# 嵌入式系统

# 实验报告



|  |  |
| --- | --- |
| 实验名称： | 贪吃蛇交互游戏 |
| 姓 名： |  |
| 学 号： |  |
| 学 院(系)： |  |
| 专 业： |  |
| 指导教师： |  |

2023年12月23日

# 实验目的

结合 STM32的库，描述如何设置 STM32 的 GPIO 口作为输入使用。通过 FS-STM32F407 载有 4 个按键（KEY3、KEY4、KEY5、KEY6）来进行用户交互。

熟悉并了解TFT彩屏开发流程，理解彩屏显示图片或字符的原理。

# 实验环境

1. FS-STM32F407开发平台
2. ST-Link仿真器
3. RealView MDK5.23集成开发环境
4. STM32CUBEMX图形开发软件
5. PC机XP，Windows7/8/10(32/64bit)

# 实验要求

1. 功能一：按下RESET后可以实现主菜单页的加载，主菜单页中显示图片和文字，有供用户选择的模式，还可以选择加载游戏规则页。
2. 功能二：在主菜单页按下K6选择简单模式，再按一次开始游戏。
3. 功能三：在主菜单页按下K5选择困难模式，再按一次开始游戏。
4. 功能四：在主菜单页按下K4选择地狱模式，再按一次开始游戏。
5. 功能五：在主菜单页按下K3选择游戏说明，再按一次打开规则说明界面。
6. 功能六：在规则说明界面有规则的详细说明文字。按下K3关闭规则说明界面并返回主菜单。
7. 功能七：开始游戏后蛇从屏幕左上角向屏幕右边移动，屏幕上有随机生成的食物可以被蛇吃到。蛇吃到食物后食物消失，蛇的身体变长，屏幕上出现新生成的食物。
8. 功能八：开始游戏后按下K6可以控制蛇向左移动。
9. 功能九：开始游戏后按下K5可以控制蛇向下移动。
10. 功能十：开始游戏后按下K4可以控制蛇向上移动。
11. 功能十一：开始游戏后按下K3可以控制蛇向左移动。
12. 功能十二：开始游戏后，蛇头碰到边框就会死，蛇咬到自身也会死。
13. 功能十三：蛇死亡后，游戏显示结束提示，输出玩家得到的分数，按K6可以退出，按K5可以重新开始。

# 实验原理

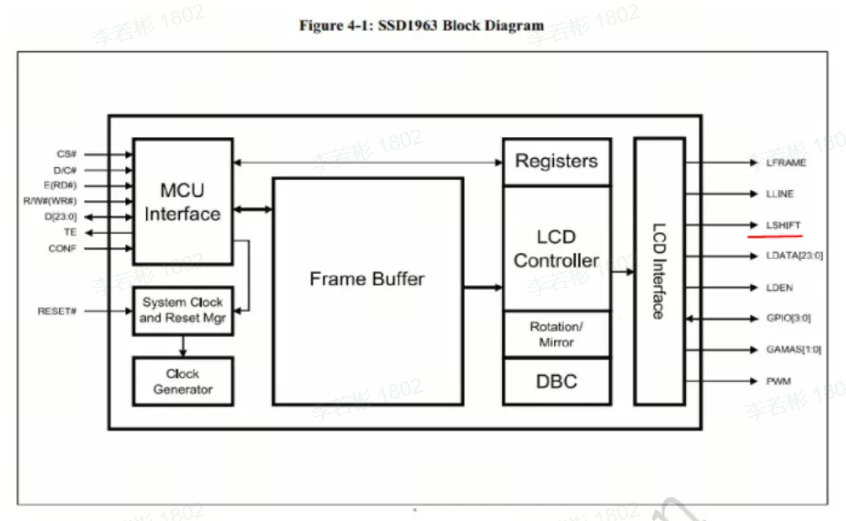
1. 按键控制原理

FS-STM32F407 开发板总共有 4 个功能按键，在不使用第二功能的情况下，这四个按键都可以作为通用的按键，有用户自定义其功能。这四个按键分别与 PI9、PF11、PC13、和 PA0 四个管脚连接，当按键按下时，对应的 GPIO 管脚为低电平，反之，当没有按键按下时，对应的GPIO 管脚为高电平。本实验中所有的按键均作为普通 IO 使用。

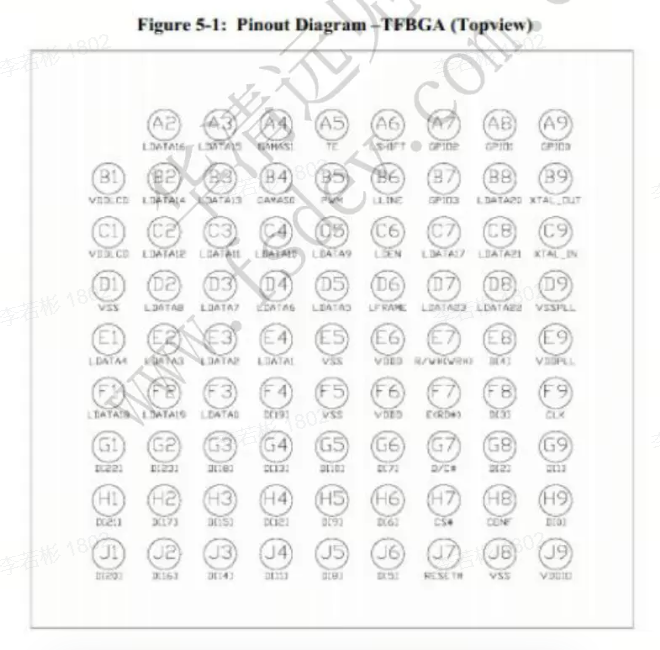
1. 彩屏显示原理

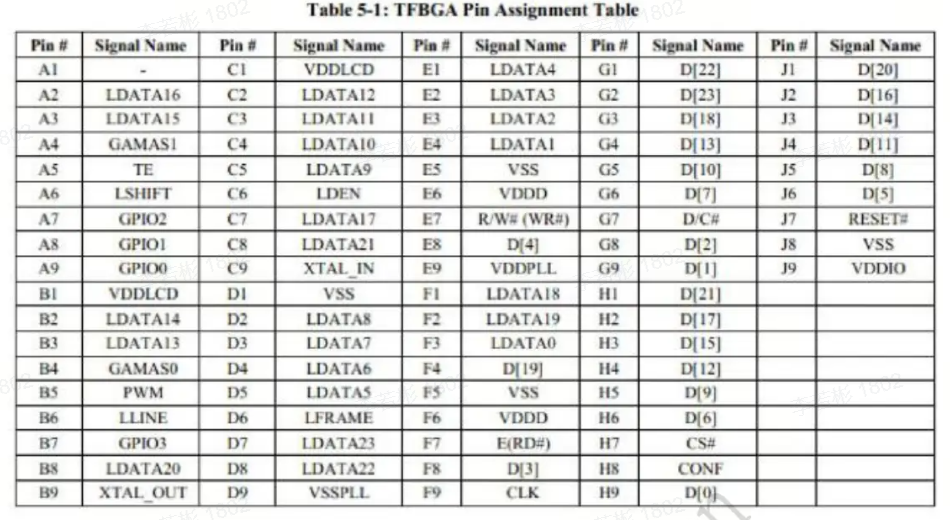
彩屏使用到的显示控制器是 ssd1963 支持 864 x 480 x 24 位图形内容。它也配有不同宽度并行接口总线来接收图形数据和命令从单片机。它的显示界面，支持常见的内存更少的 LCD 驱动器, 每—像素的颜色深度可达 24 比特。

结构图为：



引脚排列：





本实验 TFT 液晶使用的是 4.3 寸的 480\*282 分辨率（像素)，16 位真彩显示（接近自然色)共用 16 位的数据线，所以最低 5 位代表蓝色，中间 6 位为绿色，最高 5 位为红色。数值越大，表示该颜色越深。

1. 实现绘制API时使用到的算法

Bresenham算法是一种用于在计算机屏幕或图像上绘制直线的算法。它的主要目的是通过尽可能高效地确定线条上的像素点来减少计算工作。

Bresenham算法最初设计用于绘制直线，但后来扩展到绘制其他形状，如圆、椭圆等。以下是解释Bresenham算法的基本思想：

离散化直线：计算机屏幕是由离散的像素组成的，而直线是连续的。该算法将直线划分为若干个离散的像素，以在屏幕上表示一条直线。

决策变量：该算法维护决策变量，该变量帮助决定应该在直线上的哪个像素点。该决策在每一步的迭代中更新。

整数运算：该算法使用整数运算而不是浮点数，来提高计算效率。这是通过避免使用浮点数乘法和除法来实现的。

八个对称性：由于直线在平面上有八个对称方向（上下左右和四个对角线），Bresenham算法只需计算其中一个方向上的像素，然后通过对称性获得其他方向上的像素。

增量算法：Breseham算法是增量算法，每次只需要增加或减少一个像素，而不是对整个直线进行完整计算。

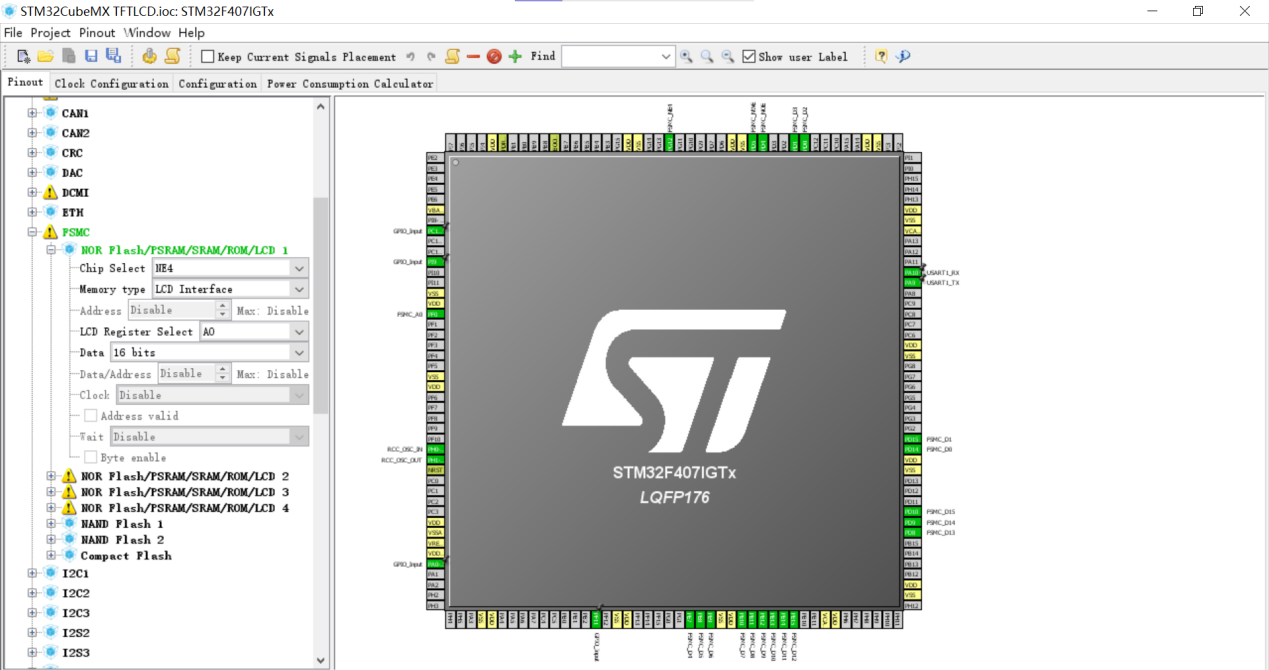
递推公式：该算法使用递推公式来计算决策变量的变化，并根据该变化来决定下一个像素点的位置。

# 实验步骤

1. 芯片选择与管脚分配：

芯片选择STM32F407IGTx

管脚分配如下：



根据原理图分配片选线，配置LCD标准接口，分配LCD寄存器，数据长度设置为16位。

1. 程序编写

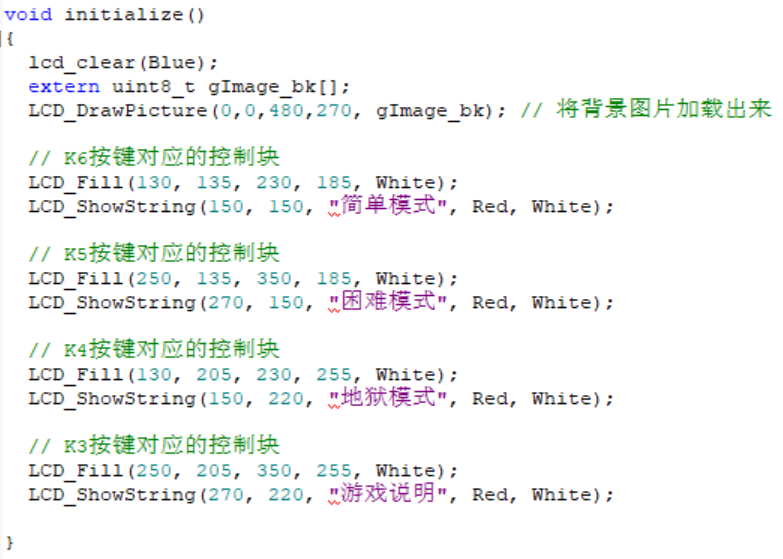
（1）文字库和图形库的增加：

将实验14中的font.h添加到项目，在其中的字模表中添加我们需要显示的汉字。将汉字内码作为键，将点阵码数据作为值。点阵码的得到可以根据免费的计算工具得到。

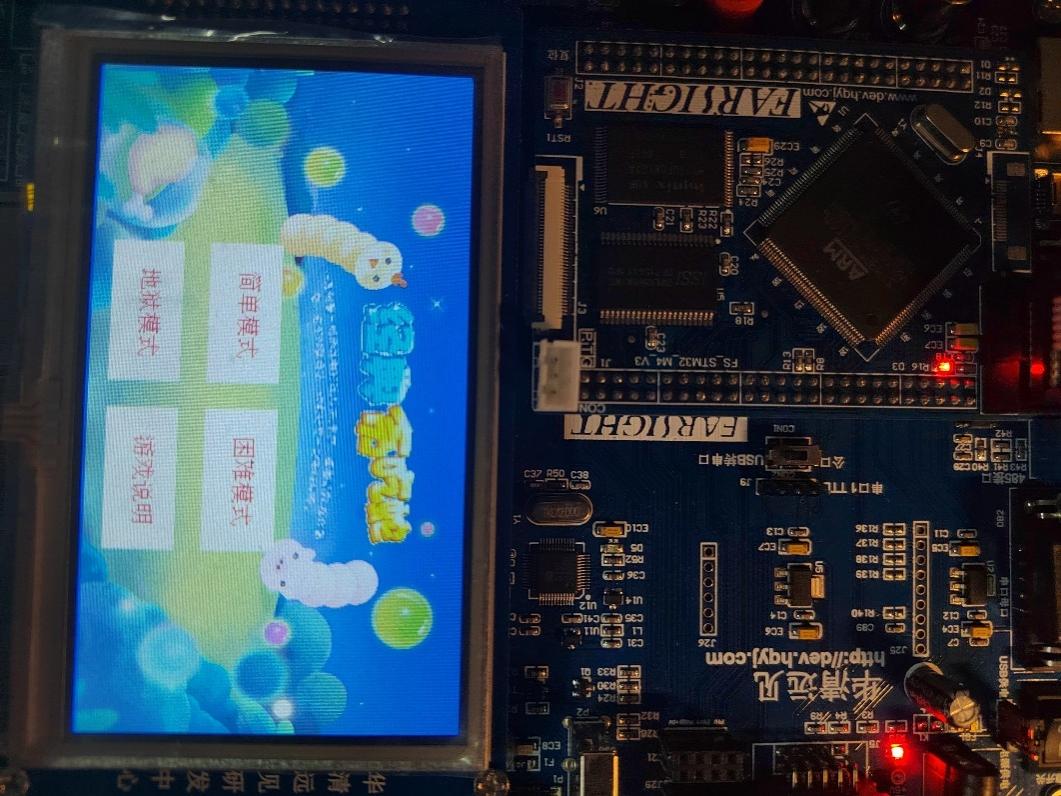
在其中的图片库中添加我们需要显示的图片数组。图片数组由工具软件Img2Lcd得到，导出水平扫描、十六位真彩色、最大宽度480、最大高度282的c数组，将其添加到font.h中。

（2）菜单页的编写：

菜单页首先通过调用API函数加载背景图片，然后加载4个填充块，对应K6-K3键来进行操控。



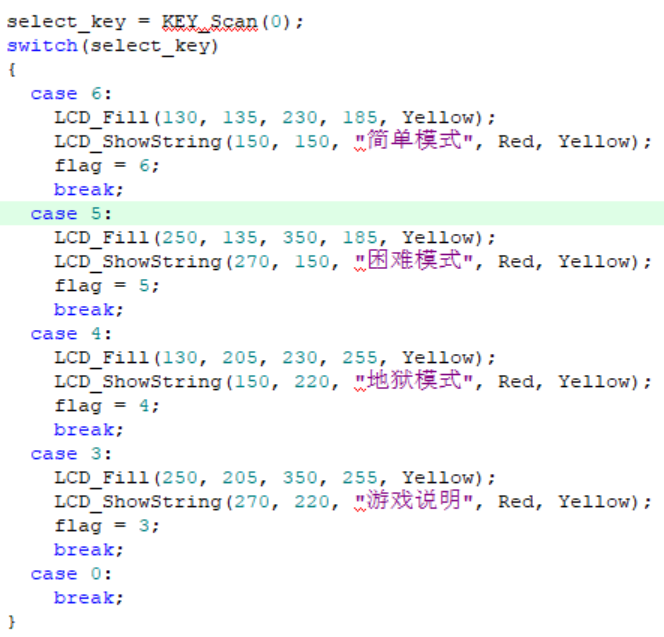
加载后的菜单栏如下图所示：



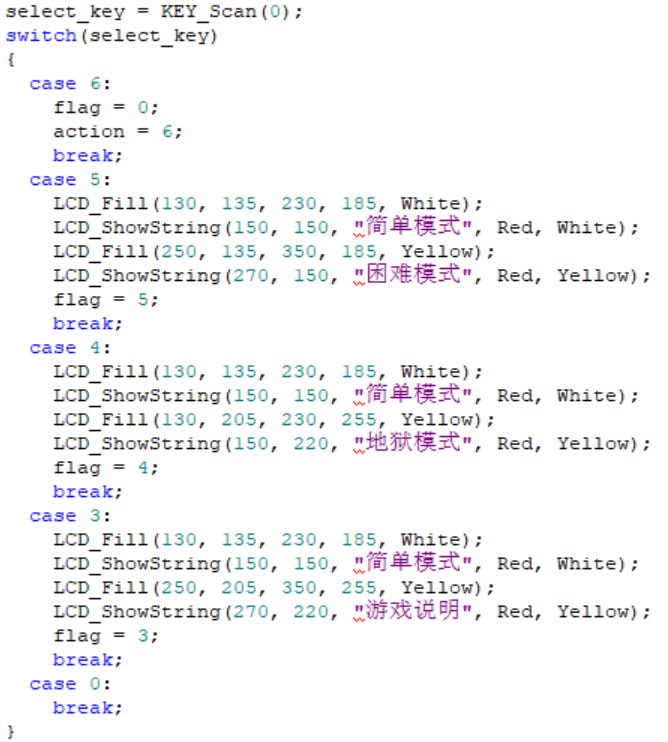
（3）用户模式选择部分代码编写：

在用户按下K6后，“简单模式”块被选中，该块变为黄色；在用户按下K5后，“困难模式”块被选中，该块变为黄色；在用户按下K4后，“地狱模式”块被选中，该块变为黄色；在用户按下K3后，“游戏说明”块被选中，该块变为黄色。一次只能有一个块被选中。

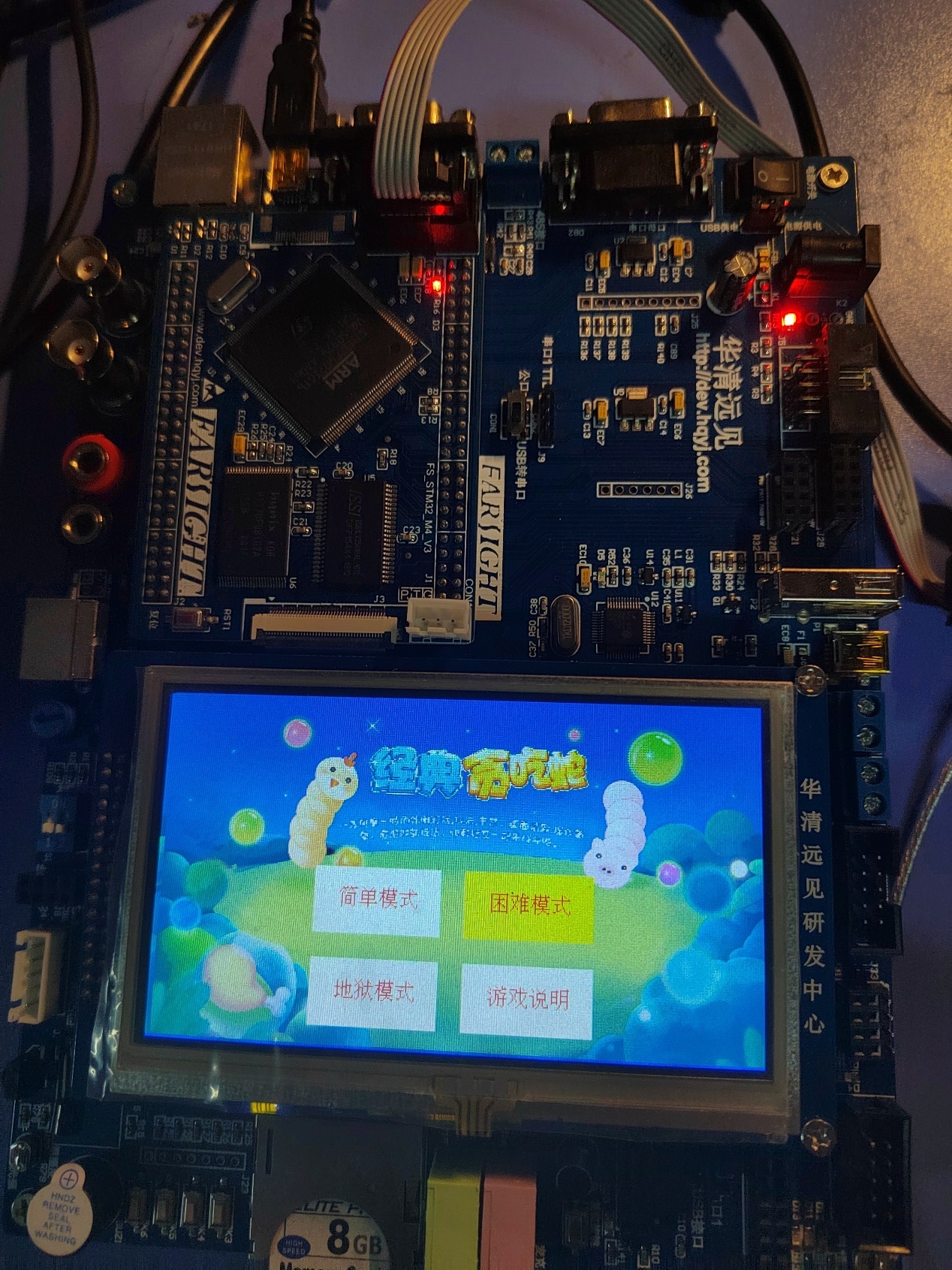
建立循环，不断获取用户按下的按键的编号，并将其块变为黄色。



由于一次只能选中一个块，引入标志位flag来记录上一次选中的块，flag初始为0，按下按键后变为对应按键编号，下一次循环进入对应编号的选择语句中，将上个编号的块还原为白色。以上一次按下K6为例：

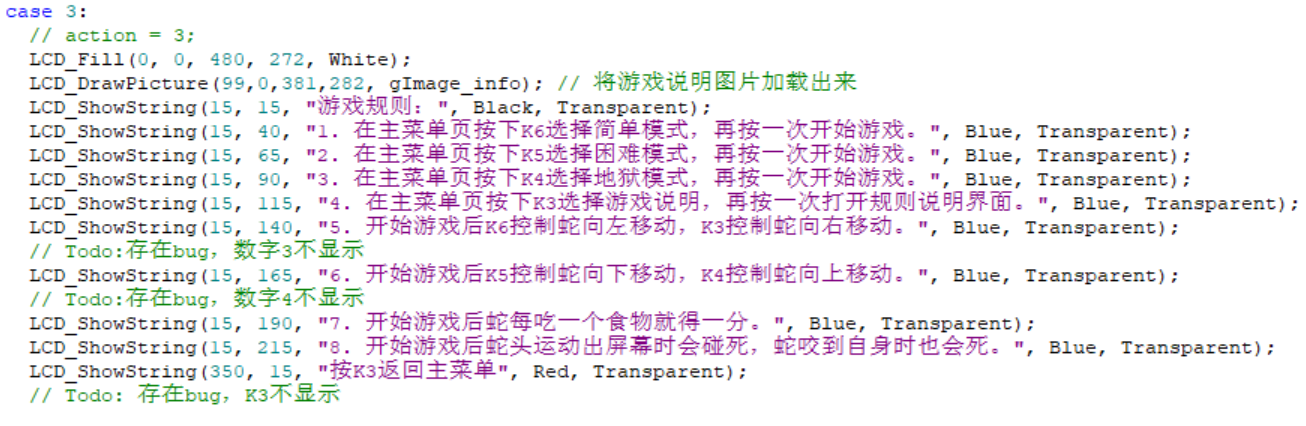


将“简单模式”块还原为白色，并将此时按下的按键变为黄色。效果如下：

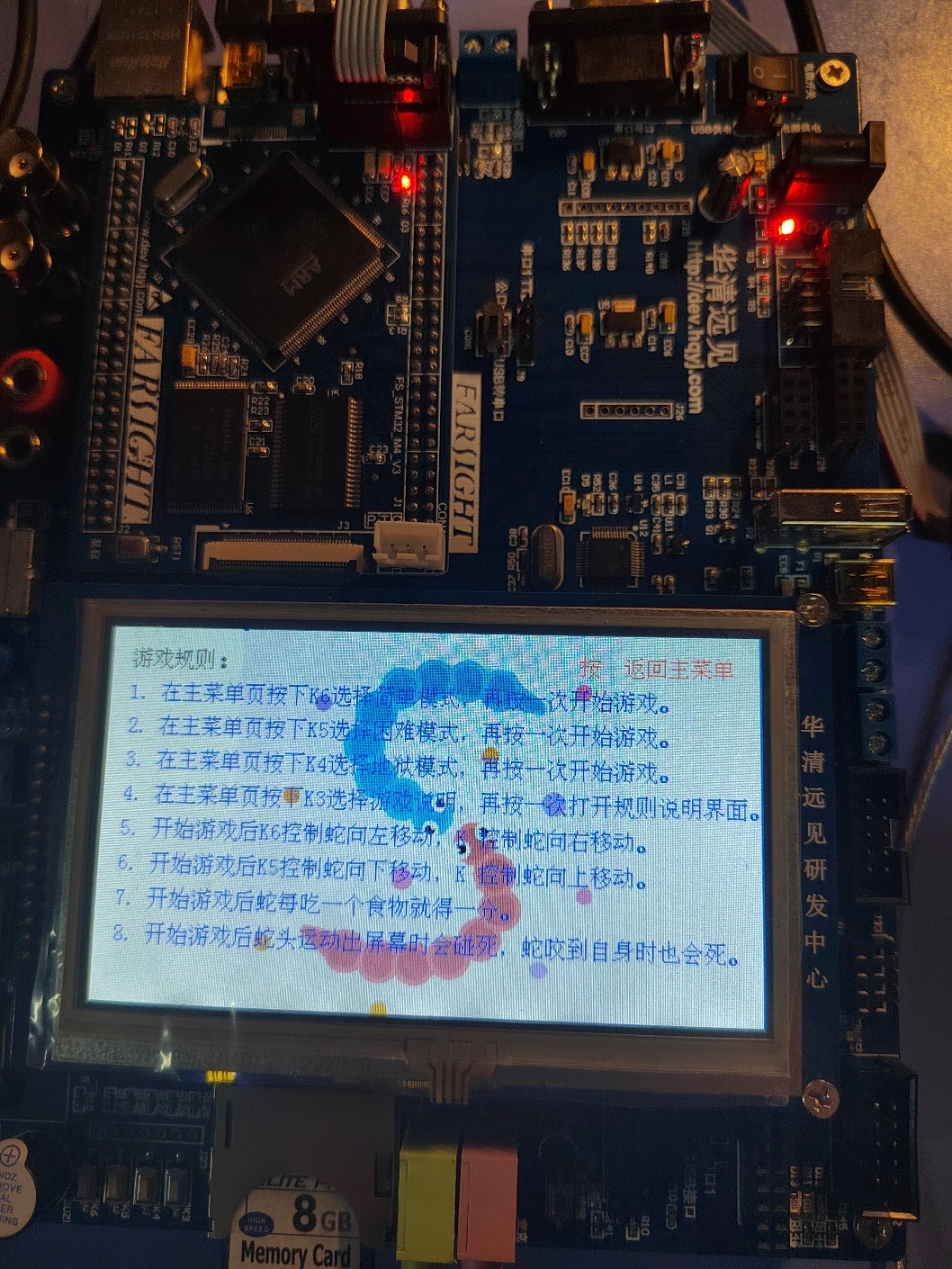


（3）游戏说明页的编写：

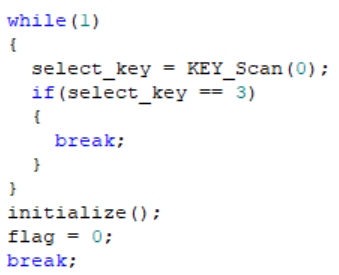
在菜单页选中K3后，再按下K3会进入游戏说明页，即在flag等于3的时候再按下K3，触发游戏说明页的显示：



显示效果如下：



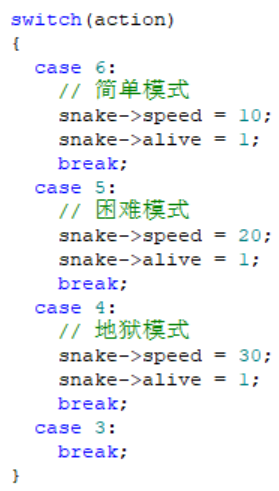
进入游戏说明页后，进入循环，不断获取用户按键，当用户按下K3时退出循环，将flag置为0，并加载标签页。代码如下：



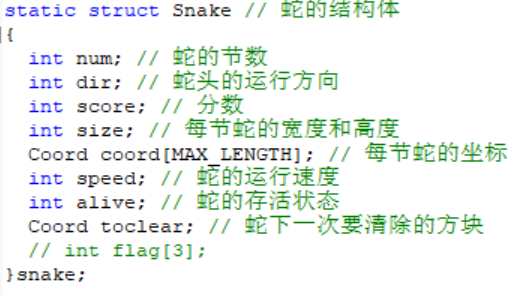
（4）游戏模式选择部分代码编写：

初始化action为0代表游戏模式。

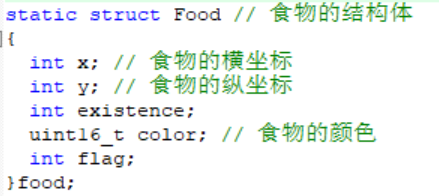
当用户首次按下K6-K4中的一个，将对应编号赋值给Flag，在下一次循环时，如果按下了相同的按键，则将action置为对应编号，循环退出的条件是action不为0，因此按下两次相同的按键后就将action置为对应编号，对action进行选择来选择不同的游戏模式，并将蛇的状态置为alive来启动游戏。游戏模式越简单，蛇的速度越慢，单节蛇的大小越小。



（5）蛇结构体及食物结构体定义



蛇的结构体如上，包含节数，每个节有对应的坐标，在LCD上绘制蛇时，根据每个节的坐标和大小绘制各节的矩形，各节连在一起形成蛇的样子。



食物的结构体如上，食物在LCD上是一个填充圆，根据食物的坐标和蛇的大小来绘制。有状态existence来表示食物被吃与否。

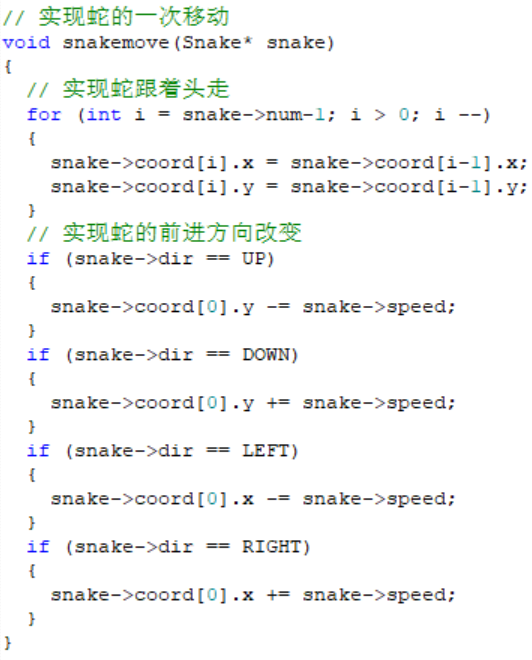
1. 游戏过程代码：

游戏开始时首先填充背景色，代表进入游戏。随后初始化蛇和食物。蛇初始化为3节，从屏幕左上角，行动方向水平向右。食物初始化为与蛇尺寸想当的随机位置的填充圆，并将食物状态existence变为1。

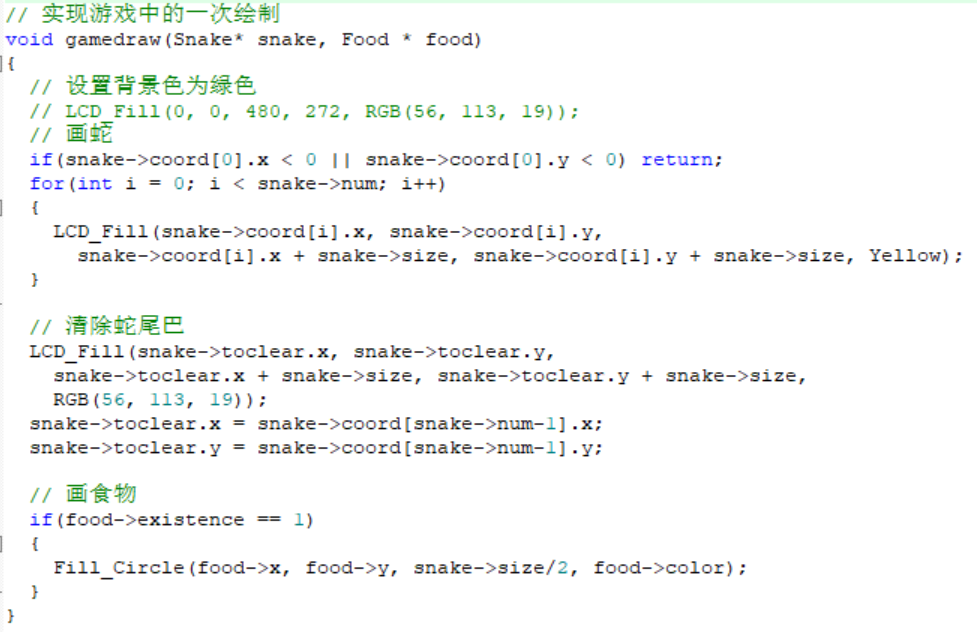
游戏过程是一个循环，退出循环条件是蛇死亡，游戏即结束。

一次游戏过程完成蛇的移动逻辑、画一次蛇和食物、蛇吃食物逻辑、获取按键逻辑、判断死亡逻辑。

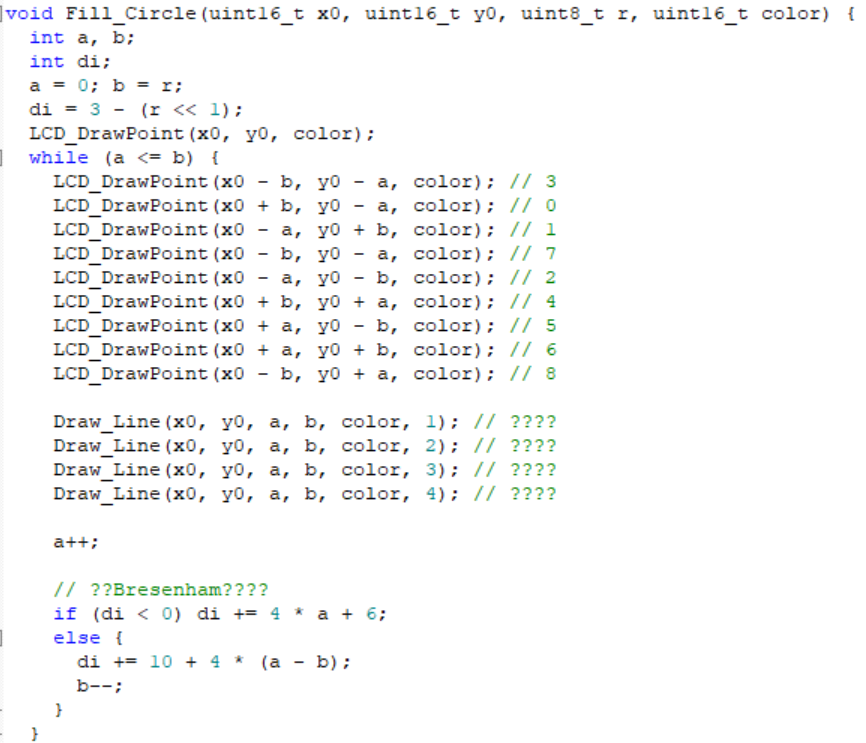
蛇的移动逻辑要求蛇的各节的坐标随着蛇头前进方向做改变，实现根据现在蛇头的移动方向改变蛇头的坐标。



画一次蛇和食物逻辑实现并没有清空上一次画好的蛇，而是画新的蛇，去掉上次画的蛇的蛇尾。再判断食物状态existence为1后画食物。

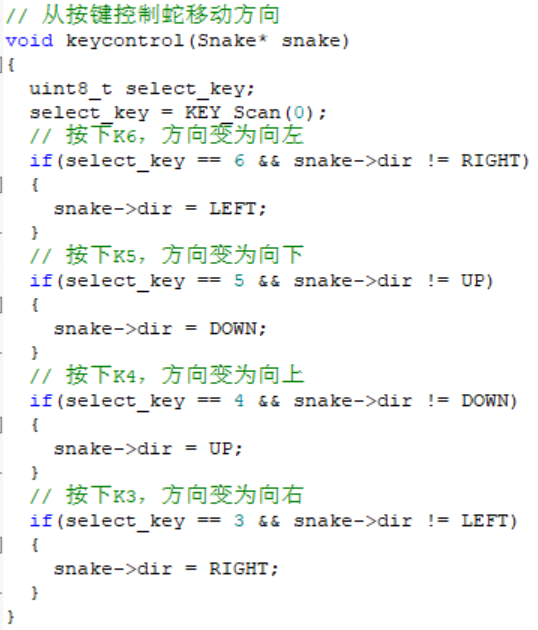


画食物的函数Fill\_Circle函数由提供给我们的API函数Draw\_Cricle函数改进。Fill\_Circle函数首先初始化了 Bresenham 算法所需的参数，然后通过循环绘制了圆上的八个对称点，并在每一步中通过 Draw\_Line 函数绘制了四个象限的线。整个循环过程通过 Bresenham 算法更新圆上的点的位置，直到整个圆被绘制完成。这种方法通过充分利用圆的对称性和 Bresenham 算法的高效性，实现了填充整个圆的效果。

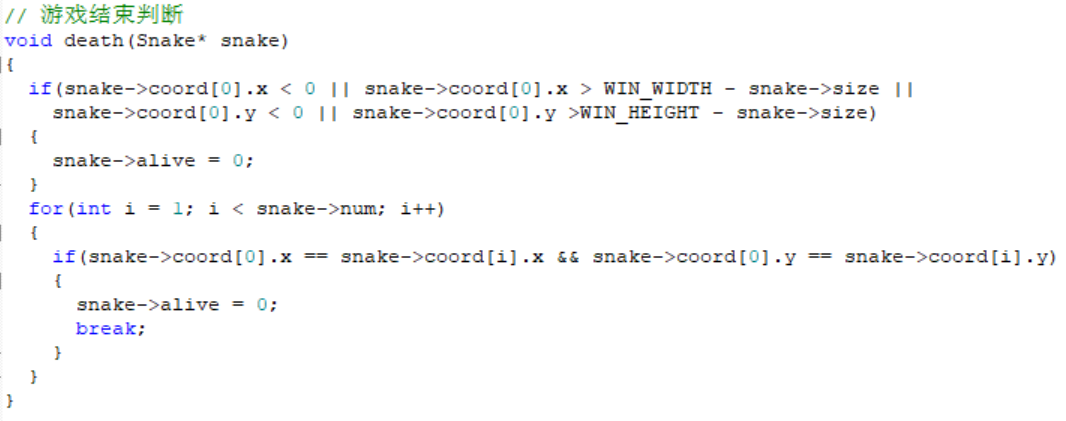


蛇吃食物逻辑是判断蛇头中心坐标是否与食物中心坐标是否重合，如果重合则将食物吃掉，将食物existence置为0，将蛇的节数加1，将本局游戏分数加一。吃掉食物后判断existence为0则初始化新的食物。初始化食物后进行逻辑判断，如果食物出现在蛇身上则重新初始化食物，直到初始化的蛇出现在其他位置为止。

获取按键响应逻辑是用来获取K6-K3其中一个按键来改变蛇的前进方向，蛇只能实现90°转向而不能实现180°转向。以蛇现在向右前进为例，按下向左键后无响应，按下上键或下键才能转向活动。

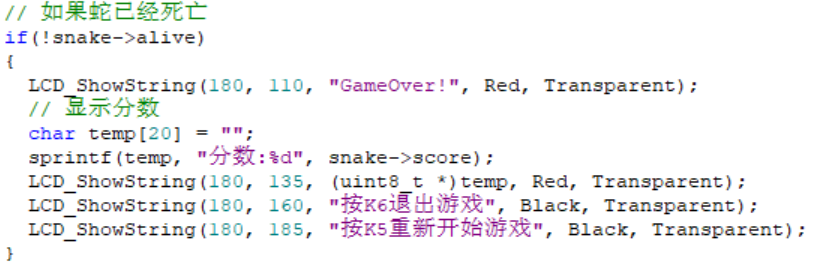


蛇的判断死亡逻辑是蛇头坐标出屏幕外，即蛇不能碰壁，同时，蛇不能碰到自身。

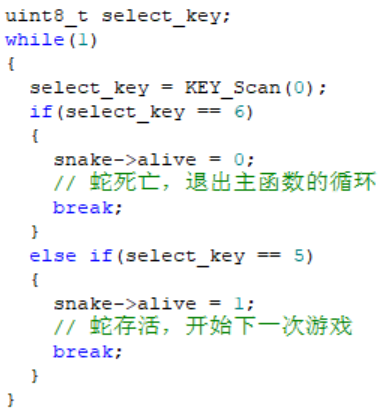


1. 游戏结束逻辑实现。

当蛇死亡后，当前游戏结束。此时屏幕上显示所得分数，并显示提示，提示用户可以按下K6退出游戏，按下K5重新开始游戏。



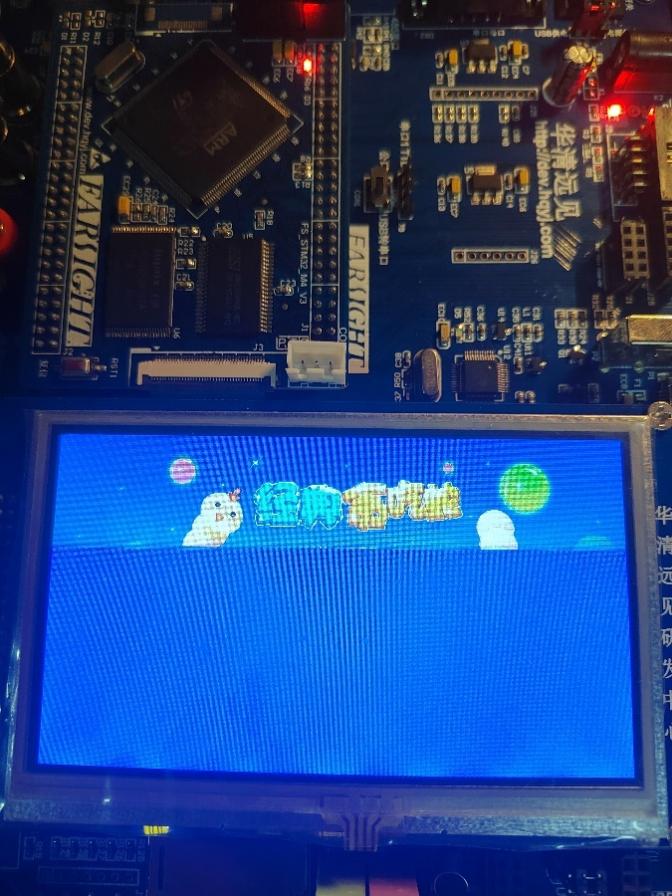
此时进入死循环，不断读取用户按键，如果用户按下了K6，则将使蛇继续死亡，从而退出游戏过程；如果用户按下了K5，则将蛇复活，直接进入下一游戏过程。退出游戏过程后，将直接通过goto语句跳转到主菜单界面。



1. 游戏过程典型截图：

菜单栏加载截图：

先加载背景图片，再加载模式块显示。



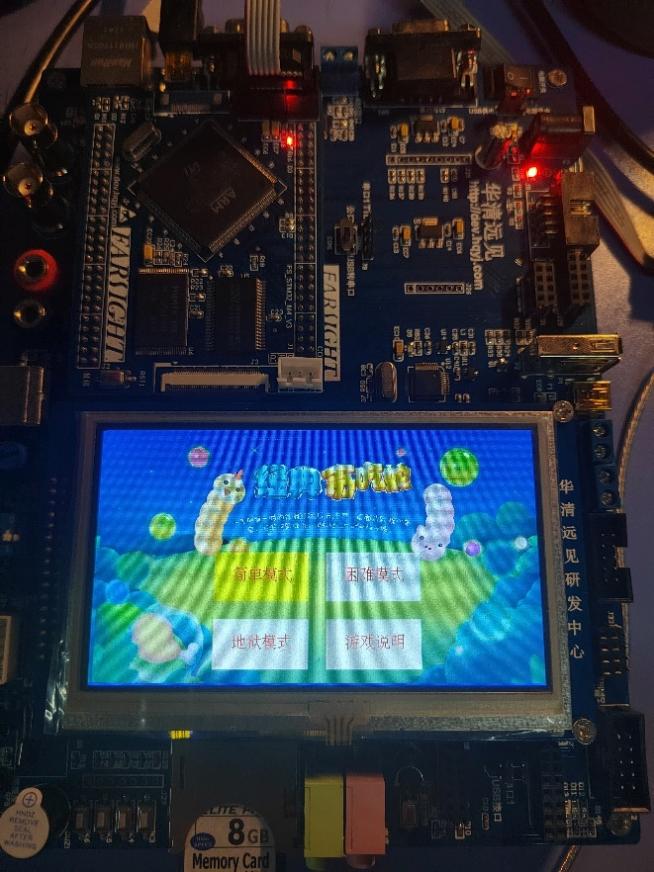
菜单栏截图：

显示背景图片，图片上有四个控制块，分别是“简单模式”、“困难模式”、“地狱模式”、“游戏说明”。



模式块选中截图：

按下K6，“简单模式”块被选中。



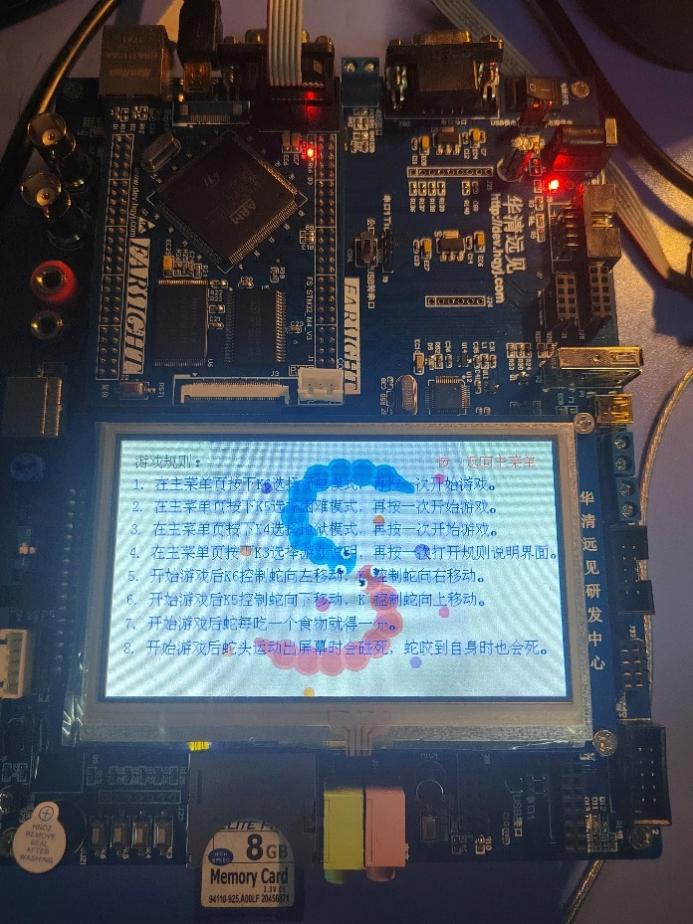
按下K5，“困难模式”块被选中。



按下K3，“游戏说明”块被选中。

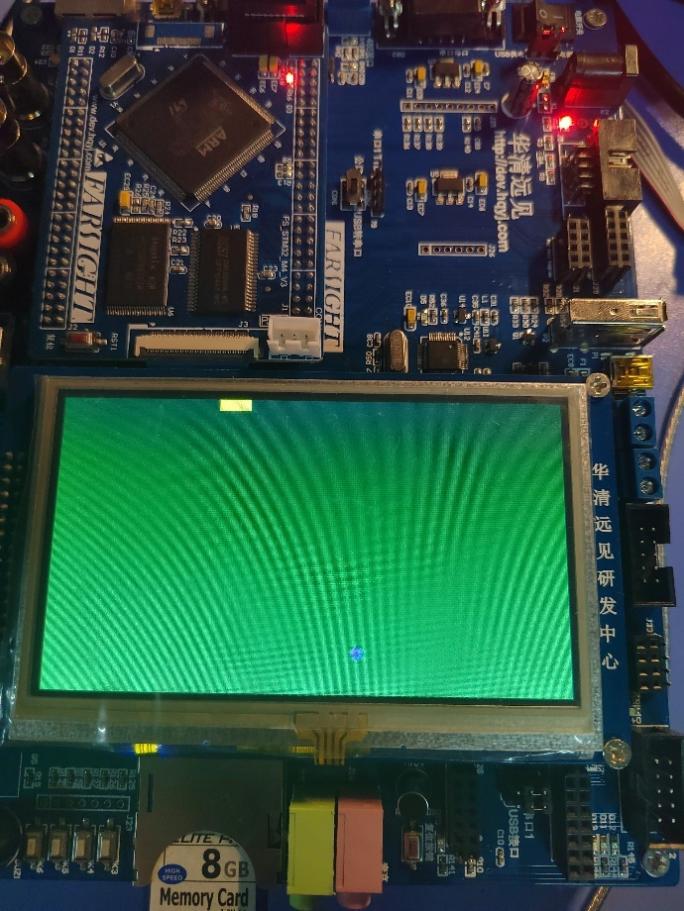


游戏规则页截图：

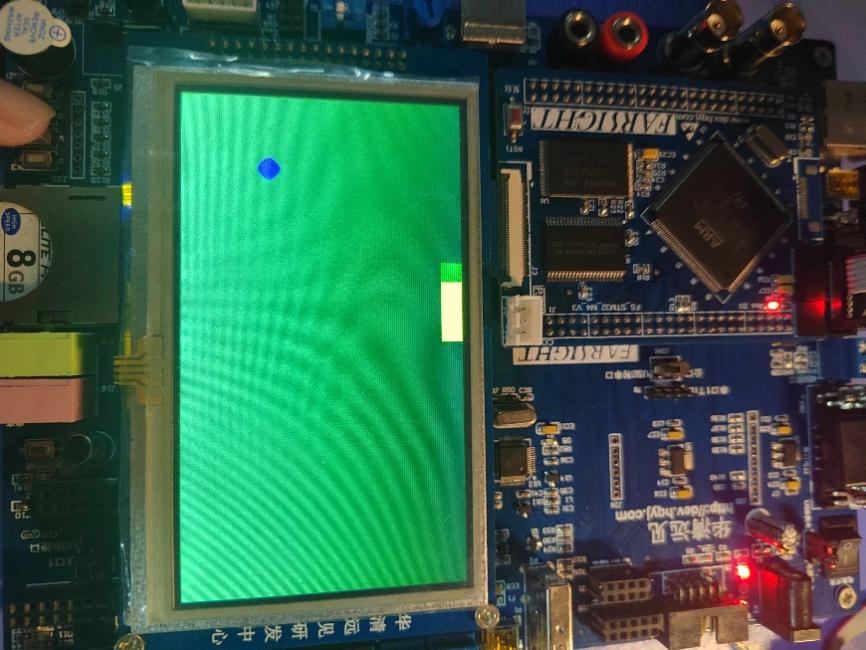


游戏运行时截图：

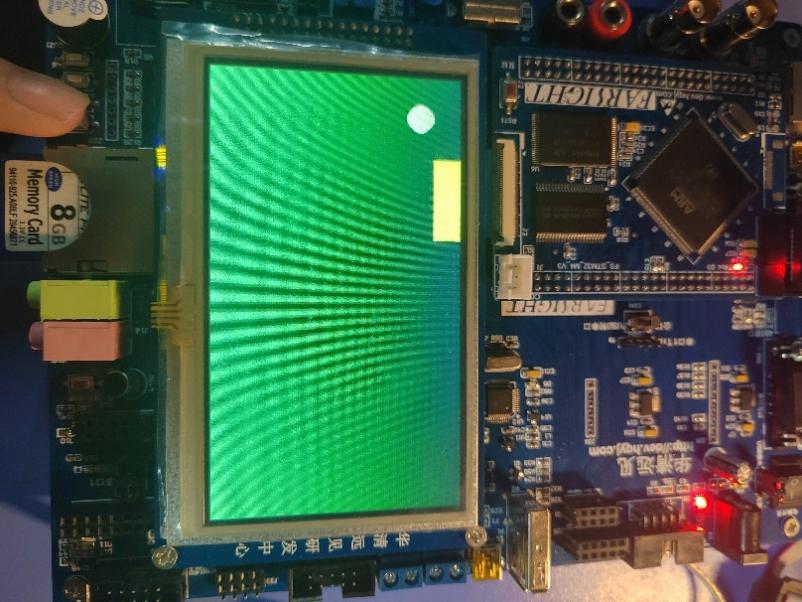
简单模式中，蛇的尺寸最细、运行速度最慢。



困难模式中，蛇的尺寸和速度都比较大。

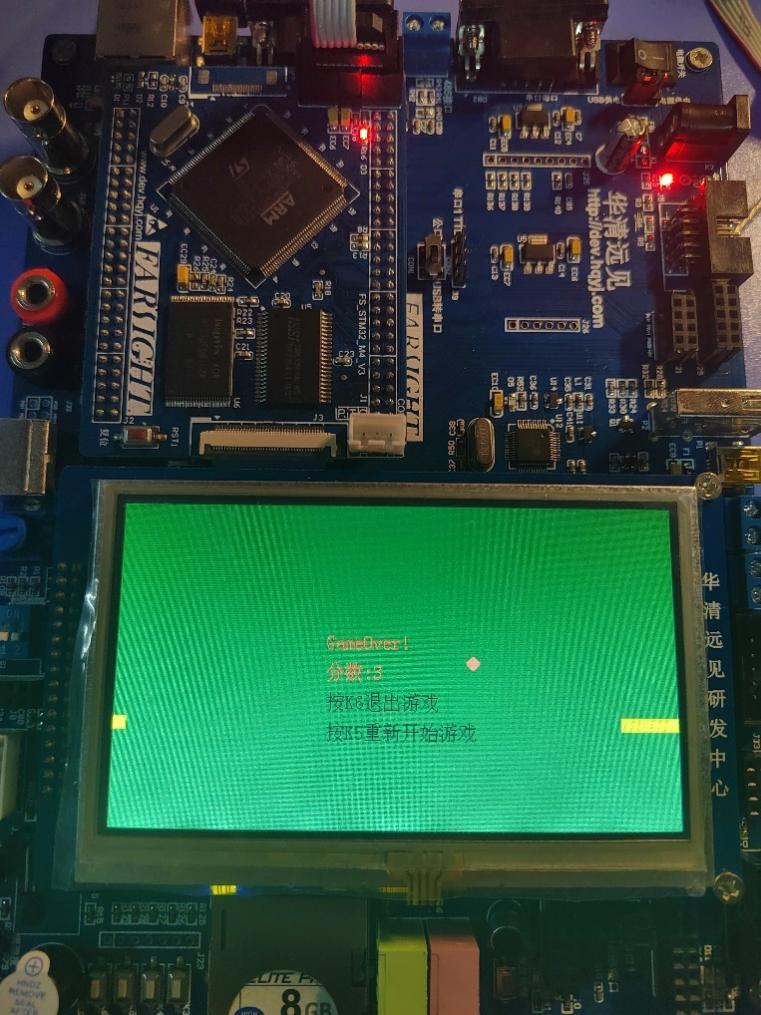


地狱模式中，蛇的尺寸和速度最大。



游戏结束时截图：





# 实验思考与总结

在这次综合实验中，我深刻体会到了嵌入式系统开发的魅力。与单纯的软件开发不同，嵌入式开发需要将软件和硬件紧密结合，通过不断的调试和优化，才能实现理想的效果。这个过程锻炼了我的编程能力，也让我熟悉了嵌入式系统的工作原理。

为了提高编程效率，我充分利用了嵌入式库提供的API函数，它们可以帮助我快速实现一些常用的功能。同时，我也尝试对一些API函数进行改进，创造出自己的API函数，满足更多的需求。

例如，我改进了字符串显示函数，让它可以支持透明背景。我通过牺牲一个RGB值作为标志，让字符显示函数在绘制背景色时，跳过这个RGB值，从而实现透明效果。我还改进了画圆函数，让它可以绘制实心圆。我利用了Bresenham算法，它是一种高效的画圆算法，它可以在圆的八个对称位置画出八个点。我在此基础上，利用圆的对称性，画出四条线，填充圆的内部，从而形成实心圆。