

c0b23036a5 / ProjExD_Groupe13

<> Code

Pull requests

Actions

Projects

Wiki

Security

Insights

ProjExD_Groupe13 / README.md



c0b23036a5 readme更新

1 minute ago

39 lines (28 loc) · 1.11 KB

PreviewCodeBlame

RawCopyDownloadEditDropdownMenu

ProjExD_Groupe13

概要

前回の授業で作成した鳥無双を改良する。

仕様を固めようそう！そう「しよう」（やったこと）

- ボス（若林）1分に一つ使える。おっきい。たくさん球を出す。HPの概念を追加
- アイテム（田端）1分にひとつ出現する。こうかトンに触れると、アイテムフラグが立ち、すでに実装済みの必殺技をひとつ使える。
- 音の実装（山田）たまの発射音 被弾音の追加 BGMを再生
- こうかとん残機（織田）被弾するとハートがひとつ減る。画面上部にハートを表示
- jsonファイルに最高点を記載。ワールドレコードをクラウドにアップロード（村田）ゲームオーバー時に最高得点も同時に表示する。最高点を更新したら誉める。

todo

- アイテム
- スタート画面

import packs

以下のパッケージを利用しています。

math,os,random,sys,time,pygame,firebase_admin

c0b23036a5 /
ProjExD_Groupe13

<> Code

Pull requests

Actions

Projects

Wiki

Security

Insights



C0B23036/kouka... ▾

ProjExD_Groupe13 / p_musou_koukaton.py



c0b23036a5 readme更新

now



506 lines (430 loc) · 17.9 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, HEIGHT = 1600, 900 # ゲームウィンドウの幅, 高さ
10 os.chdir(os.path.dirname(os.path.abspath(__file__)))
11
12
13  def check_bound(obj_rct:pg.Rect) -> tuple[bool, bool]:
14      """
15      Rectの画面内外判定用の関数
16      引数: こうかとんRect, または, 爆弾Rect, またはビームRect
17      戻り値: 横方向判定結果, 縦方向判定結果 (True: 画面内 / False: 画面外)
18      """
19      yoko, tate = True, True
20      if obj_rct.left < 0 or WIDTH < obj_rct.right: # 横方向のはみ出し判定
21          yoko = False
22      if obj_rct.top < 0 or HEIGHT < obj_rct.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      ゲームキャラクター（こうかとん）に関するクラス
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  state="normal" #こうかとんの状態の変数
50  hyper_life=500 #無敵時間(単位:frame)
51
52  def __init__(self, num: int, xy: tuple[int, int]):
53      """
54      こうかとん画像Surfaceを生成する
55      引数1 num : こうかとん画像ファイル名の番号
56      引数2 xy : こうかと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/{num}.png"), 0, 2.0)
60      img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
61      self.imgs = {
62          (+1, 0): img, # 右
63          (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
64          (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
65          (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
66          (-1, 0): img0, # 左
67          (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
68          (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
69          (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 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
77  def change_img(self, num: int, screen: pg.Surface):
78      """
79      こうかとん画像を切り替え、画面に転送する
80      引数1 num : こうかとん画像ファイル名の番号
81      引数2 screen : 画面Surface
82      """
83      self.image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
84      screen.blit(self.image, self.rect)
85
86  def update(self, key_lst: list[bool], screen: pg.Surface):
87      """
88      押下キーに応じてこうかとんを移動させる
89      引数1 key_lst : 押下キーの真理値リスト
90      引数2 screen : 画面Surface
91      """
```

```

92     sum_mv = [0, 0]
93     for k, mv in __class__.delta.items():
94         if key_lst[k]:
95             sum_mv[0] += mv[0]
96             sum_mv[1] += mv[1]
97     self.rect.move_ip(self.speed*sum_mv[0], self.speed*sum_mv[1])
98     if check_bound(self.rect) != (True, True):
99         self.rect.move_ip(-self.speed*sum_mv[0], -self.speed*sum_mv[1])
100     if not (sum_mv[0] == 0 and sum_mv[1] == 0):
101         self.dire = tuple(sum_mv)
102         self.image = self.imgs[self.dire]
103
104     if self.state == "hyper":      #無敵状態の場合
105         if self.hyper_life <= 0:  #時間が0の場合
106             self.state="normal"   #ノーマルモードにする
107         else:
108             self.hyper_life -= 1   #時間を1フレームごとに減らす
109             self.image = pg.transform.laplacian(self.image) #特殊状態にする
110
111     screen.blit(self.image, self.rect)
112
113     class Life(pg.sprite.Sprite):
114         """
115         こうかとの残機に関するクラス
116         """
117         def __init__(self, initial_lives: int):
118             """
119             こうかとん画像Surfaceを生成する
120             引数1 initial_lives:初期値
121             """
122             self.lives = initial_lives #初期ライフというだけ
123             self.life_image = pg.image.load("fig/koukaton_life.png") #赤いハート画像
124             self.lost_life_image=pg.image.load("fig/lost_life.png") #輝きを失ったハート画像
125             self.neta_life=pg.image.load("fig/koukaton_buti.png") #こうかとんカットイン画像
126
127             def lose_life(self): #こうかとのライフ減少を行う関数
128                 if self.lives > 0:
129                     self.lives -= 1
130
131             def draw(self, screen: pg.Surface): #こうかとのライフを表示する関数
132                 for i in range(3):
133                     if i < self.lives:
134                         screen.blit(self.life_image, (10 + i * (self.life_image.get_width() + 10),
135                     else:
136                         screen.blit(self.lost_life_image, (10 + i * (self.life_image.get_width() +
137             """
138             def huzake(self, screen: pg.Surface): #こうかとのカットインを表示する関数
139                 screen.blit(self.neta_life, (-300, 0))
140             """
141
142
143     class Bomb(pg.sprite.Sprite):
144         """
145         爆弾に関するクラス

```

```
146     """
147     colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255,
148
149     def __init__(self, emy: "Enemy", bird: Bird):
150         """
151         爆弾円Surfaceを生成する
152         引数1 emy : 爆弾を投下する敵機
153         引数2 bird : 攻撃対象のこうかとん
154         """
155         super().__init__()
156         rad = random.randint(10, 50) # 爆弾円の半径 : 10以上50以下の乱数
157         self.image = pg.Surface((2*rad, 2*rad))
158         color = random.choice(__class__.colors) # 爆弾円の色 : クラス変数からランダム選択
159         pg.draw.circle(self.image, color, (rad, rad), rad)
160         self.image.set_colorkey((0, 0, 0))
161         self.rect = self.image.get_rect()
162         # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
163         self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
164         self.rect.centerx = emy.rect.centerx
165         self.rect.centery = emy.rect.centery+emy.rect.height/2
166         self.speed = 6
167
168     def update(self):
169         """
170         爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
171         引数 screen : 画面Surface
172         """
173         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
174         if check_bound(self.rect) != (True, True) :
175             self.kill()
176
177
178
179     class Beam(pg.sprite.Sprite):
180         """
181         ビームに関するクラス
182         """
183     def __init__(self, bird: Bird, angle0 =0):
184         """
185         ビーム画像Surfaceを生成する
186         引数 bird : ビームを放つこうかとん
187         """
188         super().__init__()
189         self.vx, self.vy = bird.dire
190         angle= math.degrees(math.atan2(-self.vy, self.vx)+angle0) #こうかとんの角度にangle0
191         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 2.0)
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 = 10
198
199     def update(self):
```

```
200     """
201     ビームを速度ベクトルself.vx, self.vyに基づき移動させる
202     引数 screen : 画面Surface
203     """
204     self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
205     if check_bound(self.rect) != (True, True):
206         self.kill()
207
208     '''
209     Beamを複数つくる弾幕クラス
210     class NeoBeam():
211
212         def __init__(self):
213             self.b_lst = []
214
215         def gen_beams(self):
216             for i in range(10):
217                 self.b_lst.append(-50+i*10)
218             return self.b_lst
219     '''
220
221
222
223     class Explosion(pg.sprite.Sprite):
224         """
225         爆発に関するクラス
226         """
227     def __init__(self, obj: "Bomb|Enemy", life: int):
228         """
229         爆弾が爆発するエフェクトを生成する
230         引数1 obj : 爆発するBombまたは敵機インスタンス
231         引数2 life : 爆発時間
232         """
233         super().__init__()
234         img = pg.image.load(f"fig/explosion.gif")
235         self.imgs = [img, pg.transform.flip(img, 1, 1)]
236         self.image = self.imgs[0]
237         self.rect = self.image.get_rect(center=obj.rect.center)
238         self.life = life
239
240     def update(self):
241         """
242         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
243         爆発エフェクトを表現する
244         """
245         self.life -= 1
246         self.image = self.imgs[self.life//10%2]
247         if self.life < 0:
248             self.kill()
249
250
251     class Enemy(pg.sprite.Sprite):
252         """
253         敵機に関するクラス
```

```
254     """
255     imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
256
257     def __init__(self):
258         super().__init__()
259         self.image = random.choice(__class__.imgs)
260         self.rect = self.image.get_rect()
261         self.rect.center = random.randint(0, WIDTH), 0
262         self.vy = +6
263         #self.bound = random.randint(50, int(HEIGHT/2)) # 停止位置
264         self.bound = random.randint(50, int(HEIGHT/2)) # 停止位置
265         self.state = "down" # 降下状態or停止状態
266         self.interval = random.randint(50, 300) # 爆弾投下インターバル
267
268     def update(self):
269         """
270         敵機を速度ベクトルself.vyに基づき移動（降下）させる
271         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
272         引数 screen : 画面Surface
273         """
274         if self.rect.centery > self.bound:
275             self.vy = 0
276             self.state = "stop"
277             self.rect.centery += self.vy
278
279
280     class Score:
281         """
282         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
283         爆弾 : 1点
284         敵機 : 10点
285         """
286     def __init__(self):
287         self.font = pg.font.Font(None, 50)
288         self.color = (0, 0, 255)
289         self.value = 0
290         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
291         self.rect = self.image.get_rect()
292         self.rect.center = 100, HEIGHT-50
293
294     def update(self, screen: pg.Surface):
295         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
296         screen.blit(self.image, self.rect)
297
298     class Gravity(pg.sprite.Sprite):
299         """
300         エンターキーを押すと画面内の敵を倒す
301         """
302     def __init__(self, life):
303         super().__init__()
304         self.image = pg.Surface((1600, 900))
305         pg.draw.rect(self.image, (0, 0, 0), (0, 0, 1600, 900))
306         self.image.set_alpha(200)
307         self.rect = self.image.get_rect()
```



```
308         self.rect.center = (WIDTH/2, HEIGHT/2)
309         self.life = life      #発動時間の定義
310     def update(self):
311         self.life -= 1
312         if self.life <= 0:
313             self.kill()
314
315     class Sheeld(pg.sprite.Sprite):
316         """
317         こうかとの向いている向きに長方形のシールドを表示するクラス
318         """
319         is_not_shield = True
320     def __init__(self, bird: Bird, life: int):
321         #シールドが展開中はFalseのクラス変数
322
323
324         """
325         長方形の壁を生成する
326         """
327         super().__init__()
328         # 壁 幅20 高さ こうかとの半分の高さ
329         self.image = pg.Surface((20, bird.rect.height*2))
330         self.image.fill((0, 0, 255))
331         self.rect = self.image.get_rect()
332         self.rect.center = bird.rect.center
333         pg.draw.rect(self.image, (0, 0, 255), self.rect)
334
335         # birdの向いている方向に合わせて回転
336         vx , vy = bird.dire
337         angle = math.degrees(math.atan2(-vy, vx))
338         self.image = pg.transform.rotate(self.image, angle)
339         self.rect = self.image.get_rect(center=self.rect.center)
340
341         # シールドをBirdの向いている先に設置
342         self.rect.centerx += bird.rect.width*vx
343         self.rect.centery += bird.rect.height*vy
344
345         # シールドの耐久値
346         self.life = life
347
348         # シールド展開中はFalse
349         __class__.is_not_shield = False
350
351     def update(self, bird: Bird):
352         #print(self.life)
353         self.life -= 1
354         if self.life < 0:
355             __class__.is_not_shield = True
356             self.kill()
357
358
359
360
361     def main():
```

```
361     def main():
362
363
364     pg.display.set_caption("真！こうかとん無双")
365     screen = pg.display.set_mode((WIDTH, HEIGHT))
366     bg_img = pg.image.load(f"fig/pg_bg.jpg")
367     score = Score()
368     #neobeam = NeoBeam
369     #score.value = 900000000
370
371     bird = Bird(3, (900, 400))
372     life = Life(3) # こうかとんの残機を3に設定
373     bombs = pg.sprite.Group()
374     beams = pg.sprite.Group()
375     exps = pg.sprite.Group()
376     emys = pg.sprite.Group()
377     gravity = pg.sprite.Group()
378     sheelds = pg.sprite.Group()
379
380     tmr = 0
381     clock = pg.time.Clock()
382     while True:
383         key_lst = pg.key.get_pressed()
384         for event in pg.event.get():
385             if event.type == pg.QUIT:
386                 return 0
387
388         #もし左シフトキーおよびスペースキーが押されたら五個のビームを生成
389         if event.type == pg.KEYDOWN and event.key == pg.K_LSHIFT and pg.K_SPACE:
390             for i in range(5):
391                 if i != 2:
392                     #1 = neobeam.gen_beams()
393                     beams.add(Beam(bird, -50+i*25))
394
395             if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
396                 beams.add(Beam(bird))
397             if event.type == pg.KEYDOWN and event.key == pg.K_RETURN and score.value >= 200:
398                 gravity.add(Gravity(400))
399                 score.value -= 200
400
401
402             if (event.type == pg.KEYDOWN) and (event.key == pg.K_RSHIFT) and (score.value >= 200):
403                 bird.state = "hyper" #無敵状態にする
404                 score.value -= 100 #スコアを減らして500フレーム分無敵にする
405                 bird.hyper_life = 500
406
407         screen.blit(bg_img, [0, 0])
408
409         if key_lst[pg.K_LSHIFT]:
410             bird.speed = 20
411         else:
412             bird.speed = 10
413
414         if tmr%200 == 0: # 200フレームに1回, 敵機を出現させる
```

```
415     emys.add(enemy())
416
417     for emy in emys:
418         if emy.state == "stop" and tmr%emy.interval == 0:
419             # 敵機が停止状態に入ったら、intervalに応じて爆弾投下
420             bombs.add(Bomb(emy, bird))
421
422     for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
423         exps.add(Explosion(emy, 100)) # 爆発エフェクト
424         score.value += 10 # 10点アップ
425         bird.change_img(6, screen) # こうかたん喜びエフェクト
426
427     for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
428         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
429         score.value += 1 # 1点アップ
430
431     for emy in pg.sprite.groupcollide(emys, gravity, True, False).keys():
432         exps.add(Explosion(emy, 100))
433         score.value += 10
434         bird.change_img(6, screen)
435
436     for bomb in pg.sprite.groupcollide(bombs, gravity, True, False).keys():
437         exps.add(Explosion(bomb, 50))
438         score.value += 1
439
440
441
442     # Cボタンを押すとシールドを展開
443     #if key_lst[pg.K_c] & score.value >= 50:
444     if key_lst[pg.K_c] & Sheeld.is_not_shield & (score.value >= 50):
445         #if key_lst[pg.K_c]:
446             print(score.value >= 50)
447             print(Sheeld.is_not_shield)
448             #score.value -= 50
449             print("シールド展開")
450             # スコアが50減る
451             score.value -= 50
452             sheelds.add(Sheeld(bird, 400))
453
454     # 壁と爆弾が衝突したら爆発
455     for bomb in pg.sprite.groupcollide(bombs, sheelds, True, False).keys():
456         exps.add(Explosion(bomb, 50))
457         score.value += 1
458
459     touch_bomb=pg.sprite.spritecollide(bird, bombs, False)
460     if len(touch_bomb) != 0:
461         if (bird.state == "hyper"):
462             exps.add(Explosion(touch_bomb[0], 50)) # 爆発エフェクト
463             score.value += 1 # 1点アップ
464             touch_bomb[0].kill()
465         else:
466             touch_bomb[0].kill()
467             bird.change_img(8, screen) # こうかたん悲しみエフェクト
468         """
```

```
469         if life.lives != 1: #こうかとの被弾カットインの条件：残りライフが1ではないとき
470             life.huzake(screen) #被弾カットイン表示関数実行
471         """
472         score.update(screen)
473         pg.display.update()
474         time.sleep(0.4)
475         life.lose_life() #こうかとのライフが一つ減る
476         if life.lives == 0: #こうかとのライフが0ならば
477             time.sleep(2) #2秒待つ
478             return
479
480     bird.update(key_lst, screen)
481     beams.update()
482     beams.draw(screen)
483     emys.update()
484     emys.draw(screen)
485     bombs.update()
486     bombs.draw(screen)
487     exps.update()
488     exps.draw(screen)
489     score.update(screen)
490     gravity.draw(screen)
491     gravity.update()
492     life.draw(screen) # 残機を画面に表示
493
494     if not Sheeld.is_not_shield:
495         sheelds.update(bird)
496         sheelds.draw(screen)
497     pg.display.update()
498     tmr += 1
499     clock.tick(50)
500
501
502 if __name__ == "__main__":
503     pg.init()
504     main()
505     pg.quit()
506     sys.exit()
```