汇编语言与接口设计

分组实验：贪吃蛇游戏设计

小组成员：龙水彬（1120161842）、吴兴勇（1120161849）、向思疑（1120161850）

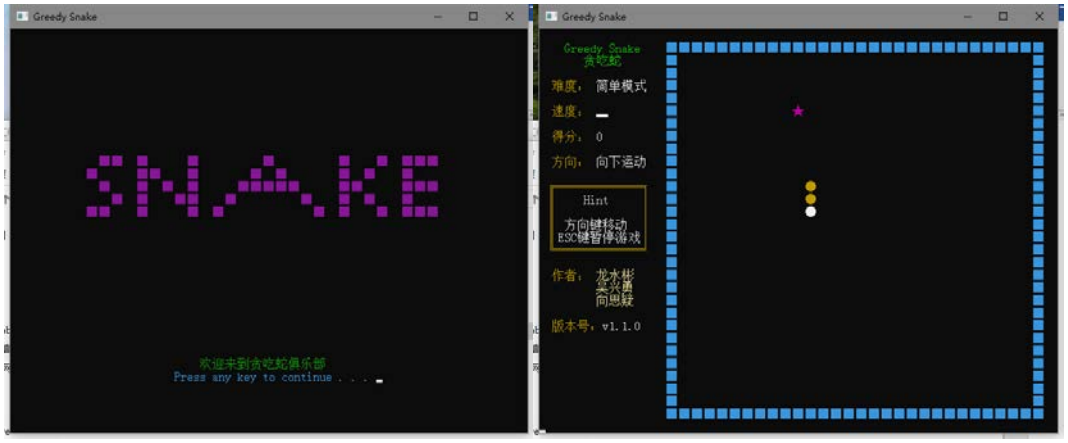
一、 实验分工

- **龙水彬：**游戏的逻辑构思与代码编写，游戏框架设计，参与地图绘制、食物绘制、蛇身绘制等代码实现，完成汇讲工作。
- **吴兴勇：**参与游戏框架、游戏模式设计，参与软件测试，设计并实现贪吃蛇 AI 自动寻路功能。
- **向思疑：**查阅并学习基础汇编语言函数，并传授给同组同学，参与界面交互与选择设计、参与地图绘制，参与软件测试。

二、 实验内容与概述

本实验是基于汇编语言，使用 Visual Studio 2019 完成的经典游戏《贪吃蛇》的设计与实现工作。其游戏内容包括：

- 每时刻，蛇头朝运动方向位移一定距离，由速度决定，同时方向键（上、下、左、右）控制蛇头移动方向。
- 同时地图上会随机生成食物，需要控制蛇觅食，每吃到一颗果子，积分增加，蛇身伸长，当蛇吃到食物的同时，地图上重新生成食物。
- 当然地图上有一定几率生成奖励食物，该食物有时间限制，吃到该食物的奖励积分为$\frac{\text{食物剩余时间}\times\text{难度系数}}{\text{比例系数}}$。
- 游戏包括多种梯度型难度模式，不同难度下，蛇头的速度不同，食物的积分不同。
- 游戏有评价系统，每轮游戏根据得分结合难度系数综合评价。



（图 2-1，游戏部分画面）

三、 实验环境

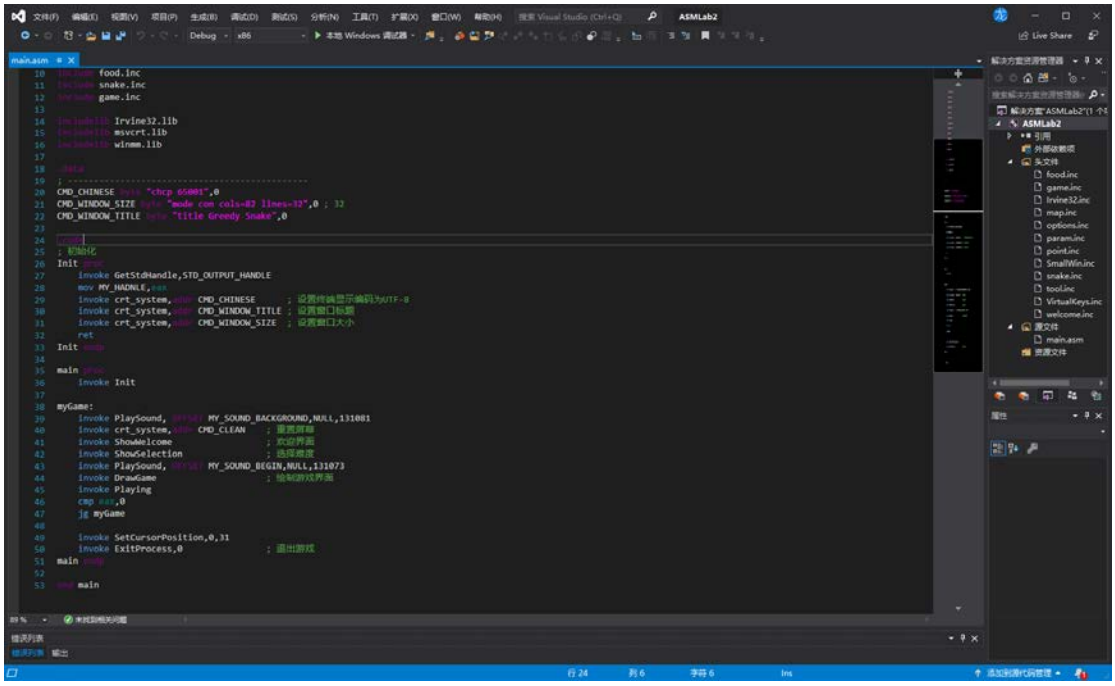
本实验使用的操作系统是 Windows 10，其配置如下：

- **系统：**Windows 10 教育版
- **处理器：**Intel(R) Core(TM) i5-6300HQ CPU @ 2.30GHz
- **内存：**16.0 GB
- **显卡：**NVDIA GeForce GTX 965M

本实验使用的汇编语言版本是基于 Windows 的 masm SDK 控件，它的下载地址为：http://www.masm32.com/

本实验使用的 Visual Studio 版本配置如下：

- **版本：**Community 16.1.3
- **插件：**
  - **AsmHighlighter 功能：**高亮汇编语言代码，增强辨识度，效果如图 3-1 所示
  - **Live Share 功能：**便于多人同时编写贪吃蛇游戏项目，提高效率



（图 3-1，汇编高亮插件渲染的 IDE 主界面）

四、 实验步骤

4.1 终端绘图

- 设置光标位置 *SetConsoleCursorPosition*，函数如图 4-1 所示。

```
; 设置光标位置
_SetCursorPosition proc X:word,Y:word
    local pos:coord
    mov ax,X
    mov pos.X,ax
    mov ax,Y
    mov pos.Y,ax
    invoke SetConsoleCursorPosition,MY_HADNLE,pos
    ret
_SetCursorPosition endp
```

（图 4-1，设置光标位置）

- 设置终端输出颜色：*SetConsoleTextAttribute*

```
; 设置文本颜色
SetFontColor proc ColorID:word
    invoke SetConsoleTextAttribute,MY_HADNLE,ColorID
    ret
SetFontColor endp
```

（图 4-2，设置输出文本颜色）

- 输出对应字符：*crt_printf*，部分 Unicode 字符长占两个光标单位，下图 4-3 是基于输出字符串函数完成界面基本元素的输出工作，PrintBlock 完成输出方形字符，用于绘制界面墙，PrintCircle 完成输出圆形字符，用于绘制蛇身，PrintFood 用于输出食物字符，用于绘制生成的食物，PrintBlank 用于擦除地图上存在的字符（实质上是在原字符位置用空字符完成覆盖）
  - **参数：**字符串首地址

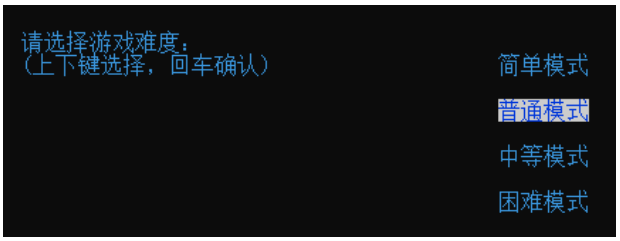
```
.data
MY_BLOCK    byte "■",0
MY_CIRCLE   byte "●",0
MY_FOOD     byte "★",0
MY_BLANK    byte "  ",0

.code
PrintBlock proc X:word,Y:word
    invoke SetCursorPosition,X,Y
    invoke crt_printf,addr MY_BLOCK
    ret
PrintBlock endp

PrintCircle proc X:word,Y:word
    invoke SetCursorPosition,X,Y
    invoke crt_printf,addr MY_CIRCLE
    ret
PrintCircle endp

PrintFood proc X:word,Y:word
    invoke SetCursorPosition,X,Y
    invoke crt_printf,addr MY_FOOD
    ret
PrintFood endp

PrintBlank proc X:word,Y:word
    invoke SetCursorPosition,X,Y
    invoke crt_printf,addr MY_BLANK
    ret
PrintBlank endp
```

（图 4-3，输出字符）

4.2 交互模式

本游戏的交互模式包括键盘输入与屏幕反馈，函数描述如下：

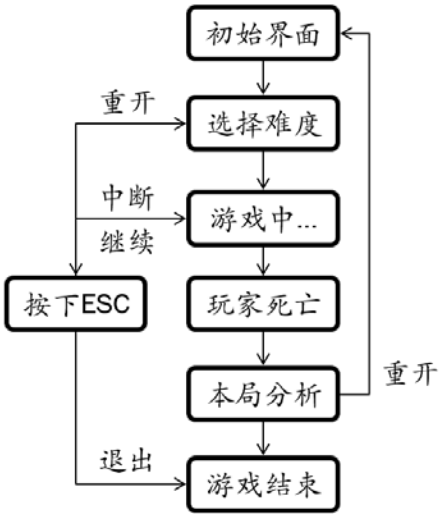
- 键盘输入函数：*ReadChar*、*ReadKey*
  - *ReadChar:* 等待从键盘输入字符，用于交互选择时
  - *ReadKey:* 无需等待输入，立即从键盘输入缓冲区中读取一个字符，用于蛇头的运动方向调整



（图 4-4，交互界面之一）

如图 4-4，在选择关卡时候需要用于通过上下键调整模式选择。该部分逻辑是通过变量维护当前高亮的区域，设置该区域的背景颜色以达到高亮显示的目的。

4.3 游戏逻辑



（图 4-5，游戏逻辑图）

根据图 4-5 描述的游戏逻辑设计，贪吃蛇部分需要三个游戏界面，即初始界面用于选择游戏难度，还有游戏中画面，用于实时绘制贪吃蛇，最后是游戏结束画面，用于与玩家交互，同时也用于评价系统的展示，提高游戏趣味度。

4.4 游戏初始界面

该界面使用点阵图绘制，需要等待任意键输入，输入完成后，立即进入“选择难度”，选择难度的模式如上图 4-4 所示，需要高亮当前选项。本部分代码如图 4-6 所示，详细代码部分请参考附件代码中的 welcome.inc 和 options.inc。

```
.code
DrawBoard proc
    mov ecx,53
    mov edx,0
myWelDraw:
    push ecx
    push edx
    mov ebx,offset MY_LAB
    add ebx,edx
    mov esi,0
    mov ax,word ptr[ebx+esi]
    mov WEL_X,ax
    add esi,2
    mov ax,word ptr[ebx+esi]
    mov WEL_Y,ax
    invoke PrintBlock,WEL_X,WEL_Y
    pop edx
    pop ecx
    add edx,4
    loop myWelDraw
    ret
DrawBoard endp

ShowWelcome proc
    invoke SetFontColor,5
    invoke DrawBoard
    invoke SetFontColor,2
    invoke SetCursorPosition,15,26
    invoke crt_printf,addr MY_WELCOME
    invoke SetFontColor,3
    invoke SetCursorPosition,13,27
    invoke crt_system,addr CMD_PAUSE
    ret
ShowWelcome endp
```

（图 4-6，初始界面绘制代码逻辑）

4.5 游戏界面

游戏界面包括三要素：绘制地图、绘制菜单面板和绘制贪吃蛇。下面依次介绍各部分的代码逻辑工作。详细的代码内容部分请参考附件代码的 map.inc 文件。

4.5.1 绘制地图

根据地图特征预处理点阵信息用于定位游戏区域的矩形，我们使用二维数组定位地图上的每个像素点，使用 PrintBlock 的函数方法输出带颜色的边框区域，效果图如 2-1 所示，点阵代码如 4-7 所示：

```
.data
MY_MAP  word 10,1,11,1,12,1,13,1,14,1,15,1,16,1,17,1,18,1,19,1,20,1,21,1,22,1,23,1,24,1,25,1,26,1,27,1,28,1,29,1
        word 30,1,31,1,32,1,33,1,34,1,35,1,36,1,37,1,38,1,39,1
        word 10,2,39,2,10,3,39,3,10,4,39,4,10,5,39,5,10,6,39,6,10,7,39,7,10,8,39,8,10,9,39,9,10,10,39,10,10,11,39,11
        word 10,12,39,12,10,13,39,13,10,14,39,14,10,15,39,15,10,16,39,16,10,17,39,17,10,18,39,18,10,19,39,19,10,20,39,20
        word 10,21,39,21,10,22,39,22,10,23,39,23,10,24,39,24,10,25,39,25,10,26,39,26,10,27,39,27,10,28,39,28,10,29,39,29
        word 10,30,11,30,12,30,13,30,14,30,15,30,16,30,17,30,18,30,19,30,20,30,21,30,22,30,23,30,24,30,25,30,26,30
        word 27,30,28,30,29,30,30,30,31,30,32,30,33,30,34,30,35,30,36,30,37,30,38,30,39,30
```

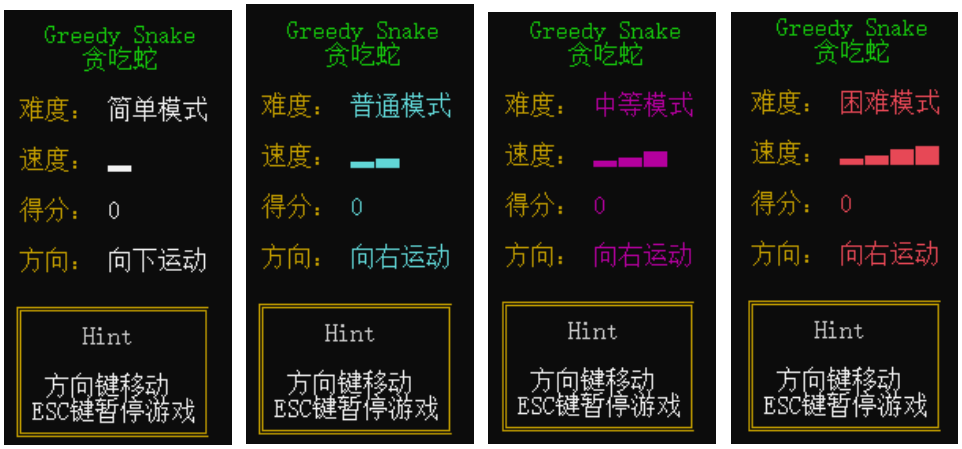
（图 4-7-a，边界区域点阵数据）

```
.code
InitMap proc
    mov ecx,117
    mov edx,0
myMapDraw:
    push ecx
    push edx
    mov ebx,offset MY_MAP
    add ebx,edx
    mov esi,0
    mov ax,word ptr[ebx+esi]
    mov MAP_X,ax
    add esi,2
    mov ax,word ptr[ebx+esi]
    mov MAP_Y,ax
    invoke PrintBlock,MAP_X,MAP_Y
    mov eax,5
    call Delay
    pop edx
    pop ecx
    add edx,4
    loop myMapDraw
    ret
InitMap endp
```

（图 4-7-b，绘制墙壁的部分代码）

4.5.2 绘制菜单面板

菜单面板我们通过前置选择的难度关卡调整菜单面板的相应颜色，效果如下图 4-8 所示，其实现方法也是基于点阵的输出，由于代码过于冗长，这里不做展示，详细请参考代码附件的 map.inc 的 DrawMainMenu 函数，里面也包含了我们编写时候的注释，主要是实现 switch-case 功能。



（图 4-8，颜色调整）

4.5.3 绘制贪吃蛇

该部分工作包括维护蛇身、蛇头，还有绘制对应位置。

维护蛇身同样使用的二维数组存储对应坐标位置，但是蛇身每一个画面帧都会运动，而且运动具有特征性，即头部沿着运动方向伸长，尾部沿着运动方向缩短，这个特征正好符合队列这种数据结构，如果使用队列维护它，每帧画面只需要将队尾出列，并再次计算蛇头新

坐标并放入队头即可实现这个工作。但是汇编语言中并没有直接的库函数，需要自己手动维护，我们基于二维连续数组变量，维护了静态连续内存块（用于标识队列内容），同时增加两个变量用于标识队头和队尾。代码部分如下图 4-9：

```
141  Growing proc    ; 蛇身前进一格
142      mov ebx,offset MY_SNAKE_POS
143      movzx esi,MY_SNAKE_LEN
144      dec esi
145      imul esi,4
146      mov cx,word ptr[ebx+esi]
147      mov dx,word ptr[ebx+esi+2]
148      add esi,4
149      .if MY_SNAKE_DIR==0     ; up
150          dec dx
151      .elseif MY_SNAKE_DIR==1 ; down
152          inc dx
153      .elseif MY_SNAKE_DIR==2 ; left
154          dec cx
155      .elseif MY_SNAKE_DIR==3 ; right
156          inc cx
157      .endif
158      mov [ebx+esi],cx
159      mov [ebx+esi+2],dx
160      mov ax,MY_SNAKE_LEN
161      inc ax
162      mov MY_SNAKE_LEN,ax
163      invoke DrawSnake
164      ret
165  Growing endp
```

```
167  Moving proc
168      local X:word,Y:word
169      ; 删除蛇尾
170      mov esi,offset MY_SNAKE_POS
171      mov ax,[esi]
172      mov X,ax
173      mov ax,[esi+2]
174      mov Y,ax
175      invoke PrintBlank,X,Y
176
177      ; 原蛇身平移一格
178      movzx ecx,MY_SNAKE_LEN
179      dec ecx
180      mov edx,0
181  myMovLoop:
182      mov esi,offset MY_SNAKE_POS
183      add esi,edx
184      mov ax,[esi+4]
185      mov [esi],ax
186      mov ax,[esi+6]
187      mov [esi+2],ax
188      add edx,4
189      loop myMovLoop
190      mov ax,MY_SNAKE_LEN
191      dec ax
192      mov MY_SNAKE_LEN,ax
193
194      invoke Growing  ; 蛇头前进
195      ret
196  Moving endp
```

（图 4-9，维护蛇身移动）

而蛇头就是队列中的队头坐标啦，这里为了区分蛇头，使用不同于黄色的蛇身的颜色——白色用来标识蛇头位置。

而有了二维蛇身的坐标，接下来只要在每帧画面中循环遍历整个蛇身数组逐个像素点绘制即可，当然也可以优化一下，因为每次移动实质上只有蛇头和蛇尾的变化，中间的元素是没有发生位置移动和变化的，这样可以将绘制蛇身的时间复杂度从$O(n)$降低到$O(1)$。

4.5.4 绘制贪吃蛇的食物

对于蛇的食物，整个过程包括随机食物坐标、判断该坐标的合法性和最后的食物绘制。其中我们还需要避免食物坐标生成到已知的地图元素中，例如该坐标本来就是蛇身，再例如该坐标已经存在了食物。本部分代码如图 4-10 所示。

当然，对于奖励食物，为了标识它的不同，我们在每一帧画面都赋予奖励食物随机的颜色，这样还实现了奖励食物的闪烁绘制。

生成食物之后还需要检测蛇是否吃中了食物，本部分不需要遍历蛇的整个身位，因为蛇每时每刻都在移动，而吃到食物的瞬间蛇一定只有蛇头和食物重合，因此只需要检测蛇头的坐标即可，当蛇头吃到食物后，就触发蛇身增长，同时更新游戏积分并刷新新一轮的食物。而蛇身增长的逻辑，也和蛇身移动逻辑类似，只不过蛇尾不需要出队，这里不再重复赘述其逻辑部分。

对于奖励食物，我们的机制还包括了奖励食物吃到它的时间，因此我们还维护了一个倒计时机制，若游戏中存在奖励食物我们将触发它的闪烁效果，同时触发它的倒计时时钟装置，倒计时用于积分，也就是蛇身越早吃到奖励食物，它获得的积分就越高。同时不同难度下奖

励积分的加倍效果也不同。

```
DrawFood proc
    local flag:byte
    mov flag,1
    .while flag
        invoke Random,11,38
        mov MY_FOOD_X,ax
        invoke Random,2,29
        mov MY_FOOD_Y,ax
        mov flag,0
        movzx ecx,MY_SNAKE_LEN
        mov edx,0
myDrawCheck:
        push ecx
        push edx
        mov esi,offset MY_SNAKE_POS
        add esi,edx
        mov ax,word ptr[esi]
        mov cx,ax
        mov ax,word ptr[esi+2]
        .if flag==0 && MY_FOOD_X==cx && MY_FOOD_Y==ax
            mov flag,1
        .endif
        pop edx
        pop ecx
        add edx,4
        loop myDrawCheck
    .endw
    invoke SetFontColor,13
    invoke PrintFood,MY_FOOD_X,MY_FOOD_Y
    mov ax,MY_FOOD_CNT
    inc ax
    mov MY_FOOD_CNT,ax
    ; 是否生成奖励食物
    mov dx,0
    mov ax,MY_FOOD_CNT
    mov cx,5
    div cx
    .if dx==0
        invoke DrawBigFood
    .endif
    ret
DrawFood endp
```

（图 4-10，绘制食物）

4.6 游戏特效

为了有别于其它小组的贪吃蛇，增强特性，我们增加了几个贪吃蛇游戏玩家经常需要的体验感：

- **游戏加速：**在慢速条件下，虽然可以很谨慎的玩游戏，但是也容易不耐烦，因此增加了蛇身加速功能，只需要长按相同的方向键，就会触发蛇身加速，这样提高了蛇身运动效率。
  - 本功能的实现是通过检测多次采样的方向键是否相同，如果相同则以提高游戏延时帧率的方法加速蛇身运动。
- **评价系统：**在游戏结束后，系统会根据玩家的得分与游戏模式的难度综合给出评价，效果如图 4-11 所示。
  - **5 分以下：**一星
  - **6 分~14 分：**二星
  - **15 分~29 分：**三星
  - **30 分~49 分：**四星
  - **50 分以上：**五星
- **游戏音乐：**音乐是游戏的灵魂之一，我们百般收集，查阅了相当多的游戏资料，进行了不同游戏音效的测试与比对，综合选择了最适合本游戏的贪吃蛇音效，整个游戏效果我们也录制了 demo.mp4 文件，建议打开观看，体验感极佳。
- **自动寻路：**对于这类游戏，我们综合近些年非常火热的 AI 算法，实现了贪吃蛇自动寻路功能，展示效果也在 demo.mp4 文件中，算法主要通过 A* 实现的启发式搜

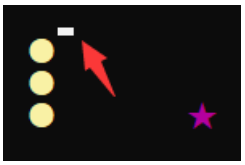
素。如果蛇身当前可以直接到达食物位置，则立即规划最短路线前往食物，如果食物不在最短路线中则让它随机走动或者追随尾巴，直到适当的时机再奔向食物。

（图 4-11，评价系统）

五、 实验总结

5.1 实验遇到的问题与解决方法

- **典型问题：**终端光标严重影响画面体验感。本程序是基于终端界面绘图实现的动画，在输出图形后光标将默认停止在字符末尾的位置，看起来十分出戏，不够美观。



（图 5-1，终端光标乱）

  - **解决方案：**每次绘图后，强行设置光标位置为屏幕左下角位置，这样蛇身运动的动画较为连贯和谐。但是也要注意不要设置过于频繁，否则容易影响程序运行效率。

5.2 心得体会

实验需要实现的功能较为复杂，从界面绘制到后端逻辑设计，再到龙水彬同学突发奇想，希望和吴兴勇同学一块研制贪吃蛇 AI，我们小组成员之间分工合作，一步步完成了这个经典的游戏，期间也遇到了不少问题，经过我们反复的讨论和重复的实验最终逐一解决。不仅仅加深了我们对汇编语言的知识的学习与巩固，还提高了我们程序设计能力，从底层上理解了高级程序设计语言背后的汇编语言的特征与魅力。最后希望老师保佑我们考试顺利！