Insights  Settings

 C0A22033/Crow ▾

ProjExD\_05 / scrole\_kokaton.py



 Go to file t ...



gatimokey 追加機能カラス

6 minutes ago



411 lines (359 loc) · 13.9 KB

```
1  import math
2  import os
3  import random
4  import sys
5  import time
6  from typing import Any
7  import pygame as pg
8  from pygame.sprite import AbstractGroup
9
10
11  WIDTH = 1200 # ゲームウィンドウの幅
12  HEIGHT = 750 # ゲームウィンドウの高さ
13  #WIDTH = 700
14  #HEIGHT = 500
15  MAIN_DIR = os.path.split(os.path.abspath(__file__))[0]
16  MV_FIELD = False
17  MV_JUMP = False
18  MV_MOVE = False
19  SOKO = 0
20  COUNT = 0
21
22
23  ▾ def check_bound(obj: pg.Rect) -> tuple[bool, bool]:
24      """
25      オブジェクトが画面内か画面外かを判定し、真理値タプルを返す
26      引数 obj : オブジェクト (爆弾, こうかとん, ビーム) SurfaceのRect
27      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)
28      """
29      yoko, tate = True, True
30      if obj.left < 0 or WIDTH < obj.right: # 横方向のはみ出し判定
31          yoko = False
32      if obj.top < 0 or HEIGHT + SOKO < obj.bottom: # 縦方向のはみ出し判定
33          tate = False
34      return yoko, tate
35
36
37  ▾ def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
38      """
39      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
40      引数1 org : 爆弾SurfaceのRect
41      引数2 dst : こうかとんSurfaceのRect
```

```

42     戻り値 : orgから見たdstの方向ベクトルを表すタプル
43     """
44     x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
45     norm = math.sqrt(x_diff**2+y_diff**2)
46     return x_diff/norm, y_diff/norm
47
48
49     class Bird(pg.sprite.Sprite):
50         """
51         ゲームキャラクター（こうかとん）に関するクラス
52         """
53     delta = { # 押下キーと移動量の辞書
54         pg.K_UP: (0, -1),
55         pg.K_DOWN: (0, +1),
56         pg.K_LEFT: (-1, 0),
57         pg.K_RIGHT: (+1, 0),
58     }
59
60     def __init__(self, num: int, xy: tuple[int, int]):
61         """
62         こうかとん画像Surfaceを生成する
63         引数1 num : こうかとん画像ファイル名の番号
64         引数2 xy : こうかとん画像の位置座標タプル
65         """
66         super().__init__()
67         img0 = pg.transform.rotozoom(pg.image.load(f"{MAIN_DIR}/fig/{num}.png"), 0, 2.0)
68         img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
69         self.imgs = {
70             (+1, 0): img, # 右
71             (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
72             (0, -1): img, # 上
73             (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
74             (-1, 0): img0, # 左
75             (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
76             (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
77             (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
78         }
79         self.dire = (+1, 0)
80         self.image = self.imgs[self.dire]
81         self.rect = self.image.get_rect()
82         self.rect.center = xy
83         self.speed = 10
84
85     def change_img(self, num: int, screen: pg.Surface):
86         """
87         こうかとん画像を切り替え、画面に転送する
88         引数1 num : こうかとん画像ファイル名の番号
89         引数2 screen : 画面Surface
90         """
91         self.image = pg.transform.rotozoom(pg.image.load(f"{MAIN_DIR}/fig/{num}.png"), 0, 2.0)
92         screen.blit(self.image, self.rect)
93
94     def update(self, key_lst: list[bool], screen: pg.Surface):
95         """
96         押下キーに応じてこうかとんを移動させる
97         引数1 key_lst : 押下キーの真理値リスト
98         引数2 screen : 画面Surface
99         """

```

```
100     global MV_FIELD,COUNT,MV_MOVE
101     sum_mv = [0, 0]
102     moto_center = self.rect.center
103     for k, mv in __class__.delta.items():
104         if key_lst[k]:
105             self.rect.move_ip(+self.speed*mv[0], +self.speed*mv[1])
106             sum_mv[0] += mv[0]
107             sum_mv[1] += mv[1]
108             if k == pg.K_UP:
109                 COUNT += 1
110                 if 20 <= COUNT < 35:
111                     self.rect.move_ip(-self.speed*mv[0], -self.speed*mv[1])
112                     MV_MOVE = False
113                 elif 35 <= COUNT:
114                     COUNT = 0
115     self.rect.move_ip(0,2)
116     if check_bound(self.rect) != (True, True):
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                 MV_MOVE = False
121     if self.rect.right > WIDTH/13*5: #画面推移のための線引き
122         self.rect.move_ip(-self.speed*mv[0],0)
123         MV_FIELD = True
124     if MV_MOVE == True:
125         for k, mv in __class__.delta.items():
126             if key_lst[k]:
127                 self.rect.move_ip(-self.speed*mv[0], -self.speed*mv[1])
128                 self.rect.move_ip(0,10)
129     if not (sum_mv[0] == 0 and sum_mv[1] == 0):
130         self.dire = tuple(sum_mv)
131         self.image = self.imgs[self.dire]
132     screen.blit(self.image, self.rect)
133
134
135     class Bomb(pg.sprite.Sprite):
136         """
137         爆弾に関するクラス
138         """
139         colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
140
141     def __init__(self, emy: "Enemy", bird: Bird):
142         """
143         爆弾円Surfaceを生成する
144         引数1 emy : 爆弾を投下する敵機
145         引数2 bird : 攻撃対象のこうかとん
```



C0A22033/Crow ▾

ProjExD\_05 / scrole\_kokaton.py

↑ Top

Code

Blame

Raw



```
151     pg.draw.circle(self.image, color, (rad, rad), rad)
152     self.image.set_colorkey((0, 0, 0))
153     self.rect = self.image.get_rect()
154     # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
155     self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
156     self.rect.centerx = emy.rect.centerx
157     self.rect.centery = emy.rect.centery+emy.rect.height/2
158     self.speed = 6
```

```

158         self.speed = 0
159
160     def update(self):
161         """
162         爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
163         引数 screen : 画面Surface
164         """
165         if MV_FIELD == True:
166             self.rect.move_ip(-5,0)
167             self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
168             if check_bound(self.rect) != (True, True):
169                 self.kill()
170
171
172     class Beam(pg.sprite.Sprite):
173         """
174         ビームに関するクラス
175         """
176     def __init__(self, bird: Bird):
177         """
178         ビーム画像Surfaceを生成する
179         引数 bird : ビームを放つこうかとん
180         """
181         super().__init__()
182         self.vx, self.vy = bird.dire
183         angle = math.degrees(math.atan2(-self.vy, self.vx))
184         self.image = pg.transform.rotozoom(pg.image.load(f"{MAIN_DIR}/fig/beam.png"), angle, 2.0)
185         self.vx = math.cos(math.radians(angle))
186         self.vy = -math.sin(math.radians(angle))
187         self.rect = self.image.get_rect()
188         self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
189         self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
190         self.speed = 10
191
192     def update(self):
193         """
194         ビームを速度ベクトルself.vx, self.vyに基づき移動させる
195         引数 screen : 画面Surface
196         """
197         self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
198         if check_bound(self.rect) != (True, True):
199             self.kill()
200
201
202     class Explosion(pg.sprite.Sprite):
203         """
204         爆発に関するクラス
205         """
206     def __init__(self, obj: "Bomb|Enemy", life: int):
207         """
208         爆弾が爆発するエフェクトを生成する
209         引数1 obj : 爆発するBombまたは敵機インスタンス
210         引数2 life : 爆発時間
211         """
212         super().__init__()
213         img = pg.image.load(f"{MAIN_DIR}/fig/explosion.gif")
214         self.imgs = [img, pg.transform.flip(img, 1, 1)]
215         self.image = self.imgs[0]
216         self.rect = self.image.get_rect(center=obj.rect.center)

```

```

217         self.life = life
218
219     def update(self):
220         """
221         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
222         爆発エフェクトを表現する
223         """
224         self.life -= 1
225         self.image = self.imgs[self.life//10%2]
226         if self.life < 0:
227             self.kill()
228
229
230     class Enemy(pg.sprite.Sprite):
231         """
232         カラスに関するクラス
233         """
234         #imgs = [pg.image.load(f"{MAIN_DIR}/fig/crow_{i}.png") for i in range(1, 3)]
235         tmr = 0
236         i = 0
237
238     def __init__(self, arrive):
239         super().__init__()
240         img0 = pg.transform.rotozoom(pg.image.load(f"{MAIN_DIR}/fig/crow_1.png"), 0, 0.2)
241         img1 = pg.transform.rotozoom(pg.image.load(f"{MAIN_DIR}/fig/crow_2.png"), 0, 0.2)
242         self.crow_list = [img0, img1]
243         self.rect = self.crow_list[0].get_rect()
244         self.rect.center = 1100, 0
245         self.vy = +6
246         self.bound = 40 # 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         self.arrive = arrive
250
251
252
253
254     def update(self):
255         """
256         敵機を速度ベクトルself.vyに基づき移動（降下）させる
257         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
258         引数 screen : 画面Surface
259         """
260         if self.rect.centery > self.bound:
261             self.vy = 0
262             self.state = "stop"
263             self.rect.centery += self.vy
264
265         self.arrive -= 1
266         self.image = self.crow_list[self.arrive//10%2]
267         if self.arrive < 0:
268             self.arrive += 1
269
270
271
272     class Score:
273         """
274         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

```

```
275     爆弾 : 1点
276     敵機 : 10点
277     """
278     def __init__(self):
279         self.font = pg.font.Font(None, 50)
280         self.color = (0, 0, 255)
281         self.value = 0
282         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
283         self.rect = self.image.get_rect()
284         self.rect.center = 100, HEIGHT-50
285
286     def update(self, screen: pg.Surface):
287         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
288         screen.blit(self.image, self.rect)
289
290
291     class Field(pg.sprite.Sprite):
292         """
293         足場に関するクラス
294         """
295     def __init__(self, left_L = 100, top_L = HEIGHT-50, yoko = 50, tate = 50):
296         super().__init__()
297         self.left = left_L
298         self.top = top_L
299         self.image = pg.Surface((yoko, tate))
300         pg.draw.rect(self.image, (255, 0, 0), (0, 0, yoko, tate))
301         self.rect = self.image.get_rect()
302         self.rect.centerx = left_L
303         self.rect.centery = top_L
304
305     def update(self):
306         """
307         足場の移動と消去の更新に関する関数
308         """
309         if MV_FIELD == True:
310             self.rect.move_ip(-5, 0)
311             if self.rect.right < 0:
312                 self.kill()
313
314
315     def main():
316         global MV_FIELD, SOKO, MV_JUMP, MV_MOVE
317         pg.display.set_caption("真！ どうかとん無双")
318         screen = pg.display.set_mode((WIDTH, HEIGHT))
319         bg_img = pg.image.load(f"{MAIN_DIR}/fig/pg_bg.jpg")
320         score = Score()
321
322
323         bird = Bird(3, (50, 50))
324         bombs = pg.sprite.Group()
325         beams = pg.sprite.Group()
326         exps = pg.sprite.Group()
327         emys = pg.sprite.Group()
328         fields = pg.sprite.Group()
329         fields.add(Field())
330         fields.add(Field(0, HEIGHT-20, 1000, 20))
331         fields.add(Field(1200, HEIGHT-20, 200, 20))
332         fields.add(Field(1000, HEIGHT/2))
```



```
333
334     tmr = 0
335     clock = pg.time.Clock()
336     emys.add(Enemy(100000))
337     while True:
338         key_lst = pg.key.get_pressed()
339         for event in pg.event.get():
340             if event.type == pg.QUIT:
341                 return 0
342             if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
343                 beams.add(Beam(bird))
344
345         screen.blit(bg_img, [0, 0])
346
347
348         for emy in emys:
349             if emy.state == "stop" and tmr%emy.interval == 0:
350                 # 敵機が停止状態に入ったら, intervalに応じて爆弾投下
351                 bombs.add(Bomb(emy, bird))
352
353         for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
354             exps.add(Explosion(emy, 100)) # 爆発エフェクト
355             score.value += 10 # 10点アップ
356             bird.change_img(6, screen) # こうかたん喜びエフェクト
357
358         for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
359             exps.add(Explosion(bomb, 50)) # 爆発エフェクト
360             score.value += 1 # 1点アップ
361
362         if len(pg.sprite.spritecollide(bird, bombs, True)) != 0:
363             print("爆発死")
364             bird.change_img(8, screen) # こうかたん悲しみエフェクト
365             score.update(screen)
366             pg.display.update()
367             time.sleep(2)
368             return
369
370         if pg.sprite.spritecollide(bird, fields, False):
371             cc = pg.sprite.spritecollideany(bird, fields)
372             print(cc.rect.center)
373             if cc.rect.centery+20 <= bird.rect.top <= cc.rect.bottom:
374                 bird.rect.move_ip(0,10)
375             if cc.rect.top <= bird.rect.bottom <= cc.rect.centery+20:
376                 bird.rect.move_ip(0,-12)
377             bird.rect.move_ip(0,-2)
378             MV_MOVE = True
379
380         if bird.rect.top < 1 or HEIGHT -1 < bird.rect.bottom:
381             print("画面外死")
382             bird.change_img(8, screen) # こうかたん悲しみエフェクト
383             score.update(screen)
384             pg.display.update()
385             time.sleep(2)
386             return
387
388         bird.update(key_lst, screen)
389         beams.update()
390         beams.draw(screen)
```

```
391         emys.update()
392         emys.draw(screen)
393         bombs.update()
394         bombs.draw(screen)
395         exps.update()
396         exps.draw(screen)
397         fields.update()
398         fields.draw(screen)
399         score.update(screen)
400         pg.display.update()
401         MV_FIELD = False
402         MV_MOVE = False
403         tmr += 1
404         clock.tick(50)
405
406
407 if __name__ == "__main__":
408     pg.init()
409     main()
410     pg.quit()
411     sys.exit()
```