**MATLAB程序设计**

**基于Matlab的贪吃蛇游戏设计**

****

**班 级：** 211704

**学 号：** 21170424

**姓 名： 吴沛林**

**教 师： 曹弋**

**成绩评定：**

**南京师范大学电气与自动化工程学院**

目录

[1、游戏介绍 3](#_Toc27007880)

[1.1贪吃蛇游戏的历史 3](#_Toc27007881)

[1.2贪吃蛇的操作指南 4](#_Toc27007882)

[1.3贪吃蛇的文化背景 4](#_Toc27007883)

[2、基于Matlab的贪吃蛇游戏设计 5](#_Toc27007884)

[2.1 GUI界面设计 5](#_Toc27007885)

[2.2 GUI界面交互函数设计 7](#_Toc27007886)

[（1）关闭游戏 7](#_Toc27007887)

[（2）开始游戏 7](#_Toc27007888)

[（3）难度调选框 7](#_Toc27007889)

[2.3 各个功能函数的设计及其实现 8](#_Toc27007890)

[（1）图形的初始化函数 8](#_Toc27007891)

[（2）运动功能函数实现 9](#_Toc27007892)

[（3）状态功能实现方法函数 11](#_Toc27007893)

[（4）记分系统 13](#_Toc27007894)

[（4）按键获取实现方法 14](#_Toc27007895)

[（5）主函数实现方法 15](#_Toc27007896)

[3.游戏测试和总结 17](#_Toc27007897)

[3.1测试画面 17](#_Toc27007898)

[3.2总结收获 18](#_Toc27007899)

[引用文献： 19](#_Toc27007900)

## 1、游戏介绍

## 1.1贪吃蛇游戏的历史

1976年，Gremlin平台推出了一款经典街机游戏Blockade。游戏中，两名玩家分别控制一个角色在屏幕上移动，所经之处砌起围栏。角色只能向左、右方向90度转弯，游戏目标保证让对方先撞上屏幕或围栏。听起来有点复杂，其实就是下面这个样子：

基本上就是两条每走一步都会长大的贪吃蛇比谁后完蛋，玩家要做的就是避免撞上障碍物和越来越长的身体。Blockade 很受欢迎，类似的游戏先后出现在 Atari 2600、TRS-80、苹果 2 等早期游戏机、计算机上。

虽然贪吃蛇游戏诞生较早，但是真正让这种游戏形式红遍全球的还是21年后随诺基亚手机走向世界的贪吃蛇游戏《Snake》（如图1所示）。1997年，诺基亚工程师 Taneli Armanto 为诺基亚N6610手机写了一款贪吃蛇程序，直接命名为Snake，中译贪吃蛇。嗯，所以“贪吃蛇”虽然概念久远，但这个名字的确是诺基亚公司创造的。

一般认为，Snake是有史以来第一款手机游戏。



图 1

已知最早的个人电脑版本，是TRS-80型电脑上的Worm程序，作者Peter Trefonas。TRS-80型，坦迪公司（Tandy Corporation）出品，1977年上市，1981年停产，最早的大规模生产个人电脑之一。它的本体售价约600美元，拥有一枚1.77M主频的CPU，4k-48k的内存，全键盘，TRS-DOS操作系统，浮点BASIC语言。可外接磁带、软盘或者硬盘存储（那时的软盘容量85k到180k不等；硬盘容量5兆，价格1500美元）。

随机附送一台黑白电视机改装的显示器，但因为只有1k显存，所以只能显示64\*16的字母/符号矩阵，不能单独显示真正的像素点。不过利用符号模拟的方法，可以达成128\*48的虚拟分辨率。坦迪公司1963年收购了著名电子元器件零售商Radioshack，2000年公司名称改为了Radioshack。[1]

## 1.2贪吃蛇的操作指南

玩家使用方向键（方向键或者W,A,S,D对应的方向）操控一条长长的蛇，同时蛇身随着时间不断变长，当蛇头撞到蛇身时游戏结束。本游戏设计中，为了降低游戏难度，设定蛇可以穿过墙从另一侧出现。

在游戏难度上，玩家可以通过难度选项按钮进行调节，调节的项目主要为贪吃蛇的移动速度大小。

与传统吃蛇蛋增长不同，本游戏设计的方式是按照生存时间来计分，且随着使劲按的增长蛇的身体也逐渐增长，每两秒增加一个单位长度。这样设计异于传统贪吃蛇游戏，达到让人耳目一新的效果。尤其是在困难模式，想得到高分需要一直集中精力才能实现。

## 1.3贪吃蛇的文化背景

衔尾蛇可以视作是贪吃蛇的原型。这个自古代流传，经常出现于阿兹特克文明、中东地区及美洲原住民，以及其他古老地区的古老神话之中的蛇，其形象大多为一条蛇正在吞食自己的尾巴，结果形成了一个圆环。在柏拉图看来，衔尾蛇为一头处 于自我吞食状态的宇宙始祖生物，它是不死之身，并拥有完美的生物结构。或许正是这个存在于全球大多数古老神话中的影响，贪吃蛇的粉丝们才会在第一次接触它的时候，就莫名地有一种亲切感。 而贪吃蛇的最高境界也是让自己咬上自己的尾巴。

在贪吃蛇风靡的时代，很多人发明了无数种贪吃蛇攻略，想要获得最晚死的荣耀，而最简单有效的方法则是让这条蛇按照“Z字形” 路线前进，如果顺利，就有可能在占满屏幕时最终咬着自己的尾巴。但大多数人都有同样的感慨，贪吃蛇尽管简单，但却鲜有拿下最高分的，以诺基亚手机上初版贪吃蛇为例，在一块20×11的长方形区域驰骋，按照分数计算公式总分=屏幕点数(212)\*关 数(1~9)+100，贪吃蛇第一关的最高分是 312，而第九关的最高分是2008。但即使是游戏设计师Taneli Armanto本人，也只拿到过1125分。为了实现贪吃蛇世界观里的最终梦想，无数的人进行了挑战，甚至在芬兰，从1999年开始就举办贪吃蛇大赛，想要在全球找到最后死的贪吃蛇玩家，据说也还没有发现实现最高分的高手。其实也正因为要达成最后的Game Over难于上青天，才让贪吃蛇这款简单却有世界观的游戏能够耐玩如此之久而不衰。贪吃蛇甚至被它的喜好者从虚拟搬到了现实中，诺基亚在2010年哥伦比亚举办了一次快闪活动，他们在论坛里招募网友，而乐意来参加活动的出奇的多，因为这个活动是要爽一 把真人版贪吃蛇。在活动中，扮演食物的网友多得让贪吃蛇都有点吞不下了，负责引领前进方向的贪吃蛇头部的几个人经过简单的培训。同其他快闪行动一样，在贪吃蛇撞到自己身体导致Game Over之后，这些网友解散回家。 这样的快闪往复循环了许多次，就算是诺基亚收工了，网友们也还是乐此不疲地继续找地方去快闪，理由很简单，这个游戏是挚爱，能够有机会在 现实中玩上一把，那可是千载难逢的好机会。作为全球发行量最大的游戏，贪吃蛇没有如今移动游戏那么炫目，但却长盛不衰，并深得玩家所喜爱，它的成功秘诀其实也很简单——简单。[2]。

# 2、基于Matlab的贪吃蛇游戏设计

## 2.1 GUI界面设计

本游戏设计界面实现了功能有：游戏的难度选择、游戏的使用说明、游戏开始和结束按钮、计分板（如图2所示）。

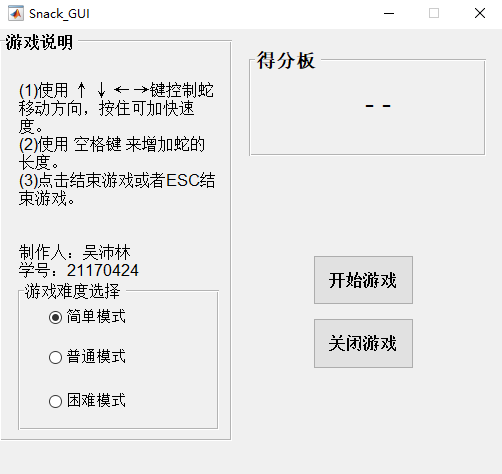


图 2

GUI管理界面中的对象有2个按钮，2个面板，其中一个面板中包含一个静态文本以及一个三按钮单选按钮，另一个面板包含一个静态文本，以及一个图纸。综上简述可有“对象浏览器”获得（如图3所示）。

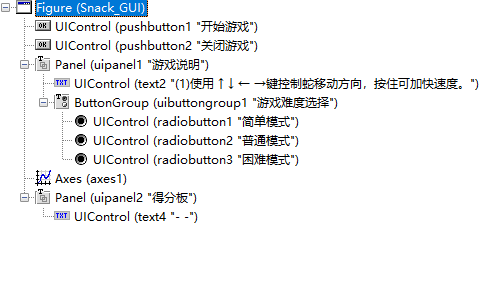


图 3

且游戏界面与管理界面不同，是单独跳出的窗口，如图4所示。

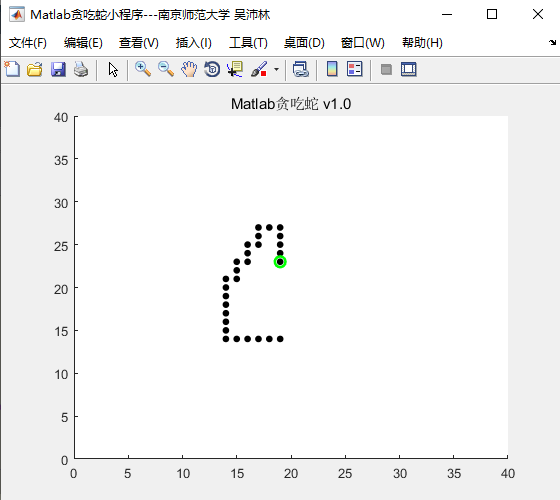


图 4

## 2.2 GUI界面交互函数设计

在GUI界面中，用户最主要的交互按钮分为一下三部分。

### （1）关闭游戏

由于关闭按钮设计非常简单，故最先说明。本游戏当按下“关闭游戏按钮的时候，窗口自动关闭，同时游戏关闭。其代码如下：

function pushbutton2\_Callback(hObject, eventdata, handles)

close all

end

此处需要说明的是需要使用close all来关闭所有窗口，如果仅使用close则会导致管理界面关闭后生成的游戏界面不会关闭。

### （2）开始游戏

当按下开始游戏的时候，游戏开始。在本设计中，将“开始游戏”按钮设计为类似于主函数的形式来实现整个程序的正常运行。而其他功能均写成函数的形式在主函数中进行调用。

### （3）难度调选框

众所周知，贪吃蛇正是因为简单而闻名。在正常的游戏体验中，如果设计过于单一，会导致用户很快对于游戏感到疲倦乏味。在本设计中，正是考虑到这点，才设计了难度选择。

考虑到软件的更新换代，暂且只添加了速度选择框。从简单到困难模式中，蛇运动的速度依次递增，具体实现方法为改变每一帧循环中暂停的时间。每一帧暂停的时间越短，反应出蛇运动的速度越快，反之运动速度越慢。

部分关键代码如下：

… …

pause\_time = 0;

rad\_bu1=get(handles.radiobutton1,'value');

rad\_bu2=get(handles.radiobutton2,'value');

rad\_bu3=get(handles.radiobutton3,'value');

if rad\_bu1 == 1

pause\_time=0.3;

end

if rad\_bu2 ==1

pause\_time=0.2;

end

if rad\_bu3==1

pause\_time=0.05;

end

… …

pause(pause\_time)

## 2.3 各个功能函数的设计及其实现

### （1）图形的初始化函数

在函数功能中基本有两个绘制函数，即绘制背景的函数和绘制蛇的函数。在主函数中调用如下：

init\_figure(snake);

draw\_snake(snake);

其中init\_figure()函数为绘制背景，draw\_snake（）为绘制蛇的函数。具体定义如下，

… …

% 初始化figure

snake.fig\_hnd = figure('NumberTitle', 'off', 'Name', 'Matlab贪吃蛇南京师范大学 吴沛林');

hold on;

snake.plot\_hnd = plot(0, 0, 'k.', 'MarkerSize', 15);

snake.plot\_head\_hnd = plot(0, 0, 'bo', 'MarkerSize', 8, 'LineWidth', 2);

… …

在主函数中首先初始化一个窗口，且窗口名称设置为：Matlab贪吃蛇南京师范大学 吴沛林。

并且设置窗口常开保持，并且更新数据。snake.plot\_hnd设置了贪吃蛇的身体点形状颜色为黑色点“k.”，且大小为15；snake.plot\_head\_hnd设置为的是贪吃蛇的头部信息为蓝色的圆圈“bo”，大小为8，线宽为2；

function init\_figure(snake)，

% 初始化figure

set(gcf,'keypressfcn',@(src,eventdata)key\_pressed\_fcn(src, eventdata));

xlim([0, snake.xmax]);

ylim([0, snake.ymax]);

title('Matlab贪吃蛇游戏v1.0')

end

该函数主要定义了网格图的长（xlim）、宽（ylim），以及图标的标题（title设置为Matlab贪吃蛇游戏v1.0）。相关的数值大小被封装成结构体，在主函数中修改添加。绘制蛇的函数如下，其设计相对复杂。

%%

function draw\_snake(snake)

set(snake.plot\_hnd, 'XData', snake.segments(:,1));

set(snake.plot\_hnd, 'YData', snake.segments(:,2));

set(snake.plot\_head\_hnd, 'XData', snake.segments(1,1));

set(snake.plot\_head\_hnd, 'YData', snake.segments(1,2));

if snake.collision == 1

set(snake.plot\_head\_hnd, 'Color', 'r');

else

set(snake.plot\_head\_hnd, 'Color', 'g');

end

end

其中snack.segments是主函数中定义的结构体。具体参数如下：

% 贪吃蛇的运动状态

snake.segments = [snake.xmax/2 snake.ymax/2; ...

snake.xmax/2-1 snake.ymax/2; ...

snake.xmax/2-2 snake.ymax/2];

一共三组数据分别代表着下一状态，目前状态和上一状态。适用于所有的点

在应用的时候默认snake.segments(1,1) ，snake.segments(1,2)是贪吃蛇头部的位置，其余为身体的位置。

snake.collision是判断贪吃蛇是否咬到自己的函数变量，其判断方法放在后面说明。

### （2）运动功能函数实现

在运动函数中，分为左转、右转、前进三个运动函数。左转函数和右转函数类似，都是基于蛇目前的状态，即运动方向的上下左右来进行改变运动方向。函数中其中E、W、S、N分别表示右、左、下、上。首先判断目前状态snack.dir，然后修改运动方向，并转变当前状态snack.dir。

unction snake = turn\_left(snake)

snake.segments(2:end, :) = snake.segments(1:end-1, :);

if snake.dir == 'E' % horizontal snake, face east

snake.segments(1, :) = [snake.segments(1, 1) snake.segments(1, 2)+1];

snake.dir = 'N';

elseif snake.dir == 'W' % horizontal snake, face west

snake.segments(1, :) = [snake.segments(1, 1) snake.segments(1, 2)-1];

snake.dir = 'S';

elseif snake.dir == 'N' % vertical snake, face north

snake.segments(1, :) = [snake.segments(1, 1)-1 snake.segments(1, 2)];

snake.dir = 'W';

elseif snake.dir == 'S' % vertical snake, face south

snake.segments(1, :) = [snake.segments(1, 1)+1 snake.segments(1, 2)];

snake.dir = 'E';

end

end

右转方向函数如下。原理同左转函数，故此处介绍省略。

function snake = turn\_right(snake)

snake.segments(2:end, :) = snake.segments(1:end-1, :);

if snake.dir == 'E' % horizontal snake, face east

snake.segments(1, :) = [snake.segments(1, 1) snake.segments(1, 2)-1];

snake.dir = 'S';

elseif snake.dir == 'W' % horizontal snake, face west

snake.segments(1, :) = [snake.segments(1, 1) snake.segments(1, 2)+1];

snake.dir = 'N';

elseif snake.dir == 'S' % vertical snake, face south

snake.segments(1, :) = [snake.segments(1, 1)-1 snake.segments(1, 2)];

snake.dir = 'W';

elseif snake.dir == 'N' % vertical snake, face north

snake.segments(1, :) = [snake.segments(1, 1)+1 snake.segments(1, 2)];

snake.dir = 'E';

end

end

前进函数，其中v是以下一状态坐标-当前状态坐标，得到的矢量可近似看作是速度，用v表示。

function snake = forward(snake)

v = snake.segments(1, :) - snake.segments(2, :);

% Toroidal grid

if v(1,1) == -snake.xmax

v(1,1) = 1;

elseif v(1,1) == snake.xmax

v(1,1) = -1;

end

if v(1,2) == -snake.ymax

v(1,2) = 1;

elseif v(1,2) == snake.ymax

v(1,2) = -1;

end

% 加快移动速度，每次双倍速度。

snake.segments(2:end, :) = snake.segments(1:end-1, :);

snake.segments(1, :) = snake.segments(1, :) + v;

% Toroidal grid again

if snake.segments(1, 1) == snake.xmax + 1

snake.segments(1, 1) = 0;

elseif snake.segments(1, 1) == - 1

snake.segments(1, 1) = snake.xmax;

end

if snake.segments(1, 2) == snake.ymax + 1

snake.segments(1, 2) = 0;

elseif snake.segments(1, 2) == - 1

snake.segments(1, 2) = snake.ymax;

end

snake = check\_collision(snake);

end

### （3）状态功能实现方法函数

状态表示模块中，主要有检查是否碰撞、蛇增长函数。

其中检查是否碰撞函数如下所示，snake.cllision表示是是否发生碰撞，0代表没有碰撞而1代表碰撞。计算方法为：首先将头不位置状态与其他点的位置进行数值等价判断，并且将所有判断结果求和。可知当有一个其余点与头重合时其和>0，因此判断为碰撞。

当判断为碰撞后，令状态值snake.collision = 1;并且让运动状态snake.running = 0；并弹出警告窗口，提示游戏结束，如图5所示。

%%

function snake = check\_collision(snake)

snake.collision = 0;

r = sum(snake.segments(2:end, 1) == snake.segments(1, 1) & snake.segments(2:end, 2) == snake.segments(1, 2));

if r > 0

snake.collision = 1;

errordlg('真不小心，再接再厉！','YOU GO DIE');

pause(3);

snake.running = 0；

end

end

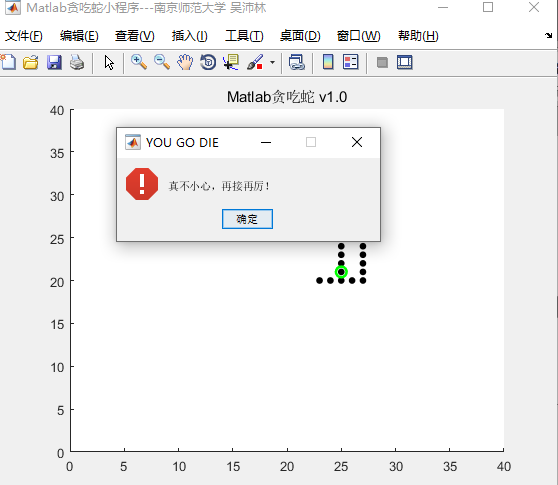


图 5

蛇体增长函数如下，基本思路为首先获得最末端的位置及次末端位置，并且通过两个位置只差获得尾部运动方向，适量看作v。进而在末端再添加一个新位置参数，再把上一状态的末端位置赋值给新末端状态实现延申的效果。

function snake = increase(snake)

v = snake.segments(end, :) - snake.segments(end-1, :);

%获得末端状态

if v(1,1) == -snake.xmax

v(1,1) = 1;

elseif v(1,1) == snake.xmax

v(1,1) = -1;

end

if v(1,2) == -snake.ymax

v(1,2) = 1;

elseif v(1,2) == snake.ymax

v(1,2) = -1;

end

%增加一个点

snake.segments(end+1,:) = snake.segments(end,:) + v;

% 延续总体的点

if snake.segments(end,1) == snake.xmax + 1

snake.segments(end, 1) = 0;

elseif snake.segments(end, 1) == - 1

snake.segments(end, 1) = snake.xmax;

end

if snake.segments(end, 2) == snake.ymax + 1

snake.segments(end, 2) = 0;

elseif snake.segments(end, 2) == - 1

snake.segments(end, 2) = snake.ymax;

end

end

### （4）记分系统

由于本游戏设计异于传统吃蛋模式，而是一种类似与竞速的模式。所以增加的最高长度即为分数。计算的长度和时间均在循环函数中实现。具体如下所示：

其中count\_time为计时变量，count为计数变量。且两个参数再ClickButton1中初始化，由于初始化时候蛇的身子为3格，因此起始值为3。且每当帧数取余5等于3的时候，即增加一节长度。其判断公式为：mod(count\_time,5) ==3

function snake\_loop(snake,pause\_time,handles,count\_time,count)

% The main loop

while 1

snake = guidata(snake.fig\_hnd);

snake = forward(snake);

draw\_snake(snake);

count\_time = count\_time+1;

if mod(count\_time,5) ==3

snake = increase(snake);

count = count +1;

end

guidata(snake.fig\_hnd, snake);

pause(pause\_time);

set(handles.text4,"string",count);

if snake.running == 0

close;

end

end

close all;

end

### （4）按键获取实现方法

按键获取通过设置函数，不停地读取键盘上的值。'uparrow'，'rightarrow'，'leftarrow'，'downarrow' 代表四个方向键的值，当检测到相关按键的时候，蛇的状态发生改变。此外，还设置了按住空格键快速增加长度的设置，通过按“ESC”键退出游戏的设置。

function [] = key\_pressed\_fcn(H, E)

snake = guidata(H);

% Figure keypressfcn

switch E.Key

case 'rightarrow'

if (snake.dir == 'N')

snake = turn\_right(snake);

elseif (snake.dir == 'S')

snake = turn\_left(snake);

elseif (snake.dir == 'E')

snake = forward(snake);

end

case 'leftarrow'

if (snake.dir == 'N')

snake = turn\_left(snake);

elseif (snake.dir == 'S')

snake = turn\_right(snake);

elseif (snake.dir == 'W')

snake = forward(snake);

end

case 'uparrow'

if (snake.dir == 'E')

snake = turn\_left(snake);

elseif (snake.dir == 'W')

snake = turn\_right(snake);

elseif (snake.dir == 'N')

snake = forward(snake);

end

case 'downarrow'

if (snake.dir == 'E')

snake = turn\_right(snake);

elseif (snake.dir == 'W')

snake = turn\_left(snake);

elseif (snake.dir == 'S')

snake = forward(snake);

end

case 'space'

snake = increase(snake);

case 'escape'

snake.running = 0;

disp('再见！');

otherwise

end

snake = check\_collision(snake);

guidata(snake.fig\_hnd, snake);

draw\_snake(snake);

end

### （5）主函数实现方法

在本游戏设计中，主函数主函数由PushButton按键作为触发键。其内容如下所示：

function pushbutton1\_Callback(hObject, eventdata, handles)

%初始化相关参数

count\_time = 0;

count = 3;

snake.running = 1;

snake.collision = 0;

pause\_time = 0;

rad\_bu1=get(handles.radiobutton1,'value');

rad\_bu2=get(handles.radiobutton2,'value');

rad\_bu3=get(handles.radiobutton3,'value');

if rad\_bu1 == 1

pause\_time=0.2;

end

if rad\_bu2 ==1

pause\_time=0.1;

end

if rad\_bu3==1

pause\_time=0.03;

end

% 地图大小初始化

snake.xmax = 40;

snake.ymax = 40;

% 绘制蛇初始化

snake.fig\_hnd = figure('NumberTitle', 'off', 'Name', 'Matlab贪吃蛇 南京师范大学 吴沛林);

hold on;

snake.plot\_hnd = plot(0, 0, 'k.', 'MarkerSize', 15);

snake.plot\_head\_hnd = plot(0, 0, 'bo', 'MarkerSize', 8, 'LineWidth', 2);

% 蛇身体参数的初始化

snake.segments = [snake.xmax/2 snake.ymax/2; ...

snake.xmax/2-1 snake.ymax/2; ...

snake.xmax/2-2 snake.ymax/2];

snake.dir = 'E';

% 从GUI更新参数

guidata(snake.fig\_hnd, snake);

%绘制

init\_figure(snake);

draw\_snake(snake);

%主循环函数

snake\_loop(snake,pause\_time,handles,count\_time,count);

end

# 3.游戏测试和总结

## 3.1测试画面

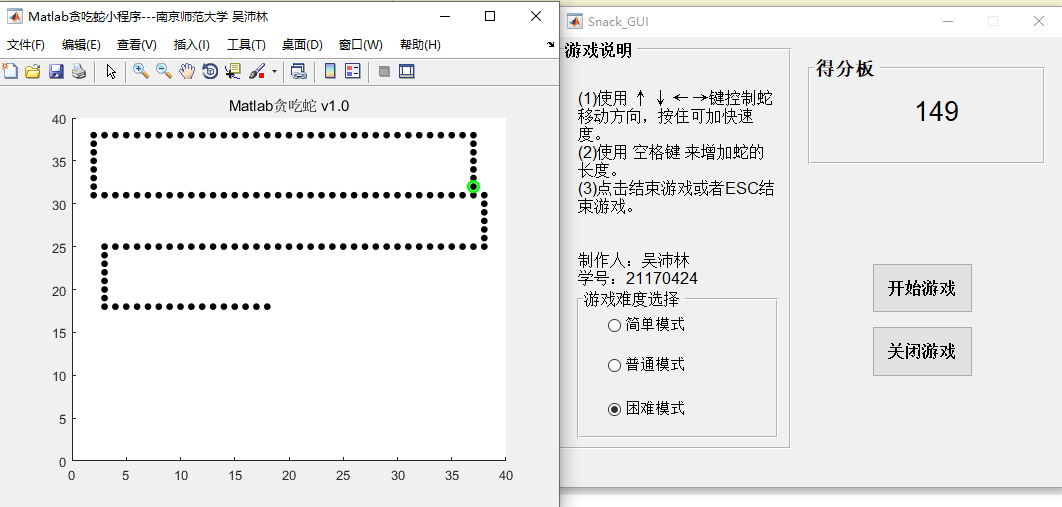


图 6

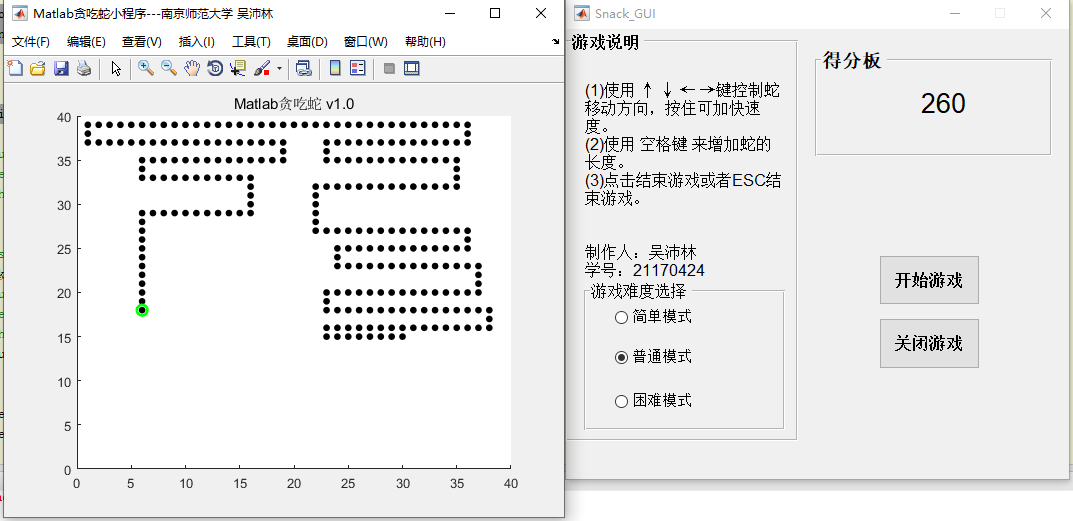


图 7

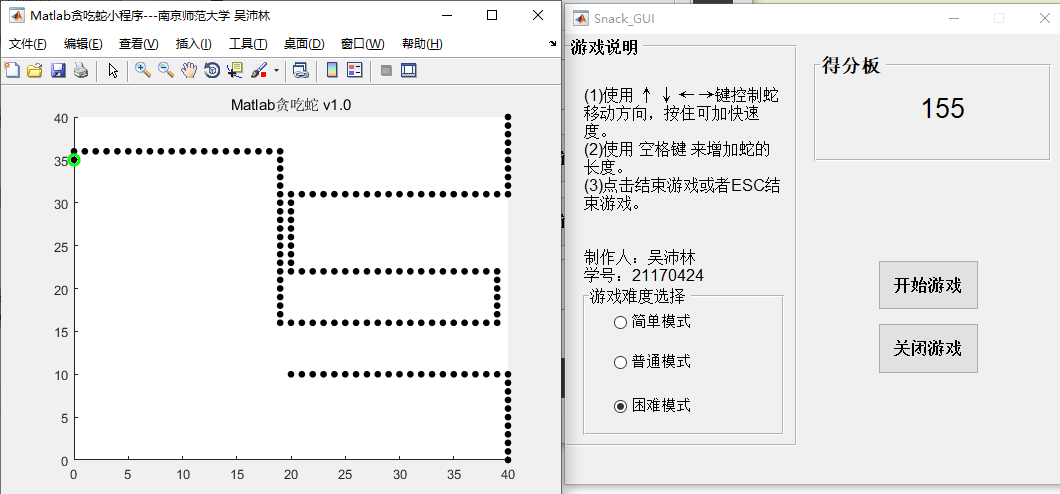


图 8

## 3.2总结收获

在本次游戏设计中，尽量使用了本学期Mablab课程中所学的知识，包括GUI界面的设计、子函数的定义和调用、数据可视化、结构体数据的使用以及循环嵌套等语句。基本囊括了本学期所学的内容，当完成本项目后，感觉对于Matlab更加熟悉，在今后一定对于自己的科研方面也有积极影响。

除了掌握了基本的操作，在完成本项目的过程中也通过解决无数次的报错，熟悉了相关数据之间的关系。尤其是对于GUI数据更新的语句，起初非常难理解为什么要存在这类语句，感觉不如C/C++语言直观。但是后来意识到，抽象化程度较高的语句，虽然理解起来不如其他语句容易，但是在实现功能方面却会让程序设计更加容易。

熟练的掌握搜索工具对于编程也是极为有用的。在编程的初期，遇到很多问题在百度上直接搜索，往往会出现很多不相关的结果。后来发现一些Matlab的论坛中也有很多资料可以查询，甚至有别人实现的相关函数功能可以借鉴思考，大大增加了编程的效率问题。

“工欲善其事必先利其器”，我们在科研学习的过程中，对于很多数学问题或者系统设计问题往往在大脑中已经有了初步的解决方案。但是如何将“简单的”方案落实到计算机上确是比较困难的，在选题之初我选择游戏设计是因为想当然地认为像“贪吃蛇”这种小游戏应该很简单。但是当落实到实际以后发现，虽然基本的语法是掌握的，但是在系统实现的时候掌握基本的语法是远远不够的。除了基本的语法之外，还需要设计好背后的数学原理，更需要时时考虑数据存放的问题。此外，还需要掌握一些更加高级的语法来实现功能，否则真的就是“报错连篇”了啊。由此看来，在Matlab学习的道路上还是要永不停息，通过实践来掌握这门语言。

# 引用文献：

[1]刘港毓,龙波,欧阳朝煌,张磊涛.浅析《贪吃蛇》游戏设计与制作[J].科技风,2017(05):68.

[2]张书乐.贪吃蛇必须死[J].电脑爱好者,2013(07):20-21.