






c0a2302985 /
ProjExD_Group04



[Code](#) [Issues](#) [Pull requests](#) **1** [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insight](#)

ProjExD_Group04 / README.md 

c0a2302985 基本機能

198846c · 3 days ago



35 lines (23 loc) · 1.39 KB

Preview

Code

Blame

Raw



エアホッケー

実行環境の必要条件

- python >= 3.10
- pygame >= 2.1

ゲームの概要

- こうかとんを操作するエアホッケー

ゲームの遊び方

- 矢印キーかWASDキーでこうかとんを操作し、特定のキーを押すことによりスキル発動
- 一定の時間でゲーム終了とし、その時点で点数が多い方が勝利

ゲームの実装

共通基本機能

- 背景画像と1P・2Pのキャラクター描画

分担追加機能

- フェイクのたまを出すスキル(担当: やたべ) : どっかのボタン押したらスキル発動・フェイクのほうは一定時間たったら消える
- たま拡大/縮小(担当: いさな) : j キーを押したらスキル発動(発動時間あり)・一定時間たったら元の大きさに戻る
- 無敵の壁(担当: るな) : どっかのボタン押したらスキル発動・ゴールの色変える・一定時間たったら元に戻る



- スコア二倍（担当：ませ）：10%の確率でスコア2倍のボールを出現させる
- 爆弾とこうかとんの衝突判定（担当：おがわ）：爆弾がこうかとんに衝突した際に正しい挙動で反射するようにする

ToDo






- 持ち手拡大
- ゴール幅狭くなる

メモ

- 「count_?」でクールタイムの設定



c0a2302985 /
ProjExD_Group04



<> Code Issues Pull requests 1 Actions Projects Wiki Security Insight

ProjExD_Group04 / main.py



c0a2302985 コメントの修正

23af174 · 11 minutes ago



308 lines (271 loc) · 10.4 KB

Code Blame Raw Copy Download Edit

```
1  import math
2  import os
3  import random
4  import sys
5  import time
6  import pygame as pg
7
8
9  WIDTH = 800 # ゲームウィンドウの幅
10 HEIGHT = 600 # ゲームウィンドウの高さ
11 os.chdir(os.path.dirname(os.path.abspath(__file__)))
12
13
14 def check_bound(obj_rct: pg.Rect) -> tuple[bool, bool]:
15     """
16     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
17     引数：こうかとんや爆弾、ビームなどのRect
18     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）
19     """
20     yoko, tate = True, True
21     if obj_rct.left < 0 or WIDTH < obj_rct.right:
22         yoko = False
23     if obj_rct.top < 0 or HEIGHT < obj_rct.bottom:
24         tate = False
25     return yoko, tate
26
27
28 class Bird:
29     """
30     ゲームキャラクター（こうかとん）に関するクラス
31     """
32     delta = { # 押下キーと移動量の辞書
33         pg.K_UP: (0, -5),
34         pg.K_DOWN: (0, +5),
35         pg.K_LEFT: (-5, 0),
36         pg.K_RIGHT: (+5, 0),
37     }
38     img0 = pg.transform.rotozoom(pg.image.load("fig/3.png"), 0, 0.9)
39     img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん（右向き）
40     imgs = { # 0度から反時計回りに定義
41         (+5, 0): img, # 右
42         (+5, -5): pg.transform.rotozoom(img, 45, 0.9), # 右上
```

```

43         (0, -5): pg.transform.rotozoom(img, 90, 0.9), # 上
44         (-5, -5): pg.transform.rotozoom(img0, -45, 0.9), # 左上
45         (-5, 0): img0, # 左
46         (-5, +5): pg.transform.rotozoom(img0, 45, 0.9), # 左下
47         (0, +5): pg.transform.rotozoom(img, -90, 0.9), # 下
48         (+5, +5): pg.transform.rotozoom(img, -45, 0.9), # 右下
49     }
50
51     def __init__(self, xy: tuple[int, int]):
52         """
53         こうかとおん画像Surfaceを生成する
54         引数 xy: こうかとおん画像の初期位置座標タプル
55         """
56         self.img = __class__.imgs[(-5, 0)]
57         self.rct: pg.Rect = self.img.get_rect()
58         self.rct.center = xy
59         self.dire = (+5, 0)
60
61     def change_img(self, num: int, screen: pg.Surface):
62         """
63         こうかとおん画像を切り替え、画面に転送する
64         引数1 num: こうかとおん画像ファイル名の番号
65         引数2 screen: 画面Surface
66         """
67         self.img = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 0.9)
68         screen.blit(self.img, self.rct)
69
70     def update(self, key_lst: list[bool], screen: pg.Surface):
71         """
72         押下キーに応じてこうかとおんを移動させる
73         引数1 key_lst: 押下キーの真理値リスト
74         引数2 screen: 画面Surface
75         """
76         sum_mv = [0, 0]
77         for k, mv in __class__.delta.items():
78             if key_lst[k]:
79                 sum_mv[0] += mv[0]
80                 sum_mv[1] += mv[1]
81         self.rct.move_ip(sum_mv)
82         if check_bound(self.rct) != (True, True):
83             self.rct.move_ip(-sum_mv[0], -sum_mv[1])
84         if not (sum_mv[0] == 0 and sum_mv[1] == 0):
85             self.img = __class__.imgs[tuple(sum_mv)]
86         screen.blit(self.img, self.rct)
87         if sum_mv != [0, 0]:
88             self.dire = (sum_mv[0], sum_mv[1])
89
90     class Bird2:
91         """
92         ゲームキャラクター（こうかとおん）に関するクラス
93         """
94         delta = { # 押下キーと移動量の辞書
95             pg.K_w: (0, -5),
96             pg.K_s: (0, +5),
97             pg.K_a: (-5, 0),
98             pg.K_d: (+5, 0),
99         }
100         img0 = pg.transform.rotozoom(pg.image.load("fig/3.png"), 0, 0.9)
101         img = pg.transform.flip(img0, True, False) # デフォルトのこうかとおん（右向き）

```

```

102  imgs = { # 0度から反時計回りに定義
103      (+5, 0): img, # 右
104      (+5, -5): pg.transform.rotozoom(img, 45, 0.9), # 右上
105      (0, -5): pg.transform.rotozoom(img, 90, 0.9), # 上
106      (-5, -5): pg.transform.rotozoom(img, -45, 0.9), # 左上
107      (-5, 0): img0, # 左
108      (-5, +5): pg.transform.rotozoom(img, 45, 0.9), # 左下
109      (0, +5): pg.transform.rotozoom(img, -90, 0.9), # 下
110      (+5, +5): pg.transform.rotozoom(img, -45, 0.9), # 右下
111  }
112
113  def __init__(self, xy: tuple[int, int]):
114      """
115      こうかとん画像Surfaceを生成する
116      引数 xy: こうかとん画像の初期位置座標タプル
117      """
118      self.img = __class__.imgs[(+5, 0)]
119      self.rct: pg.Rect = self.img.get_rect()
120      self.rct.center = xy
121      self.dire = (+5, 0)
122
123  def change_img(self, num: int, screen: pg.Surface):
124      """
125      こうかとん画像を切り替え、画面に転送する
126      引数1 num: こうかとん画像ファイル名の番号
127      引数2 screen: 画面Surface
128      """
129      self.img = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 0.9)
130      screen.blit(self.img, self.rct)
131
132  def update(self, key_lst: list[bool], screen: pg.Surface):
133      """
134      押下キーに応じてこうかとんを移動させる
135      引数1 key_lst: 押下キーの真理値リスト
136      引数2 screen: 画面Surface
137      """
138      sum_mv = [0, 0]
139      for k, mv in __class__.delta.items():
140          if key_lst[k]:
141              sum_mv[0] += mv[0]
142              sum_mv[1] += mv[1]
143      self.rct.move_ip(sum_mv)
144      if check_bound(self.rct) != (True, True):
145          self.rct.move_ip(-sum_mv[0], -sum_mv[1])
146      if not (sum_mv[0] == 0 and sum_mv[1] == 0):
147          self.img = __class__.imgs[tuple(sum_mv)]
148      screen.blit(self.img, self.rct)
149      if sum_mv != [0, 0]:
150          self.dire = (sum_mv[0], sum_mv[1])
151
152
153  class Bomb:
154      """
155      爆弾に関するクラス
156      """
157  def __init__(self, color: tuple[int, int, int], rad: int):
158      """
159      引数に基づき爆弾円Surfaceを生成する

```

```
160     引数1 color: 爆弾円の色タプル
161     引数2 rad: 爆弾円の半径
162     """
163     self.img = pg.Surface((2*rad, 2*rad))
164     pg.draw.circle(self.img, color, (rad, rad), rad)
165     self.img.set_colorkey((0, 0, 0))
166     self.rct = self.img.get_rect()
167     self.rct.center = random.randint(0, WIDTH), random.randint(0, HEIGHT)
168     self.vx, self.vy = +5, +5
169
170     def update(self, screen: pg.Surface):
171         """
172         爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
173         引数 screen: 画面Surface
174         """
175         yoko, tate = check_bound(self.rct)
176         if not yoko:
177             self.vx *= -1
178         if not tate:
179             self.vy *= -1
180         self.rct.move_ip(self.vx, self.vy)
181         screen.blit(self.img, self.rct)
182
183
184     class Score:
185     def __init__(self):
186         self.fonto = pg.font.SysFont("hgp創英角ゴ 体", 30)
187         self.score = 0
188         self.img = self.fonto.render(f"スコア: {self.score}", 0, (0, 0, 255))
189         self.rct = self.img.get_rect()
190         self.rct.center = (100, HEIGHT-50)
191
192     def update(self, screen):
193         self.img = self.fonto.render(f"スコア: {self.score}", 0, (0, 0, 255))
194         screen.blit(self.img, self.rct)
195
196
197     class Explosion:
198     def __init__(self, bomb: Bomb):
199         self.img1 = pg.image.load(f"fig/explosion.gif")
200         self.img2 = pg.transform.flip(self.img1, True, True)
201         self.imgs = [self.img1, self.img2]
202         self.rct = self.img1.get_rect()
203         self.rct.center = bomb.rct.center
204         self.life = 50
205
206     def update(self, screen):
207         self.life -= 1
208         if self.life > 0:
209             ind = (self.life // 10) % 2
210             screen.blit(self.imgs[ind], self.rct)
211
212
213     class Limit:
214     def __init__(self):
215         self.fonto = pg.font.SysFont("hgp創英角ゴ 体", 30)
216         self.time = 1000
217         self.img = self.fonto.render(f"制限時間: {self.time}", 0, (255, 0, 0))
```

```
218         self.rct = self.img.get_rect()
219         self.rct.center = (100, 50)
220
221     def update(self, screen):
222         self.img = self.fonto.render(f"制限時間：{self.time}", 0, (0, 0, 255))
223         screen.blit(self.img, self.rct)
224
225
226     def check_coll(bomb: Bomb, bird: Bird) -> None:
227         """
228         爆弾とこうかとの衝突処理を行う関数
229
230         引数:
231             bomb (Bomb): 衝突対象の爆弾オブジェクト
232             bird (Bird): 衝突対象のこうかとんオブジェクト
233
234         戻り値:
235             None
236         """
237         hit_margin = 10
238         if abs(bomb.rct.bottom - bird.rct.top) < hit_margin and bomb.vy > 0:
239             bomb.vy *= -1
240         elif abs(bomb.rct.top - bird.rct.bottom) < hit_margin and bomb.vy < 0:
241             bomb.vy *= -1
242         elif abs(bomb.rct.left - bird.rct.right) < hit_margin and bomb.vx < 0:
243             bomb.vx *= -1
244         elif abs(bomb.rct.right - bird.rct.left) < hit_margin and bomb.vx > 0:
245             bomb.vx *= -1
246
247
248     def main():
249         NUM_OF_BOMBS = 1
250         pg.display.set_caption("たたかえ！こうかとん")
251         screen = pg.display.set_mode((WIDTH, HEIGHT))
252         bg_img = pg.image.load("fig/pg_bg.jpg")
253         bird = Bird((WIDTH-100, HEIGHT/2))
254         bird2 = Bird2((100, HEIGHT/2))
255         bomb = Bomb((255, 0, 0), 10)
256         bombs = [Bomb((255, 0, 0), 10) for _ in range(NUM_OF_BOMBS)]
257         clock = pg.time.Clock()
258         score = Score()
259         expls = []
260         limit = Limit()
261         tmr = 0
262         while True:
263             for event in pg.event.get():
264                 if event.type == pg.QUIT:
265                     return
266             screen.blit(bg_img, [0, 0])
267
268             if limit.time == 0:
269                 fonto = pg.font.Font(None, 80)
270                 txt = fonto.render("end", True, (255, 0, 0))
271                 screen.blit(txt, [WIDTH//2-80, HEIGHT//2])
272                 pg.display.update()
273                 time.sleep(1)
274                 return
275
```

```
276     # 爆弾とこうかとん1の衝突判定
277     for bomb in bombs:
278         if bird.rct.colliderect(bomb.rct):
279             check_coll(bomb, bird)
280
281     # 爆弾とこうかとん2の衝突判定
282     for bomb in bombs:
283         if bird2.rct.colliderect(bomb.rct):
284             check_coll(bomb, bird2)
285
286     key_lst = pg.key.get_pressed()
287     bird.update(key_lst, screen)
288     bird2.update(key_lst, screen)
289     bombs = [bomb for bomb in bombs if bomb is not None] # Noneでないもののリスト
290     for bomb in bombs:
291         bomb.update(screen)
292     score.update(screen)
293     expls = [expl for expl in expls if expl.life > 0]
294     for expl in expls:
295         expl.update(screen)
296     if (tmr != 0) and (tmr % 50 == 0):
297         limit.time -= 1
298     limit.update(screen)
299     pg.display.update()
300     tmr += 1
301     clock.tick(50)
302
303
304 if __name__ == "__main__":
305     pg.init()
306     main()
307     pg.quit()
308     sys.exit()
```