**贪吃蛇设计报告**

**221300066**

**季千焜**

**qkjiai@smail.nju.edu.cn**

**一．选题描述：**

本项目是一个基于BFS等算法的贪吃蛇小游戏，利用python语言编写，采用pygame库实现图形化界面，玩家版通过键盘控制贪吃蛇的移动，AI版实现了自动寻路，小蛇的初始长度为2，吃到食物后蛇的长度增加，得分加1，撞到墙壁或自身则游戏结束，最终目标是使贪吃蛇的长度尽可能长，尽可能占满空间。

**二．设计方案：**

整个项目可以分为图形化界面、蛇与食物的生成、小蛇的运动、游戏逻辑、算法套路、分数统计等模块，各模块功能如下：

**1.图形界面模块：**

使用pygame库实现游戏窗口、各元素(如屏幕大小，背景颜色，蛇与果实的颜色，网格线等)的布局。

优化：添加开始与结束模块。

**2.蛇体运动模块：**

根据键盘键控制蛇的运动，形成蛇身体的位置变化，判定游戏是否结束等。

**3.食物生成模块：**

随机生成食物，若蛇的头部碰到食物，则认为蛇吃食物成功，蛇身体长度增加，同时重新随机生成食物。

**4.游戏逻辑模块：**

根据蛇与食物的交互情况，判断游戏是否结束，同时维护分数等信息。

**5.算法套路模块：**

以BFS广度搜索为基础算法，加入虚拟的小蛇，探索吃到食物的最短路径，最安全的路径（DFS最远路径，不进入死路的路径，头追着尾的路径等），为继续运行而随便走的路径(哈密顿路径三选一)等。

**6. 分数统计模块：**

使用文本框实现分数的实时统计，通过Show\_Score函数处理得分。

**三．代码模块的功能划分与描述：**

（一些准备代码省略）

**玩家版**

1. 显示蛇

def Show\_Snake(coords):

x = coords[0]['x'] \* Cell\_Size

y = coords[0]['y'] \* Cell\_Size

Snake\_head\_Rect = pygame.Rect(x, y, Cell\_Size, Cell\_Size)

pygame.draw.rect(Main\_Display, (0, 80, 255), Snake\_head\_Rect)

Snake\_head\_Inner\_Rect = pygame.Rect(x+4, y+4, Cell\_Size-8, Cell\_Size-8)

pygame.draw.rect(Main\_Display, (0, 80, 255), Snake\_head\_Inner\_Rect)

for coord in coords[1:]:

x = coord['x'] \* Cell\_Size

y = coord['y'] \* Cell\_Size

Snake\_part\_Rect = pygame.Rect(x, y, Cell\_Size, Cell\_Size)

pygame.draw.rect(Main\_Display, (0, 155, 0), Snake\_part\_Rect)

Snake\_part\_Inner\_Rect = pygame.Rect(x+4, y+4, Cell\_Size-8, Cell\_Size-8)

pygame.draw.rect(Main\_Display, (0, 255, 0), Snake\_part\_Inner\_Rect)

2. 显示果实

def Show\_Apple(coord):

x = coord['x'] \* Cell\_Size

y = coord['y'] \* Cell\_Size

apple\_Rect = pygame.Rect(x, y, Cell\_Size, Cell\_Size)

pygame.draw.rect(Main\_Display, (255, 0, 0), apple\_Rect)

3. 画网格

def draw\_Grid():

# 垂直方向

for x in range(0, Window\_Width, Cell\_Size):

pygame.draw.line(Main\_Display, (40, 40, 40), (x, 0), (x, Window\_Height))

# 水平方向

for y in range(0, Window\_Height, Cell\_Size):

pygame.draw.line(Main\_Display, (40, 40, 40), (0, y), (Window\_Width, y))

4. 显示开始与结束界面

def Show\_Start\_Interface():

title\_Font = pygame.font.Font('simkai.ttf', 100)

title\_content = title\_Font.render('贪吃蛇', True, (255, 255, 255), (128,0,128))

angle = 0

while True:

Main\_Display.fill(Background\_Color)

rotated\_title = pygame.transform.rotate(title\_content, angle)

rotated\_title\_Rect = rotated\_title.get\_rect()

rotated\_title\_Rect.center = (Window\_Width/2, Window\_Height/2)

Main\_Display.blit(rotated\_title, rotated\_title\_Rect)

pressKey\_content = Main\_Font.render('按任意键开始游戏！', True, (255, 255, 255))

pressKey\_Rect = pressKey\_content.get\_rect()

pressKey\_Rect.topleft = (Window\_Width-200, Window\_Height-30)

Main\_Display.blit(pressKey\_content, pressKey\_Rect)

if Check\_PressKey():

# 清除事件队列

pygame.event.get()

return

pygame.display.update()

Snake\_Clock.tick(Display\_Clock)

angle -= 5

def Show\_End\_Interface():

title\_Font = pygame.font.Font('simkai.ttf', 100)

title\_game = title\_Font.render('Game', True, (233, 150, 122))

title\_over = title\_Font.render('Over', True, (233, 150, 122))

game\_Rect = title\_game.get\_rect()

over\_Rect = title\_over.get\_rect()

game\_Rect.midtop = (Window\_Width/2, 70)

over\_Rect.midtop = (Window\_Width/2, game\_Rect.height+70+25)

Main\_Display.blit(title\_game, game\_Rect)

Main\_Display.blit(title\_over, over\_Rect)

pressKey\_content = Main\_Font.render('按任意键开始游戏！', True, (255, 255, 255))

pressKey\_Rect = pressKey\_content.get\_rect()

pressKey\_Rect.topleft = (Window\_Width-200, Window\_Height-30)

Main\_Display.blit(pressKey\_content, pressKey\_Rect)

pygame.display.update()

pygame.time.wait(500)

# 清除事件队列

Check\_PressKey()

while True:

if Check\_PressKey():

pygame.event.get()

return

5. 检测玩家的按键是否合理

def Check\_PressKey():

if len(pygame.event.get(QUIT)) > 0:

close\_game()

KeyUp\_Events = pygame.event.get(KEYUP)

if len(KeyUp\_Events) == 0:

return None

elif KeyUp\_Events[0].key == K\_ESCAPE:

close\_game()

return KeyUp\_Events[0].key

6. 获得果实位置

def Get\_Apple\_Location(snake\_Coords):

flag = True

while flag:

apple\_location = {'x': random.randint(0, Cell\_W-1), 'y': random.randint(0, Cell\_H-1)}

if apple\_location not in snake\_Coords:

flag = False

return apple\_location

7. 显示当前得分

def Show\_Score(score):

score\_Content = Main\_Font.render('得分：%s' % (score), True, (255, 255, 255))

score\_Rect = score\_Content.get\_rect()

score\_Rect.topleft = (Window\_Width-120, 10)

Main\_Display.blit(score\_Content, score\_Rect)

8. 运行游戏

def Run\_Game():

# 蛇的出生地

start\_x = random.randint(5, Cell\_W-6)

start\_y = random.randint(5, Cell\_H-6)

snake\_Coords = [{'x': start\_x, 'y': start\_y},

{'x': start\_x-1, 'y': start\_y}]

direction = 'right'

apple\_location = Get\_Apple\_Location(snake\_Coords)

while True:

for event in pygame.event.get():

if event.type == QUIT:

close\_game()

elif event.type == KEYDOWN:

if (event.key in (K\_a, K\_LEFT)) and (direction != 'right'):

direction = 'left'

elif (event.key in (K\_d, K\_RIGHT)) and (direction != 'left'):

direction = 'right'

elif (event.key in (K\_w, K\_UP)) and (direction != 'down'):

direction = 'up'

elif (event.key in (K\_s, K\_DOWN)) and (direction != 'up'):

direction = 'down'

elif event.key == K\_ESCAPE:

close\_game()

# 碰到墙壁或者自己则游戏结束

if (snake\_Coords[Head\_index]['x'] == -1) or (snake\_Coords[Head\_index]['x'] == Cell\_W) or \

(snake\_Coords[Head\_index]['y'] == -1) or (snake\_Coords[Head\_index]['y'] == Cell\_H):

return

if snake\_Coords[Head\_index] in snake\_Coords[1:]:

return

if (snake\_Coords[Head\_index]['x'] == apple\_location['x']) and (snake\_Coords[Head\_index]['y'] == apple\_location['y']):

apple\_location = Get\_Apple\_Location(snake\_Coords)

else:

del snake\_Coords[-1]

if direction == 'up':

newHead = {'x': snake\_Coords[Head\_index]['x'],

'y': snake\_Coords[Head\_index]['y']-1}

elif direction == 'down':

newHead = {'x': snake\_Coords[Head\_index]['x'],

'y': snake\_Coords[Head\_index]['y']+1}

elif direction == 'left':

newHead = {'x': snake\_Coords[Head\_index]['x']-1,

'y': snake\_Coords[Head\_index]['y']}

elif direction == 'right':

newHead = {'x': snake\_Coords[Head\_index]['x']+1,

'y': snake\_Coords[Head\_index]['y']}

snake\_Coords.insert(0, newHead)

Main\_Display.fill(Background\_Color)

draw\_Grid()

Show\_Snake(snake\_Coords)

Show\_Apple(apple\_location)

Show\_Score(len(snake\_Coords)-2)

pygame.display.update()

Snake\_Clock.tick(Display\_Clock)

9. 主函数

def main():

global Main\_Display, Main\_Font, Snake\_Clock

pygame.init()

Snake\_Clock = pygame.time.Clock()

Main\_Display = pygame.display.set\_mode((Window\_Width, Window\_Height))

Main\_Font = pygame.font.Font('simkai.ttf', 18)

pygame.display.set\_caption('Normal\_snake')

Show\_Start\_Interface()

while True:

Run\_Game()

Show\_End\_Interface()

**AI版**

算法主要思路：

1.蛇每走一步，就使用BFS计算游戏界面中每个位置（蛇身除外）到达食物的最短路径长；

2.将蛇的安全路径定义为蛇可以跟着蛇尾运动，即蛇头和蛇尾间存在路径；

3.蛇每次行动前先利用虚拟的蛇进行探路，若虚拟的蛇吃完食物后是安全的，蛇才行动；

4.若蛇和食物之间不存在路径或者吃完食物后并不安全，就跟着蛇尾走，保证不死；

5.若蛇和食物之间、蛇和蛇尾之间均不存在路径，就随便挑一步来走；

6.需要注意目标是食物时蛇走最短路径，目标是蛇尾时蛇走DFS的最长路径（保证不死）。

具体代码：

1.每走一步都要重置board

def board\_reset(psnake, pboard, pfood):

temp\_board = pboard[:]

pfood\_idx = pfood['x'] + pfood['y'] \* Cell\_W

for i in range(FIELD\_SIZE):

if i == pfood\_idx:

temp\_board[i] = FOOD

elif Is\_Cell\_Free(i, psnake):

temp\_board[i] = FREE\_PLACE

else:

temp\_board[i] = SNAKE\_PLACE

return temp\_board

2.BFS最短路径吃果实

def board\_refresh(psnake, pfood, pboard):

temp\_board = pboard[:]

pfood\_idx = pfood['x'] + pfood['y'] \* Cell\_W

queue = []

queue.append(pfood\_idx)

inqueue = [0] \* FIELD\_SIZE

found = False

while len(queue) != 0:

idx = queue.pop(0)

if inqueue[idx] == 1:

continue

inqueue[idx] = 1

for move\_direction in ['left', 'right', 'up', 'down']:

if is\_move\_possible(idx, move\_direction):

if (idx+move\_directions[move\_direction]) == (psnake[Head\_index]['x'] + psnake[Head\_index]['y']\*Cell\_W):

found = True

# 该点不是蛇身(食物是0才可以这样子写)

if temp\_board[idx+move\_directions[move\_direction]] < SNAKE\_PLACE:

if temp\_board[idx+move\_directions[move\_direction]] > temp\_board[idx]+1:

temp\_board[idx+move\_directions[move\_direction]] = temp\_board[idx] + 1

if inqueue[idx+move\_directions[move\_direction]] == 0:

queue.append(idx+move\_directions[move\_direction])

return (found, temp\_board)

# 根据board中元素值

# 从蛇头周围4个领域点中选择最短路径

def choose\_shortest\_safe\_move(psnake, pboard):

best\_move = ERR

min\_distance = SNAKE\_PLACE

for move\_direction in ['left', 'right', 'up', 'down']:

idx = psnake[Head\_index]['x'] + psnake[Head\_index]['y']\*Cell\_W

if is\_move\_possible(idx, move\_direction) and (pboard[idx+move\_directions[move\_direction]]<min\_distance):

min\_distance = pboard[idx+move\_directions[move\_direction]]

best\_move = move\_direction

return best\_move

3. 虚拟蛇运行一次

def virtual\_move(psnake, pboard, pfood):

temp\_snake = psnake[:]

temp\_board = pboard[:]

reset\_tboard = board\_reset(temp\_snake, temp\_board, pfood)

temp\_board = reset\_tboard

food\_eated = False

while not food\_eated:

refresh\_tboard = board\_refresh(temp\_snake, pfood, temp\_board)[1]

temp\_board = refresh\_tboard

move\_direction = choose\_shortest\_safe\_move(temp\_snake, temp\_board)

snake\_Coords = temp\_snake[:]

temp\_snake.insert(0, find\_snake\_head(snake\_Coords, move\_direction))

# 如果新的蛇头正好是食物的位置

if temp\_snake[Head\_index] == pfood:

reset\_tboard = board\_reset(temp\_snake, temp\_board, pfood)

temp\_board = reset\_tboard

pfood\_idx = pfood['x'] + pfood['y'] \* Cell\_W

temp\_board[pfood\_idx] = SNAKE\_PLACE

food\_eated = True

else:

newHead\_idx = temp\_snake[0]['x'] + temp\_snake[0]['y'] \* Cell\_W

temp\_board[newHead\_idx] = SNAKE\_PLACE

end\_idx = temp\_snake[-1]['x'] + temp\_snake[-1]['y'] \* Cell\_W

temp\_board[end\_idx] = FREE\_PLACE

del temp\_snake[-1]

return temp\_snake, temp\_board

4. 选择最远路径

def choose\_longest\_safe\_move(psnake, pboard):

best\_move = ERR

max\_distance = -1

for move\_direction in ['left', 'right', 'up', 'down']:

idx = psnake[Head\_index]['x'] + psnake[Head\_index]['y']\*Cell\_W

if is\_move\_possible(idx, move\_direction) and (pboard[idx+move\_directions[move\_direction]]>max\_distance) and (pboard[idx+move\_directions[move\_direction]]<FREE\_PLACE):

max\_distance = pboard[idx+move\_directions[move\_direction]]

best\_move = move\_direction

return best\_move

5. 蛇头朝着蛇尾运行一步

def follow\_tail(psnake, pboard, pfood):

temp\_snake = psnake[:]

temp\_board = board\_reset(temp\_snake, pboard, pfood)

# 将蛇尾看作食物

end\_idx = temp\_snake[-1]['x'] + temp\_snake[-1]['y'] \* Cell\_W

temp\_board[end\_idx] = FOOD

v\_food = temp\_snake[-1]

# 食物看作蛇身

pfood\_idx = pfood['x'] + pfood['y'] \* Cell\_W

temp\_board[pfood\_idx] = SNAKE\_PLACE

# 求得每个位置到蛇尾的路径长度

result, refresh\_tboard = board\_refresh(temp\_snake, v\_food, temp\_board)

temp\_board = refresh\_tboard

# 还原

temp\_board[end\_idx] = SNAKE\_PLACE

# temp\_board[pfood\_idx] = FOOD

return choose\_longest\_safe\_move(temp\_snake, temp\_board)

6. 找一条安全的路径

def find\_safe\_way(psnake, pboard, pfood):

safe\_move = ERR

real\_snake = psnake[:]

real\_board = pboard[:]

v\_psnake, v\_pboard = virtual\_move(psnake, pboard, pfood)

# 如果虚拟运行后，蛇头蛇尾间有通路，则选最短路运行

if is\_tail\_inside(v\_psnake, v\_pboard, pfood):

safe\_move = choose\_shortest\_safe\_move(real\_snake, real\_board)

else:

safe\_move = follow\_tail(real\_snake, real\_board, pfood)

return safe\_move

7. 各种方案均无效时，随便走一步

def any\_possible\_move(psnake, pboard, pfood):

best\_move = ERR

reset\_board = board\_reset(psnake, pboard, pfood)

pboard = reset\_board

result, refresh\_board = board\_refresh(psnake, pfood, pboard)

pboard = refresh\_board

min\_distance = SNAKE\_PLACE

for move\_direction in ['left', 'right', 'up', 'down']:

idx = psnake[Head\_index]['x'] + psnake[Head\_index]['y']\*Cell\_W

if is\_move\_possible(idx, move\_direction) and (pboard[idx+move\_directions[move\_direction]]<min\_distance):

min\_distance = pboard[idx+move\_directions[move\_direction]]

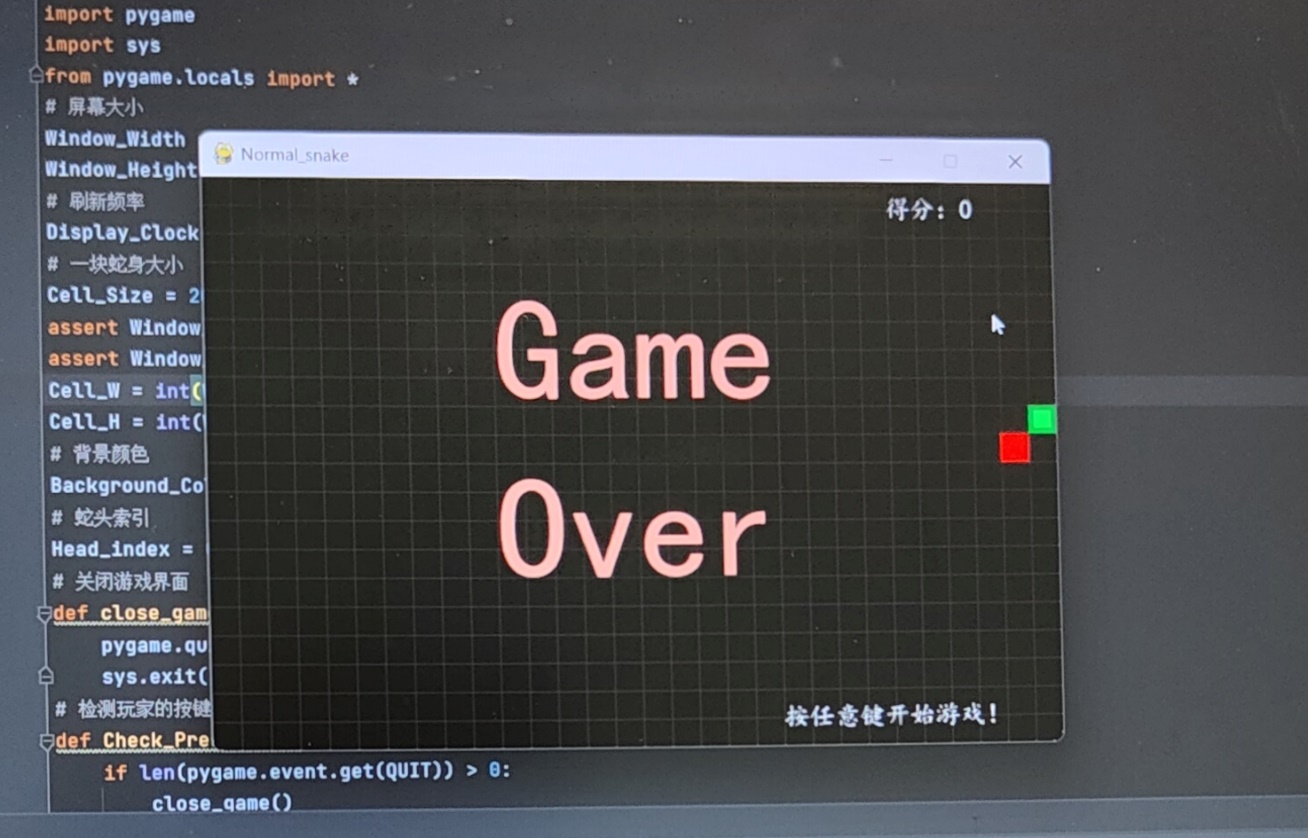
best\_move = move\_direction

return best\_move

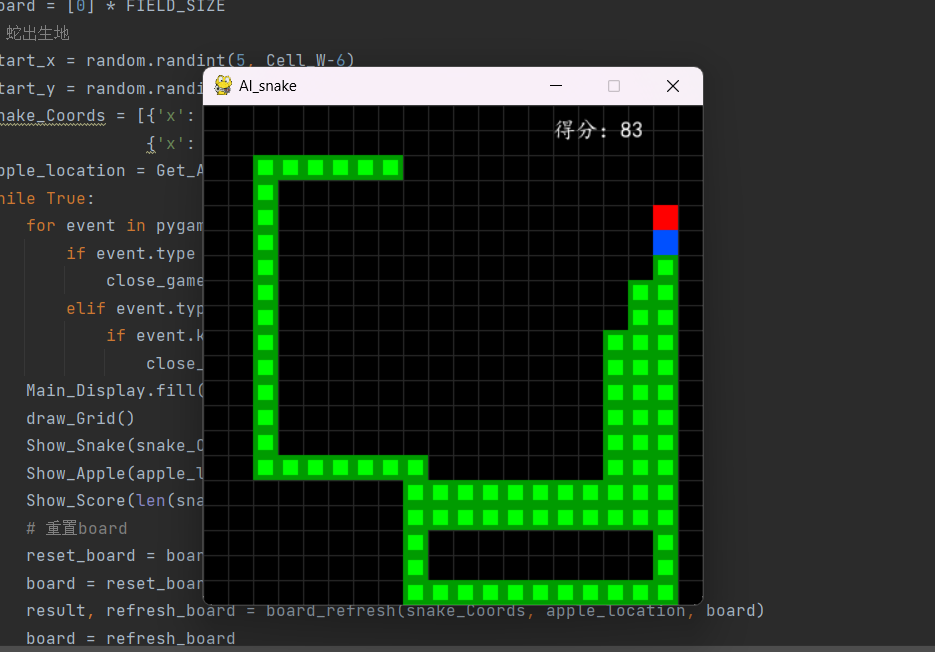
**四．实现效果：**

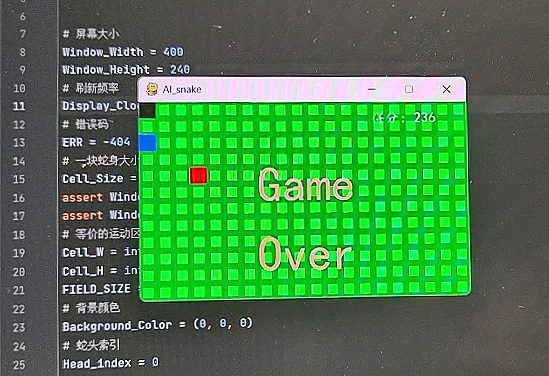
**玩家版**

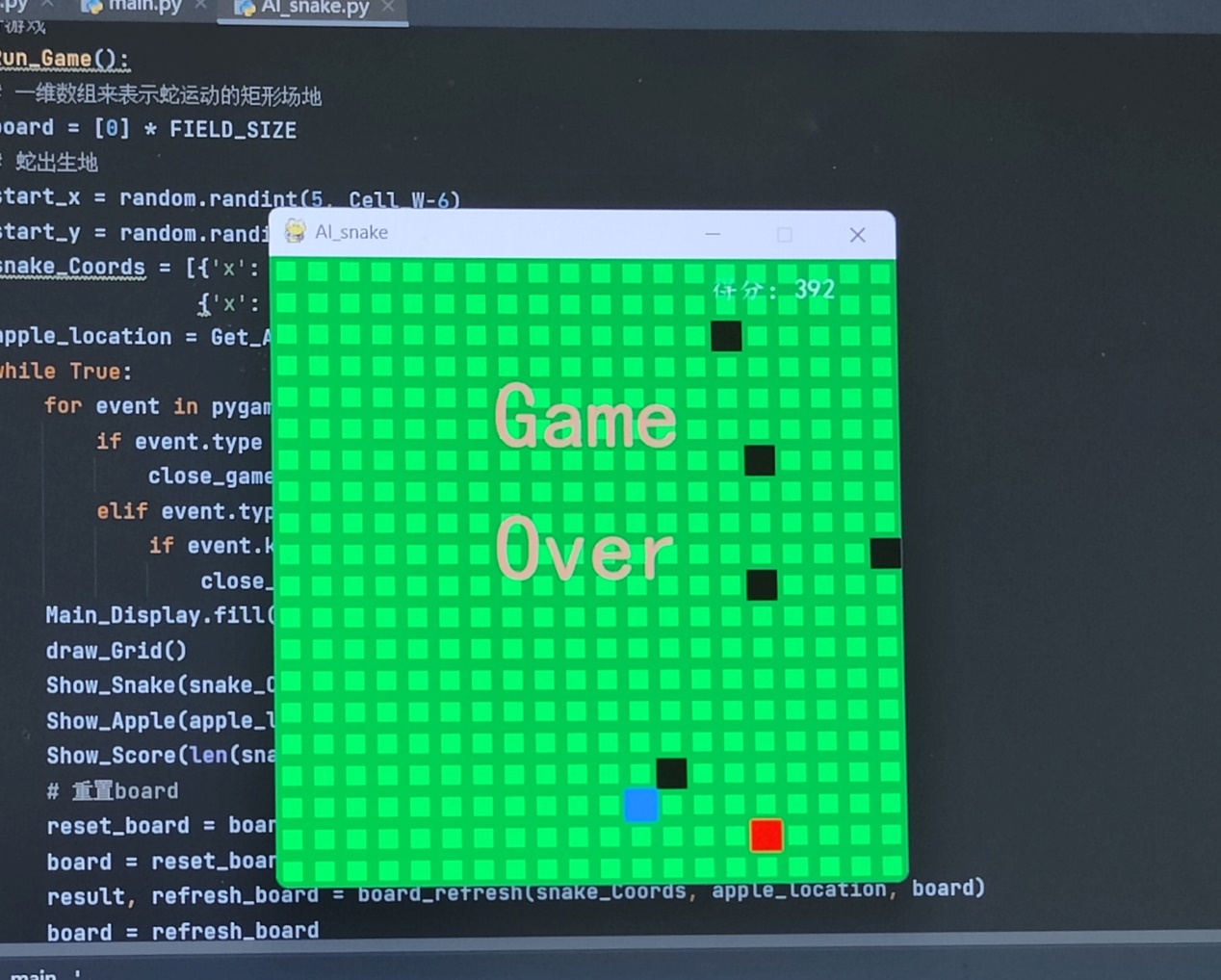


****

**AI版**







**五．一些不足：**

因为食物是随机出现的，在空格较少的时候，若虚拟的蛇跑一遍发现去吃食物是不安全的，真蛇就不会去吃食物，而是选择追着蛇尾跑，若一直如此，就陷入了死循环，蛇一直追着蛇尾跑，游戏不会停止。

P.S.我测试过程中只有一次小蛇铺满了整个区域，但是我没有截图555