第5回

たべろにょろにょろ（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，にょろにょろ（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

- 矢印キーでにょろにょ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

```
1
2  import sys
3  import random,time
4  import pygame as pg
5  import maze_maker as mm
6  """
7  WALL_COLOR = (87, 45, 24) #迷路　棒倒し法
8  FLOOR_COLOR = (181, 152, 132)
9
10 # 床1枚の幅と高さ
11 FLOOR_W = 60
12 FLOOR_H = 60
13 # 迷路の幅と高さ（床の枚数）
14 MAZE_W = 20
15 MAZE_H = 15
16 # リストの宣言と初期化
17 maze = []
18 for y in range(MAZE_H):
19     maze.append([0] * MAZE_W)
20
21 # 迷路の設計情報を自動生成する
22 def make_maze():
23     # 柱から伸ばす壁のに利用する値を定義
24     # [上，右，下，左]
25     XP = [0, 1, 0, -1]
26     YP = [-1, 0, 1, 0]
27
28     # 迷路を囲う壁を作る
29     for x in range(MAZE_W):
30         maze[0][x] = 1
31         maze[MAZE_H - 1][x] = 1
32     for y in range(1, MAZE_H - 1):
33         maze[y][0] = 1
34         maze[y][MAZE_W - 1] = 1
35
36     # 中を何もない状態にする
37     for y in range(1, MAZE_H - 1):
38         for x in range(1, MAZE_W - 1):
```

```
            maze[y][x] = 0

    # 柱を作る
    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マス飛ばししている
        for x in range(2, MAZE_W - 2, 2):
            maze[y][x] = 1

    # 各柱から壁を伸ばす
    for y in range(2, MAZE_H - 2, 2):
        for x in range(2, MAZE_W - 2, 2):
            while True:
                d = random.randint(0, 3)            # 変数dに柱から伸ばす方向を0~3で指定
                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
                    d = random.randint(0, 2)

                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
                    continue

                # 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
                # そのマス目を表すmaze[]に壁有りを示す1を代入
                maze[y + YP[d]][x + XP[d]] = 1
                break

def main():
    pygame.init()
    pygame.display.set_caption("いもむしマン")
    screen = pygame.display.set_mode((FLOOR_W * MAZE_W, FLOOR_H * MAZE_H))
    clock = pygame.time.Clock()

    make_maze()


    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_SPACE:
                    make_maze()

        # 自動生成した迷路の設計情報を使い実際に描画する
        for y in range(MAZE_H):
            for x in range(MAZE_W):
                W = FLOOR_W
                H = FLOOR_H
                X = x * W
                Y = y * H
                # 通路を描画
                if maze[y][x] == 0:
                    pygame.draw.rect(screen, FLOOR_COLOR, [X, Y, W, H])
                # 壁を描画
                if maze[y][x] == 1:
                    pygame.draw.rect(screen, WALL_COLOR, [X, Y, W, H])

        pygame.display.update()
        clock.tick(2)
```

```
"""

class Screen:
    def __init__(self, title, wh, img_path):
        pg.display.set_caption(title)
        self.sfc = pg.display.set_mode(wh)
        self.rct = self.sfc.get_rect()
        self.bgi_sfc = pg.image.load(img_path)
        self.bgi_rct = self.bgi_sfc.get_rect()

    def blit(self):
        self.sfc.blit(self.bgi_sfc, self.bgi_rct)

class Maze:
    def __init__(self,maze_lst,scr:Screen):
        self.sfc=pg.Surface((1200,900))
        self.sfc.set_colorkey((0,0,0))
        color=["white", "blue"] #迷路の色
        for y in range(len(maze_lst)):
            for x in range(len(maze_lst[y])):
                pg.draw.rect(self.sfc,color[maze_lst[y][x]],(x*50,y*50,50,50)) #迷路の一マスの大きさと間隔
        self.rct=self.sfc.get_rect()

    def blit(self,scr:Screen):
        scr.sfc.blit(self.sfc,self.rct)

def main2():
    scr = Screen("食べろにょろにょろ", (1600,900), "fig/pg_bg.jpg")

    make_lst=mm.make_maze(18,18) #マスの数
    print (make_lst)
    maze=Maze(make_lst,scr)

    color_red = pg.Color(255, 0, 0)
    color_green = pg.Color(0, 255, 0)
    screen = pg.display.set_mode((900, 1000)) #スクリーンの大きさ
    pg.display.set_caption("蛇")
    arr = [([0] * 41) for i in range(61)]
    x = 10  # 蛇の初期x座標
    y = 10  # 蛇の初期y座標
    foodx = random.randint(1, 60)  # 食べ物のx座標
    foody = random.randint(1, 40)  # 食べ物のy座標
    arr[foodx][foody] = -1
    snake_lon = 3  # 蛇の長さ
    way = 1  # 蛇の運動方向

    while True:
        scr.blit()
        maze.blit(scr)

        #screen.fill(color_white)
        time.sleep(0.1)
        for event in pg.event.get():  # 监听器
            if event.type == pg.QUIT:
                sys.exit()
            elif event.type == pg.KEYDOWN:
                if (event.key == pg.K_RIGHT) and (way != 2):  # 右
                    way = 1
```

```
                if (event.key == pg.K_LEFT) and (way != 1):  # 左
                    way = 2
                if (event.key == pg.K_UP) and (way != 4):  # 上
                    way = 3
                if (event.key == pg.K_DOWN) and (way != 3):  # 下に移動
                    way = 4
        if way == 1:
            x += 1
        if way == 2:
            x -= 1
        if way == 3:
            y -= 1
        if way == 4:
            y += 1
        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)
            sys.exit()
        arr[x][y] = snake_lon
        for a, b in enumerate(arr, 1):
            for c, d in enumerate(b, 1):
                # 食べ物は-1，空地は0，蛇の位置は正数
                if (d > 0):
                    # print(a,c) #蛇の座標を表示
                    arr[a - 1][c - 1] = arr[a - 1][c - 1] - 1
                    pg.draw.rect(screen, color_green, ((a - 1) * 10, (c - 1) * 10, 10, 10))
                if (d < 0):
                    pg.draw.rect(screen, color_red, ((a - 1) * 10, (c - 1) * 10, 10, 10))
        if (x == foodx) and (y == foody):   #蛇が食べ物を食べったら
            snake_lon += 1    #長さ+1
            while (arr[foodx][foody] != 0):    #新しい食べ物を表示
                foodx = random.randint(1, 60)
                foody = random.randint(1, 40)
            arr[foodx][foody] = -1

        pg.display.update()


if __name__ == '__main__':
    pg.init()
    main2()
    pg.quit()
    sys.exit()
```