利用 OLED 显示的贪吃蛇游戏

一、各个元件的控制

1. OLED

(1) IIC 通讯：OLED 使用的是 IIC 通讯；

```
void TWI_Init(void) //twi 接口的初始化
{ //设置 SCL 的频率：1MHz cpu-50KHz scl,2M-100K,8M-400K
    TWSR = 0x00; //最低 2 位为预分频设置(00-1,01-4,10-16,11-64)
    TWBR = 0x02; //位率设置，fscl=cpu 频率/(16+2*TWBR*预分频值)
    TWCR = (1<<TWEN); //开启 TWI
}
void TWI_Start(void) //发送 Start 信号，开始本次 TWI 通信
{ TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);//发送 Start 信号
    while(!(TWCR &(1<<TWINT))); //等待 Start 信号发出
}
void TWI_Stop(void) //发送 Stop 信号，结束本次 TWI 通信
{   TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);//发送 Stop 信号
}
void TWI_Write(unsigned char uc_data) //向 TWI 接口发送 8 位数据
{   TWDR = uc_data; //8 位数据存放在 TWDR
    TWCR = (1<<TWINT)|(1<<TWEN); //发送 TWDR 中的数据
    while(!(TWCR &(1<<TWINT))); //等待数据发出}
unsigned char TWI_Read_With_ACK(void)
{   TWCR = (1<<TWINT)|(1<<TWEA)|(1<<TWEN);//准备接收数据，并 ACK
    while(!(TWCR &(1<<TWINT))); //等待接收数据
    return TWDR; //返回接收到的数据
}
unsigned char TWI_Read_With_NACK(void)
{   TWCR = (1<<TWINT)|(1<<TWEN); //准备接收数据，并 NACK
    while(!(TWCR &(1<<TWINT))); //等待接收数据
    return TWDR; //返回接收到的数据}
unsigned char TWI_Get_State_Info(void)
{   unsigned char uc_status;
    uc_status = TWSR & 0xf8; //取状态寄存器高 5 位，低 3 位为 0
    return uc_status; //返回状态寄存器高 5 位
}
```

函数解释：

1.TWI_Init(void)函数：初始化 TWI 接口，设置 SCL 的频率并开启 TWI。

2.TWI_Start(void)函数：发送 Start 信号，开始一次 TWI 通信。

3.TWI_Stop(void)函数：发送 Stop 信号，结束一次 TWI 通信。

4.TWI_Write(unsigned char uc_data)函数：向 TWI 接口发送 8 位数据。

5.TWI_Read_With_ACK(void)函数：从 TWI 接口读取数据，并发送 ACK 信号。

6.TWI_Read_With_NACK(void)函数：从 TWI 接口读取数据，并发送 NACK 信号。

7.TWI_Get_State_Info(void)函数：获取 TWI 状态信息。

(2).OLED 的通讯和初始化

```
// OLED 写入函数，根据 cmd 参数判断发送命令还是数据
void OLED_WR_Byte(unsigned char dat, unsigned char cmd) {
    TWI_Start(); // 发送 Start 信号
    TWI_Write(OLED_ADDRESS); // 写入 OLED 地址，写入模式
    if (cmd == 0x00) {
        TWI_Write(0x00); // 数据模式
        } else {
        TWI_Write(0x40); // 命令模式
    }
     TWI_Write(dat); // 发送数据或命令
     TWI_Stop(); // 发送 Stop 信号
}
void OLED_Init(void)
{
    OLED_WR_Byte(0xAE,OLED_CMD);//--turn off oled panel
    OLED_WR_Byte(0x02,OLED_CMD);//---set low column address
    OLED_WR_Byte(0x10,OLED_CMD);//---set high column address
    OLED_WR_Byte(0x40,OLED_CMD);//--set start line address
    OLED_WR_Byte(0x81,OLED_CMD);//--set contrast control register
    OLED_WR_Byte(0xCF,OLED_CMD); // Set SEG Output Current Brightness
    OLED_WR_Byte(0xA1,OLED_CMD);//--Set SEG/Column Mapping     0xa0 左右反置 0xa1 正常
    OLED_WR_Byte(0xC8,OLED_CMD);//Set COM/Row Scan Direction   0xc0 上下反置 0xc8 正常
    OLED_WR_Byte(0xA6,OLED_CMD);//--set normal display
    OLED_WR_Byte(0xA8,OLED_CMD);//--set multiplex ratio(1 to 64)
    OLED_WR_Byte(0x3f,OLED_CMD);//--1/64 duty
    OLED_WR_Byte(0xD3,OLED_CMD);//-set display offset   Shift Mapping RAM Counter (0x00~0x3F)
```

```
    OLED_WR_Byte(0x00,OLED_CMD);//-not offset
    OLED_WR_Byte(0xd5,OLED_CMD);//--set display clock divide ratio/oscillator frequency
    OLED_WR_Byte(0x80,OLED_CMD);//--set divide ratio, Set Clock as 100 Frames/Sec
    OLED_WR_Byte(0xD9,OLED_CMD);//--set pre-charge period
    OLED_WR_Byte(0xF1,OLED_CMD);//Set Pre-Charge as 15 Clocks & Discharge as 1 Clock
    OLED_WR_Byte(0xDA,OLED_CMD);//--set com pins hardware configuration
    OLED_WR_Byte(0x12,OLED_CMD);
    OLED_WR_Byte(0xDB,OLED_CMD);//--set vcomh
    OLED_WR_Byte(0x40,OLED_CMD);//Set VCOM Deselect Level
    OLED_WR_Byte(0x20,OLED_CMD);//-Set Page Addressing Mode (0x00/0x01/0x02)
    OLED_WR_Byte(0x02,OLED_CMD);//
    OLED_WR_Byte(0x8D,OLED_CMD);//--set Charge Pump enable/disable
    OLED_WR_Byte(0x14,OLED_CMD);//--set(0x10) disable
    OLED_WR_Byte(0xA4,OLED_CMD);// Disable Entire Display On (0xa4/0xa5)
    OLED_WR_Byte(0xA6,OLED_CMD);// Disable Inverse Display On (0xa6/a7)
    OLED_WR_Byte(0xAF,OLED_CMD);//--turn on oled panel
    OLED_Set_Pos(0,0);
}
```

(3).OLED 寻址函数

在具体编写 OLED 的控制函数之前，简要了解 OLED 的寻址原理是必要的。我用下面张图简单介绍一下：

		第0列	第1列	第2列	第3列	...	第124列	第125列	第126列	第127列
第0页	第0行									
	第1行									
	第2行									
	第3行									
	第4行									
	第5行									
	第6行									
	第7行									
第1页	第8行									
	...									
...	...									
第7页	...									

OLED 的分辨率为 128*64，其中一共有 128 列，行的部分分为 8 页，每一页里面有 8 行。因此可以写出源代码：

```
void OLED_Set_Pos(unsigned char x, unsigned char y) {
    OLED_WR_Byte(0xb0 + (y & 0x07), OLED_CMD); // 设置页地址(0-7)
    OLED_WR_Byte(((x & 0xf0) >> 4) | 0x10, OLED_CMD); // 设置列地址的高 4 位
    OLED_WR_Byte((x & 0x0f), OLED_CMD); // 设置列地址的低 4 位
}
```

(4) .OLED 像素点的控制

清屏：

```
void OLED_Clear(void)
{
    unsigned char i,n;
    for(i=0;i<8;i++)
    {
        OLED_WR_Byte (0xb0+i,OLED_CMD);    //设置页地址（0~7）
        OLED_WR_Byte (0x02,OLED_CMD);      //设置显示位置—列低地址
        OLED_WR_Byte (0x10,OLED_CMD);      //设置显示位置—列高地址
        for(n=0;n<128;n++)OLED_WR_Byte(0,OLED_DATA);
    } //更新显示
}
```

在指定位置上画点：

前面已经讲过 OLED 的亮灭控制原理，如果直接 OLED_Set_Pos()定位到某个位置然后点亮可能会出现一个问题：如果想画出同一数列上的点，（同一页上）后画的点会覆盖掉前面画的。在开始时由于这个问题浪费的大量的时间。最后的解决方案：写一个 OLED 的状态数组，每次读写操作修改 OLED 相应位置上的状态，最后利用函数统一打印。

但是这个方案随之而来的就是空间上的紧张。如果想开一个 128*8 的状态数组至少需要 1024B 的存储空间。然而后来查阅手册发现，ATmega8A 的 SRAM 也只有 1024B。一开始没

有意识到这个问题的时候，写完的程序 Data Memory Usage 到达了 2402.1%,竟然超出了内存 24 倍！

最后被逼无奈，将显示区域限制在 128*6（中间 6 页）的区间上，并对贪吃蛇的程序进行了大幅改进，才保证了总内存在 1024B 以内。

```
void OLED_DRAW(unsigned char x,unsigned char y,unsigned char GRAM[128][6]){//x:0-127;y:8-56
    unsigned char n,m;
    n=y/8+1;
    m=y%8;
    GRAM[x][n]|=0x01<<m;
}
void OLED_DEL(unsigned char x,unsigned char y,unsigned char GRAM[128][6]){
    unsigned char n,m;
    n=y/8+1;
     m=y%8;
     GRAM[x][n] &= ~(0x01<<m);
}
void OLED_DRAW_BIG(unsigned char big_x,unsigned char big_y,unsigned char GRAM[128][6]){
    for(unsigned char i=0;i<2;i++)
        for(unsigned char j=0;j<2;j++)
            OLED_DRAW(big_x*2+i,big_y*2+j,GRAM);
}
void OLED_DEL_BIG(unsigned char big_x,unsigned char big_y,unsigned char GRAM[128][6]){
    for(unsigned char i=0;i<2;i++)
        for(unsigned char j=0;j<2;j++)
            OLED_DEL(big_x*2+i,big_y*2+j,GRAM);
}
void OLED_Refresh(unsigned char GRAM[128][6]){
    unsigned char i,n;
    for(i=1;i<7;i++){
        OLED_WR_Byte(0xb0+i,OLED_CMD); //设置行起始地址
        OLED_WR_Byte(0x00,OLED_CMD);   //设置低列起始地址
        OLED_WR_Byte(0x10,OLED_CMD);   //设置高列起始地址
        for(n=0;n<128;n++)
        OLED_WR_Byte(GRAM[n][i],OLED_DATA);}
}
```

这个方法通过“设置状态”的方式解决了前面提到的问题。在状态数组中画完点之后，用 OLED_Refresh()更新画的状态。

在完成初版代码后发现只用一个像素点实在太小看不清，所以后面又写了 OLED_DRAW_BIG 函数，将 4 个小像素的合成一个大点，让游戏体验大大提升。

2. 七位数码管

七位数码管用以显示分数

```
void dp_show(unsigned char grade)  {
unsigned char seg7_hex[16]={0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,0xfe,0xf6,0xee,0x3e,0x9c,0x7a,0x9e,0x8e};

    DDRB=(0xff);
    DDRC=(0x01);
    PORTC=0x00; //禁止显示
    PORTB=seg7_hex[grade];
}
```

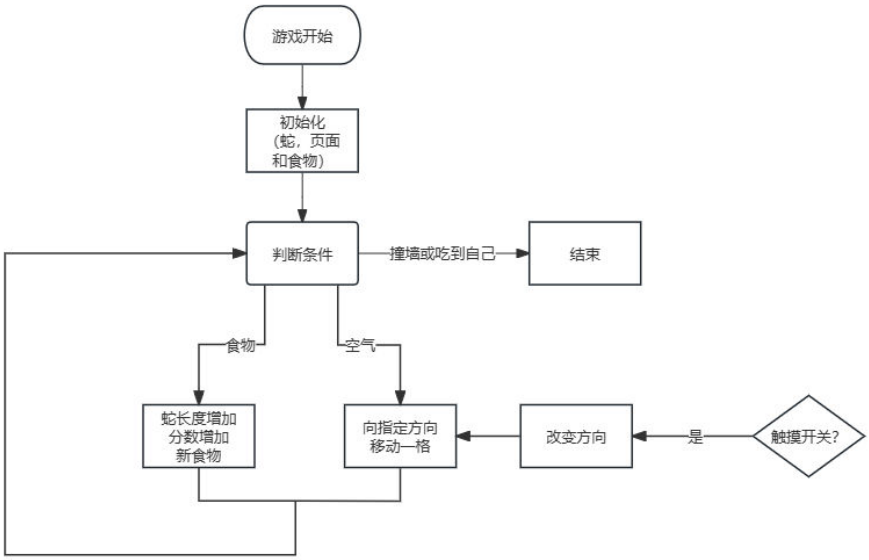
3. 触摸开关

利用中断统计按键的次数，并根据次数控制蛇的转向。

```
//main 函数中
    DDRD &= ~(1<<DDRD1);
    MCUCR |=((1<<ISC11)|(1<<ISC10));
    GICR |= (1<<INT1);
    sei();
//中断处理
ISR(INT1_vect){
    if(count < 3) count++;
    else count = 0;}
```

二、贪吃蛇游戏的控制

游戏控制的流程图如下：



1. 蛇的身体结构

```
struct Body {
    unsigned char x;
    unsigned char y;
}body[10];// 定义蛇身体节点结构体
struct Snake {
    unsigned char x;  // 蛇头的横坐标
    unsigned char y;  // 蛇头的纵坐标
    unsigned char length; // 蛇的长度
};// 定义蛇结构体
```

2. 初始化，随机食物的生成

```
void InitInterface(){
    for(unsigned char i=0;i<24;i++){
        OLED_DRAW_BIG(1,i,GRAM);
        OLED_DRAW_BIG(63,i,GRAM);}
    for(unsigned char j=1;j<63;j++){
        OLED_DRAW_BIG(j,0,GRAM);
        OLED_DRAW_BIG(j,23,GRAM);}}
void InitSnake(){
    snake.length = 3;
     snake.x = 32;
     snake.y = 12;
     body[0].x = 31;
     body[0].y =12;
     body[1].x =30;
     body[1].y =12;}
unsigned char ranNum(unsigned char x,unsigned char y){
     return rand()%(y-x)+x;}
void RandFood(){
     unsigned char i, j;
     do
     {
         i = ranNum(1,62);
         j = ranNum(1,22);
     } while (judge(i,j)!=AIR);
     foodx=i;
     foody=j;
     OLED_DRAW_BIG(foodx,foody,GRAM);
     OLED_Refresh(GRAM);}
```

3. 打印蛇与覆盖蛇

```
//打印蛇与覆盖蛇
void DrawSnake(unsigned char flag){
    if (flag == 1) //打印蛇
    {
        OLED_DRAW_BIG(snake.x,snake.y,GRAM);
        for (unsigned char i = 0; i < snake.length; i++)
            OLED_DRAW_BIG(body[i].x, body[i].y,GRAM);
        OLED_Refresh(GRAM);}
    else //覆盖蛇
     {
         OLED_DEL_BIG(snake.x,snake.y,GRAM);
         for (unsigned char i = 0; i < snake.length; i++)
             OLED_DEL_BIG(body[i].x, body[i].y,GRAM);
         OLED_Refresh(GRAM);}
}
```

4. 移动蛇

```
void run(unsigned char x,unsigned char y,unsigned char n){
    while (1){
        dp_show(grade);
        JudgeFunc(x, y);
        MoveSnake(x, y);
        if (n!=count) //若触发开关，回到 main 函数
        {break;}}
}
```

5. 结束条件

```
unsigned char judge(unsigned char x,unsigned char y){
    if(x==0||x==63||y==0||y==23) return WALL;
    for(unsigned char i=0;i<snake.length;i++)
        if(body[i].x==x&&body[i].y==y) return BODY;
    return AIR;}
void JudgeFunc(unsigned char x, unsigned char y){//判断得分与结束
    if (snake.y==foody && snake.x==foodx)//食物
    {
        RandFood();
         snake.length++;
         grade += 1;
         OLED_Refresh(GRAM);}
     else if (judge(snake.x + x,snake.y + y) == WALL || judge(snake.x + x,snake.y + y) ==  BODY || snake.length>10)   //墙或者蛇身
     {
         _delay_ms(2000);
         OLED_Clear();//game over!
         exit(0);}
}
```

6. 游戏运行

```
void run(unsigned char x,unsigned char y,unsigned char n){//运行游戏
    while (1){
        dp_show(grade);
        JudgeFunc(x, y);
        MoveSnake(x, y);
        if (n!=count) break;//若触发开关，回到 main 函数
    }
}
int main(void) {
    TWI_Init();
    OLED_Init();// 初始化 OLED
    OLED_Clear();
    DDRD &= ~(1<<DDRD1);
    MCUCR |=((1<<ISC11)|(1<<ISC10));
    GICR |= (1<<INT1);
    sei();//开中断
    InitInterface();
    InitSnake();
    srand((unsigned)time(NULL));
    RandFood();
    DrawSnake(1); //打印蛇
    while(1) {
        OLED_Refresh(GRAM);
        unsigned char n=count;
        switch(count){
            case UP:
                run(0, -1, n); //向上移动（横坐标偏移为 0，纵坐标偏移为-1）
                break;
            case DOWN:
                run(0, 1, n);
                break;
            case LEFT:
                run(-1, 0, n);
                break;
            case RIGHT:
                run(1, 0, n);
                break;
        }
        n=count;
    }
    return 0;
}
```

三、总结

1.总结

本项目实现了一个基于 OLED 显示屏的经典贪吃蛇游戏。OLED 通过 IIC 与主控芯片通信，负责显示蛇身、食物和墙。游戏在初始化的时候设定了蛇的长度和位置，随机生成食物，并在循环中控制蛇的移动。每次移动之后，可以根据相应条件判断蛇的得分、死亡或继续移动。项目外接了一个触摸开关，并利用中断来响应玩家操作，以改变蛇的移动方向；并且利用其

位数码管显示分数。在 OLED 的控制代码中，利用状态数组的方法良好的解决了直接绘图可能造成的前图覆盖问题。

2.未来可能的改进空间

虽然状态数组可以保证我们准确且不受限制的控制点的位置，但是不得不承认的是这一方法消耗了大量的空间且降低了打印程序的效率，导致了贪吃蛇“前进”缓慢。个人认为这个问题可以通过修改状态数组的数据结构来改变：装态数组本质上是利用二维数组存储了一个（较为稀疏的）图结构，导致效率低下。如果采用更为恰当的数据结构存储这个稀疏图结构，也许可以大幅的提升算法的效率。