

c0a2311499 / ProjExD_Group14

🔍

📧

👤

<> Code

🕒 Issues

🔗 Pull requests

🔄 Actions

📁 Projects

📖 Wiki

🛡 Security

⋮

📄

🔗 main ▾

ProjExD_Group14 / README.md

📄

🔍 Go to file

t

⋮

👤 c0a2311499 README.mdの編集

845164e · 1 minute ago

🕒

58 lines (47 loc) · 3.66 KB

PreviewCodeBlame

Raw📄📥✎⌵⋮

こうかとんクエスト

実行環境の必要条件

- python >= 3.10
- pygame >= 2.1

ゲームの概要

- こうかとんを十字キーで操作しスペースキーでビームを発射、敵やその爆弾を打ち落としてスコアを稼いでいくゲームで、元あるこうかとん無双に機能を追加していくことでさらにゲーム感が増している。具体的にはこうかとんの体力や攻撃力、技を使うためのスキルポイントの追加、一定のスキルを取ることで挑戦可能なボス戦の追加、画面スクロールの追加をすることでゲーム感を増させている。このゲームはクリアを目指すというより、ハイスコアを目指すものである。

ゲームの実装

共通基本機能

- 背景画像、こうかとん、こうかとんのビーム、敵のUFO、UFOの爆弾の描画
- こうかとんの各種スキルの実装
- スコアの描画

担当追加機能

- 画面の動き(担当：寺川 竣祐)：横スクロール、敵を右から表示させる機能の追加
- ライフゲージ(担当：川畑 しのすけ)：こうかとんのライフゲージの追加
- ボス追加①(担当：濱口 莉奈)：通常画面からボス画面、ボス撃破後に通常画面への移動
- ボス追加②(担当：町田 拓斗)：ボスの性能を決めるクラスの作成
- ステータス変化(担当：萩原 颯人)：打ち落とした爆弾の色によって各種ステータスを変化

画面の動き（担当：寺川 竣祐）

- 横スクロールの追加をした
- 上から敵から出てくるシステムを右から出現するように変更した
- GameOver画面の追加をした

ライフゲージ（担当：川畑伸乃輔）

- こうかとんのHPバーを追加した
- バーをまず緑色に表示
- エネミーの攻撃を一定数くらうとHPが減少も緑→黄→赤と変わっていく
- 被弾したときにHP減少に伴ってこうかとんの悲しみエフェクトも追加

ボス戦切り替え(担当：濱口莉奈)

- 1キーを押したときにボス戦に行くかどうかを確認するポップアップを表示。
- このときスコアが100以上でないといけない。
- "Yキー"を押したらボス戦の画面に切り替わり、"Nキー"のときにゲームに戻る

ボスの機能（担当：町田 拓斗）

- ボスの表示
- ボスとボスの弾とこうかとんのビームの衝突判定
- 1キーを押したときにボス戦に突入できる機能
- ボスからこうかとんへのホーミング弾
- ボスのHP設定と、ボスのHPが0になった時にボスを消す
- ボスの弾を火の玉に変更

ステータス変化（担当：萩原 颯人）

- 攻撃力の概念の追加：打ち落とした爆弾が赤色のときにこうかとんの攻撃力を+1する。攻撃力の数値を画面上に追加して可視化した。
- スキルポイントの概念の追加：打ち落とした爆弾が青色のときにスキルポイントが+1する。SPと書いて画面上に追加して可視化した。
- こうかとん無双ではスコアを消費して技を使っていたがスキルポイントを追加したので技に応じたポイントを消費して発動できるようにコードを書き直した。

ToDo

- ☐ 体力の回復機能の追加
- ☐ ポップアップを表示したときに画面で動いているゲームの一時停止
- ☐ フェードアウト処理
- ☐ YesやNoを選択したときに色を変える
- ☐ 本体が後退するプログラムの追加

c0a2311499 /
ProjExD_Group14[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

ProjExD_Group14 / musou_kokaton.py



c0a2311499 GameOverの追加

67cd993 · 36 minutes ago



743 lines (628 loc) · 27.7 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  def check_bound(obj_rct:pg.Rect) -> tuple[bool, bool]:
13      """
14      Rectの画面内外判定用の関数
15      引数: こうかとんRect, または, 爆弾Rect, またはビームRect
16      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)
17      """
18      yoko, tate = True, True
19      if obj_rct.left < 0 or WIDTH < obj_rct.right: # 横方向のはみ出し判定
20          yoko = False
21      if obj_rct.top < 0 or HEIGHT < obj_rct.bottom: # 縦方向のはみ出し判定
22          tate = False
23      return yoko, tate
24
25
26  def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
27      """
28      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
29      引数1 org: 爆弾SurfaceのRect
30      引数2 dst: こうかとんSurfaceのRect
31      戻り値: orgから見たdstの方向ベクトルを表すタプル
32      """
33      x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
34      norm = math.sqrt(x_diff**2+y_diff**2)
35      return x_diff/norm, y_diff/norm
36
37
38  class Bird(pg.sprite.Sprite):
39      """
40      ゲームキャラクター (こうかとん) に関するクラス
41      """
42      delta = { # 押下キーと移動量の辞書
43          pg.K_UP: (0, -1),
44          pg.K_DOWN: (0, +1),
```

```

45     pg.K_LEFT: (-1, 0),
46     pg.K_RIGHT: (+1, 0),
47 }
48
49 ✓ def __init__(self, num: int, xy: tuple[int, int]):
50     """
51     こうかとん画像Surfaceを生成する
52     引数1 num: こうかとん画像ファイル名の番号
53     引数2 xy: こうかとん画像の位置座標タプル
54     """
55     super().__init__()
56     img0 = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
57     img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
58     self.imgs = {
59         (+1, 0): img, # 右
60         (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
61         (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
62         (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
63         (-1, 0): img0, # 左
64         (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
65         (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
66         (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
67     }
68     self.state = "normal"
69     self.hyper_life = 0
70     self.dire = (+1, 0)
71     self.image = self.imgs[self.dire]
72     self.rect = self.image.get_rect()
73     self.rect.center = xy
74     self.speed = 10
75
76
77
78 ✓ def change_img(self, num: int, screen: pg.Surface):
79     """
80     こうかとん画像を切り替え、画面に転送する
81     引数1 num: こうかとん画像ファイル名の番号
82     引数2 screen: 画面Surface
83     """
84     self.image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
85     screen.blit(self.image, self.rect)
86
87 ✓ def update(self, key_lst: list[bool], screen: pg.Surface):
88     """
89     押下キーに応じてこうかとんを移動させる
90     引数1 key_lst: 押下キーの真理値リスト
91     引数2 screen: 画面Surface
92     """
93     sum_mv = [0, 0]
94
95     for k, mv in __class__.delta.items():
96         if key_lst[k]:
97             sum_mv[0] += mv[0]
98             sum_mv[1] += mv[1]
99         if key_lst[pg.K_LSHIFT]:
100             self.speed = 20
101         else:
102             self.speed = 10
103
104
105

```

```

105
106         self.rect.move_ip(self.speed*sum_mv[0], self.speed*sum_mv[1])
107
108         if check_bound(self.rect) != (True, True):
109             self.rect.move_ip(-self.speed*sum_mv[0], -self.speed*sum_mv[1])
110         if not (sum_mv[0] == 0 and sum_mv[1] == 0):
111             self.dire = tuple(sum_mv)
112             self.image = self.imgs[self.dire]
113         if self.state == "hyper":
114             self.image = pg.transform.laplacian(self.image)
115             self.hyper_life -= 1
116         if self.hyper_life < 0:
117             self.state = "normal"
118         screen.blit(self.image, self.rect)
119
120
121
122  ✓ class Bomb(pg.sprite.Sprite):
123     """
124     爆弾に関するクラス
125     """
126     colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
127
128  ✓ def __init__(self, emy, bird: Bird, mode:str):
129     """
130     爆弾円Surfaceを生成する
131     引数1 emy : 爆弾を投下する敵機
132     引数2 bird : 攻撃対象のこうかとん
133     """
134     super().__init__()
135     self.mode = mode
136     # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
137     if mode == "emy":
138         rad = random.randint(10, 50) # 爆弾円の半径 : 10以上50以下の乱数
139         self.image = pg.Surface((2*rad, 2*rad))
140         self.color = random.choice(__class__.colors) # 爆弾円の色 : クラス変数からランダム選択
141         pg.draw.circle(self.image, self.color, (rad, rad), rad)
142         self.image.set_colorkey((0, 0, 0))
143         self.rect = self.image.get_rect()
144         self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
145         self.rect.centerx = emy.rect.centerx
146         self.rect.centery = emy.rect.centery+emy.rect.height/2
147         self.speed = 6
148         self.hp = 1 # HPの追加
149
150     elif mode == "boss":
151         self.image = pg.image.load(f"fig/fireball.png")
152         self.image = pg.transform.scale(self.image, (130, 130))
153         self.rect = self.image.get_rect()
154         self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
155         self.rect.centerx = emy.rect.centerx
156         self.rect.centery = emy.rect.centery
157         self.speed = 6
158         self.hp = 5 # HPの追加
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

```

```

165
166
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"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"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]

```

```

225         self.image = self.image[self.rect.x // 40 // 2]
226         if self.life < 0:
227             self.kill()
228
229
230     class Enemy(pg.sprite.Sprite):
231         """
232         敵機に関するクラス
233         """
234         imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
235
236     def __init__(self):
237         super().__init__()
238         self.image = random.choice(__class__.imgs)
239         self.rect = self.image.get_rect()
240         self.rect.center = WIDTH, random.randint(0, HEIGHT)
241         self.vx = +6
242         self.bound = random.randint(WIDTH/2, WIDTH) # 停止位置
243         self.state = "LEFT" # 進行状態or停止状態
244         self.interval = random.randint(50, 300) # 爆弾投下インターバル
245         self.hp = 1 # HPの追加
246
247     def update(self):
248         """
249         敵機を速度ベクトルself.vyに基づき移動（降下）させる
250         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
251         引数 screen : 画面Surface
252         """
253         if self.rect.centerx < self.bound:
254             self.vx = 0
255             self.state = "stop"
256             self.rect.centerx -= self.vx
257
258
259     class Score:
260         """
261         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
262         爆弾 : 1点
263         敵機 : 10点
264         """
265     def __init__(self):
266         self.font = pg.font.Font(None, 50)
267         self.color = (0, 0, 255)
268         self.value = 100
269         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
270         self.rect = self.image.get_rect()
271         self.rect.center = 100, HEIGHT-50
272
273     def update(self, screen: pg.Surface):
274         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
275         screen.blit(self.image, self.rect)
276
277
278     class life_gage(pg.sprite.Sprite):
279         """
280         ライフゲージの作成
281         rectを用いてHPバーを作成し被弾するにつれてHPバーが減るようにする
282         """
283
284     def __init__(self, screen):
285         self.colors = [(255, 0, 0), (0, 255, 0), (255, 255, 0)] # ライフゲージの減少に伴って色の変化をさせる

```

```

286         self.screen = screen
287
288         # hpバーの幅・高さ
289         self.hp_width = 300 # + 6
290         self.hp_height = 40 #+ 6
291
292         # HPバーを表示させたい位置
293         self.hp_x = 160
294         self.hp_y = 700
295
296         self.hp = 100 #HPの基準
297
298         self.HP_bar = pg.Surface((self.hp_width,self.hp_height)) #HPバーの土台作り
299         self.HP_back = pg.Surface((self.hp_width +6,self.hp_height+6)) #HPバーの枠兼減少の値がわかりやすいよう
300         pg.draw.rect(self.HP_back, (0, 0, 0), (0,0,self.hp_width,self.hp_height)) #HPバーの枠作成
301         pg.draw.rect(self.HP_bar, self.colors[1], (0,0,self.hp_width,self.hp_height)) #HPバー（緑色）の作成
302         self.rect = self.HP_bar.get_rect()
303         self.rect_back = self.HP_back.get_rect()
304
305         self.rect.centerx = self.hp_x
306         self.rect.centery = self.hp_y
307
308         self.rect_back.center = (self.hp_x, self.hp_y)
309
310         self.screen.blit(self.HP_back, self.rect_back)
311         self.screen.blit(self.HP_bar, self.rect)
312
313
314     def update(self,screen):
315         # pg.draw.rect("(0,0)~(hp,10)まで表示")
316         self.HP_bar = pg.Surface((300,self.hp_height))
317         if self.hp <= 30: #HPが30以下の時表示を赤色にする
318             pg.draw.rect(self.HP_bar, self.colors[0], (0,0,self.hp_width,self.hp_height))
319         elif self.hp <= 60: #HPが60以下の時表示を黄色にする
320             pg.draw.rect(self.HP_bar, self.colors[2], (0,0,self.hp_width,self.hp_height))
321         else: #それ以外時表示を赤色にする
322             pg.draw.rect(self.HP_bar, self.colors[1], (0,0,self.hp_width,self.hp_height))
323         self.rect = self.HP_bar.get_rect()
324         self.rect.centerx = self.hp_x
325         self.rect.centery = self.hp_y
326         screen.blit(self.HP_back, self.rect_back) #画面に反映
327         screen.blit(self.HP_bar, self.rect) #画面に反映
328
329     def damages(self, damage):
330         self.damege = damage
331         self.damege*=3 #303行で300としているため100に割合をあわせるため
332         self.hp_width -= self.damege
333
334         print(damege, self.damege, self.hp_width) #ちゃんと実行されてHPが減っているかの確認
335         self.hp -= damege
336
337
338     class Shield(pg.sprite.Sprite):
339         """
340         SPを3消費してこうかとんを守る防御壁を出現させるクラス
341         Caps lock押下で出現
342         """
343
344     def __init__(self, bird : Bird, life):
345         super().__init__()

```



```
346         self.life = life
347         self.image = pg.Surface((20, bird.rect.height*2))
348         pg.draw.rect(self.image, (0, 0, 255), (0, 0, 20, bird.rect.height*2))
349
350         vx, vy = bird.dire
351         deg = math.degrees(math.atan2(-vy, vx))
352         self.image = pg.transform.rotozoom(self.image, deg, 1.0)
353         self.image.set_colorkey((0, 0, 0))
354         self.rect = self.image.get_rect()
355         self.rect.centerx = bird.rect.centerx + bird.rect.width * vx
356         self.rect.centery = bird.rect.centery + bird.rect.height * vy
357
358
359     def update(self):
360         self.life -= 1
361         if self.life < 0:
362             self.kill()
363
364
365     class Gravity(pg.sprite.Sprite):
366         """
367         画面全体を覆う重力場を発生させる
368         """
369     def __init__(self, life):
370         super().__init__()
371         self.image = pg.Surface((1600,900))
372         pg.draw.rect(self.image, (0, 0, 0), (0, 0, 1600,900))
373         self.image.set_alpha(200)
374         self.rect = self.image.get_rect()
375         self.rect.center = (WIDTH/2, HEIGHT/2)
376         self.life = life # 発動時間
377
378     def update(self):
379         self.life -= 1
380         if self.life < 0:
381             self.kill()
382
383
384
385     def update(self):
386         self.life -= 1
387         if self.life < 0:
388             self.kill()
389
390
391     class EMP(pg.sprite.Sprite):
392     def __init__(self, Enemy, Bomb, Surface): #敵機、爆弾、surfaceを与えている
393         for emy in Enemy:
394             emy.interval = math.inf
395             emy.image = pg.transform.laplacian(emy.image)
396             emy.image.set_colorkey((0, 0, 0))
397
398         for bomb in Bomb:
399             bomb.speed /= 2
400
401     def update(self):
402         self.life -= 1
403         if self.life < 0:
404             self.kill()
405
```

```
406
407  class change_Boss():
408     """
409     1キーを押したときにボス戦警告のポップアップ表示
410     Yesを押したときボス戦開始エフェクト+背景切り替え
411     """
412  def __init__(self, screen):
413      self.screen = screen
414      self.rct = pg.Surface((800, 450)) # Surfaceを生成
415      pg.draw.rect(self.rct, (128, 128, 128), (0, 0, 800, 450)) # 灰色の矩形を生成
416      # screen.blit(self.rct, (WIDTH/2, WIDTH/2))
417      # time.sleep(1)
418      # self.bg_img = pg.image.load(f"fig/aozora.jpg") # ボス戦の画像読み込み
419      # self.bg_img = pg.transform.scale(self.bg_img, (WIDTH, HEIGHT)) # ボス戦の背景画像サイズを調整
420
421      self.font = pg.font.Font(None, 130) # 文字の大きさ
422
423      # 文字色の設定
424      self.color = (255, 255, 255)
425      self.color_red = (255, 0, 0)
426
427      self.Yes = self.font.render("Yes", 0, self.color)
428      self.No = self.font.render("No", 0, self.color)
429      self.Ready = self.font.render("Are you ready ?", 0, self.color_red)
430
431      # 透明度を設定
432      self.rct.set_alpha(200)
433      self.Yes.set_alpha(200)
434      self.No.set_alpha(200)
435      self.Ready.set_alpha(200)
436
437      # 形の値を取得
438      self.rect = self.Yes.get_rect()
439      self.rect2 = self.No.get_rect()
440      self.rect3 = self.Ready.get_rect()
441
442      self.rect.center = 600, 600 # "Yes"の位置設定
443      self.rect2.center = 1000, 600 # "No"の位置設定
444      self.rect3.center = 800, 400 # "Are you ready?"の位置設定
445
446      self.change = True # ポップアップが表示されている状態に設定 (True : 表示, False: 非表示)
447
448  def update(self, screen:pg.Surface):
449      screen.blit(self.rct, (400, 300)) # Surfaceの描画、座標設定
450      screen.blit(self.Yes, self.rect) # Yesの描画
451      screen.blit(self.No, self.rect2) # Noの描画
452      screen.blit(self.Ready, self.rect3) # Are you ready?の描画
453
454  def fade_in_out(self):
455      fade_surface = pg.Surface((WIDTH, HEIGHT))
456      fade_surface.fill((0, 0, 0))
457      for i in range(0, 256, 5): # フェードイン
458          fade_surface.set_alpha(i)
459          self.screen.blit(fade_surface, (0, 0))
460          pg.display.flip()
461          pg.time.delay(30)
462
463      # self.screen.blit(self.bg_img, (0, 0)) # 背景画像を描画
464      # pg.display.flip()
465      # time.sleep(5) # 背景画像を3秒間表示
```

```
466
467
468  class Powerup:
469     """
470     攻撃力の概念の追加
471     打ち落とした爆弾が赤色だと攻撃力+1
472     """
473  def __init__(self):
474     self.font = pg.font.Font(None, 50)
475     self.color = (0, 0, 255)
476     self.value = 1
477     self.image = self.font.render(f"Power: {self.value}", 0, self.color) # 画面に攻撃力の数値を追加
478     self.rect = self.image.get_rect()
479     self.rect.center = 87, HEIGHT-100
480
481  def update(self, screen: pg.Surface):
482     self.image = self.font.render(f"Power: {self.value}", 0, self.color) # 数値を更新
483     screen.blit(self.image, self.rect)
484
485
486  class Skillpoint:
487     """
488     スキルポイントの概念の追加
489     打ち落とした爆弾が青色だとSP+1
490     """
491  def __init__(self):
492     self.font = pg.font.Font(None, 50)
493     self.color = (0, 0, 255)
494     self.value = 1
495     self.image = self.font.render(f"SP: {self.value}", 0, self.color) # 画面にスキルポイントの数値を追加
496     self.rect = self.image.get_rect()
497     self.rect.center = 58, HEIGHT-150
498
499  def update(self, screen: pg.Surface):
500     self.image = self.font.render(f"SP: {self.value}", 0, self.color) # 数値を追加
501     screen.blit(self.image, self.rect)
502
503
504  class Boss(pg.sprite.Sprite):
505  def __init__(self, ):
506     super().__init__()
507     self.boss_flag = False
508     self.image = pg.image.load(f"fig/last.png") #ボス画像の読み込み
509     self.image = pg.transform.scale(self.image, (WIDTH/3, HEIGHT*(6/10))) #画像の大きさ調整
510     self.rect = self.image.get_rect() #座標の取得
511     self.rect.center = (WIDTH*(17/20), HEIGHT/2) #座標の指定
512     self.B_HP = 100 #ボスのHP
513
514  def update(self, M_life, score: Score):
515     self.B_HP -= M_life
516     print(self.B_HP)
517     if self.B_HP <= 0:
518         score.value += 200
519         self.boss_flag = False
520         self.kill()
521
522
523  def main():
524     pg.display.set_caption("こうかとん伝説（仮）")
525     screen = pg.display.set_mode((WIDTH, HEIGHT))
```

```
526     bg_img = pg.image.load(f"fig/pg_bg.jpg")
527     # bg_img2 = pg.image.load(f"fig/aozora.jpg") # ボス戦の背景画像
528     # bg_img2 = pg.transform.scale(bg_img2, (WIDTH, HEIGHT)) # ボス戦の背景画像サイズを調整
529
530     # wait = WebDriverWait(driver=driver, timeout=60)
531     score = Score()
532     score.value = 99999 # 実行確認のために仮置き、後で消す
533     power = Powerup()
534     sp = Skillpoint()
535     sp.value = 99999 # 実行確認のために仮置き、後で消す
536     # changeBoss = None # 初期状態を設定
537     bird = Bird(3, (900, 400))
538     bombs = pg.sprite.Group()
539     beams = pg.sprite.Group()
540     exps = pg.sprite.Group()
541     emys = pg.sprite.Group()
542     shields = pg.sprite.Group()
543     gravity = pg.sprite.Group()
544     bosses = pg.sprite.Group()
545     boss = Boss()
546     hp = life_gage(screen)
547     mode = "emy"
548     bg_img_b = pg.image.load(f"fig/aozora.jpg")
549     pop_flag = False
550     #スライド
551
552     bg_img2 = pg.transform.flip(bg_img, True, False)
553     kk_img = pg.transform.flip(bg_img, True, False)
554     # kk_rct = kk_img.get_rect()
555     tmr = 0
556     clock = pg.time.Clock()
557
558     while True:
559         x=0
560         y=0
561         key_lst = pg.key.get_pressed()
562         for event in pg.event.get():
563             if event.type == pg.QUIT:
564                 return 0
565             if event.type == pg.KEYDOWN and event.key == pg.K_1 and (score.value >= 300): # 1キーを押したと
566
567                 pop_flag = True
568                 changeBoss = change_Boss(screen) # クラス呼び出し
569                 # score.value -= 100
570                 # boss.boss_flag = True
571                 # bosses.add(boss)
572             if event.type == pg.KEYDOWN and pop_flag == True and event.key == pg.K_y:
573                 # changeBoss = change_Boss(screen) # クラス呼び出し
574                 score.value -= 100
575                 boss.boss_flag = True
576                 bosses.add(boss)
577                 changeBoss.fade_in_out()
578
579                 pop_flag = False
580
581             if event.type == pg.KEYDOWN and pop_flag == True and event.key == pg.K_n:
582                 # changeBoss.change = False # ポップアップを閉じる
583                 pop_flag = False
584
585             if boss.boss_flag == True:
```

```

586         mode = "boss"
587         # changeBoss.change = False # ポップアップを閉じる
588         # if event.type == pg.KEYDOWN and event.key == pg.K_n:
589         #     changeBoss = None # ポップアップを閉じる
590
591     if boss.boss_flag == False:
592         mode = "emy"
593     if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
594         beams.add(Beam(bird))
595     if event.type == pg.KEYDOWN and event.key == pg.K_RSHIFT and (sp.value >= 5): # 右シフトキーを
596         bird.hyper_life = 500
597         sp.value -= 5 # 消費SP
598         bird.state = "hyper"
599     if event.type == pg.KEYDOWN and event.key == pg.K_RETURN and sp.value >= 10: # エンター押したと
600         gravity.add(Gravity(400))
601         sp.value -= 10 # 消費SP
602     if event.type == pg.KEYDOWN and event.key == pg.K_e:
603         if sp.value > 8:
604             EMP(emy, bombs, screen)
605             sp.value -= 8 # 消費SP
606     if event.type == pg.KEYDOWN and event.key == pg.K_w and sp.value >= 3 and len(shields) == 0: #
607         sp.value -= 3 # 消費SP
608         shields.add(Shield(bird, 400))
609         print(len(shields))
610     # screen.blit(bg_img, [0, 0])
611     print(mode)
612     #スライド
613     if mode == "emy":
614         # kk_rct.move_ip((x-1,y))
615         x=tmr%3200
616         screen.blit(bg_img, [-x, 0])
617         screen.blit(bg_img2, [-x+1600,0])
618         screen.blit(bg_img, [-x+3200,0])
619         screen.blit(bg_img2, [-x+4800,0])
620
621         # screen.blit(kk_img,kk_rct)
622     elif mode == "boss":
623         screen.blit(bg_img_b, [0, 0])
624
625     if boss.boss_flag == True:
626         if tmr%30 == 0:
627             bombs.add(Bomb(boss, bird, "boss"))
628
629     if tmr%200 == 0: # 200フレームに1回, 敵機を出現させる
630         emys.add(Enemy())
631
632     for emy in emys:
633         if emy.state == "stop" and tmr%emy.interval == 0:
634             # 敵機が停止状態に入ったら, intervalに応じて爆弾投下
635             bombs.add(Bomb(emy, bird, "emy"))
636
637     for emy in pg.sprite.groupcollide(emy, beams, True, True).keys():
638         emy.hp -= power.value # 敵のHPを自分の攻撃力分だけ削る
639         if emy.hp <= 0: # 敵のHPが0以下の時
640             exps.add(Explosion(emy, 100)) # 爆発エフェクト
641             score.value += 10 # 10点アップ
642             bird.change_img(6, screen) # こうかたん喜びエフェクト
643
644     for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
645         bomb.hp -= power.value # 爆弾の耐久力を自分の攻撃力分だけ削る

```

```

646     if bomb.hp <= 0: # 爆弾の耐久力が0以下の時
647         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
648         score.value += 1 # 1点アップ
649     if bomb.mode == "emy":
650         if bomb.color == (255, 0, 0): # 敵の爆弾の色が赤色のとき
651             power.value += 1 # 攻撃力アップ
652         if bomb.color == (0, 0, 255): # 敵の爆弾の色が青色のとき
653             sp.value += 1 # スキルポイントアップ
654     """
655     HPのクラスが追加されたら追加する
656     今回はマージできないので追加しない
657     """
658     #if bomb.color == (0, 255, 0):
659     #    hp.value += 1 # HP回復
660
661 for bomb in pg.sprite.spritecollide(bird, bombs, True):
662     if bird.state == "hyper":
663         exps.add(Explosion(bomb, 50))
664         score.value += 1
665     if bird.state == "normal":
666         bird.change_img(8, screen) # こうかたん悲しみエフェクト
667         score.update(screen)
668
669     # ダメージ判定
670     hp.dameges(10) #ダメージの割合、今回は敵の攻撃をくらったら10分の1ずつHPが減っていくようにする
671     # pg.display.update()
672     # time.sleep(2)
673     # time.sleep(1)
674     if hp.hp <= 0: #HPがなくなったらゲームオーバー
675         fonto = pg.font.Font(None, 80)
676         screen.fill((50, 0, 0))
677         txt = fonto.render("GameOver", True, (255, 0, 0))
678         screen.blit(txt, [WIDTH/2 -150, HEIGHT/2])
679         pg.display.update()
680         time.sleep(4)
681         return
682
683 # for -----
684 # if bossと当たったら チームメンバーがボスを作ったとき用
685
686 for bomb in pg.sprite.groupcollide(bombs, gravity, True, False).keys():
687     bomb.hp -= power.value # 爆弾の耐久力を自分の攻撃力分だけ削る
688     if bomb.hp <= 0: # 爆弾の耐久力が0以下の時
689         exps.add(Explosion(bomb, 50))
690         score.value += 1
691
692 for emy in pg.sprite.groupcollide(emy, gravity, True, False).keys():
693     emy.hp -= power.value # 敵のHPを自分の攻撃力分だけ削る
694     if emy.hp <= 0: # 敵のHPが0以下の時
695         exps.add(Explosion(emy, 100))
696         score.value += 10
697         bird.change_img(6, screen) # こうかたん喜びエフェクト
698
699
700 for bomb in pg.sprite.groupcollide(bombs, shields, True, False).keys():
701     exps.add(Explosion(bomb, 50))
702     score.value += 1
703 if len(pg.sprite.spritecollide(bird, bombs, True)) != 0:
704     bird.change_img(8, screen) # こうかたん悲しみエフェクト
705     score.update(screen)

```

```
706         pg.display.update()
707         time.sleep(2)
708         return
709     for beam in pg.sprite.groupcollide(beams, bosses, True, False).keys(): #ボスとビームの衝突判定、爆発:
710         exps.add(Explosion(beam, 50))
711         bosses.update(10, score)
712     hp.update(screen)
713     bird.update(key_lst, screen)
714     beams.update()
715     beams.draw(screen)
716     emys.update()
717     emys.draw(screen)
718     bosses.draw(screen)
719     bombs.update()
720     bombs.draw(screen)
721     exps.update()
722     exps.draw(screen)
723     gravity.update()
724     gravity.draw(screen)
725     score.update(screen)
726     shields.update()
727     shields.draw(screen)
728     power.update(screen)
729     sp.update(screen)
730
731     # pop up
732     if pop_flag == True :
733         changeBoss.update(screen)
734     pg.display.update()
735     tmr += 1
736     clock.tick(50)
737
738
739 if __name__ == "__main__":
740     pg.init()
741     main()
742     pg.quit()
743     sys.exit()
```