UEFI贪吃蛇

李万亭　　1652724
周裕　　1652713

目录

1、程序功能介绍

项目实现了一个界面简单的UEFI贪吃蛇小游戏，下文将对小游戏的几个基本方面进行简要描述：

1.1 SHELL应用

UEFI应用主要有三种：主要有三类：ShellAppMain类型、UefiMain类型和Main类型。这三种类型的功能丰富程度是递增的，ShellAppMain类型是Shell下的应用，UefiMain类型则具有镜像句柄、系统表等功能，而Main类型的应用只能在AppPkg环境下编译运行，可以通过StdLib包调用C语言里常用的库函数。

ShellAppMain类型应用在ShellPkg中，我们选择了直接在edk2给出的示例应用ShellCTestApp的基础上实现贪吃蛇小游戏。

1.2 界面与操作

贪吃蛇小游戏中，我们设置了三个主要界面：

欢迎界面：包含程序内容和作者信息等。

菜单界面：玩家可以移动光标，选择开始游戏、查看最高记录、重置记录和退出游戏。

游戏界面：分为贪吃蛇地图和右侧的操作指南。

玩家通过键盘操作小游戏，程序使用EFI_TEXT_INPUT_PROTOCOL处理用户按下的单个按键，涉及的扫描码有六个：

```
#define SCAN_UP          0x0001
#define SCAN_DOWN    0x0002
#define SCAN_RIGHT     0x0003
#define SCAN_LEFT       0x0004
#define SCAN_F1          0x000B
#define SCAN_ESC        0x0017
```

1.3 贪吃蛇算法

常见的贪吃蛇小游戏一般通过链表来实现，即创建蛇身结点，蛇前进一步，则头指针指向新结点，尾指针指向前一个结点。我们起初也采用了这种方法：在AppPkg中创建贪吃蛇小程序，在Library class中调用C标准库中的stdlib.h（这样就可以用malloc为新结点分配空间），最后在AppPkg的环境中编译。

但是在编译的过程中，我们遇到了尚未解决的报错：unsolved external symbol _fpclassifyd，影响了对C标准库的调用，由于目前在网络上，和UEFI编程相关的参考资料有限，我们确实至今没有解决遇到的这一问题，所以决定改变算法，将Main类型应用改为更保守的ShellApp类型。

新的贪吃蛇算法思路如下：

用一个二维数组表示地图，从蛇尾到蛇头用数字1至snakebody表示（snakebody为蛇的长度，如果蛇吃到了食物，snakebody加一）。

除去无效移动和撞墙等情况，蛇的正常移动情形分为两种：吃到了食物——头的位置变为新点（加头）。没有吃到食物——头的位置变为新点，同时尾部前移（加头去尾）。

现在假设当前蛇长为3，蛇尾到蛇头的数字为1，2，3，除食物、围墙外的其他区域为0。经过判断，蛇头的下一个位置为（5，5）。

第一种情形的实现：首先，让snakebody加一，即4。第二步，将BARS【5】【5】赋值snakebody，即可得到一条数字为1，2，3，4的增长的蛇。

第二种情形的实现：首先，用遍历算法让大于0的数字减一，这时蛇为变成0，自然成了空地。第二步同上，将BARS【5】【5】赋值snakebody，即可得到一条数字为1，2，3的长度不变的蛇。

2、核心代码展示

2.1 处理键盘事件

程序会根据游戏当前的状态，对键盘事件进行处理。

```
VOID updateKeys(VOID){ //根据键盘移动光标或蛇位置
  EFI_STATUS Status;
  EFI_INPUT_KEY Key;
  Status = gST->ConIn->ReadKeyStroke(gST->ConIn, &Key);
  switch (sys_gs)
  {
     case WELCOME:
        if (Key.ScanCode == SCAN_ESC){
            exitGame();
        }//如果ESC，退出游戏
        else if(Key.ScanCode == SCAN_F1){
           startMenu(1);
        }//如果按键向上，状态改为Menu,isfresh仍然是1。
        break;
     case MENU:
        if (Key.ScanCode == SCAN_UP)
        {
           MenuIndex --;
           if(MenuIndex <= 0)
              MenuIndex = MainMenuNum;
           isRefresh = 1;
        }
        else if(Key.ScanCode == SCAN_DOWN){
          MenuIndex ++;
          if(MenuIndex > MainMenuNum)
              MenuIndex = 1;
          isRefresh = 1;
        }
```

```
        else if(Key.ScanCode == SCAN_F1){
            switch(MenuIndex){
             case 1:   //Start
               startGame();
               break;
             case 2:   //Edit
               //viewRecord ();//查看最高纪录
               break;
             case 3:
               resetRecord();//重置记录
               break;
             case 4:   //Exit
               exitGame();
               break;
            }
        }
        break;
    case GAMING:
        if (Key.ScanCode == SCAN_ESC){
               startMenu(1);
        }
        else if(Key.ScanCode == SCAN_F1){
               sys_gs = PAUSE;
               isRefresh = 1;
        }
        else if (Key.ScanCode == SCAN_UP)
           {
               UINT32 result = moveSnake(UP);
               if(result == 2){
                  winGame();
                  return ;
               }
               else {
                  if(result != 3) totalSteps ++;
```

```
                isRefresh = 1;
            }
        }
    else if (Key.ScanCode == SCAN_DOWN)
        {
            UINT32 result = moveSnake(DOWN);
            if(result == 2){
                winGame();
                return ;
            }
            else {
                if(result != 3) totalSteps ++;
                isRefresh = 1;
            }
        }
    else if (Key.ScanCode == SCAN_LEFT)
        {
            UINT32 result = moveSnake(LEFT);
            if(result == 2){
                winGame();
                return ;
            }
            else {
                if(result != 3) totalSteps ++;
                isRefresh = 1;
            }
        }
     else if (Key.ScanCode == SCAN_RIGHT)
        {
            UINT32 result = moveSnake(RIGHT);
            if(result == 2){
                winGame();
                return ;
            }
```

```
                else {
                    if(result != 3) totalSteps ++;
                    isRefresh = 1;
                }
            }
            break;
        case PAUSE:
            if(Key.ScanCode == SCAN_F1 || Key.ScanCode == SCAN_ESC){   //Resume
                mySetCursorPos(0, ROW);
                gST->ConOut->OutputString(gST->ConOut,L"\t\t\t\t\t\t\t");
                sys_gs = GAMING;
                isRefresh = 1;
            }
        case STOP:
            if(Key.ScanCode == SCAN_F1){ //reStart
                startGame();
            }
            else if(Key.ScanCode == SCAN_ESC){
                startMenu(1);
            }
            break;
        default:
            break;
    }
}
```

2.2 食物生成

贪吃蛇食物的生成采用了随机数算法：如果上一个食物被吃掉，则分别在横、纵坐标的范围内生成随机数，再判断食物是否落在空地上，若没有，则重新生成食物。

```
VOID createFood(VOID){
    UINT32 i=0,j=0;
    i=JR_randomIn(ROW);
    j=JR_randomIn(COL);
    if(BARS[i][j]) createFood();
```

```
    else BARS[i][j]=580;
}
#define JR_RANDOM_NUM   100
//static UINT32 * RandomPool;
static UINT32 RandomPool[JR_RANDOM_NUM];
static UINT32 JR_index = JR_RANDOM_NUM;
static UINT32 RandomResult = 0;
//生成随机数
static VOID JR_InitRandom(){
    UINT32 i = 0;
    EFI_TIME  Time;
    if(JR_index != JR_RANDOM_NUM) return ;
    JR_index = 0;
    for(i=0; i<JR_RANDOM_NUM; i++){
        if(RandomResult ==0)
        {
            gRT->GetTime(&Time, NULL);
            RandomResult  = Time.Second;
        }
        RandomResult = (RandomResult<<1) |
(((RandomResult&0x80)>>7)^((RandomResult&0x40)>>6));
        RandomPool[i] = RandomResult;
    }
}
UINT32 JR_randomIn(UINT32 max){
    JR_InitRandom();
    return (RandomPool[JR_index++] % (max));
}//取一个不超过max的随机数
```

2.3 蛇身移动

通过给二位数组赋值一串连续数字来标识蛇的位置，在上文中已经进行了描述。

```
UINT32 moveSnake(UINT32 dir){
    UINT32 result = 0;
    SNAKE node;
```

```
node.x = head.x;
node.y = head.y;
if(dir == Direc) return 3;//移动不成功
else if (dir == 0){
    node.x --;
    Direc = 3;//UP
}
else if (dir == 1){
    node.y --;
    Direc = 2;//LEFT
}
else if (dir == 2){
    node.y ++;
    Direc = 1;//RIGHT
}
else if (dir == 3){
    node.x ++;
    Direc = 0;//DOWN
}
//p为移动指针，p指向头结点
result = checkSnake(node);//0是继续走。1是变长，2是撞墙或者撞自己身子。
if(result == 1){//蛇变长
    snakebody++;
    createFood();//吃掉旧食物，创造新食物
}
else{//遍历
    UINT32 i,j,k;
    for(i=1; i<ROW-1; i++){
        for(j=1; j<COL-1; j++){
            k=BARS[i][j];
            if((k<=snakebody)&&(k>0)) BARS[i][j]--;
            if(BARS[i][j]==1) {
                tail.x=i;
```

```
                tail.y=j;
            }
        }
    }
  }
  BARS[node.x][node.y]=snakebody;
  head=node;//换个头就行
  drawBars();
  return result;
}
```

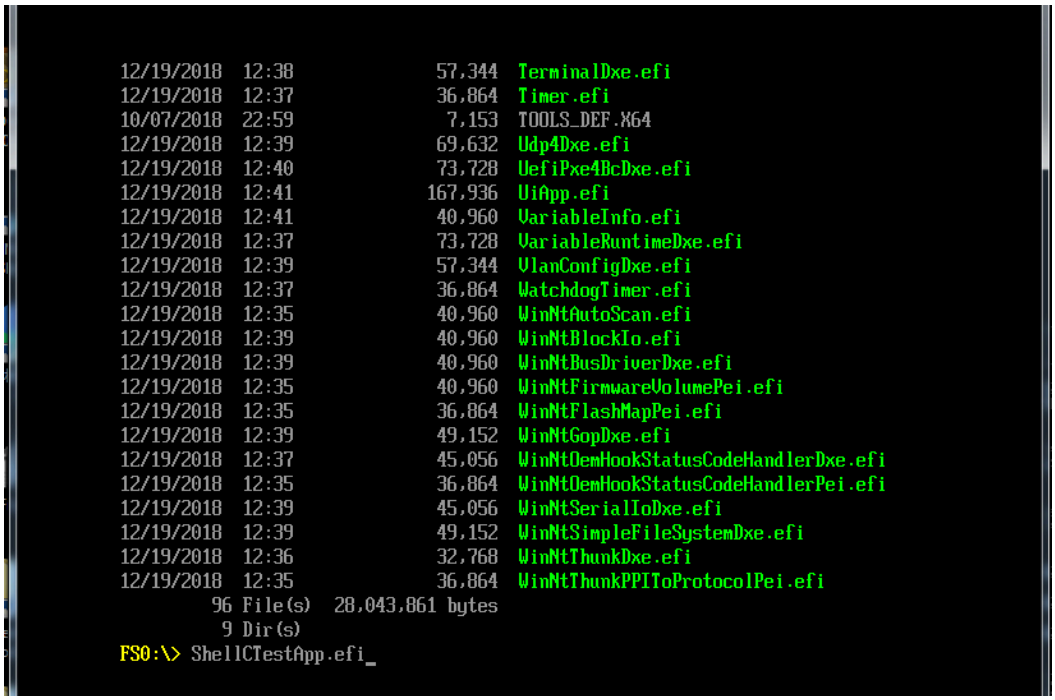
2.4 显示界面

将所有需要显示对象的坐标和内容存在同一个结构体中，再把结构体放入结构体数组中，最后统一显示。

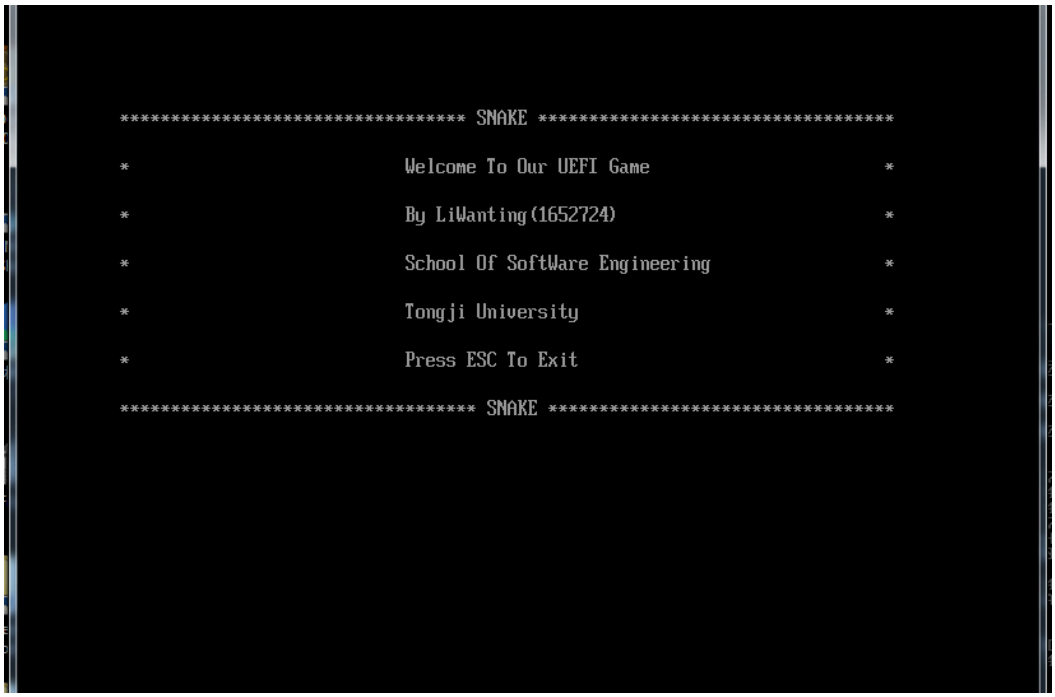
```
    VOID showInBuff(SHOW buffer){
  showBuff[BuffIndex].x = buffer.x;
  showBuff[BuffIndex].y = buffer.y;
  myStrcpy(showBuff[BuffIndex].text, buffer.text);
  BuffIndex ++;
}2.5 取得系统时间
VOID showGame(VOID){
  UINT32 i = 0;
  for(i=0; i<BuffIndex; i++){
    mySetCursorPos(showBuff[i].x, showBuff[i].y);
    if ((showBuff[i].text)[0] == '1')//墙壁
    {
        gST->ConOut->SetAttribute(gST->ConOut, EFI_TEXT_ATTR(EFI_LIGHTGRAY,
EFI_YELLOW));
    }
    if ((showBuff[i].text)[0] == ' ')//空格
    {
      gST->ConOut->SetAttribute(gST->ConOut, EFI_TEXT_ATTR(EFI_LIGHTGRAY,
EFI_BLACK));
    }
```

```
        if ((showBuff[i].text)[0] == '*')//蛇身
        {
            gST->ConOut->SetAttribute(gST->ConOut, EFI_TEXT_ATTR(EFI_LIGHTGRAY,
EFI_RED));
        }
        if ((showBuff[i].text)[0] == 'O')//蛇头
        {
            gST->ConOut->SetAttribute(gST->ConOut, EFI_TEXT_ATTR(EFI_LIGHTGRAY,
EFI_RED));
        }
        if ((showBuff[i].text)[0] == '#')//食物
        {
            gST->ConOut->SetAttribute(gST->ConOut, EFI_TEXT_ATTR(EFI_LIGHTGRAY,
EFI_BLUE));
        }
        gST->ConOut->OutputString(gST->ConOut,showBuff[i].text);
    }
    BuffIndex = 0;
    if(sys_gs == WIN){
        sys_gs = STOP;
        isRefresh = 1;
    }
}
```

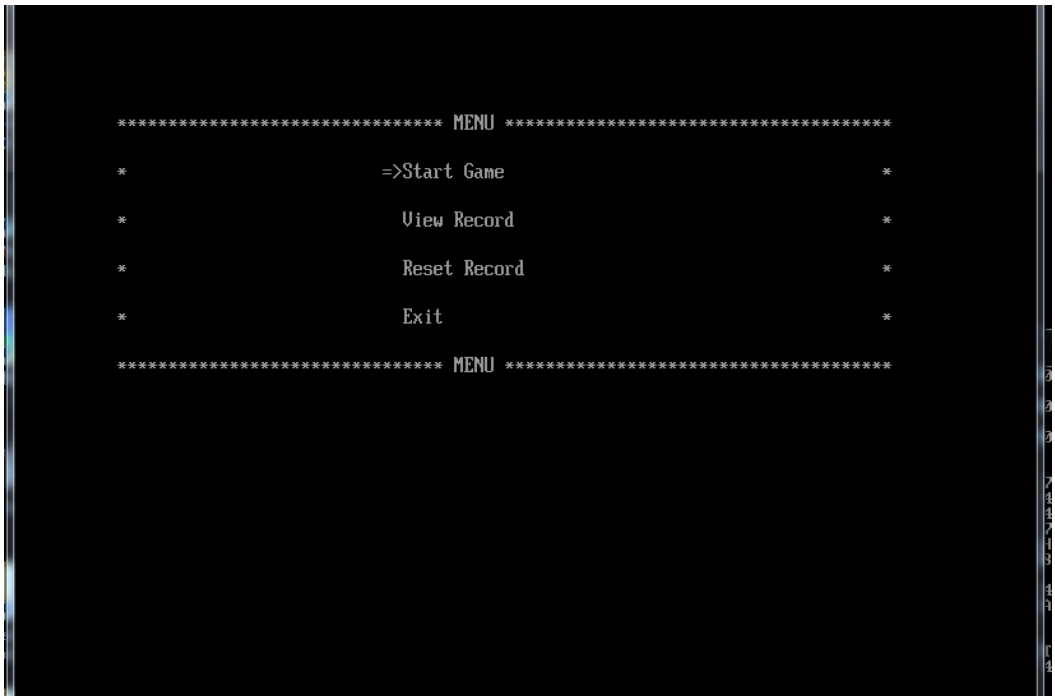
3、程序界面截图



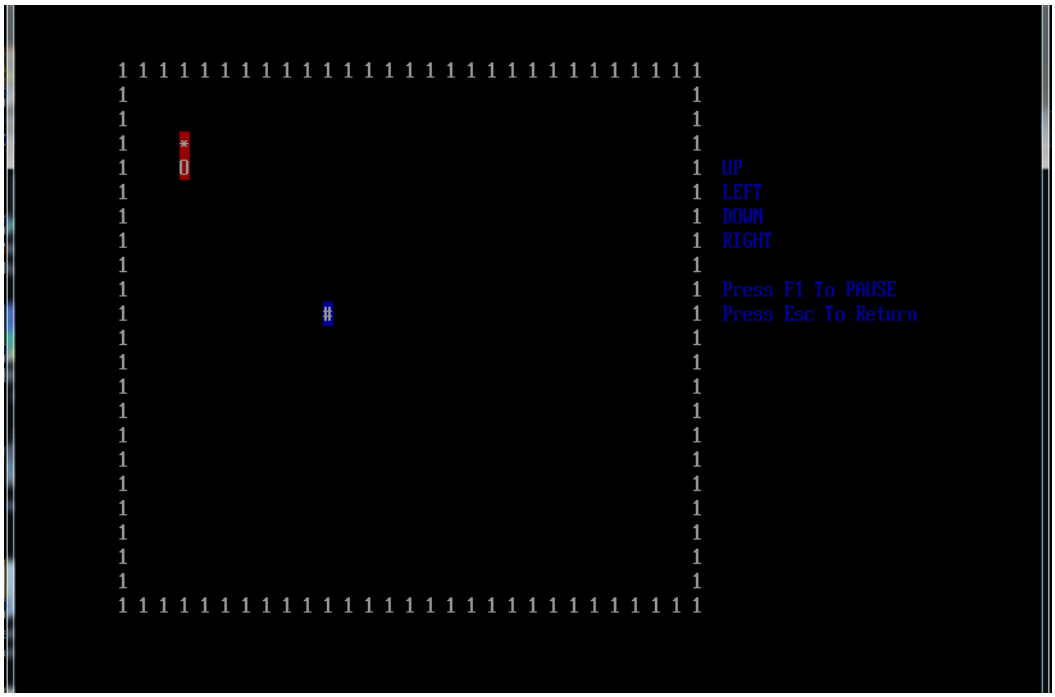
运行过程



欢迎界面


******************************** MENU **********************************
*   =>Start Game   *
*   View Record   *
*   Reset Record   *
*   Exit   *
******************************** MENU **********************************

菜单界面


UP
LEFT
DOWN
RIGHT
Press F1 To PAUSE
Press Esc To Return

游戏界面

4、项目开发心得

1、关于UEFI开发的小技巧

1、通过编写.bat批处理脚本，可以避免重复输入命令。

以下是我的.bat内容：

@echo off

build -p nt32pkg\nt32pkg.dsc -a X64 -t VS2017 -m ShellPkg\Application\ShellCTestApp\ShellCTestApp.inf

2、VS具有自动优化代码的功能，我们是在给二位数组赋值时，意外遇见了unresolved external symbol _memset 的报错，经过查阅资料发现，原因可能是编译器对代码进行了自动优化，把可以用memset代替的代码进行了处理。

阻止编译器自动优化的方法是，对编译参数进行调整，具体操作为在inf文件的Build Option中增加代码：

MSFT:*_*_IA32_CC_FLAGS = /GL - /Od

MSFT:*_*_X64_CC_FLAGS = /GL - /Od

程序编译通过，关于VS编译参数的介绍可以参考微软官网：

https://docs.microsoft.com/en-us/previous-versions/windows/embedded/aa448731(v%3dmsdn.10)

2、反思不足之处

1、由于没有采用链表算法， 而是选择了用二位数组实现蛇身移动，程序的执行速度比较慢。如果能够解决调用C标准库时所遇到的问题，小游戏的功能会更丰富和完善。

2、界面的设计欠缺人性化。

3、代码有些复杂，函数之间独立度不佳，应当进一步精简和优化。