

MomonoKawabata / ProjExD

Public

<> Code

Issues 10

Pull requests

Actions

Projects

Security

Insights

main

ProjExD / ex06 / README.md

MomonoKawabata

たべろによろによろ

History

1 contributor

13 lines (13 sloc) | 699 Bytes

第5回

たべろによろによろ (ex06/ex.py)

ゲーム概要

- ex06/ex.pyを実行すると、900x1000のスクリーンに迷路が描画され、たべろによろ（緑の物体）を敵から逃げながらアイテムを食べるゲーム
- ライフが無くなるか壁に当たるとゲームオーバー

操作方法

- 矢印キーでたべろによろを上下左右に移動する

追加機能

- 背景に迷路を追加する

ToDo（実装しようと思ったけど時間がなかった）

- ☐ 迷路の壁に当たるとぶつかる（壁を貫通しない）
- ☐ 講義外で見つけた迷路の描写方法を反映する（棒倒し法）

メモ

MomonoKawabata / ProjExD

Public

<> Code

Issues 10

Pull requests

Actions

Projects

Security

Insights

main

ProjExD / ex06 / ex.py / <> Jump to

MomonoKawabata

たべろによろよろ

History

1 contributor

195 lines (167 sloc) | 6.49 KB

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

```
import sys
import random,time
import pygame as pg
import maze_maker as mm
"""
WALL_COLOR = (87, 45, 24) #迷路 棒倒し法
FLOOR_COLOR = (181, 152, 132)

# 床1枚の幅と高さ
FLOOR_W = 60
FLOOR_H = 60
# 迷路の幅と高さ（床の枚数）
MAZE_W = 20
MAZE_H = 15
# リストの宣言と初期化
maze = []
for y in range(MAZE_H):
    maze.append([0] * MAZE_W)

# 迷路の設計情報を自動生成する
def make_maze():
    # 柱から伸ばす壁のに利用する値を定義
    # [上, 右, 下, 左]
    XP = [0, 1, 0, -1]
    YP = [-1, 0, 1, 0]

    # 迷路を囲う壁を作る
    for x in range(MAZE_W):
        maze[0][x] = 1
        maze[MAZE_H - 1][x] = 1
    for y in range(1, MAZE_H - 1):
        maze[y][0] = 1
        maze[y][MAZE_W - 1] = 1

    # 中を何も無い状態にする
    for y in range(1, MAZE_H - 1):
        for x in range(1, MAZE_W - 1):
```

https://github.com/MomonoKawabata/ProjExD/blob/main/ex06/ex.py

1/4

```
39         maze[y][x] = 0
40
41     # 柱を作る
42     for y in range(2, MAZE_H - 2, 2):           # range()は第三引数を2を指定し、ステップ機能で1マス飛ばして
        いる
43         for x in range(2, MAZE_W - 2, 2):
44             maze[y][x] = 1
45
46     # 各柱から壁を伸ばす
47     for y in range(2, MAZE_H - 2, 2):
48         for x in range(2, MAZE_W - 2, 2):
49             while True:
50                 d = random.randint(0, 3)          # 変数dに柱から伸ばす方向を0~3で指定
51                 if x > 2:                          # 2列目以降なら0~2（左を示す3を含めない）で左に伸ばさない
52                     d = random.randint(0, 2)
53
54                 if maze[y + YP[d]][x + XP[d]] == 1: # dの値が既に壁が作られた場所であればやり直し
55                     continue
56
57                 # 柱から伸ばす壁を示す値（変数d）を、定数YP、XPの添字に使い壁を伸ばすマス目を指定
58                 # そのマス目を表すmaze[]に壁有りを示す1を代入
59                 maze[y + YP[d]][x + XP[d]] = 1
60                 break
61
62 def main():
63     pygame.init()
64     pygame.display.set_caption("いもむしマン")
65     screen = pygame.display.set_mode((FLOOR_W * MAZE_W, FLOOR_H * MAZE_H))
66     clock = pygame.time.Clock()
67
68     make_maze()
69
70
71     while True:
72         for event in pygame.event.get():
73             if event.type == pygame.QUIT:
74                 pygame.quit()
75                 sys.exit()
76             if event.type == pygame.KEYDOWN:
77                 if event.key == pygame.K_SPACE:
78                     make_maze()
79
80     # 自動生成した迷路の設計情報を使い実際に描画する
81     for y in range(MAZE_H):
82         for x in range(MAZE_W):
83             W = FLOOR_W
84             H = FLOOR_H
85             X = x * W
86             Y = y * H
87             # 通路を描画
88             if maze[y][x] == 0:
89                 pygame.draw.rect(screen, FLOOR_COLOR, [X, Y, W, H])
90             # 壁を描画
91             if maze[y][x] == 1:
92                 pygame.draw.rect(screen, WALL_COLOR, [X, Y, W, H])
93
94     pygame.display.update()
95     clock.tick(2)
```

```
96  """
97
98  class Screen:
99      def __init__(self, title, wh, img_path):
100          pg.display.set_caption(title)
101          self.sfc = pg.display.set_mode(wh)
102          self.rct = self.sfc.get_rect()
103          self.bgi_sfc = pg.image.load(img_path)
104          self.bgi_rct = self.bgi_sfc.get_rect()
105
106      def blit(self):
107          self.sfc.blit(self.bgi_sfc, self.bgi_rct)
108
109  class Maze:
110      def __init__(self, maze_lst, scr: Screen):
111          self.sfc = pg.Surface((1200, 900))
112          self.sfc.set_colorkey((0, 0, 0))
113          color = ["white", "blue"] # 迷路の色
114          for y in range(len(maze_lst)):
115              for x in range(len(maze_lst[y])):
116                  pg.draw.rect(self.sfc, color[maze_lst[y][x]], (x*50, y*50, 50, 50)) # 迷路のマスの大きさと間隔
117          self.rct = self.sfc.get_rect()
118
119      def blit(self, scr: Screen):
120          scr.sfc.blit(self.sfc, self.rct)
121
122  def main2():
123      scr = Screen("食べるによろによろ", (1600, 900), "fig/pg_bg.jpg")
124
125      make_lst = mm.make_maze(18, 18) # マスの数
126      print(make_lst)
127      maze = Maze(make_lst, scr)
128
129      color_red = pg.Color(255, 0, 0)
130      color_green = pg.Color(0, 255, 0)
131      screen = pg.display.set_mode((900, 1000)) # スクリーンの大きさ
132      pg.display.set_caption("蛇")
133      arr = [[([0] * 41) for i in range(61)]
134      x = 10 # 蛇の初期x座標
135      y = 10 # 蛇の初期y座標
136      foodx = random.randint(1, 60) # 食べ物のx座標
137      foody = random.randint(1, 40) # 食べ物のy座標
138      arr[foodx][foody] = -1
139      snake_lon = 3 # 蛇の長さ
140      way = 1 # 蛇の運動方向
141
142      while True:
143          scr.blit()
144          maze.blit(scr)
145
146          # screen.fill(color_white)
147          time.sleep(0.1)
148          for event in pg.event.get(): # 监听器
149              if event.type == pg.QUIT:
150                  sys.exit()
151              elif event.type == pg.KEYDOWN:
152                  if (event.key == pg.K_RIGHT) and (way != 2): # 右
153                      way = 1
```

```
154         if (event.key == pg.K_LEFT) and (way != 1): # 左
155             way = 2
156         if (event.key == pg.K_UP) and (way != 4): # 上
157             way = 3
158         if (event.key == pg.K_DOWN) and (way != 3): # 下に移動
159             way = 4
160     if way == 1:
161         x += 1
162     if way == 2:
163         x -= 1
164     if way == 3:
165         y -= 1
166     if way == 4:
167         y += 1
168     if (x > 60) or (y > 40) or (x < 1) or (y < 1) or (arr[x][y] > 0): # 死亡(壁、自分の体をぶつかった
    ら)
169         sys.exit()
170     arr[x][y] = snake_lon
171     for a, b in enumerate(arr, 1):
172         for c, d in enumerate(b, 1):
173             # 食べ物は-1, 空地は0, 蛇の位置は正数
174             if (d > 0):
175                 # print(a,c) #蛇の座標を表示
176                 arr[a - 1][c - 1] = arr[a - 1][c - 1] - 1
177                 pg.draw.rect(screen, color_green, ((a - 1) * 10, (c - 1) * 10, 10, 10))
178             if (d < 0):
179                 pg.draw.rect(screen, color_red, ((a - 1) * 10, (c - 1) * 10, 10, 10))
180     if (x == foodx) and (y == foody): #蛇が食べ物を食べたら
181         snake_lon += 1 #長さ+1
182         while (arr[foodx][foody] != 0): #新しい食べ物を表示
183             foodx = random.randint(1, 60)
184             foody = random.randint(1, 40)
185             arr[foodx][foody] = -1
186
187     pg.display.update()
188
189
190 if __name__ == '__main__':
191     pg.init()
192     main2()
193     pg.quit()
194     sys.exit()
```