**嵌入式技术课程设计报告**

题 目： 基于嵌入式Linux的俄罗斯方块游戏设计

组 别： 第一组

团 队 成 员： 学号 2022118142 姓名 庹福利

学号 2022129145 姓名 田福敏

学号 2022117460 姓名 朱晓龙

指 导 教 师： 徐德刚

专 业： 物联网工程

班 级： 物联网2212

学 院： 计算机工程学院

（课程设计时间：2025年 6月9日——2025年 6月20日）

摘 要

本报告旨在介绍使用GEC-6818开发板基于Linux操作系统实现的俄罗斯方块游戏设计。该项目的主要目的是展示如何利用GEC-6818开发板的高性能特性，结合Linux系统的多任务处理能力，来设计和开发一款经典的俄罗斯方块游戏。在项目中，我们对GEC-6818开发板进行了详细的研究，了解其硬件资源和软件支持，以确保它能够满足游戏开发的需求。接着，我们在开发板上成功安装了Linux操作系统，并配置了必要的开发环境，包括编译器、编辑器和其他工具。通过这个项目，我们积累了宝贵的经验，并为未来类似项目的开发提供了有益的参考，也为嵌入式系统开发领域的研究和实践提供了有益的经验和启示。

关键词：GEC-6818；linux；嵌入式

目录

[一、设计任务 4](#_Toc168043454)

[二、设计方案 5](#_Toc168043455)

[2.1任务分析 5](#_Toc168043456)

[2.2方案设计 5](#_Toc168043457)

[三、系统总体设计 7](#_Toc168043458)

[四、系统模块 8](#_Toc168043459)

[五、性能分析 12](#_Toc168043460)

[六、心得体会 13](#_Toc168043461)

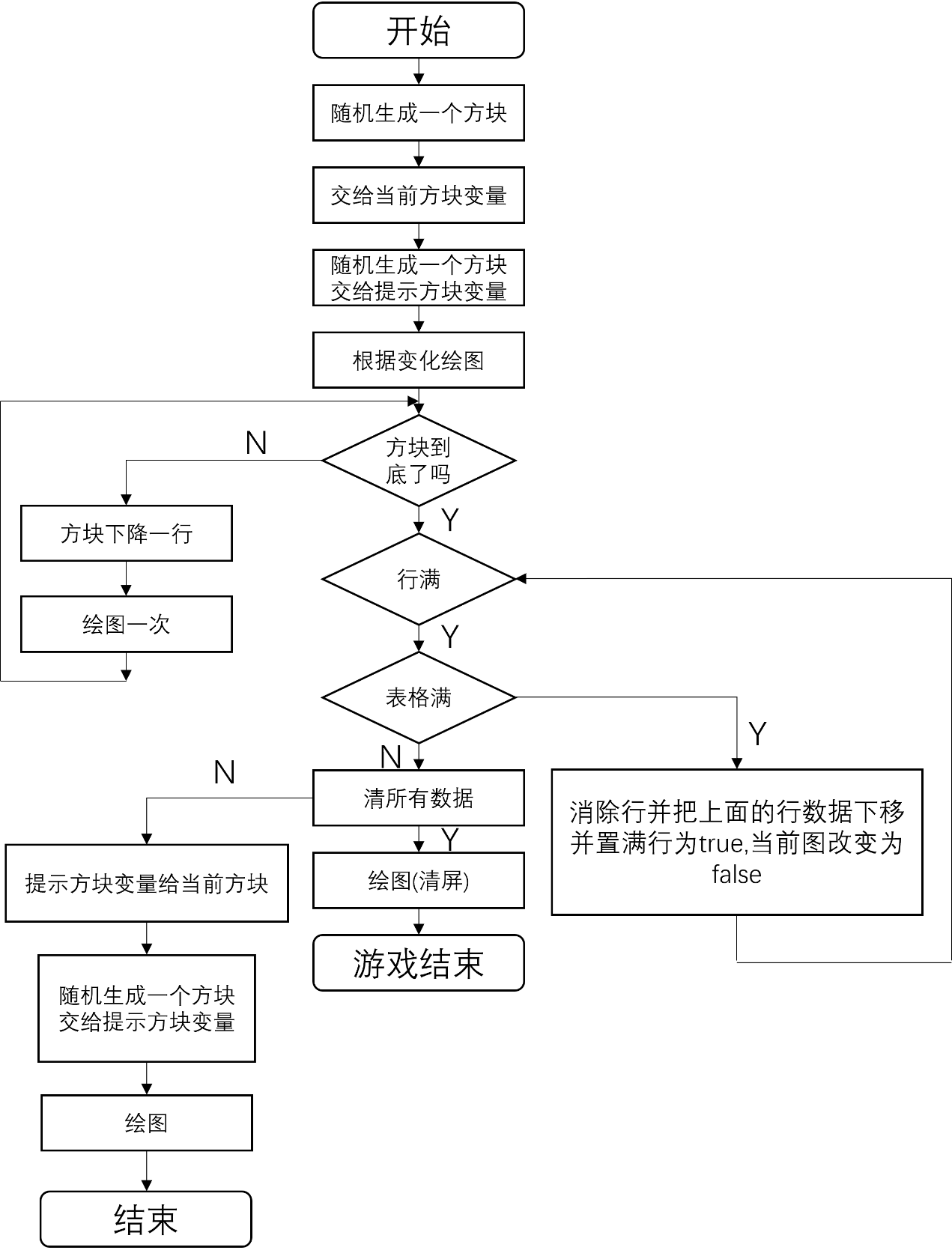
[参考文献 14](#_Toc168043462)

[附录 15](#_Toc168043463)

# 1 设计任务

在嵌入式系统开发领域，基于嵌入式Linux的俄罗斯方块游戏设计是一个具有挑战性且富有趣味性的项目。本设计任务旨在通过GEC-6818开发板实现一款功能完备、运行流畅的俄罗斯方块游戏。GEC-6818开发板以其卓越的性能和丰富的硬件资源，为嵌入式系统开发提供了理想的实验平台。

俄罗斯方块游戏作为一款经典的益智类游戏，其核心在于图形的生成、移动、旋转以及消除逻辑的实现。在嵌入式Linux环境下，需要对GEC-6818开发板的硬件资源进行精准调用，同时结合Linux系统的多任务处理能力，以确保游戏的高效运行。具体而言，游戏设计需实现以下功能：利用LinuxC编程语言实现俄罗斯方块的主要逻辑功能，确保游戏代码能够在GEC-6818开发板上稳定运行，并支持玩家通过输入设备控制游戏内图形的实时变化，具体来说，左右箭头控制方块左右移动，上下箭头控制方块的旋转和快速下移。此外，还需设计一个直观且具有吸引力的用户界面，使玩家能够沉浸在游戏中。总体流程图如图1.1所示。



图表1.1 游戏流程图

为了实现上述功能，我们首先深入研究了GEC-6818开发板的硬件特性，包括其图形处理能力、输入输出接口以及内存管理机制。随后，我们在开发板上成功安装了Linux操作系统，并配置了必要的开发环境，包括编译器、编辑器和其他工具。通过这个研究和配置过程，我们为游戏的开发奠定了坚实的基础。

在游戏的设计过程中，我们还注重了对游戏性能的优化。通过合理分配内存资源、优化图形绘制算法以及采用高效的多线程技术，我们确保了游戏在嵌入式设备上的流畅运行。游戏程序采用了模块化设计，将图形绘制、游戏逻辑、用户输入处理以及计分系统等功能模块分开实现，从而提高了代码的可读性、可维护性和可扩展性。

在游戏开发的每个阶段，我们进行了严格的测试和调试，以确保游戏的稳定性和可靠性。通过模拟玩家的各种操作，我们对游戏的响应速度、图形渲染效果以及输入设备的兼容性进行了全面评估。这确保了游戏在实际使用中的稳定性和流畅性，为玩家提供良好的游戏体验。

# 2 设计方案

## 2.1 任务分析

要成功实现基于嵌入式 Linux 的俄罗斯方块游戏设计，需遵循以下详细步骤：

（1）搭建编译环境：首要任务是安装交叉编译工具链，它能确保代码在 6818 开发板上顺利编译与运行。目前，arm-linux-gnueabi-gcc是比较常用且适配性良好的交叉编译器，安装过程中需注意遵循官方文档指引，正确配置环境变量，以便后续调用编译器时能精准定位。

（2）游戏代码编写：采用C语言实现俄罗斯方块游戏，核心模块涵盖以下几个部分：

游戏界面绘制：鉴于资源限制与开发便捷性，运用字符画在终端呈现游戏界面是合理选择。需精心设计字符组合，勾勒出清晰的游戏区域边框，同时以不同字符或字符组合标识多样化的方块形状，让玩家能直观区分。

方块生成与移动：编写算法实现方块的随机生成，依据经典俄罗斯方块规则，共有七种基础形状，要确保每种形状生成概率均衡。对于方块移动，需精准处理上下左右及旋转操作，涉及坐标变换运算，保障移动过程流畅自然，无卡顿或异常偏移。

碰撞检测：这是游戏流畅性的关键保障。要实时监测方块与游戏区域左右边界、下边界以及已堆积方块的碰撞情况，一旦检测到碰撞，立即阻止方块的对应方向移动或下落，算法需高效以避免出现检测延迟或误差。

消除行：持续监测游戏区域各横向行，当某一行方块紧密排列无间隙时，迅速执行消除操作。消除过程要注意行内方块的移除动画效果，以及上方方块整体下落的过渡处理，使得视觉效果连贯，同时及时更新游戏区域数据结构，反映当前状态。

计分系统 ：依据玩家消除的行数给予相应得分奖励。可设定单行消除、双行同时消除、三行消行等阶梯式计分规则，激发玩家挑战高分欲望，计分过程要及时在界面更新分数显示。

将编写好的代码编译成适用于6818开发板的可执行文件。在终端中输入以下命令进行编译：

bash复制代码运行

arm-linux-gnueabi-gcc -o tetris tetris.c

编译期间，留意可能出现的警告与错误提示，针对性地回到代码中排查语法错误、逻辑漏洞或函数调用不当等问题，直至编译顺利通过。将编译好的可执行文件传输到6818开发板上，并在开发板上运行游戏。可以使用scp命令将编译好的可执行文件从宿主机传输至6818开发板指定目录。随后，运用ssh工具连接到开发板，进入文件所在目录，执行 “./tetris” 命令启动游戏，在开发板终端开启俄罗斯方块游戏之旅。

控制游戏内图形的变化。为进一步提升游戏趣味性与个性化，可深入代码，修改相关参数达成图形变化效果。例如，调整方块形状参数，创造出特殊造型的方块；或者更改颜色值（尽管在字符界面呈现颜色受限于终端支持，但可预先做好代码适配，方便后续拓展到支持图形界面与彩色显示的平台），为游戏增添视觉新鲜感。

## 2.2 方案设计

### 2.2.1 硬件平台选择与准备

在本次设计中，我们选择了GEC6818开发板作为硬件平台。GEC6818开发板以其卓越的性能和丰富的功能，成为了嵌入式系统开发领域的理想选择。它基于先进的ARM架构，具备高效的处理能力和低功耗特性，这不仅确保了游戏的流畅运行，还延长了设备的使用寿命。此外，GEC6818开发板配备了强大的图形处理单元（GPU），能够高效地处理图形绘制和渲染任务，这对于俄罗斯方块游戏中各种形状方块的实时显示和动态更新至关重要。

开发板的输入输出接口丰富多样，包括GPIO、串口、USB等，为游戏的输入控制和外围设备的连接提供了极大的便利。例如，通过GPIO接口，我们可以轻松地连接按钮和传感器，实现玩家对游戏的精准控制；而串口则可以用于调试信息的输出，帮助开发人员及时发现和解决问题。这些丰富的接口选项为游戏的开发和扩展提供了广阔的空间。

在开始开发之前，我们对GEC6818开发板进行了全面的检查和测试，以确保所有硬件资源都能正常工作。这包括对开发板的电源管理、内存、存储以及各种接口的测试。通过这些准备工作，我们为后续的软件开发和游戏实现奠定了坚实的基础。

### 2.2.2 软件平台选择与准备

在软件平台的搭建过程中，我们首先选择了适配GEC6818开发板的Linux发行版。经过仔细评估和对比，我们最终选择了Ubuntu作为操作系统。Ubuntu以其稳定性和强大的社区支持而闻名，它提供了丰富的软件包和开发工具，能够极大地简化开发流程。

在GEC6818开发板上安装Ubuntu操作系统并非简单的复制粘贴操作，而是需要经过一系列的配置和优化步骤。我们首先下载了适合开发板硬件配置的Ubuntu镜像文件，并使用专用的工具将其烧录到开发板的存储设备中。在安装过程中，我们特别注意了文件系统的配置，选择了适合嵌入式系统的轻量级文件系统，以减少存储空间的占用并提高系统的启动速度。

此外，后续还可以对操作系统的内核进行定制，移除不必要的驱动和服务，以减少系统资源的消耗。通过这些优化措施，我们成功地在GEC6818开发板上搭建了一个稳定、高效的Linux运行环境，为游戏的开发提供了坚实的基础。

在操作系统的支持下，进一步搭建了适用于Linux的C语言开发环境。安装gcc编译器、make构建工具以及其他必要的开发库和调试工具。

gcc编译器作为Linux平台上最常用的C语言编译器之一，以其强大的功能和良好的兼容性而受到广泛认可。我们通过配置gcc的编译选项，使其能够生成针对ARM架构优化的可执行文件，从而提高游戏程序的运行效率。

make构建工具则用于简化项目的编译和构建过程。我们编写了详细的Makefile文件，定义了项目的源文件、头文件、编译选项以及生成目标等信息。通过make命令，开发者可以一键完成项目的编译和链接工作，大大提高了开发效率。

除了编译和构建工具，还可以安装一系列调试工具，如gdb调试器和valgrind内存检测工具。gdb调试器可以帮助开发者在程序运行过程中设置断点、查看变量值、分析调用栈等，从而快速定位和修复问题。valgrind工具则用于检测程序中的内存泄漏和非法访问等问题，确保程序的稳定性和可靠性。为了提高代码的可读性和可维护性，可以考虑引入代码编辑器和版本控制系统。代码编辑器提供了语法高亮、代码补全、自动格式化等功能，帮助开发者更高效地编写和编辑代码。版本控制系统则用于跟踪项目的开发历史，管理代码的变更和分支，确保团队协作的顺利进行。

# 3 系统总体设计

## 3.1 俄罗斯方块游戏实现

（1）游戏逻辑设计

游戏逻辑是俄罗斯方块游戏的核心，它决定了游戏的玩法和体验。在本项目中，我们对游戏逻辑进行了精心设计和实现。

我们首先定义了游戏中的基本元素，包括不同形状的方块（如I形、L形、T形、Z形等）和游戏区域。每个方块由四个小方格组成，具有独特的形状和颜色。游戏区域是一个固定的矩形网格，方块在其中下落、移动和旋转。

游戏逻辑主要包括以下几个方面：

1.方块生成与下落：游戏开始时，随机生成一个方块并将其放置在游戏区域的顶部中央位置。方块按照预设的速度自动向下移动，玩家可以通过输入指令控制方块的移动速度和方向。

2.方块移动与旋转：玩家可以通过方向键控制方块在游戏区域内左右移动和旋转。我们实现了方块的边界检测逻辑，确保方块不会移出游戏区域或与其他已固定的方块发生碰撞。当方块旋转时，还考虑了旋转后的边界检查，以防止方块超出游戏区域或重叠。

3.方块消除与计分：当方块下落到游戏区域底部或与其他固定方块接触后，会检查当前行是否被完全填满。如果玩家成功消除一行或多行，将根据消除的行数获得相应的分数。同时设计了一套合理的计分规则，鼓励玩家尽可能多地消除行以获得更高的分数。

4.游戏结束判断：当新生成的方块无法在游戏区域内放置时，判断游戏结束。此时，游戏会显示玩家的最终得分，并提供重新开始的选项。

为了提高游戏的可玩性，还可以引入了难度递增机制。随着游戏时间的推移或玩家得分的增加，方块的下落速度会逐渐加快，对玩家的反应速度和操作技巧提出更高的要求。

（2）图形绘制与界面设计

在嵌入式Linux环境下，利用GEC6818开发板的图形处理能力和Linux图形库实现了游戏的图形绘制和界面设计。

采用双缓冲技术来绘制游戏画面，有效避免了屏幕刷新时可能出现的闪烁问题。双缓冲技术通过在内存中维护两个缓冲区，一个用于显示当前画面，另一个用于绘制下一帧画面。当新的一帧画面绘制完成后，通过交换两个缓冲区的角色，实现画面的快速切换，从而为玩家提供流畅、舒适的视觉体验。

游戏界面设计简洁直观，主要包括以下几个部分：

1.游戏区域显示：在屏幕中央显示游戏区域，方块在其中下落、移动和旋转。每个方块使用不同的颜色进行区分，便于玩家识别。

2.下一个方块提示：在游戏区域右侧显示下一个即将出现的方块的预览，帮助玩家提前规划操作策略。

3.分数显示：在屏幕顶部显示玩家的当前得分，激励玩家不断挑战更高分数。

4.游戏控制按钮：在屏幕底部或侧面设置游戏控制按钮，如开始、暂停、重新开始等，方便玩家操作。

为了增强游戏的视觉效果，还可以添加了一些动画效果，如方块消除时的爆炸效果和得分增加时的闪光效果。这些动画效果不仅增加了游戏的趣味性，还为玩家提供了即时的反馈，提升了游戏的沉浸感。

（3）输入控制与交互设计

在游戏控制方面，要充分利用GEC6818开发板上的IO接口，如GPIO、串口等，实现了多样化的输入控制方式。

设计通过GPIO接口连接按钮的输入控制系统。玩家可以使用四个方向按钮控制方块的移动和旋转，以及一个暂停按钮来暂停或继续游戏。在实现过程中，对输入信号进行了滤波和去抖处理，以确保输入的准确性和稳定性。

此外，还尝试通过串口连接外部设备（如游戏手柄）进行游戏控制。这为玩家提供了更加灵活和舒适的操作方式，特别是对于长时间游戏而言。后续通过编写相应的串口通信程序，可以实现对手柄输入信号的解析和转换，使其能够与游戏逻辑无缝对接。同时，为了提高游戏的兼容性和可扩展性，也可以设计一个输入抽象层，将具体的输入设备与游戏逻辑分离。这样，玩家可以根据自己的喜好选择不同的输入设备，而无需修改游戏的核心代码。这也为后续增加新的输入设备支持提供了便利。

（4）多线程与并发处理

为了充分发挥嵌入式Linux系统的多任务处理能力，在游戏实现中可以采用多线程技术。

创建以下几个主要线程：

1.游戏主线程：负责游戏的整体流程控制，包括方块的生成、下落、移动、旋转以及碰撞检测等核心逻辑。游戏主线程按照固定的时间间隔更新游戏状态，并触发相应的图形绘制和界面更新操作。

2.输入处理线程：专门负责监听和处理玩家的输入事件。该线程实时监测输入设备的状态变化，并将输入指令传递给游戏主线程进行处理。通过将输入处理与游戏逻辑更新分离到不同的线程，能够确保游戏对玩家输入的快速响应，即使在游戏逻辑较为复杂的情况下也能保持良好的交互体验。

3.图形绘制线程：负责游戏画面的绘制工作。该线程根据游戏主线程提供的数据，利用Linux图形库将游戏界面实时渲染到屏幕上。通过独立的图形绘制线程，能够实现游戏逻辑更新与图形绘制的并行执行，从而提高游戏的整体性能和流畅度。

在多线程实现过程中，需要特别注意线程之间的同步和通信问题。通过使用互斥锁、信号量等同步机制，我们确保了线程之间的数据一致性，避免了因并发访问导致的数据冲突和错误。

## 3.2 与开发板的交互

GEC6818开发板丰富的IO接口为游戏的输入控制提供了多种实现方式。

GPIO接口是实现游戏输入控制的基础。我们通过配置GPIO引脚的方向和电平状态，将外部按钮连接到开发板上。当玩家按下按钮时，GPIO引脚的电平状态会发生变化，我们编写了相应的中断服务程序来检测这些变化，并将其转换为游戏内的控制指令，如方块的移动、旋转或暂停游戏等操作。这种基于中断的输入处理方式能够及时响应玩家的操作，提高了游戏的实时性和交互性。

除了GPIO接口，还能利用开发板的串口实现与外部设备的通信。例如，连接一个游戏手柄通过串口与开发板进行数据传输。在开发板上，编写串口通信程序，用于接收和解析手柄发送的控制信号。通过这种方式，不仅可以扩展游戏的输入方式，还为玩家提供了更加丰富和灵活的操作体验。

在实现IO接口控制的过程中，我们遇到了一些挑战，例如如何处理多个输入设备的同时输入、如何提高输入信号的稳定性和可靠性等。通过查阅相关技术资料和反复实验，我们采用了合理的硬件连接方式和软件滤波算法，成功解决了这些问题，确保了游戏输入控制的准确性和稳定性。

显示输出是游戏体验的重要组成部分。我们通过GEC6818开发板连接的显示设备，将游戏界面实时输出到屏幕上。开发板支持多种显示接口，如HDMI、LVDS等，我们可以根据实际需求选择合适的显示设备进行连接。在本项目中，我们选择了一款支持HDMI接口的液晶显示器，将其与开发板的HDMI输出端口相连。通过配置Linux操作系统的显示驱动程序，我们成功地将游戏画面完整、清晰地显示在屏幕上。

在显示输出的实现过程中，我们注重了画面的质量和性能优化。我们选择了合适的分辨率和刷新率设置，在保证画面清晰度的同时，尽量减少对系统资源的占用。此外，我们还对图形绘制算法进行了优化，例如采用批量绘制技术、减少不必要的绘制操作等，以提高画面的更新速度和流畅度。

为了增强游戏的视觉效果，我们还在显示输出中添加了一些特效，如渐变过渡、阴影效果等。这些特效不仅提升了游戏的美观度，还为玩家提供了更加沉浸式的游戏体验。

## 3.3 调试与优化

在设计的最后阶段，调试与性能优化是确保游戏在GEC-6818开发板上流畅运行的关键环节。调试工作通过运行游戏程序并利用开发板上的调试工具，如gdb等，以及输出信息来识别和修复程序中的潜在错误。性能优化则着重于针对嵌入式系统资源有限的特点，对代码进行深入分析和优化，通过优化关键算法、改进资源分配和利用多线程技术等方法，提高游戏的性能和流畅度。

### 3.3.1调试

调试工作贯穿于游戏开发的整个过程。我们主要采用了以下几种调试方法：

1.使用调试工具：在开发板上运行游戏程序时，我们利用gdb调试器进行调试。通过设置断点、单步执行、查看变量值等操作，我们能够深入分析程序的运行状态，快速定位问题所在。例如，当游戏出现异常崩溃或逻辑错误时，我们通过gdb查看调用栈和相关变量的值，迅速找到了问题的根源并加以解决。

2.输出调试信息：在代码中添加了大量的调试信息输出语句，将程序的关键状态和运行信息输出到终端或日志文件中。这些信息包括方块的位置、游戏得分、线程状态等。通过分析这些输出信息，我们能够了解程序的执行流程和各个模块的工作情况，及时发现潜在的问题。例如，在游戏逻辑复杂的情况下，我们通过输出调试信息找到了方块消除逻辑中的一个边界条件处理错误，并进行了修正。

3.远程调试：由于嵌入式设备的硬件限制，有时直接在开发板上进行调试会受到诸多限制。因此，我们采用了远程调试的方式，将开发板与主机相连，通过主机的强大计算能力和丰富的调试工具进行辅助调试。例如，我们使用ssh工具将开发板与主机建立连接，然后在主机上运行调试器对开发板上的程序进行调试。这种方式不仅提高了调试的效率，还使我们能够更全面地分析程序的性能和资源占用情况。

### **3.3.2 性能优化**

针对嵌入式系统资源有限的特点，我们对游戏程序进行了全面的性能优化。

1.代码优化：我们对游戏的核心算法进行了深入分析和优化。例如，在图形绘制方面，我们采用了更高效的算法来减少计算量和内存消耗。我们优化了方块的绘制顺序，尽量减少对已固定方块的重复绘制，同时利用图形缓存技术避免了不必要的图形重绘操作。在游戏逻辑方面，我们改进了碰撞检测算法，通过减少不必要的碰撞检测计算，提高了游戏的运行效率。

2.内存管理优化：嵌入式系统的内存资源相对有限，因此我们对游戏的内存管理进行了优化。我们通过合理分配和回收内存，减少了内存泄漏和碎片化的问题。例如，我们采用了内存池技术，预先分配一定大小的内存块用于存储游戏对象，避免了频繁的内存分配和释放操作。同时，我们定期检查和清理不再使用的内存，确保内存资源的高效利用。

3.多线程优化：为了充分发挥多线程的优势，我们对线程的创建、调度和同步机制进行了优化。我们根据游戏的实际需求合理设置了线程的优先级，确保关键任务能够及时得到处理。同时，我们减少了线程之间的锁竞争，通过优化锁的粒度和使用方式，提高了线程的并发执行效率。例如，我们将一些非关键的计算任务分配到低优先级线程中，在不影响游戏主逻辑的前提下，充分利用了系统的空闲资源。

4.硬件加速利用：我们充分利用了GEC6818开发板的硬件加速能力，如GPU加速。在图形绘制方面，我们尽可能地利用GPU的并行计算能力，将一些复杂的图形计算任务交给GPU处理，从而减轻了CPU的负担，提高了整体性能。例如，我们采用了OpenGL ES图形库进行图形绘制，通过调用GPU的专用指令集，实现了高效、流畅的图形渲染效果。

通过对游戏程序的全面调试和性能优化，我们不仅解决了游戏在运行过程中出现的各种问题，还显著提高了游戏的性能和流畅度。这不仅提升了玩家的游戏体验，也为嵌入式Linux环境下其他类型游戏的开发提供了宝贵的经验和参考。通过这些细致的工作，我们成功地将俄罗斯方块游戏从设计阶段推进到了实现和优化阶段，使其在GEC-6818开发板上能够稳定、高效地运行。

# 4 系统模块

**1.用户界面模块**：负责创建游戏窗口、显示游戏画面、接收用户输入等。

**2.游戏逻辑模块**：负责处理游戏的主要逻辑，包括方块的生成、移动、旋转、消除等。可以根据游戏规则编写相应的算法。

（1）游戏初始化：创建游戏界面和游戏区域。初始化得分、速度和其他游戏参数。生成第一个方块并放置在游戏的起始位置。

（2）主循环：

检测用户输入：监听键盘或触摸事件，如左右移动、旋转方块等。

处理用户输入：根据用户输入执行相应的方块操作，如移动、旋转等。

方块下落：使当前活跃的方块自动下落一行，除非已经到达底部或者其他方块阻挡。

边界和碰撞检测：检查方块是否超出游戏区域边界，或者与其他方块发生碰撞。

消除满行：扫描游戏区域，检查并消除完整的行，提升上方所有行的方块。

更新得分：根据消除的行数更新玩家得分。

游戏结束判断：如果新生成的方块无法放置，则游戏结束。

（3）方块生成：当前方块到达底部或被消除时，生成新的方块。新方块随机选择类型，并放置在游戏区域的顶部。

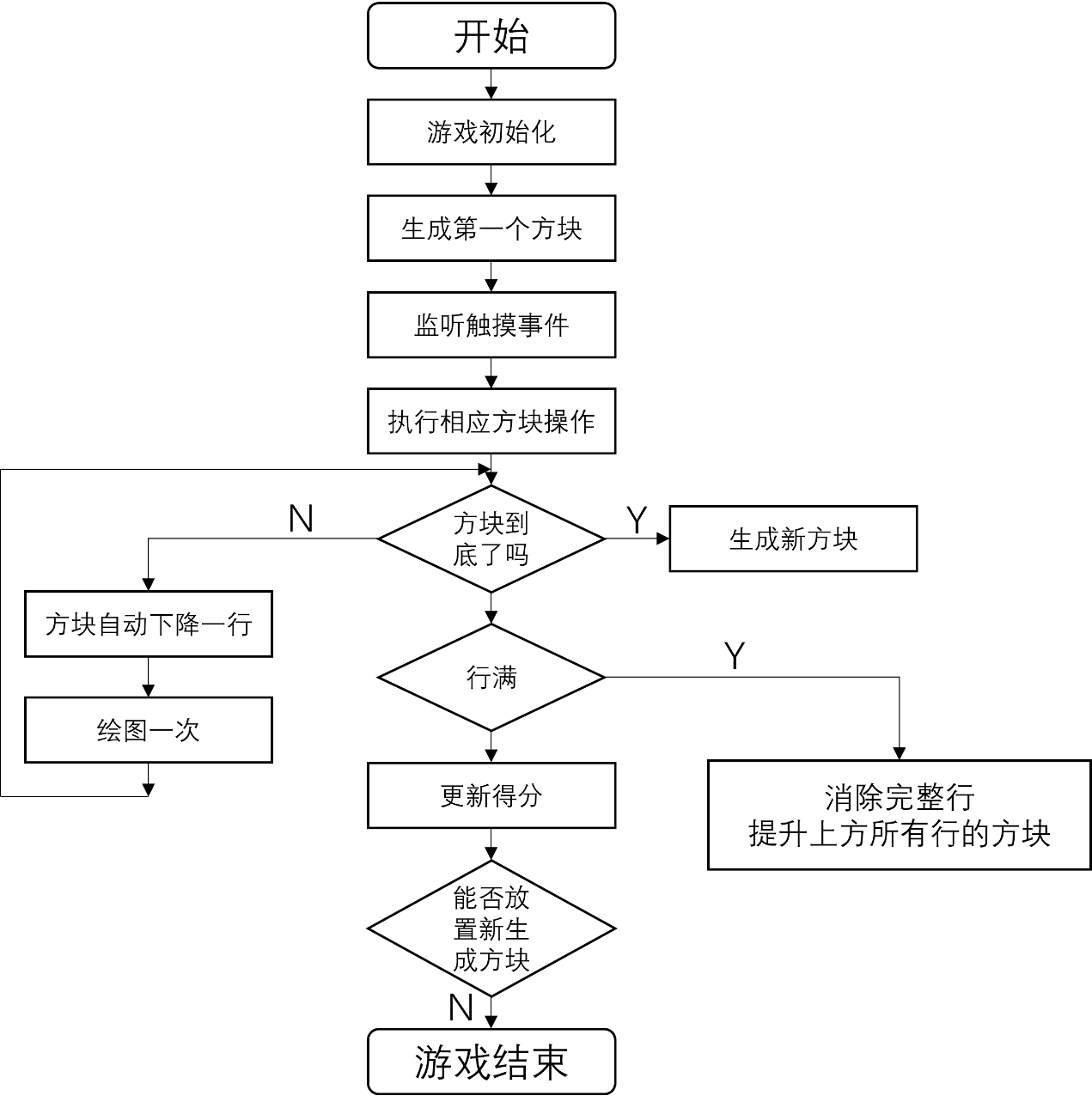


图4.2 游戏逻辑模块流程图

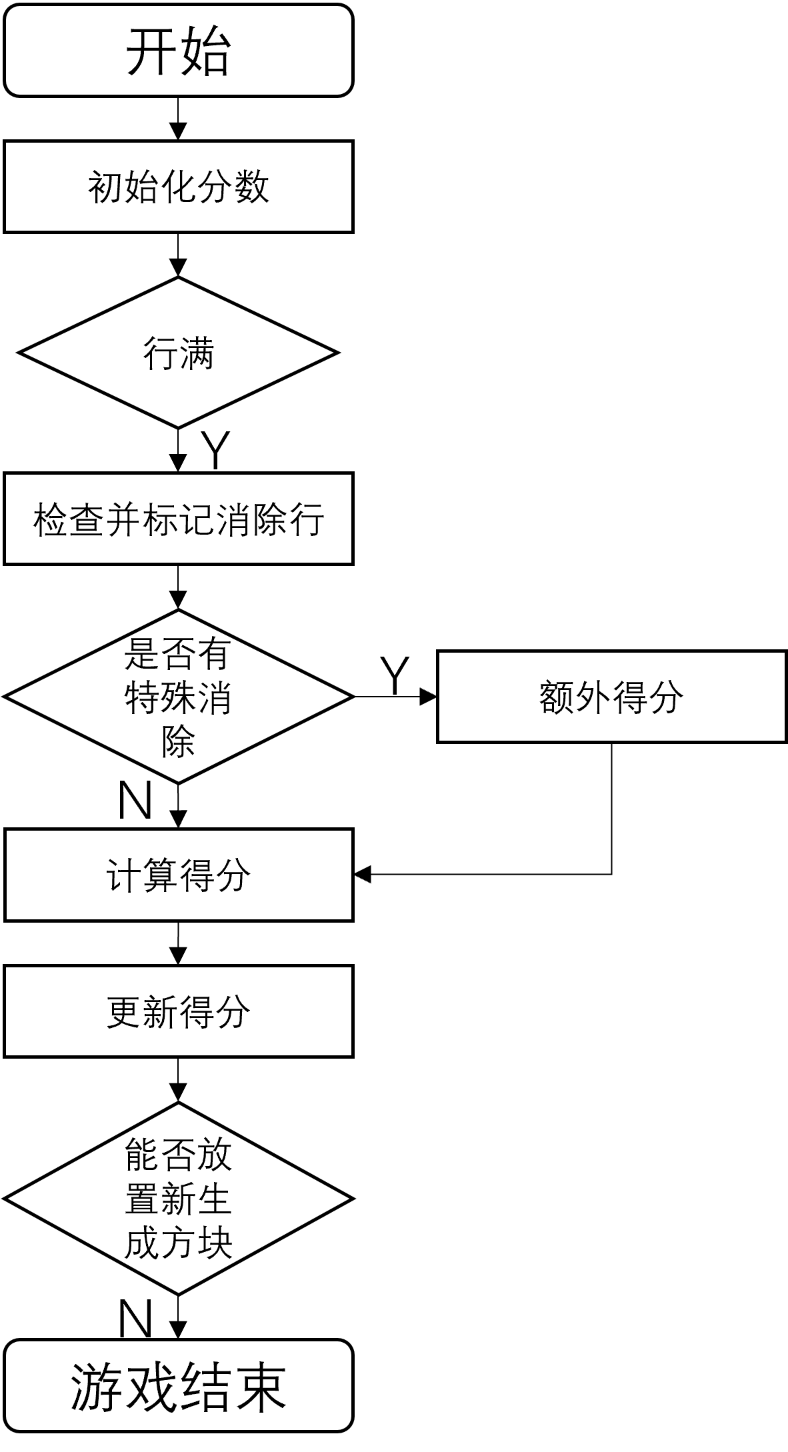
**3.计分模块：**

（1）用于记录玩家的得分，根据消除的行数或其他指标计算分数，并在游戏界面上显示。

（2）消除判断：当一行被填满时，触发消除判断。扫描游戏区域，检查并标记可消除的行。

（3）计算得分：根据消除的行数和游戏的级别，计算基础得分。考虑是否有特殊消除（如一次性消除多行），给予额外得分。更新玩家的总得分。

更新得分显示：在游戏界面上实时更新得分显示



图表4.3计分模块流程图

1. **控制模块：**处理用户的键盘或触摸屏事件，转换为游戏命令。

（1）初始化：

在游戏开始时，控制模块进行初始化设置，包括检测按键或触摸屏幕等输入设备的工作状态。设定方块的初始位置和形状。

（2）输入检测：

控制模块持续监听用户输入，这可以是键盘上的WASD或方向键、鼠标点击或者触摸屏的操作。识别不同的输入信号，如左移、右移、下移、旋转等。

（3）输入处理：

根据用户的输入执行相应的操作，例如移动方块、旋转方块或改变方块下落的速度。在执行操作前，进行碰撞检测以确保方块不会超出边界或与其他方块重叠。

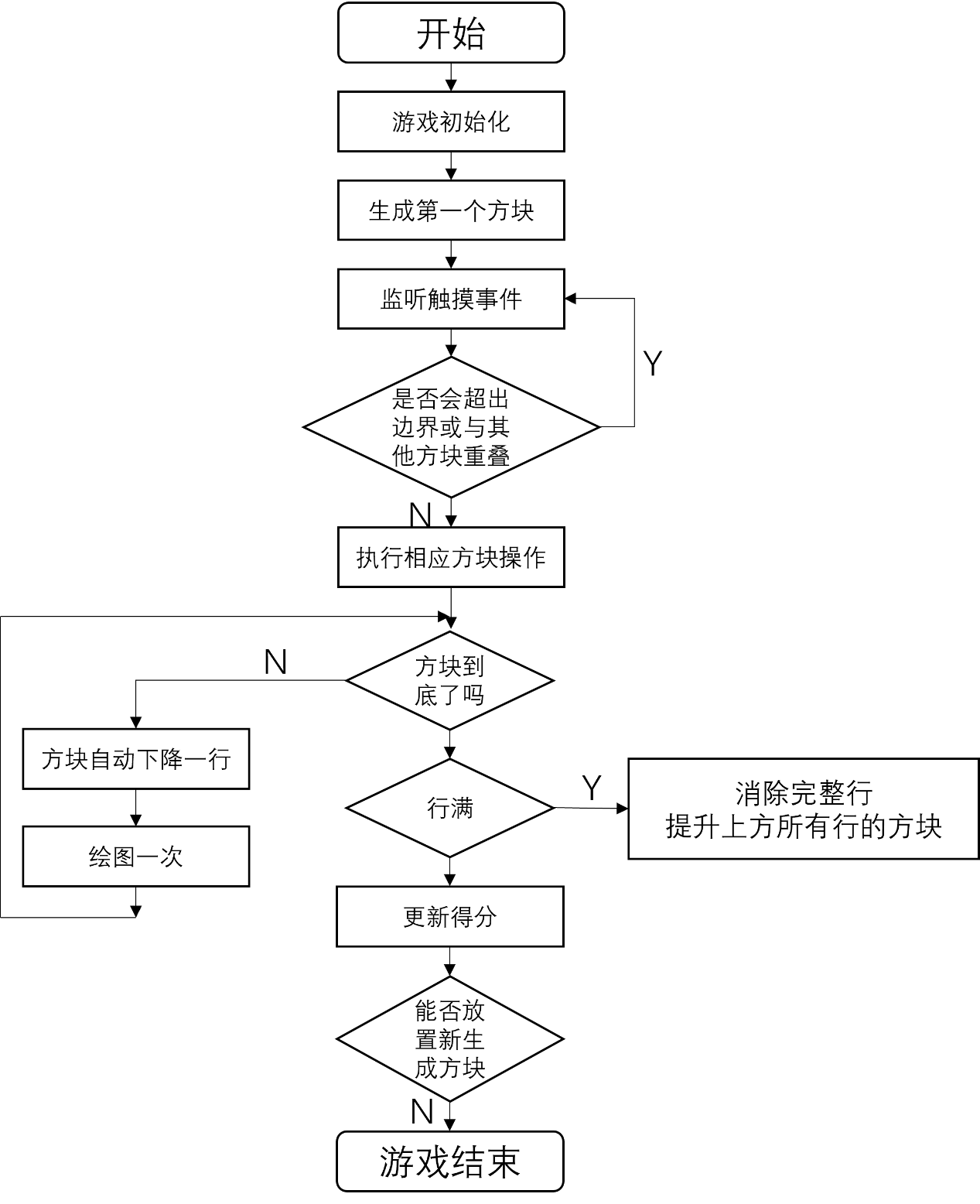
（4）方块控制：

控制当前活跃的方块根据用户指令移动或旋转。

（5）自动下落：使方块每隔一定时间自动下移一格，直到无法继续下落为止。

（6）消除逻辑：

当方块下落到无法再移动时，检查是否有满行，如果有则消除这些行并给玩家加分。消除满行后，将上方所有行整体下移。



图表4.4控制模块流程图

1. **数据存储模块：**利用文件系统或数据库存储玩家得分

# 5 性能分析

**1.性能分析的必要性：**性能分析关注游戏的响应时间、内存使用、CPU负载等关键指标。对于GEC6818这样的嵌入式设备来说，资源有限，因此优化代码以减少资源消耗是非常重要的。通过分析工具监测游戏运行时的各项指标，可以帮助开发者找到性能瓶颈，进而进行针对性的优化。

**2.优化的策略：**针对发现的性能瓶颈，开发者可以采取多种策略进行优化。例如，优化算法复杂度、减少不必要的资源加载、改进数据处理流程等。对于图形渲染，可以考虑简化图形元素的复杂性，减少渲染次数。同时，合理的内存管理也是提升性能的关键，避免内存泄漏和过度的内存分配可以显著改善游戏的稳定性和响应速度。

**3.测试的方法：**采用单元测试和集成测试来验证各个模块的功能正确性。单元测试关注游戏中的单个组件，如方块的生成、移动和旋转等功能。集成测试则检验这些组件在一起工作时的整体行为。通过自动化测试框架，可以持续地进行回归测试，确保代码的每次更改都不会引入新的问题。

**4.用户反馈：**邀请用户体验游戏并提供反馈，了解游戏的操作是否顺畅，界面是否友好，以及是否有引起困惑的地方。

适应性测试：考虑到不同的硬件配置可能对游戏性能有影响，可以在不同配置的GEC6818开发板上测试游戏，确保良好的适应性。

**5.功耗优化：**对于嵌入式设备，功耗是一个重要指标。通过优化代码和合理调度系统资源，可以减少设备的能耗，延长使用寿命。

# 6 心得体会

通过这次使用GEC-6818开发板的经验，我们对嵌入式硬件的特性和限制有了更深入的认识。在项目开发过程中，我们意识到高质量的代码对于维护和扩展项目有着至关重要的作用。在团队协作中，我们明白了有效的沟通和明确的分工是成功的关键。通过定期的会议和代码审查，我们保持了项目的进度和质量。与团队成员们一起解决问题，让我们领悟到集体智慧的力量。从队友身上学到了许多之前不熟悉的技术和问题解决方法。通过本次基于Linux的俄罗斯方块游戏设计项目，不仅提升了我们的技术能力，也对团队合作有了更深的理解。这些经验和教训将成为我们未来职业生涯中宝贵的财富。

# 参考文献

[1]张玉书. 嵌入式Linux系统开发从入门到精通：基于S3C2440 [M]. 北京： 清华大学出版社， 2010.

[2]李昕. 基于Qt的嵌入式GUI设计与实现 [J]. 电子设计工程， 2011, 19(13): 56-58.

[3]刘洪涛， 罗陶. 基于ARM的嵌入式系统开发实战 [M]. 北京： 人民邮电出版社， 2012.

[4]王晓薇. 基于ARM的嵌入式Linux系统研究与应用 [D]. 哈尔滨工业大学， 2013.

[5]陈海波， 李晓明. 基于ARM的嵌入式系统开发教程 [M]. 北京： 电子工业出版社， 2014.

[6]赵志荣. 基于ARM的嵌入式系统设计与实现 [J]. 计算机工程与设计， 2015, 36(1): 176-178.

[7]周兆熊. 基于ARM的嵌入式Linux系统开发实践 [M]. 北京： 电子工业出版社， 2016.

[8]李华平. 基于ARM的嵌入式系统开发与应用 [M]. 北京： 机械工业出版社， 2017.

# 附录

#include "move.h"

#include <pthread.h>

unsigned int shp,shp\_next; //shape

static int dir; //控制移动方向

static int paused=0;

static int running=0;

struct block \*bk,\*bk\_next; //类型方块结构体——》当前移动及下一个提示

struct block \* move\_check(struct ls\_all \*head,int dir);

void \*auto\_down(void \*arg);

void \*time\_out(void \*arg);

int main(int argc,char \*argv[]){

int x,y;

struct ls\_all \*head;

//显示背景图片

bmp\_show\_mix(0,0,800,480,"./tetris\_pic/bck.bmp");

int rt;

pthread\_t idt, idr;

//获取两种随机形状并初始化，得到初始化结构体

srand((unsigned int) time(NULL));

shp = ((unsigned int) rand()) % 7 + 1;

shp\_next = ((unsigned int) rand()) % 7 + 1;

bk = bk\_init(shp);

bk\_next = bk\_init(shp\_next);

//初始化掉落方块结构体

head = ls\_init();

//初始化分数和速度

score = 0;

speed = 0;

//显示移动方块 及 提示方块

the\_show(bk);

the\_show\_next(bk\_next);

score\_show(0); //显示成绩

//创建控制方块移动线程

pthread\_create(&idt, NULL, auto\_down, NULL);

//时间更新线程，时间到且无操作自动更新dir为下落状态

pthread\_create(&idr, NULL, time\_out, NULL);

system("madplay ./tetris\_pic/bgm.mp3 -r &");//音频

while(1) {

label :

if(running == 1){

pthread\_cancel(idt);

pthread\_cancel(idr);

bmp\_show\_mix(0, 0, 800, 480, "./tetris\_pic/bck.bmp");

// 获取两个随机形状并初始化方块

srand((unsigned int)time(NULL));

shp = ((unsigned int)rand()) % 7 + 1;

shp\_next = ((unsigned int)rand()) % 7 + 1;

bk = bk\_init(shp);

bk\_next = bk\_init(shp\_next);

// 初始化掉落方块的结构

head = ls\_init();

// 初始化分数和速度

score = 0;

speed = 0;

// 显示当前移动的方块和下一个方块

the\_show(bk);

the\_show\_next(bk\_next);

score\_show(0); // 显示分数

// 创建自动下落方块和超时的线程

pthread\_create(&idt, NULL, auto\_down, NULL);

pthread\_create(&idr, NULL, time\_out, NULL);

running = !running;

}

if (!paused) {

dir = -2; //the random number except -1 & 0 & 1 & 2;

while(dir == -2){

usleep(1000);

}

if(dir== -1){ //变形

change\_type(bk);

the\_show\_bck\_type(bk);

}

else { //移动

change\_dir(bk->p,dir);

the\_show\_bck\_dir(bk->p,dir);

}

//移动检查是否越界及掉落到底部

bk = move\_check(head,dir);

if(bk == NULL){

running =1;

goto label;

}

//显示方块形状

the\_show(bk);

} else {

//printf("游戏已暂停，请按空格键恢复...\n");

while (1){

if(paused==1);

break;

};

//printf("游戏恢复...\n");

}

}

}

//移动检查是否越界及掉落到底部

struct block \* move\_check(struct ls\_all \*head,int dir){

int rt;

//1.检查是否掉落底部

if(ls\_check(head,bk->p) == -1){

//1.1 变形越界则恢复原来（再变4次）

if(dir == -1){

change\_type(bk);

change\_type(bk);

change\_type(bk);

//the\_show(bk);

}

else{

//1.2 移动越界 则恢复原来

change\_dir\_off(bk->p,dir);

if(dir == 0){ //1.2.1 向下移动越界则已经到底把它加入到链表

the\_show(bk);

if(ls\_updata(head,bk) == -1){ //加入链表，包括重新显示整个链表

return NULL; //game over

}

shp\_next = ((unsigned int)rand())%7+1;

bk = bk\_next; //有新的方块产生

the\_show\_bck\_next(bk\_next->p);

bk\_next = bk\_init(shp\_next);//有新的下一个方块产生

the\_show\_next(bk\_next);

}

//1.2.2 左右移动越界 正常显示原来的

else if(dir == 1||dir ==2){

//the\_show(bk);

}

}

return bk;

}

//2. 越界检查

rt = bound\_check(bk->p);

//2.1 掉落底部

if(rt==0){

change\_dir\_off(bk->p,0);

the\_show(bk);

ls\_updata(head,bk);

shp\_next = ((unsigned int)rand())%7+1;

bk = bk\_next;

the\_show\_bck\_next(bk\_next->p);

bk\_next = bk\_init(shp\_next);

the\_show\_next(bk\_next);

}

//2.2 左右移动越界

else if(rt == -1){

while(bound\_check(bk->p)== -1){

change\_dir\_off(bk->p,1);

}

}

else if(rt == -2){

while(bound\_check(bk->p)== -2){

change\_dir\_off(bk->p,2);

}

}

return bk;

}

void \*auto\_down(void \*arg){

pthread\_setcanceltype(PTHREAD\_CANCEL\_ASYNCHRONOUS,NULL);

int x,y;

while(1){

get\_xy(&x,&y);

if(x>422&&x<558&&y>383&&y<480){

dir =1;

}

else if(x>536&&x<636&&y>448&&y<595){

dir = 0;

}

else if(x>652&&x<798&&y>383&&y<480){

dir = 2;

}

else if(x>536&&x<636&&y>218&&y<365){

dir = -1;

}

else if(x>0&&x<50&&y>0&&y<50){

paused = !paused;

}

else if(x>270&&x<340&&y>0&&y<80){

running = 1;

}

}

}

void \*time\_out(void \*arg){

pthread\_setcanceltype(PTHREAD\_CANCEL\_ASYNCHRONOUS,NULL);

while(1){

usleep((400-speed\*5)\*1000);

dir =0;

}

}

void \*play\_music(void \*arg){//音乐

pthread\_setcanceltype(PTHREAD\_CANCEL\_ASYNCHRONOUS, NULL);

system("madplay ./tetris\_pic/bgm.mp3 -r &");

}