**目录**

**[贪吃蛇设计思路](#_Toc27184_WPSOffice_Level1)** **[2](#_Toc27184_WPSOffice_Level1)**

**[一、设计要求](#_Toc30725_WPSOffice_Level1)** **[2](#_Toc30725_WPSOffice_Level1)**

[1、 游戏需要实现的基本因素](#_Toc30725_WPSOffice_Level2) [2](#_Toc30725_WPSOffice_Level2)

[2、 游戏规则细化](#_Toc4348_WPSOffice_Level2) [2](#_Toc4348_WPSOffice_Level2)

[3、进阶：实现人机交互界面](#_Toc9731_WPSOffice_Level2) [2](#_Toc9731_WPSOffice_Level2)

[4、具体界面如图所示](#_Toc23832_WPSOffice_Level2) [3](#_Toc23832_WPSOffice_Level2)

**[二、 游戏主界面的实现](#_Toc4348_WPSOffice_Level1)** **[4](#_Toc4348_WPSOffice_Level1)**

[1、 主界面维持](#_Toc21388_WPSOffice_Level2) [4](#_Toc21388_WPSOffice_Level2)

[2、 蛇的方向](#_Toc25223_WPSOffice_Level2) [4](#_Toc25223_WPSOffice_Level2)

[3、 蛇的移动](#_Toc20030_WPSOffice_Level2) [5](#_Toc20030_WPSOffice_Level2)

[4、 食物的产生](#_Toc12818_WPSOffice_Level2) [6](#_Toc12818_WPSOffice_Level2)

[5、 墙的形成](#_Toc32476_WPSOffice_Level2) [6](#_Toc32476_WPSOffice_Level2)

[6、 碰撞判断](#_Toc10276_WPSOffice_Level2) [6](#_Toc10276_WPSOffice_Level2)

[7、 键盘输入及控制](#_Toc21514_WPSOffice_Level2) [7](#_Toc21514_WPSOffice_Level2)

[8、 程序自启动](#_Toc5388_WPSOffice_Level2) [7](#_Toc5388_WPSOffice_Level2)

[9、 clear() 、 wclear(WINDOW\* win) 与 refresh() 的应用](#_Toc4117_WPSOffice_Level2) [7](#_Toc4117_WPSOffice_Level2)

**[三、 其他界面设计](#_Toc9731_WPSOffice_Level1)** **[7](#_Toc9731_WPSOffice_Level1)**

[1、 分窗口显示](#_Toc60_WPSOffice_Level2) [7](#_Toc60_WPSOffice_Level2)

[2、 设置三个bool量 控制相对于的界面](#_Toc31159_WPSOffice_Level2) [8](#_Toc31159_WPSOffice_Level2)

**贪吃蛇设计思路**

**一、设计要求**

1. 游戏需要实现的基本因素
2. 游戏开始，蛇会向着初始方向移动
3. 按下上下左右健时，蛇会变换方向
4. 如果蛇撞到墙，游戏结束
5. 如果蛇撞到自己的身体，游戏结束
6. 随机出现食物
7. 碰到食物加分
8. 游戏规则细化
9. 如果按下的方向健与前进的方向相反，不予改变方向
10. 只有蛇碰到食物时，食物才能变化位置
11. 分数越高，蛇的速度越快

3、进阶：实现人机交互界面

1. 拥有游戏开始界面

\* 按下“P”健开始游戏

\* 按下“Esc”健退出游戏

1. 拥有游戏运行时的主界面

\* 按下上下左右控制方向

\* 按下“Esc”健结束游戏

1. 拥有游戏结束后的计分界面

\* 输出游戏中的最终得分

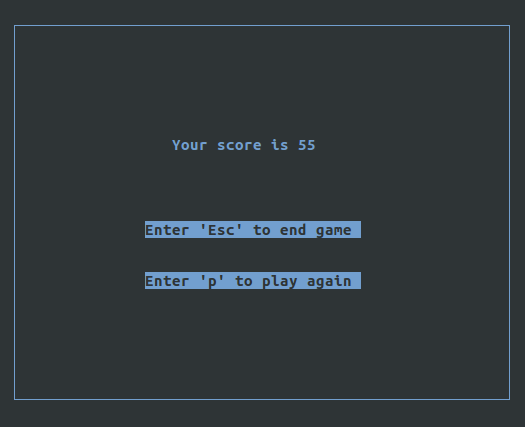
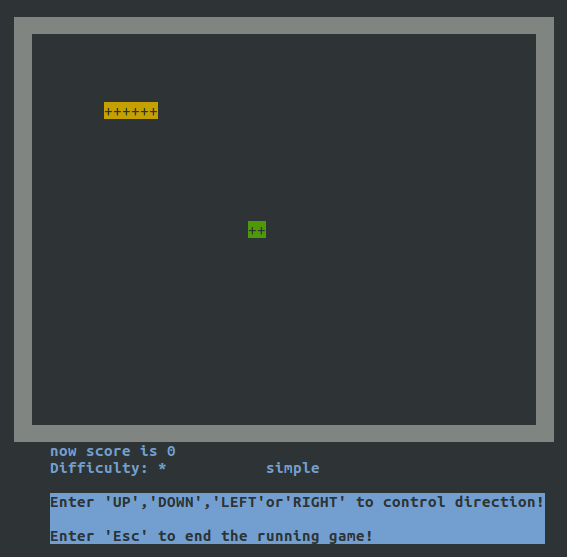
\* 按下“P”健在玩一遍

\* 按下“Esc”健退出游戏

1. 在各个界面下说明操作规则

4、具体界面如图所示





1. **游戏主界面的实现**
2. 主界面维持

设置while循环，

结束循环条件：

\*撞到墙

\*撞到蛇身

\*按下“Esc”健

1. 蛇的方向

设置一个变量F表示方向

当F = 1 ，表示方向为 上

当F = 2 ，表示方向为 下

当F = 3 ，表示方向为 左

当F = 4 ，表示方向为 右

1. 蛇的移动

蛇的移动我们应用坐标法来实现。蛇的移动可以理解为改变坐标，再重新打印出蛇的位置。如下图：

原位置：（1，1），（1，2），（2，2）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 2 | 3 |
| 1 | \* | \* |  |
| 2 |  | \* |  |
| 3 |  |  |  |

若（2，2）为头，方向向下，则下一刻新位置为：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 2 | 3 |
| 1 |  | \* |  |
| 2 |  | \* |  |
| 3 |  | \* |  |

新位置变成：（1，2），（2，2），（3，2）

可以看出蛇的移动实际上，只是删除尾坐标，并重新定义头坐标。而其他的坐标继承他们前一个位置的坐标。

若用数组来解释：

Int snack[3][2]; // 3表示蛇的长度 、 2用来储存坐标 Y 、 X 。

T = t 0 时，snack[1][0] = 1 , snack[1][1] = 1

T = t 1 时，snack[1][0] = 1 , snack[1][1] = 2

即snack[k][0] = snack[k+1][0] , snack[k][1] = snack[k+1][1]

1. 食物的产生

rand()函数产生两个随机数 X , Y

随机数表示坐标（X , Y）

1. 墙的形成

给WINDOW 加上边框

加边框函数：

（1）box(WINDOW\* win , ACS\_VLINE,ACS\_HLINE);

（2）int b = ACS\_CKBOARD;

wborder(midscreen1,b,b,b,b,b,b,b,b);

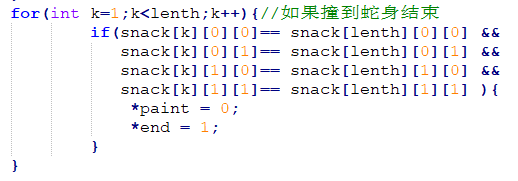
1. 碰撞判断
2. 碰到墙

|  |  |  |
| --- | --- | --- |
|  | up |  |
| left | （X , Y） | right |
|  | down |  |

得出： left < X < right ; down < Y < up;

1. 碰到蛇身

如果没碰到蛇身，储存的坐标是不可能重复的。所以可以历遍自身看看是否有坐标重合。

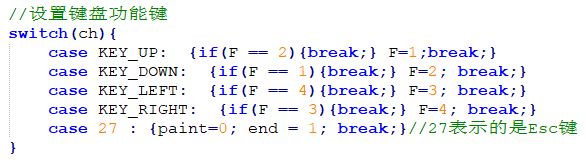


1. 键盘输入及控制

首先开启功能健

keypad(WINDOW\* win , TURE);

通过getch()得到的字符进行判断



1. 程序自启动

由于程序中有getch()，这会照成程序强制暂停，等待返回值。在curses库中有一个函数可以专门解决这个问题。这就是 half delay( int time );

halfdelay()函数会启用半延时模式（halfdelaymode）。和cbreak()函数一样，当程序需要用户输入字符时，输入的每个字符都是可用的。给halfdelay()传递一个整型参数（以0.1 秒为单位），它就会在这个参数时间内等待输入。如果没有有效的输入，则返回ERR。 一般来说，这个函数在需要等待输入的程序中会用到。如果用户没有在规定时间内给出有效输入，程序就可以去处理其它事情。最常见例子就是在输入密码时做出超时响应。

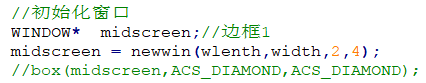
1. clear() 、 wclear(WINDOW\* win) 与 refresh() 的应用

由于在游戏过程中，墙得一直存在，而蛇的位置却是实时变化的。所以可以给墙内的空间设置个窗口，给墙设置一个窗口。在运行过程中wclear() 墙内空间窗口即可。如果不分开，墙和蛇一起打印会出现严重的闪屏问题。

1. **其他界面设计**
2. 分窗口显示

分别给各个界面一个窗口，方便各个窗口的clear() 与 printf()。

并且他们之间不会相互影响。在程序中，gstch() 与wgetch() 同样是获取字符，但wgetch(WINDOW\* win) 可以避免同个getch返回值在三个界面中混用的情况。所以程序中，有两个“Esc”健的功能并不混淆，而且第二个“Esc”并不会对其他两个的“Esc”照成影响。



1. 设置三个bool量 控制相对于的界面

bool量的设立可以实现人机交互的目的。

