 c0a22074e1 / ProjExD_05 Public

<> Code

Issues

Pull requests

Actions


Projects


Wiki

Security

Insights

Settings


 main



ProjExD_05 / Documents / ProjExD2023 / ex05 / README.md 

Go to file

t

...

 c0a22074e1 完成

11 minutes ago  

22 lines (17 loc) · 592 Bytes

Preview

Code

Blame

Raw



ゲームのタイトル

実行環境の必要条件

- python >= 3.10
- pygame >= 2.1

ゲームの概要

主人公キャラクター工科丸をキーボード上下操作でコインを獲得するゲームで、敵機に当たったら終了します。

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
- コインに関するクラス
- 敵機に関するクラス

担当追加機能

- ボーナスコインに関するクラス

ToDo

- ☐ ボーナス画面に完全変化

メモ

- BonusはCoinと同様の使い方

c0a22074e1 / ProjExD_05 Public[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)[main](#) [ProjExD_05 / Documents / ProjExD2023 / ex05](#)
/ game.py

Go to file

t



c0a22074e1 完成

12 minutes ago



199 lines (163 loc) · 5.8 KB

Code

Blame

Raw



```
1  import pygame as pg
2  import sys
3  import random
4
5  WIDTH = 1600
6  HEIGHT = 900
7
8
9  def check_bound(area: pg.Rect, obj: pg.Rect) -> tuple[bool, bool]:
10     yoko, tate = True, True
11     if obj.left < area.left or area.right < obj.right: # 横方向のはみ出し判定
12         yoko = False
13     if obj.top < area.top or area.bottom < obj.bottom: # 縦方向のはみ出し判定
14         tate = False
15     return yoko, tate
16
17
18  class Bird(pg.sprite.Sprite):
19     """
20     ゲームキャラクター（こうかとん）に関するクラス
21     """
22     delta = {
23         pg.K_UP: (0, -1),
24         pg.K_DOWN: (0, +1),
25     }
26
27  def __init__(self, xy: tuple[int, int]):
28     """
29     こうかとん画像Surfaceを生成する
30     引数1 num: こうかとん画像ファイル名の番号
31     引数2 xy: こうかとん画像の位置座標タプル
32     """
33     self.image = pg.transform.flip(pg.image.load("ex05/fig/3.png"), True, False)
34
35     self.rect = self.image.get_rect()
36     self.rect.center = xy
37     self.speed = 10
38
39  def update(self, key_lst: list[bool], screen: pg.Surface):
40     """
41     押下キーに応じてこうかとんを移動させる
42     引数1 key_lst: 押下キーの真値リスト
43     引数2 screen: 画面Surface
```

```
44         """
45         for k, mv in __class__.delta.items():
46             if key_lst[k]:
47                 self.rect.move_ip(mv)
48             if check_bound(screen.get_rect(), self.rect) != (True, True):
49                 for k, mv in __class__.delta.items():
50                     if key_lst[k]:
51                         self.rect.move_ip(-mv[0], -mv[1])
52
53         screen.blit(self.image, self.rect)
54
55
56 ✓ class Coin(pg.sprite.Sprite):
57     """
58     コインに関するクラス
59     """
60 ✓ def __init__(self, color: tuple[int, int, int], rad: int):
61     """
62     引数に基づきコインSurfaceを生成する
63     引数1 color : コインの色タプル
64     引数2 rad : コインの半径
65     """
66     super().__init__()
67     self.image = pg.Surface((2*rad, 2*rad))
68     pg.draw.circle(self.image, color, (rad, rad), rad)
69     self.image.set_colorkey((0, 0, 0))
70     self.rect = self.image.get_rect()
71     self.rect.center = (WIDTH-30, random.randint(0, HEIGHT))
72     self.vx, self.vy = +1, +1
73
74     def update(self):
75         self.rect.move_ip(-1, 0)
76
77
78 ✓ class Bonus(pg.sprite.Sprite):
79     """
80     ボーナスコインに関するクラス
81     """
82 ✓ def __init__(self, color: tuple[int, int, int], rad: int):
83     """
84     引数に基づきボーナスコインSurfaceを生成する
85     引数1 color : ボーナスコインの色タプル
86     引数2 rad : ボーナスコインの半径
87     """
88     super().__init__()
89     self.image = pg.Surface((2*rad, 2*rad))
90     pg.draw.circle(self.image, color, (rad, rad), rad)
91     self.image.set_colorkey((0, 0, 0))
92     self.rect = self.image.get_rect()
93     self.rect.center = (WIDTH, 450)
94     self.vx = +1
95
96     def update(self):
97         self.rect.move_ip(-1, 0)
98
99
100 ✓ class Enemy(pg.sprite.Sprite):
101     """
102     敵機に関するクラス
103     """
```

```
104 ✓ def __init__(self):
105     super().__init__()
106     self.image = pg.image.load(f"ex05/fig/alien1.png")
107     self.rect = self.image.get_rect()
108     self.rect.center = (WIDTH-40, random.randint(0, HEIGHT))
109
110 def update(self):
111     self.rect.move_ip(-1, 0)
112
113
114 ✓ def main():
115     pg.display.set_caption("はばたけ！ こうかとん")
116     screen = pg.display.set_mode((WIDTH, HEIGHT))
117     clock = pg.time.Clock()
118     bg_img = pg.image.load("ex05/fig/pg_bg.jpg")
119     bg_imgs = pg.transform.flip(bg_img, True, False)
120
121     bird = Bird([100, 200])
122     coins = pg.sprite.Group()
123     bonusC = pg.sprite.Group()
124     emys = pg.sprite.Group()
125
126     tmr = 0
127     flag = False
128
129     while True:
130         for event in pg.event.get():
131             if event.type == pg.QUIT: return
132
133         # フラグに応じてコインの生成を制御
134         if not flag and tmr % 700 == 0:
135             for i in range(3):
136                 coins.add(Coin((255, 0, 0), 30))
137
138         # フラグに応じて敵機の生成を制御
139         if not flag and tmr % 700 == 0:
140             emys.add(Enemy())
141
142         if tmr % 3000 == 0:
143             flag = False
144         elif tmr % 7000 == 0:
145             flag = True
146             coins.empty() # コインの削除
147             emys.empty() # 敵機の削除
148
149         if flag and tmr % 7000 == 0:
150             bonusC.add(Bonus((0, 255, 0), 200))
151
152         tmr += 1
153         x = tmr%3200
154         screen.blit(bg_img, [-x, 0])
155         screen.blit(bg_imgs, [1600-x, 0])
156         screen.blit(bg_img, [3200-x, 0])
157
158         if flag and len(pg.sprite.spritecollide(bird, bonusC, True)) != 0:
159             flag = False
160
161         # 工科丸とコインの衝突判定
162         if len(pg.sprite.spritecollide(bird, coins, True)) != 0:
163             pg.display.update()
```

```
164
165     # 工科丸とボーナスコインの衝突判定
166     if len(pg.sprite.spritecollide(bird, bonusC, True)) != 0:
167         pg.display.update()
168
169     # 工科丸と敵機の衝突判定
170     if len(pg.sprite.spritecollide(bird, emys, True)) != 0:
171         pg.display.update()
172         return
173
174     # コインが外に出たら削除
175     for coin in coins:
176         if False in check_bound(screen.get_rect(), coin.rect):
177             coin.kill()
178     # 敵機が外に出たら削除
179     for emy in emys:
180         if False in check_bound(screen.get_rect(), emy.rect):
181             emy.kill()
182
183     key_lst = pg.key.get_pressed()
184     bird.update(key_lst, screen)
185     coins.update()
186     coins.draw(screen)
187     bonusC.update()
188     bonusC.draw(screen)
189     emys.update()
190     emys.draw(screen)
191     pg.display.update()
192     clock.tick(200)
193
194
195 if __name__ == "__main__":
196     pg.init()
197     main()
198     pg.quit()
199     sys.exit()
```