上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

《嵌入式及 FPGA 实验》课程设计

基于 STM-32 的贪吃蛇游戏

学年学期：　　2024-2025 学年第二学期

小组成员：　　马悦钊（组长）　吴神宇

罗承煜　唐博宇　郑昊咏

专　　业：　　信息安全

学　　院：　　计算机学院（网络空间安全学院）

指导老师：　　周志洪

完成时间：　　2025 年 6 月 1 日

摘　要

本文主要研究了基于 STM32F103VET6 的贪吃蛇游戏的设计与实现，也是对本学期《嵌入式及 FPGA 实验》课程设计的总结报告。贪吃蛇游戏作为经典的计算机游戏之一，具有简单易懂的规则和较强的娱乐性，适合作为嵌入式系统开发的实践项目。本课程设计小组以 STM32F103VET6 野火指南者开发板为硬件平台，利用其丰富的外设资源与强大的处理能力，通过对游戏逻辑的分析与嵌入式系统开发技术的应用，构建了一个具有良好用户体验的游戏系统。本文首先介绍了项目背景及意义，接着详细阐述了系统的硬件架构与软件设计，最后通过实验验证了系统的可行性与性能，展示了项目成果。该项目不仅实现了贪吃蛇游戏的基本功能，还通过优化代码结构与资源管理，提升了系统的响应速度与稳定性。游戏界面友好，操作简单，适合各年龄段用户使用，同时也为嵌入式系统的教学与实践提供了有益的参考与借鉴。

关键词： STM32F103VET6；野火指南者；贪吃蛇游戏；嵌入式系统；用户体验

目 录

第 1 章　项目概述

1.1　项目背景

在当今数字化浪潮汹涌澎湃的时代，快节奏的生活旋律与高强度的学业压力，如同无形的枷锁，悄然重塑着儿童的课余生活图景。短视频、网络游戏等碎片化信息与被动式娱乐内容，正逐渐占据孩子们的闲暇时光。当儿童长时间沉浸于电子屏幕前，单向接收大量未经深度思考的信息，不仅会导致思维拓展受限，久而久之，更会对其心理健康与创造力发展形成潜在威胁。据相关调查显示，过度依赖此类娱乐方式的儿童，在专注力、逻辑思维能力等方面的表现，相较参与主动式、创造性活动的儿童存在明显差距。因此，开发一款集教育性与趣味性于一体的应用程序，引导儿童在玩乐中学习与成长，已然成为顺应时代需求的紧迫任务。

与此同时，嵌入式系统技术正以惊人的速度向前迈进，在工业控制、智能家居、智能交通等领域掀起智能化变革的浪潮。从自动化生产线的精准控制，到智能家居设备的互联互通，嵌入式设备已成为推动各行业技术革新的核心力量。这一领域的开发工作，不仅要求开发者熟练掌握硬件电路设计、驱动程序编写等专业技能，更需要具备在内存容量有限、处理器性能相对较低等严苛条件下，进行高效软件开发的能力。在此背景下，经典的贪吃蛇游戏以其独特的适配性，成为嵌入式系统开发实践的理想场景。将这一广为人知的游戏移植到嵌入式平台，不仅能够激发开发者的创新思维，还能让理论知识在实际项目中落地生根，转化为实实在在的技术成果。

贪吃蛇游戏虽然规则简明易懂，但其背后却蕴含着图形绘制、用户交互、游戏逻辑处理、定时器应用等多项嵌入式开发的核心技术要点。在将游戏移植到嵌入式平台的过程中，开发者需要攻克诸多挑战：如何在有限的内存空间中高效存储游戏数据？怎样优化代码以适配性能较低的处理器？如何精准设计硬件驱动程序（如显示屏驱动、按键驱动），实现与软件算法的无缝协同？这些问题的解决过程，能够帮助开发者深入理解嵌入式系统的运行机制，熟练掌握资源优化与合理分配的方法。对于参与开发的小组成员而言，每一次对游戏功能的完善、每一次对性能瓶颈的突破，都是对嵌入式系统软硬件协同开发能力的深度打磨。

本课程小组开发的贪吃蛇小程序，不仅仅是一款娱乐应用，更是一座连接理论与实践的桥梁。团队成员通过完整参与从游戏需求分析、功能设计，到代码实现、系统调试优化的全流程开发，将课堂中所学的嵌入式系统理论知识转化为实际开发能力。即便未来部分成员不再从事嵌入式相关研究，这段经历也将成为他们学习生涯中一段快乐且难

忘的宝贵回忆，其中所培养的问题解决能力、团队协作精神，都将成为伴随我们未来发展的重要财富。

1.2　开发环境

本项目的开发环境主要包括以下几个方面：

- **实验平台**：野火 STM32 指南者开发板
- **芯片选择**：STM32F103VET6
- **芯片架构**：ARM Cortex-M3
- **晶振频率**：72MHz
- **开发工具**：Keil5[1]、STM32CubeMX、VSCode、Git
- **编程语言**：C 语言
- **固件库**：标准库 V3.5

1.3　整体结构

1.3.1　硬件驱动层

硬件驱动层主要负责与硬件设备的交互，包括对外设的初始化、控制和数据交换等功能。在这个项目中，硬件驱动层的设计涉及多个模块，每个模块对应一个具体的硬件功能。以下是各个模块的简要说明：

- **ADC**：模拟数字转换器驱动代码，在本项目中主要用于生成初始化随机数，为食物生成提供随机位置。
- **LCD**：电阻触摸液晶屏驱动代码，ILI9341 实现对液晶屏的初始化、显示内容的绘制和刷新等功能，XPT2046 实现对触摸屏的坐标采集和处理。
- **GPIO**：包含 LED 和 KEY 相关驱动代码，LED 用于控制板载 LED 的亮灭状态，指示系统状态或调试信息；KEY 用于处理按键输入，控制贪吃蛇的移动和转向。
- **SysTick**：提供系统滴答定时器的相关功能，用于系统时间管理。
- **TIM**：实现通用定时器的功能，用于实现游戏中的定时任务，定时扫描 GPIO 输入和触摸屏输入，实现 SPI 交互触摸屏，以及实现了全彩 LED 灯泡的 RGB 渐变灯效。

1.3.2　应用层

应用层主要负责实现具体的应用逻辑和功能模块。在本项目中，应用层的设计包括以下几个重要文件：

- **main**：项目主程序文件，是程序执行的入口点，负责初始化各个模块，调用相关函数实现系统整体功能，并协调各硬件模块的协同工作。
- **snake**：用于管理贪吃蛇游戏的相关功能，包括游戏逻辑、状态管理、碰撞检测、食物生成等，为游戏的顺利进行提供支持。

1.3.3　中断处理层

中断处理层主要负责处理系统中的各种中断事件，确保系统在中断发生时能够及时响应并执行相应的处理逻辑。该层的设计包括以下文件：

- **stm32f10x_it**：定义 STM32 中断向量表以及各中断服务函数，处理外部中断、定时器中断等各类中断事件，确保系统在中断发生时能及时响应并处理。

1.3.4　配置层

配置层主要负责系统的硬件配置和库函数的设置。该层的设计包括以下文件：

- **stm32f10x_conf**：用于配置 STM32F10x 系列微控制器的库函数头文件，可通过宏定义开启或关闭相关外设功能，选择系统时钟源等配置项，是系统硬件配置的关键文件。

1.4　游戏设计

贪吃蛇游戏的核心逻辑主要包括以下几个方面：

游戏初始化时，首先完成界面搭建，创建矩形游戏窗口并划分边界作为后续游戏中的碰撞检测基础。接着构建网格（不显性化，但作为游戏的基本长度单位），左上角窗口显示当前得分，右上角显示游戏得分最高记录。然后设置游戏初始状态，包括蛇的初始位置、方向和移动速度等。最后，生成初始食物位置。

游戏运行过程中，核心交互逻辑包括方向控制、蛇身移动、食物判定和碰撞检测等。经过测试，我们发现野火指南者开发板的触屏响应速度较慢，不足以支撑较好的游戏体验，因此我们放弃了使用触屏控制方向，而是使用了开发板上的两个按键分别控制蛇头的左右转向。贪吃蛇的移动逻辑是蛇头按当前方向移动一个单位，随后所有蛇身节点依次移动到前一节点位置。食物判定则是在蛇头与食物坐标重合时，贪吃蛇的长度和当前

得分加一，并在场上生成新的食物；随着得分的增加，游戏难度也会相应提升，贪吃蛇的移动速度逐渐增加，使游戏更加具有挑战性。碰撞检测包括边界碰撞和自碰撞，当贪吃蛇的头部触碰到游戏窗口边界或与任意自身部分时，游戏结束。游戏过程中，得分最高记录会被实时更新在游戏界面右上角。

除此之外，我们还实现了一些有意思的小功能，如游戏开始时的欢迎和提示界面、游戏过程中通过触屏实现暂停和继续、开机后 LED 灯泡的 RGB 渐变灯效等功能。而在基础的贪吃蛇游戏之上，我们还设计了一些游戏中的小道具，如吃掉后可以使当前得分翻倍的“双倍果实”①，可以增加血量上限的“生命水晶”②，可以使蛇身变短的“缩短药水”等③，以及可以降低移动速度的“减速药水”④。这些小道具的设计不仅增加了游戏的趣味性，也使得游戏的策略性更强，玩家需要根据当前具有的效果选择合适的操作。

① 游戏中贪吃蛇的初始得分为 0，吃掉普通食物后得分加 1，吃掉双倍果实后得分翻倍，若当前的分为 0，则得分变为 1

② 游戏中贪吃蛇的初始血量为 2，吃掉生命水晶后血量加 1，蛇头每次与边界或者自身碰撞则血量减 1，减到 0 则判定失败，游戏结束，若不为零则会将蛇头传送至任意空白区域，并保持原有移动方向不变

③ 游戏中贪吃蛇的初始长度为 5，吃掉普通食物后长度加 1，吃掉缩短药水后长度减半

④ 游戏中贪吃蛇的移动速度会随着得分的增长而加快，每次吃掉减速药水后当前速度均降低至原来的 0.8 倍

1.5　项目分工

成员	分工内容
马悦钊[1]	• 完成 STM32F103 开发板与外部设备（如 LCD 显示屏、按键等）的硬件接口设计与连接调试。 • 基于 ST 固件库 3.5 开发 LCD 驱动程序、按键扫描驱动及定时器驱动，协助其他成员排查硬件相关问题，提供技术支持。 • 撰写项目报告、整理项目文档，负责项目整体进度把控与协调。
吴神宇[2]	• 设计游戏数据结构（如蛇身、食物、游戏状态结构体），规划游戏运行逻辑框架。 • 编写蛇身移动、食物生成、碰撞检测（边界碰撞、自碰撞）等核心算法代码。 • 实现游戏难度调整、得分系统及游戏结束判定功能。
郑昊咏[3]	• 制定测试计划，检测游戏运行中的性能瓶颈（如内存占用、帧率稳定性），提出优化方案。 • 收集测试数据，记录并反馈功能缺陷与 Bug，协助开发成员修复问题。
罗承煜[4]	• 设计游戏界面布局，规划菜单、得分显示、游戏区域等 UI 元素。 • 编写 LCD 绘图函数，实现蛇身、食物、文字信息的绘制与刷新。 • 处理按键输入逻辑，实现蛇移动方向控制与游戏暂停、开始功能。
唐博宇[5]	• 统筹项目进度，制定开发计划表，协调小组成员工作安排。 • 撰写项目文档、整理开发文档、用户手册等。 • 整理项目代码、报告等资料，完成项目归档。

[1] 马悦钊　学号：523031910684
[2] 吴神宇　学号：523031910465
[3] 郑昊咏　学号：523031910010
[4] 罗承煜　学号：523031910624
[5] 唐博宇　学号：523031910747

表 1.1　项目成员分工表

第 2 章　硬件设计

2.1　硬件概述

在本次课程设计的项目中，我们使用了 STM32F103VET6 野火指南者开发板作为硬件平台。STM32F103 系列芯片是意法半导体（ST）公司推出的基于 ARM Cortex-M3 内核的 32 位微控制器，广泛应用于嵌入式系统中。

STM32 系列产品有固定命名规则，本开发板名称含义见表2.1。[2]

表 2.1　STM32 系列芯片命名规则

项目	值	含义
家族	STM32	32bit MCU
产品类型	F	基础型
引脚数目	V	100pin
FLASH 大小	E	512KB
封装	T	QFP 封装
温度等级	6	A：-40~85℃

2.2　主控芯片

STM32F103VET6 芯片属于 STM32F1 系列，基于 ARM Cortex-M3 内核[3]，主频最高可达 72MHz，具备 512KB Flash 和 64KB SRAM，能够满足贪吃蛇游戏逻辑处理、数据存储及图形显示的性能需求。其丰富的外设资源，如定时器、通用定时器、SPI、USART 等，为游戏功能拓展和外部设备连接提供了便利。该芯片的规格参数见附录5.1，时钟设置和引脚设计见图2.1和图2.2。

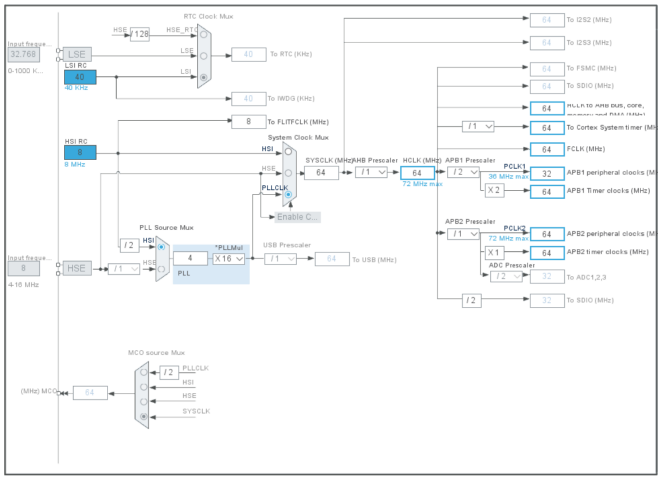


图 2.1　STM32F103VET6 时钟设置

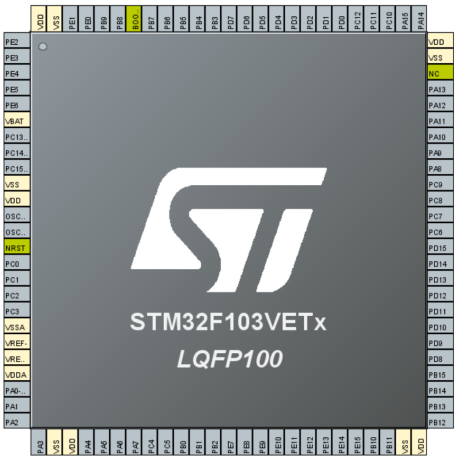


图 2.2　STM32F103VET6 引脚设计

2.3　按键模块

本项目使用了开发板上的两个按键作为游戏的输入控制。野火指南者开发板连接的按键带有硬件消抖功能，按键原理见图2.3，它利用电容充放电的延时，消除了波纹，从而简化软件的处理，软件只需要直接检测引脚的电平即可。[4]

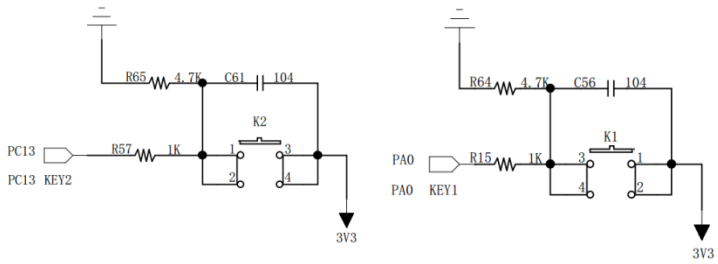


图 2.3　按键原理图

2.4　LED 指示灯模块

本项目使用了开发板上的 LED 指示灯来显示游戏状态。我们使用的野火指南者开发板上有一个 RGB LED 灯，由红、绿、蓝三个小灯组成，可以通过 PWM 控制来混合成 256 种不同的颜色。LED 灯的连接方式见图2.4。这些 LED 灯的阴极都是连接到 STM32 的 GPIO 引脚，只要我们控制 GPIO 引脚的电平输出状态，即可控制 LED 灯的亮灭。[4]

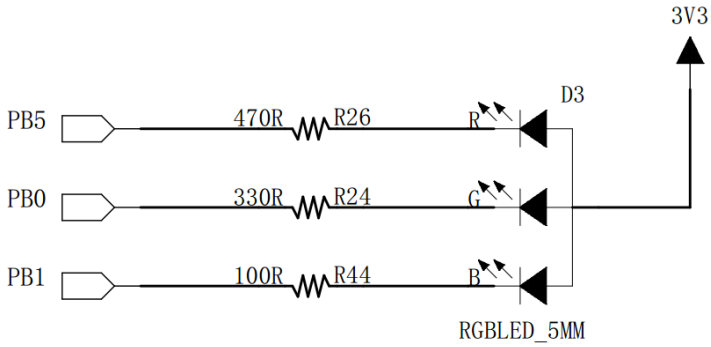
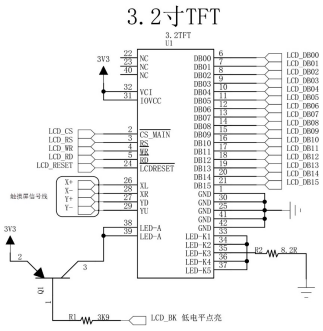


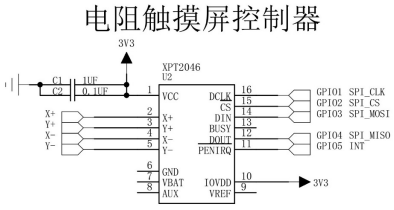
图 2.4　LED 灯的连接方式

2.5　显示及触摸模块

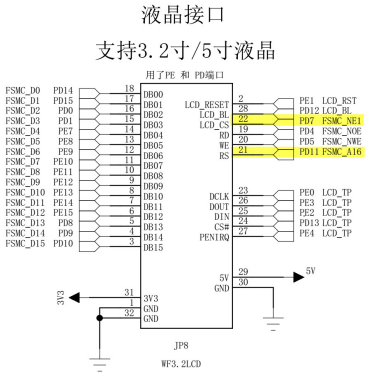
野火指南者开发板配备了一个分辨率为 320 * 240 的 3.2 寸电阻触摸液晶屏，该面板由液晶屏和触摸屏组成，液晶屏内部包含有一个液晶控制芯片 ILI9341，使用 8080 接口与单片机通讯；PCB 底板上还包含了一个电阻触摸屏的控制器 XPT2046，该控制器实质上是一个 ADC 芯片，通过检测电压值来计算触摸坐标。液晶屏背面的 PCB 电路对应图2.5。[4]



屏幕 PCB 底板原理图



屏幕 PCB 底板的触摸部分原理图



开发板与屏幕的连接的信号说明

图 2.5　液晶屏背面的 PCB 电路

借助液晶屏的显示功能，我们可以在屏幕上绘制游戏界面、显示游戏状态和分数等信息。液晶屏的分辨率为 320 × 240，能够提供清晰的图形显示效果。同时，触摸屏的输入功能使得玩家可以通过触摸屏幕来控制游戏操作，如开始游戏、暂停等。

2.6　调试模块

本次使用的野火指南者开发板上具有 SWD 接口和 JTAG 接口。其中，SWD（Serial Wire Debug）即串行单线接口，是一种用于 ARM Cortex-M 系列微控制器的调试接口，提供了高效的调试功能。通过 SWD 接口，我们可以连接到开发板进行代码下载、调试和监控。而 JTAG（Joint Test Action Group）接口则是另一种常用的调试接口，支持更复杂的调试功能。本次项目中，我们使用配套的 Fire-Debugger 仿真器以及 SW-JTAG 转换器连接到 SWD 接口进行调试。

2.7　本章小结

本章介绍了本项目所使用的硬件平台和主要硬件模块。我们选择了 STM32F103VET6 野火指南者开发板作为主控芯片，利用其丰富的外设资源和强大的处理能力来实现贪吃蛇游戏的功能。按键模块、LED 指示灯模块、显示及触摸模块以及调试模块等硬件组件的设计和实现，为游戏的交互和显示提供了必要的支持。通过这些硬件模块的协同工作，我们成功地将贪吃蛇游戏移植到嵌入式平台上，为后续的软件设计奠定了坚实的基础。

第 3 章　软件设计

3.1　软件概述

本项目的软件设计基于 STM32F103VET6 开发板，采用分层架构实现游戏逻辑与硬件控制的解耦。软件核心流程围绕游戏状态机展开，涵盖初始化、逻辑循环、交互处理及状态反馈等关键环节。游戏程序的主要流程如下图所示：

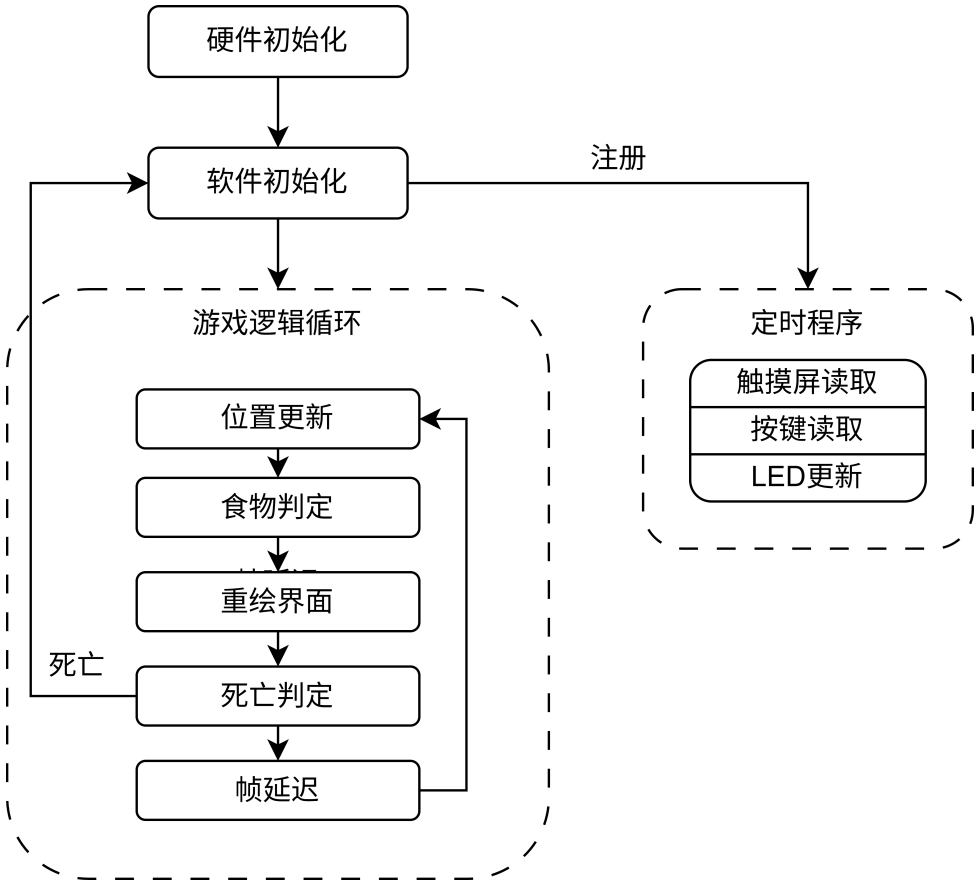


图 3.1　贪吃蛇游戏软件流程图

游戏开始前的初始化阶段包含对硬件与软件的双重配置。硬件初始化包括对 LCD 显示屏、按键、LED 小灯泡、定时器等外设的驱动配置，确保游戏界面能够正确显示并响应用户输入。软件初始化则涉及游戏逻辑的设置，主要包含对变量、数据结构进行初始化，如蛇身节点的定义、食物坐标的生成、游戏状态标志、游戏得分以及蛇的血量的

初始化等。

游戏逻辑循环是游戏的核心部分，它不断执行以下步骤：

1. **位置更新**：这个部分中，首先创建一个新的蛇头节点 nexthead，然后根据当前蛇头 head 和状态 status 计算新蛇头的位置。如果蛇没有吃到食物，则移动蛇身，即更新蛇的各个节点位置，并移除蛇尾；如果吃到食物，则增加蛇的长度，并根据食物类型更新分数、速度等因素。
2. **食物判定**：检查新蛇头是否与食物位置重合。如果是，则根据食物类型执行相应的效果，如增加得分、同步历史最高分、改变贪吃蛇的移动速度、修改贪吃蛇的长度等，并重新生成食物。
3. **重绘界面**：刷新显示界面，通过使用 LCD 相关函数绘制蛇的身体、头部眼睛以及标题等元素，以显示最新的游戏状态。
4. **死亡判定**：通过 `judge_alive()` 函数进行检查，判断蛇是否与墙壁或自身发生碰撞。如果发生碰撞，则将 die 标志置为 1；接着对贪吃蛇的血量进行操作，如果血量为 0，则游戏结束。
5. **帧延迟**：通过 `Delay(current_delay())` 函数控制游戏的帧率，调节蛇移动的速度，使游戏运行更加流畅。`current_delay()` 函数根据当前游戏状态（如速度、得分等）动态调整延迟时间，以实现游戏难度的逐步增加。

定时程序负责周期性地触发游戏逻辑循环，读取触摸屏和按键的状态，以便玩家控制蛇的方向和游戏暂停。同时还会控制 LED 灯显示颜色以反映游戏状态。

当蛇死亡时，会在屏幕上显示“Game over”信息，并提示玩家点击任意按键重新开始。此时，释放内存中的蛇节点，重置游戏状态，如分数、蛇的位置等，然后重新进入游戏逻辑循环。

在后面的部分中，我们将对软件设计的各个模块进行详细介绍，核心代码见附录5.2。

3.2　界面绘制模块

界面绘制模块主要负责游戏界面的显示与更新，包括游戏开始界面、游戏进行中的界面以及游戏结束界面。该模块通过调用 LCD 相关函数实现图形的绘制与刷新，确保游戏信息能够及时、准确地反馈给玩家。单片机自带的 2.8 寸显示屏分辨率为 320×240。我们将之分割为了上方 20×240 的文字与下方 300×240 的游戏界面。

文字显示由 `draw_title()` 函数实现，它会打印当前的成绩、当前血量与历史最高成绩。

游戏部分以 10×10 为一格，共 30×24 格。格子通过 `draw_square_snake()` 与

draw_square() 两个函数绘制；他们是 ILI9341 显示控制器驱动的包装，以更统一简单地绘制界面。

蛇头的眼睛绘制由 draw_head_eyes() 函数实现。它通过硬编码不同方向的眼睛位置信息，在不同方向时绘制对应眼睛图像。

核心代码见附录5.2.2。

3.3　食物生成模块

食物模块主要负责游戏中食物的生成与管理。食物的生成遵循随机性原则，确保每次游戏开始时食物位置的随机性。食物的类型包括普通食物、加速食物和减速食物，分别对应不同的游戏效果。定义食物类型的枚举类型 food_type 如下：

```
1  enum food_type {
2      food_normal, // 普通食物
3      food_double_score, // 双倍果实
4      food_undead, // 生命水晶
5      food_half_length, // 缩短药水
6      food_speed_low, // 减速药水
7  };
```

食物的生成由 gen_random_pos() 与 gen_food() 两个函数实现。其中，gen_random_pos() 函数会随机生成边界一格范围之外的游戏界面内的格子坐标，若该处已经被贪吃蛇占据，则会重新生成，从而实现食物位置的随机性与合理性；gen_food() 函数在随机生成食物位置之后，会按照 6:1:1:1 的概率比例随机生成食物类型，并绘制对应食物。核心代码见附录5.2.3。

3.4　状态更新模块

游戏的状态更新模块由蛇位置更新、食物判定、死亡判定、方向更新等部分构成。其中，蛇的位置更新通过 snake_step() 函数实现。它会根据当前蛇头的位置和方向，计算出新的蛇头位置，具体代码如下：

```
1  static void snake_step(struct snake* new, struct snake* old, enum
       direction dir) {
2      new->x = old->x;
3      new->y = old->y;
4      switch (dir) {
5      case RIGHT: new->x += 1; break;
```

```
6       case LEFT:  new->x -= 1; break;
7       case UP:    new->y -= 1; break;
8       case DOWN:  new->y += 1; break;
9       }
10  }
```

食物判定通过比较食物与更新后蛇头所在位置是否相同来实现。如果相同，则视为贪吃蛇吃到了该食物，并根据对应的食物类型更新 score, highest_score, snake_lives, speed_factor 等信息，从而更新当前得分、历史最高分、蛇的血量和速度因子等。此外，食物类型的不同也会导致不同的效果，如双倍得分、增加血量、缩短蛇身长度或降低速度等，这些效果的实现也在该模块中完成。

死亡判定通过 judge_alive() 函数实现。它会检查更新后蛇头的位置是否与墙壁或蛇身其他部分重合，从而判定贪吃蛇是否死亡。如果判定死亡，则减少贪吃蛇的血量；若此时血量仍大于零，则复活贪吃蛇并将蛇头更新到随机位置，方向保持不变，形成“传送复活”的效果。

方向更新由定时任务调用。当判定按键按下时，由 button_down_handler() 函数更新方向；按下左键则逆时针旋转，按下右键则顺时针旋转。

核心代码见附录5.2.4。

3.5　帧更新延迟模块

为了更有趣的游戏体验，游戏的帧更新速度被设定为根据游戏进程的不同而变化，受 current_delay() 函数调控；它随分数升高而变快，同时会受到 speed_factor 的影响。整体速度被控制在初始速度的一倍到三倍之间。帧延迟的计算公式如代码所示。

```
1   static unsigned current_delay() {
2       // speedup: (3s + 3)/(s + 3)
3       unsigned sc = speed_factor * score / 0xfffff;
4       return (0x7ffff / 3)*(sc + 3)/(sc + 1);
5   }
```

3.6　按键与触摸检测模块

按键与触摸检测由定时器作为 Interrupt Handler 定时调用。其内部做了消抖处理，即多次检测均为按下时才会判定为按下，并触发对应更新代码。按键消抖检测算法见算

法3.1。该算法通过计数器记录连续相同状态的次数，当达到设定的阈值时，才认为按键状态稳定，并触发相应的事件处理函数。

算法 3.1 按键消抖检测算法

Data: 按键扫描函数 $ReadKey()$，消抖计数阈值 $N_{\text{threshold}}$
Result: 稳定的按键状态输出

初始化：$count \leftarrow 0$，$key_{\text{stable}} \leftarrow 0$，$key_{\text{previous}} \leftarrow 0$;
while 系统运行 **do**
    $key_{\text{current}} \leftarrow$ ReadKey(); // 读取当前按键状态
    if $key_{current} = key_{previous}$ **then**
        $count \leftarrow count + 1$; // 相同状态计数递增
    else
        $count \leftarrow 0$; // 状态改变，重置计数
    end
    if $count \geq N_{threshold}$ **then**
        if $key_{current} \neq key_{stable}$ **then**
            $key_{\text{stable}} \leftarrow key_{\text{current}}$; // 更新稳定状态
            if $key_{stable} = 1$ **then**
                触发按键按下事件处理函数; // 执行按键响应代码
            end
        end
        $count \leftarrow N_{\text{threshold}}$; // 防止计数溢出
    end
    $key_{\text{previous}} \leftarrow key_{\text{current}}$;
    延时 T_{scan} ms; // 扫描周期延时
end

3.7　流光彩色 LED 灯模块

为了美观，游戏还实现了流光彩色 LED 灯，它具有平滑的 RGB 颜色渐变效果，由 `flow_color_led()` 函数实现，该函数使用位图 `light_bitmap` 数组实现 16 级亮度的 PWM 控制，数组中每个值代表在 16 个时间片中 LED 的开关模式，通过控制占空比模拟不同亮度，从而利用 8 色 LED 灯实现了 12bit 色深的颜色，平滑了颜色转换。同时，在当前颜色 `current` 和上一个颜色 `last` 之间进行线性插值，实现平滑的颜色过渡。当一次颜色渐变完成后，使用随机数生成新的目标颜色，每个 RGB 分量的亮度范围是 0 ~ 16 级，具体实现代码见附录5.2.5。

3.8　本章小结

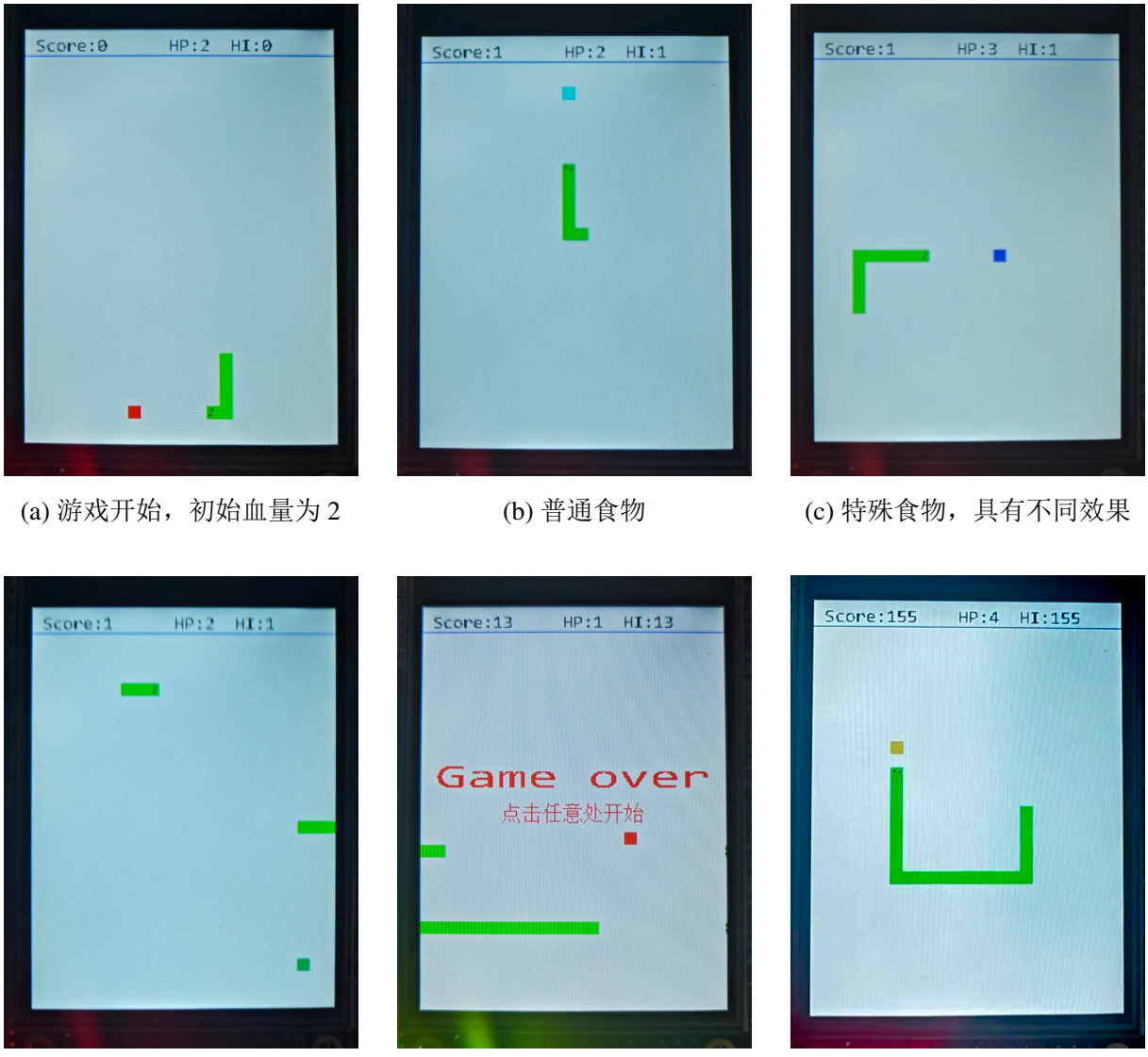
本章介绍了贪吃蛇游戏的核心软件设计与实现。通过分层架构，我们将游戏逻辑与硬件控制解耦，确保代码的可维护性与可扩展性。界面绘制模块负责游戏界面的显示与更新，食物生成模块实现了食物的随机生成与管理，状态更新模块处理了游戏逻辑的核心部分，包括蛇的位置更新、食物判定、死亡判定等。帧更新延迟模块控制了游戏的速度与流畅度，按键与触摸检测模块实现了用户输入的响应，流光彩色 LED 灯模块则为游戏增添了视觉效果。通过这些模块的协同工作，我们成功实现了一个完整的贪吃蛇游戏。

第 4 章　项目总结

4.1　项目成果

4.1.1　项目成果展示

本项目的实机运行效果展示见图4.1：



(a) 游戏开始，初始血量为 2　(b) 普通食物　(c) 特殊食物，具有不同效果

(d) 死亡扣血并传送　(e) 游戏结束　(f) 开发测试人员刷新纪录

图 4.1　贪吃蛇游戏实机运行效果展示

贪吃蛇游戏在 STM32F103VET6 野火指南者开发板上成功运行，游戏界面清晰，操作流畅。玩家可以通过按键控制蛇头的转向，游戏过程中会实时更新得分与最高记录，场上会随机生成食物供玩家吃掉，增加游戏的趣味性与挑战性。游戏结束时会显示“Game Over”界面，并提供重新开始的选项。LED 指示灯根据游戏状态变化，提供视觉反馈，增强游戏体验。

本项目代码托管在 GitHub 上，仓库地址为：https://github.com/youyeyejie/NIS2332_GreedySnake_STM32.git。

4.1.2　项目功能实现

本项目成功实现了基于 STM32F103VET6 野火指南者开发板的贪吃蛇游戏，具备完整的游戏逻辑与交互功能。游戏界面通过 LCD 显示屏显示，支持触摸操作与按键控制，能够实时响应玩家输入。游戏包括以下核心功能：

- **游戏初始化**：创建游戏窗口，设置初始蛇身位置与方向，生成初始食物。
- **实时交互**：支持按键控制蛇头转向，触摸屏可用于暂停游戏。
- **碰撞检测**：实现边界检测与自碰撞检测，当蛇头触碰到边界或自身时，游戏结束并显示“Game Over”界面。
- **道具系统**：引入多样化道具（如双倍果实、生命水晶等），增加游戏策略性。
- **分数系统**：实时更新得分与最高记录，并在界面上显示。
- **动态难度**：随着得分增加，贪吃蛇的移动速度逐渐加快，提升游戏挑战性。
- **LED 指示灯**：通过 RGB LED 指示灯显示游戏状态（全彩 RGB 渐变灯效运行、红色结束、蓝色暂停）。

4.1.3　项目代码工程化

本项目采用四层软件架构，显著提升系统可维护性与扩展性：

- **硬件驱动层**：封装 LCD、GPIO、TIM 等外设操作，提供 `LCD_WritePoint()`、`KEY_Scan()` 等标准化接口，隔离硬件细节。
- **应用层**：以 main.c 为入口，协调 snake.c（游戏逻辑）、ui.c（界面渲染）等模块，实现状态机管理（欢迎界面 → 游戏运行 → 暂停 → 结束）。
- **中断处理层**：在 stm32f10x_it.c 中实现定时器中断（TIM2 控制游戏刷新、TIM3 更新 LED PWM）与预留外部中断接口，通过优先级配置（TIM2 抢占优先级 1）确保关键任务实时响应。
- **配置层**：通过 stm32f10x_conf.h 宏定义管理外设时钟、中断使能等参数，例如开启 GPIOE 时钟以支持按键功能，选择外部晶振经 PLL 倍频至 72MHz 系统时钟。

代码工程化方面，在本次项目中我们使用 Keil5 与 STM32CubeMX 混合开发，标准库 V3.5 实现底层驱动，C 语言编写核心逻辑，代码量约 2000 行，Flash 占用 420KB（总容量 512KB），SRAM 占用 52KB（总容量 64KB）。通过 Git 进行版本管理，建立 develop 分支与 main 分支，实现代码迭代与团队协作。项目文档化程度较高，包含详细的代码注释、模块说明与使用手册，便于后续维护与二次开发。

4.2　心得体会

4.2.1　项目收获

本次项目的开发过程让我们深刻体会到嵌入式系统开发的复杂性与挑战性。通过将贪吃蛇游戏移植到 STM32F103VET6 平台，我们不仅巩固了对嵌入式系统架构的理解，还提升了在有限资源下进行高效编程的能力。特别是在硬件驱动层的设计中，我们学会了如何将底层硬件操作抽象为高层接口，使得应用层代码更加简洁易懂。此外，团队协作让我们认识到跨领域合作的重要性，硬件与软件工程师之间的紧密配合是项目成功的关键。

此外，项目中我们还学习了如何在实际开发中应用版本控制工具（如 Git），通过分支管理与代码合并，确保团队成员之间的协作高效有序。代码的工程化管理使得我们能够更好地追踪问题、回溯历史版本，并在多人协作时减少冲突，提高开发效率。

在项目开发的过程中，我们也遇到了许多挑战。比如在实现触摸屏输入时，由于响应速度较慢，我们不得不放弃了触摸控制方向的设计，转而使用按键控制，这让我们意识到在嵌入式开发中，硬件特性对软件设计的影响是多么深远。此外，在游戏逻辑的实现中，随着游戏功能的丰富，我们需要不断调整碰撞检测算法，推翻重构上一版本的代码，这让我们深刻体会到软件开发中的迭代与优化过程是多么重要。

当然，项目的成功离不开团队成员之间的密切合作与沟通。每个人都在自己的领域发挥了重要作用，从硬件设计、驱动开发，到游戏逻辑实现、界面设计，大家各司其职，相互支持，共同克服了一个又一个技术难题。通过这次项目，我们不仅提升了个人技能，更增强了团队协作能力，为未来的嵌入式系统开发打下了坚实的基础。

实际上，基于 STM32F103VET6 的贪吃蛇游戏原本只是我们小组的备选项目，我们原计划开发一个基于 STM32F103VET6 的电脑助手，实现电脑资源如 CPU 、GPU 、内存、硬盘等的监控，以及音量、亮度等的参数的快捷控制和应用的快捷启动。然而，我们在开发中遇到了许多挑战，其中单片机与上位机通信及数据解析等问题甚至一度让我们束手无策。经过多次讨论与尝试，我们最终决定从零开始，转向贪吃蛇游戏的开发，这不仅让我们重新审视了嵌入式系统开发的复杂性，也让我们意识到在面对技术挑战时，

灵活调整思路与方向的重要性。贪吃蛇游戏的成功实现，既是对我们团队协作能力的考验，也是对我们技术水平的全面提升。

4.2.2　改进方向

在本项目的基础上，我们为未来留下了进一步改进与拓展的空间。以下是一些可能的改进方向：

- **交互体验升级**：将当前的电阻触摸液晶屏升级为电容触摸屏，提高响应速度与跟手性，真正实现触摸控制方向以及更多交互功能（如双指缩放、滑动手势等），提升游戏体验。
- **代码可移植性增强**：将标准库迁移至 HAL 库，利用 STM32CubeMX 图形化配置工具生成初始化代码，便于适配 STM32F4/F7 等高端芯片，支持更高分辨率屏幕（如 800× 480）。
- **联网功能拓展**：通过 USART 接口集成 ESP8266 WiFi 模块，实现与上位机的交互，进而实现在线排行榜、多人对战等功能，提升游戏的社交性与竞争性。
- **游戏内容丰富化**：增加更多游戏元素，如不同类型的食物（如加速果实、减速果实）、障碍物、敌对蛇等，提升游戏的多样性与趣味性；增加“无尽模式”、“时间挑战”等多种游戏模式，丰富游戏玩法。
- **性能优化**：针对游戏逻辑与渲染进行进一步优化，减少 CPU 占用率，提升帧率表现，确保在高分辨率下依然流畅运行。

4.3　本章小结

本次课程设计是对嵌入式系统开发全流程的沉浸式实践，团队成员在需求分析、方案设计、调试优化中积累了宝贵经验，每一次挑战都伴随着技术能力的提升。我们在实际操作中深入理解了嵌入式系统的工作原理与开发流程，通过将理论知识应用于实际项目，我们不仅提升了编程能力，还增强了团队协作与问题解决能力。项目成果不仅是一款功能完备的贪吃蛇游戏，更是对“严谨、协作、创新”工程思维的培养。尽管在开发过程中遇到了许多挑战，但最终的成果证明了我们的努力与坚持是值得的。未来，我们期待在此基础上继续探索嵌入式技术在教育、物联网等领域的更多可能性，创造出更具创新性与实用性的嵌入式应用程序，将理论知识转化为推动技术进步的动力。

第 5 章　附　录

5.1　STM32F103VET6 芯片规格参数

表 5.1　STM32F103VET6 芯片主要规格参数

项目	参数
产品种类	ARM 微控制器 MCU
系列	STM32F103VE
核心	ARM Cortex-M3
程序存储器大小	512 KB Flash
数据 RAM 大小	64 KB SRAM
数据总线宽度	32 位
ADC 分辨率	12 位
最大时钟频率	72 MHz
工作电源电压	2V ~ 3.6V
工作温度范围	-40°C ~ 85°C
接口类型	CAN, I2C, SPI, USART, USB
A/D 转换器数量	3
ADC 通道数	16
D/A 转换器数量	2
I2C 通道数	2
SPI 通道数	3
UART 通道数	2
USART 通道数	3
PWM 通道数	16
计时器/计数器数量	8
I/O 数量	80
可编程 I/O 数量	80
振荡器类型	内部
封装	LQFP-100
封装尺寸	14.2mm × 14.2mm × 1.45mm
安装风格	贴片

数据来源：STM32F103VET6 官方数据手册

5.2 贪吃蛇游戏核心代码

5.2.1 基础类型定义

```
enum direction {
    LEFT = 0,
    DOWN,
    RIGHT,
    UP,
} __attribute__((__packed__));

typedef struct snake {
    int x, y;
    union {
        struct snake *next; // as snake
        enum food_type food_type; // as food
    };
} snake;

typedef struct {
    int x, y;
    enum direction dir;
} snake_head;
```

5.2.2 界面绘制模块

```
static void draw_square_snake(snake* s) {
    ILI9341_DrawRectangle(s->x*10,s->y*10+20,10,10,1);
}

static void draw_square(uint16_t color, int x, int y) {
    LCD_SetTextColor(color);
    ILI9341_DrawRectangle(x*10,y*10+20,10,10,1);
}

static void draw_head_eyes() {
    int hx = head->x*10;
    int hy = head->y*10+20;
    const uint8_t o[4][4] = {
        {3,2,2,6}, // left = 0
        {2,5,6,6}, // down = 1
        {6,2,5,6}, // right = 2
        {2,2,6,3}, // up = 3
    }
    int i;
    const uint8_t oo[4] = o[status];
    LCD_SetColors(CL_BLACK,CL_WHITE);
    ILI9341_DrawRectangle(hx+oo[0],hy+oo[1],2,2,1);
    ILI9341_DrawRectangle(hx+oo[2],hy+oo[3],2,2,1);
}
```

5.2.3 食物生成模块

```
1  static void gen_random_pos(int* px, int* py) {
2      int x,y;
3      snake* p;
4  redo:
5      srand(ADC_ConvertedValue);
6      x = rand()%22 + 1; // 1-22
7      y = rand()%28 + 1; // 1-28
8      p = head;
9      while(p->next) {
10         if(x==p->x && y==p->y) goto redo;
11         p = p->next;
12     }
13     *px = x; *py = y;
14 }
15
16 static void gen_food(void) {
17     uint16_t food_color;
18     gen_random_pos(&food1.x, &food1.y);
19     switch (rand()%10) {
20         case 0: food1.food_type = food_double_score; food_color =
       CL_ORANGE; break;
21         case 1: food1.food_type = food_undead; food_color =
       CL_GREEN2; break;
22         case 2: food1.food_type = food_half_length; food_color =
       CL_BLUE; break;
23         case 3: food1.food_type = food_speed_low; food_color =
       CL_CYAN; break;
24         default: food1.food_type = food_normal; food_color =
       CL_RED; break;
25     }
26     draw_square(food_color, food1.x, food1.y);
27 }
```

5.2.4 状态更新模块

```
static void consume_food() {
    switch (food1.food_type) {
        case food_normal: score += 1; break;
        case food_double_score: score = score==0 ? 1 : score*2;
   break;
        case food_half_length: snake_half_length(); break;
        case food_speed_low: speed_factor = speed_factor*4/5;
   break;
        case food_undead: snake_lives += 1;
    }
    if(score > hiscore) hiscore = score;
}

void judge_alive() {
    snake *q = head->next;  // hit boundary
    if(head->x<0||head->y<0||head->x>=24||head->y>=30)
        die = 1;
    while(die != 1 && q != NULL) {  // hit self
        if(q->x == head->x && head->y == q->y)
            die = 1;
        q = q->next;
    };
    if(die == 1 && snake_lives != 1) {  // revive
        die = 0; snake_lives --;
        gen_random_pos(&head->x, &head->y);
    }
}

void button_down_handler(enum button b) {
    switch (b) {
    // counter clock wise
    case button_left:  status = (status+1)%4; break;
    // clock wise
    case button_right: status = (status+3)%4; break;
    }
}
```

5.2.5 流光彩色 LED 灯模块

```
static void flow_color_led() {
    // color: 4 bit each
    static uint8_t current[3] = {16,16,6};
    static uint8_t last[3] = {16,16,16};
    const unsigned max_count = 25;
    static unsigned count = max_count, frame_count = 15;
    const uint16_t light_bitmap[17] = {
        0x0000, 0x8000, 0x8080, 0x8888, 0xA8A8, 0xAAAA,
        0xAAAE, 0xEEEE, 0xEEEF, 0xEFEF, 0xFFEF, 0xFFF7,
        0xFFFB, 0xFFFD, 0xFFFE, 0xFFFF, 0xFFFF
    };
    uint16_t r,g,b; unsigned tmp;
    r = ((uint16_t)current[0]*count + (uint16_t)last[0]*(max_count
    -count)) / max_count;
    g = ((uint16_t)current[1]*count + (uint16_t)last[1]*(max_count
    -count)) / max_count;
    b = ((uint16_t)current[2]*count + (uint16_t)last[2]*(max_count
    -count)) / max_count;
    tmp = 1<<frame_count;
    if(light_bitmap[r] & tmp) {LED1_ON} else {LED1_OFF}
    if(light_bitmap[g] & tmp) {LED2_ON} else {LED2_OFF}
    if(light_bitmap[b] & tmp) {LED3_ON} else {LED3_OFF}
    if (++frame_count >= 16) {
        frame_count = 0;
        if(++count >= max_count) {
            count = 0;
            last[0] = current[0];
            last[1] = current[1];
            last[2] = current[2];
            tmp = rand();
            current[0] = tmp%17; tmp/=17;
            current[1] = tmp%17; tmp/=17;
            current[2] = tmp%17;
        }
    }
}
```

参考文献

[1] 野火电子. KEIL5 安装与使用指南[Z]. 随书配套资料.

[2] 意法半导体（STMicroelectronics）. STM32F10X-中文参考手册[Z]. 在线文档. 文档编号：RM0008.

[3] ARM 公司. Cortex-M3 权威指南[Z]. 在线文档.

[4] 野火电子. STM32 库开发实战指南——基于野火指南者开发板[M]. 东莞: 野火电子, 2024.