### **2048 游戏实训报告目录**

# 一、实验要求

## 1.1 项目背景与目标

### 项目背景

2048是一款流行于全球的数字拼图游戏，玩家通过合并相同数值的方块来生成更大的数值，直到达到2048或更高的分数。该游戏简单易学但具有挑战性，适合各个年龄段的人群。本项目旨在使用C++和EasyX图形库实现一个桌面版的2048游戏。

#### **目标**

本次实训的目标是让学生们通过实现2048游戏来加深对C++编程语言的理解，掌握图形界面开发的基本技能，并熟悉游戏逻辑设计的方法。具体目标包括：

使用C++和EasyX图形库创建交互式的游戏界面。

实现2048游戏的核心玩法，包括方块移动、合并、随机生成新方块等。

提供用户友好的操作体验，如键盘控制、得分显示等。

掌握基本的图形绘制技术，如矩形、文本输出等。

## 1.2 功能需求分析

为了确保2048游戏能够满足用户预期并提供良好的游戏体验，以下是对其核心功能的需求分析：

****网格布局****：采用4x4的网格作为游戏区域，每个格子可以为空或者包含一个带有数字的方块。

****随机生成新方块****：每次玩家操作后，在空闲位置随机出现一个新的“2”或“4”的方块，其中“4”的概率为10%。

****移动与合并****：根据用户的输入（上、下、左、右），所有非空方块朝指定方向滑动；若两个相同数字的方块相遇，则它们合并成一个新的方块，其值等于这两个方块之和，同时增加玩家得分。

****胜利条件****：当任意一个方块的值达到2048时，游戏判定为胜利。（可选：继续游戏以追求更高分数）

****失败条件****：如果网格已满且无法再进行有效的移动，则游戏结束。

****得分系统****：记录当前得分，并在玩家每次成功合并方块时更新。

****用户交互****：支持键盘方向键（或WASD键）来控制方块的移动方向，提供“New Game”按钮以重新开始游戏。

****视觉效果****：根据不同数值设置不同颜色的方块，以增强视觉区分度；在得分板上实时显示当前得分。

## 1.3 技术选型与环境准备

#### **技术选型**

为了高效地构建2048游戏，以下技术栈被推荐用于本项目：

****编程语言****：C++ 是首选语言，它提供了强大的性能和直接访问硬件的能力，非常适合游戏开发。

****图形库****：EasyX 图形库简化了Windows平台下的图形程序开发，支持基本图形绘制、图像处理等功能，特别适用于教学和小型项目。

****数学函数库****：<math.h> 库用于执行数学运算，例如计算对数以确定方块的颜色索引。

****标准库****：<stdlib.h> 和 <stdio.h> 等标准库提供了内存分配、输入输出等基础功能。

****随机数生成****：rand() 函数结合 srand() 初始化种子来产生伪随机数，用于决定新方块的位置和值。

#### **环境准备**

为了顺利开展项目，需要做好如下准备工作：

****开发工具****：安装Visual Studio 或 Dev-C++ 等集成开发环境（IDE），这些工具包含了编译器和支持EasyX图形库的必要组件。

****图形库安装****：下载并正确配置EasyX图形库，确保能够调用图形相关的API。

****调试环境****：利用Visual Studio内置的调试工具，帮助开发者实时查看和修改网页内容。

****版本控制系统****：Git 是必备的版本控制工具，GitHub/GitLab 用作远程仓库托管平台，便于团队成员间的协作开发。

****持续集成/部署(CI/CD)****：对于个人项目可能不需要复杂的CI/CD设置，但对于多人合作或频繁迭代，可以考虑配置简单的自动化测试流程。

# 二、设计思路

## 2.1 系统架构设计

系统架构的设计是为了确保2048游戏的各个组成部分能够高效地协同工作。考虑到这是一个相对简单的单机游戏，我们采用的是一个较为直接的分层架构模式，主要包括以下几个层次：

****表现层****：负责与用户交互，包括图形界面的绘制和用户输入的处理。这部分主要使用EasyX库来实现。

****逻辑层****：包含游戏的核心算法和业务逻辑，如方块移动规则、合并机制、胜利或失败条件判断等。

****数据层****：存储游戏状态信息，例如当前的游戏网格（map[MAX\_SIZE][MAX\_SIZE]）、得分(score)、最高分(Last\_Max\_Score)等。

这种分层结构有助于将不同职责分离，使得代码更易于维护和扩展。

## 2.2 数据结构设计

在2048游戏中，核心的数据结构是表示游戏板面的二维数组map[MAX\_SIZE][MAX\_SIZE]，其中每个元素代表一个方格中的数字（或者为空）。此外，还有几个辅助变量用于记录游戏的状态：

int map[MAX\_SIZE][MAX\_SIZE];：存储游戏网格中每个位置的数值，初始化为0表示空位。

int score;：记录玩家当前获得的总分数。

int Last\_Max\_Score;：保存玩家历史上的最高得分。

char sscore[10];：用来临时转换得分值为字符串形式，以便于显示。

为了提高性能和简化逻辑，我们还定义了一个枚举类型color来管理不同的颜色配置，以及一个颜色数组arr[]，它根据方块上的数字映射到相应的颜色。

## 2.3 游戏逻辑设计

游戏逻辑是整个项目中最关键的部分，它决定了游戏如何响应用户的操作并更新游戏状态。以下是游戏逻辑的主要组成部分：

****初始化****：当开始新游戏时，需要重置所有变量，并随机生成两个初始方块（通常是“2”或“4”）。

****移动规则****：根据用户选择的方向（上、下、左、右），所有非零方块将朝该方向滑动直到遇到边界或其他方块。如果两个相同数字的方块相遇，则它们会合并成一个新的方块，其值等于这两个方块之和，并且玩家的得分相应增加。

****随机生成****：每次有效移动后，在空闲的位置随机出现一个新的方块（“2”或“4”，其中“4”的概率为10%）。

****胜负判定****：检查是否有任意方块达到目标值（例如2048），或者是否还有可用的移动步骤。如果有任意一个方块达到了设定的目标值，则玩家获胜；若网格已满且无法再进行有效的移动，则游戏结束。

****得分计算****：每当两个相同数字的方块合并时，新的方块值会被加到得分中。此外，还可以考虑加入额外的奖励机制来激励玩家追求更高的分数。

## 2.4 用户界面设计

用户界面设计旨在提供直观的操作体验和清晰的信息展示。以下是UI设计的一些关键点：

****布局****：游戏窗口分为两大部分，左侧为主游戏区域，显示4x4的网格及其中的方块；右侧为辅助区域，包含了得分板、最高分记录、“New Game”按钮等功能组件。

****视觉风格****：通过定义不同的颜色方案来区分各种方块，使得游戏界面更加生动有趣。同时，保持整体色调和谐统一，避免过于刺眼的颜色组合影响用户体验。

****交互性****：支持键盘方向键（或WASD键）控制方块移动，鼠标点击“New Game”按钮可以重新开始游戏。对于触摸屏设备，还需确保触摸手势同样能够触发上述操作。

****动画效果****：虽然本项目未详细提及，但可以在方块移动、合并过程中添加简单的动画效果，以增强游戏的真实感和趣味性。

****反馈机制****：每当玩家执行一次有效操作后，立即刷新屏幕并给出即时反馈（如得分变化、新方块出现等），让玩家感受到自己的行动对游戏产生了影响。

# 三、实验步骤

## 3.1 环境搭建与工具准备

#### **开发环境设置**

****操作系统****：Windows（考虑到EasyX图形库的兼容性）

****IDE选择****：推荐使用Visual Studio，它不仅支持C++开发，而且内置了对EasyX的良好支持。

****安装EasyX****：前往[EasyX官方网站](http://www.easyx.cn/" \t "https://tongyi.aliyun.com/_blank)下载并安装最新版本。按照官方文档说明进行配置，确保项目能够正确链接到EasyX库。

#### **工具准备**

****编译器****：确保安装了适用于C++的编译器（如MSVC），这是Visual Studio自带的。

****图形库****：完成EasyX的安装后，在项目属性中添加必要的包含路径和库文件路径。

****调试工具****：利用Visual Studio自带的强大调试功能，可以方便地跟踪程序执行过程、检查变量值等。

## 3.2 游戏初始化

void gameInit () { // 初始化游戏网格为全0 for (int i = 0; i < MAX\_SIZE; i++) { for (int j = 0; j < MAX\_SIZE; j++) { map[i][j] = 0; } } // 初始化得分为0 score = 0; // 设置随机数种子 srand(GetTickCount()); // 在随机位置生成两个初始数字块（2或4） createNum(); createNum(); }

map[][] 数组被清零，表示所有格子为空。

score 被设为0，作为起始分数。

使用当前时间戳来设置随机数种子，保证每次启动游戏时产生的随机序列不同。

createNum() 函数调用两次以在游戏开始时放置两个初始方块。

## 3.3 绘制游戏界面

void gameView() { // 绘制背景 setbkcolor(INTERVRL\_COLOR); cleardevice(); // 绘制得分板 drawScoreBoard(); // 遍历每个格子，根据其数值绘制相应颜色的矩形，并显示数字 for (int i = 0; i < MAX\_SIZE; i++) { for (int j = 0; j < MAX\_SIZE; j++) { int x = j \* GRID\_W + (j + 1) \* INTERVAL; int y = i \* GRID\_W + (i + 1) \* INTERVAL; drawTile(x, y, map[i][j]); } } // 更新屏幕 FlushBatchDraw(); } void drawScoreBoard() { // 绘制得分板矩形及文本 setfillcolor(SCORE\_COLOR); solidroundrect(530, 40, 670, 100, 5, 5); settextstyle(30, 0, "Microsoft YaHei UI"); outtextxy(565, 40, "SCORE"); // 显示当前得分 sprintf\_s(sscore, "%d", score); settextcolor(WHITE); drawtext(sscore, &RECT{530, 65, 670, 100}, DT\_CENTER | DT\_VCENTER | DT\_SINGLELINE); } void drawTile(int x, int y, int value) { if (value == 0) { setfillcolor(BLANK\_GRID); } else { int index = log2(value); setfillcolor(arr[index]); settextcolor((value < 8) ? RGB(119, 110, 101) : WHITE); } solidroundrect(x, y, x + GRID\_W, y + GRID\_W, 5, 5); if (value != 0) { char snum[10]; sprintf\_s(snum, "%d", value); int ntw = textwidth(snum); int nth = textheight(snum); outtextxy(x + (GRID\_W - ntw) / 2, y + (GRID\_W - nth) / 2, snum); } }

map[][] 数组被清零，表示所有格子为空。

score 被设为0，作为起始分数。

使用当前时间戳来设置随机数种子，保证每次启动游戏时产生的随机序列不同。

createNum() 函数调用两次以在游戏开始时放置两个初始方块

## 3.4 实现基本移动操作

void moveUp() { bool isMoved = false; for (int col = 0; col < MAX\_SIZE; ++col) { int now = 0; for (int row = 1; row < MAX\_SIZE; ++row) { if (map[row][col] != 0) { if (map[now][col] == 0) { map[now][col] = map[row][col]; map[row][col] = 0; isMoved = true; } else if (map[now][col] == map[row][col]) { map[now][col] \*= 2; score += map[now][col]; map[row][col] = 0; isMoved = true; now++; } else { if (++now != row) { map[now][col] = map[row][col]; map[row][col] = 0; isMoved = true; } } } } } if (isMoved) { createNum(); } gameView(); }

moveUp() 是四个方向移动函数之一，其他三个（moveDown()、moveLeft() 和 moveRight()）遵循类似的逻辑结构。

当发生有效移动时，会调用 createNum() 来生成新的方块，并通过 gameView() 刷新界面。

## 3.5 合并相同数字块逻辑

## 合并逻辑已经嵌入到了各个移动函数中（如上面的moveUp()）。每当两个相同数字的方块相遇时，它们会合并成一个新的方块，其值等于这两个方块之和，并且玩家的得分相应增加。 3.6 随机生成新数字块

void createNum() { while (true) { int x = rand() % MAX\_SIZE; int y = rand() % MAX\_SIZE; if (map[x][y] == 0) { map[x][y] = (rand() % 10 == 0) ? 4 : 2; break; } } }  
3.7 检测游戏胜利或结束条件

int isWin() { // 检查是否有任意方块达到目标值（例如2048） for (int i = 0; i < MAX\_SIZE; i++) { for (int j = 0; j < MAX\_SIZE; j++) { if (map[i][j] == 2048) return 1; } } // 检查是否还有空位或者可合并的方块 for (int i = 0; i < MAX\_SIZE; i++) { for (int j = 0; j < MAX\_SIZE; j++) { if (map[i][j] == 0 || (i > 0 && map[i][j] == map[i-1][j]) || (j > 0 && map[i][j] == map[i][j-1])) return 0; } } return 2; // 游戏失败 }

如果有任意一个方块达到了设定的目标值，则返回1表示胜利；如果有空位或相邻的相同方块，则返回0表示游戏继续；否则返回2表示游戏失败。

## 3.8 实现得分系统

得分系统已经在各个移动函数中实现。每当两个相同数字的方块合并时，新的方块值会被加到得分中。

## 3.9 添加动画效果和过渡

void drawTile(int x, int y, int value, int duration = 0) { if (duration > 0) { // 假设我们有一个简单的线性插值函数lerp // 这里只是一个示例，实际需要根据具体需求实现 for (float t = 0; t <= 1.0f; t += 0.05f) { int nx = lerp(x, x + GRID\_W, t); // 插值计算新的x坐标 int ny = lerp(y, y + GRID\_W, t); // 插值计算新的y坐标 setfillcolor(BLANK\_GRID); solidroundrect(nx, ny, nx + GRID\_W, ny + GRID\_W, 5, 5); Sleep(16); // 控制帧速率 FlushBatchDraw(); } } // ... 继续原来的绘图逻辑 ... } // 线性插值辅助函数 float lerp(float a, float b, float t) { return a + t \* (b - a); }

# 四、实验成果与展示

## 4.1 成果概述

在本次2048游戏的开发实训中，我们成功实现了基于C++和EasyX图形库的桌面版2048游戏。通过这个项目，参与者不仅掌握了C++编程语言的核心概念，还深入了解了图形界面开发的基本技巧。最终的产品具备以下特点：

****完整的功能实现****：游戏包括了所有标准2048游戏的功能，如方块移动、合并、随机生成新数字块、得分计算等。

****用户友好的界面设计****：采用了美观大方的颜色搭配，并且提供了直观的操作体验，支持键盘方向键控制和“New Game”按钮重置游戏。

****良好的交互性****：玩家可以通过键盘或触摸屏轻松地进行游戏操作，并实时看到自己的得分变化。

****视觉效果优化****：虽然没有复杂的动画系统，但通过简单的绘图逻辑为方块移动添加了基本的过渡效果，增强了游戏的真实感。

## 4.2 关键技术点解析

****图形绘制****：使用EasyX库提供的API（如setfillcolor()、solidroundrect()、outtextxy()等）来创建游戏所需的图形元素。这些函数简化了图形绘制的过程，使得开发者可以专注于游戏逻辑的设计。

****随机数生成****：为了确保每次启动游戏时产生的序列不同，我们使用了srand(GetTickCount())来初始化随机数种子，并利用rand()函数生成新的方块值（2或4）及其位置。

****游戏逻辑处理****：核心的游戏逻辑被分解成多个函数，每个函数负责特定的任务，例如moveUp()用于处理向上移动操作，createNum()负责生成新的数字块等。这种方式提高了代码的模块化程度，便于理解和维护。

****性能优化****：考虑到游戏运行过程中需要频繁更新屏幕内容，我们采用了批量绘图模式(BeginBatchDraw() 和 FlushBatchDraw())以减少不必要的刷新，从而提升程序效率。

****事件监听与响应****：通过getmessage()函数捕获用户的输入事件（如按键或鼠标点击），并根据不同的事件类型调用相应的处理函数，实现了流畅的人机交互体验。

## 4.3 成果截图

#### **五、PPT汇报**

5.1 PPT内容规划  
5.2 汇报要点  
5.3 PPT模板选择与设计

# 六、总结

## 6.1 项目回顾

在本次2048游戏开发实训中，我们从零开始构建了一个功能完整且用户友好的桌面版游戏。整个过程涵盖了需求分析、设计思路、编码实现、测试优化等多个阶段。以下是项目回顾的重点：

****成功之处****：

成功实现了所有预期的功能，包括方块移动、合并、随机生成新数字块、得分计算等。设计并实现了直观的用户界面，提供了良好的用户体验。通过模块化编程提升了代码的可读性和维护性，为后续扩展打下了基础。

****遇到的问题及解决方案****：

****图形库兼容性问题****：最初遇到了EasyX与Visual Studio集成时的一些小问题，通过查阅官方文档和社区资源得以解决。****性能优化挑战****：为了确保游戏运行流畅，尤其是在高分辨率屏幕上，我们采用了批量绘图模式，并对动画效果进行了简化处理。

****逻辑复杂度管理****：随着功能的增加，游戏逻辑变得越来越复杂。我们通过将逻辑分解成多个函数来提高代码的模块化程度，使得每个部分都易于理解和维护。

****不足与改进方向****：

游戏的动画效果较为简单，未来可以考虑引入更复杂的动画库或框架来增强视觉体验。

在多平台支持方面还有待加强，目前仅限于Windows环境下的运行

## 6.2 学习心得与体会

通过这次实训，团队成员不仅掌握了C++编程语言的核心概念，还深入了解了图形界面开发的基本技巧。以下是我们的主要学习心得：

****技术技能提升****：熟悉了C++语法结构，尤其是面向对象编程的思想；掌握了EasyX图形库的使用方法，能够创建基本的图形元素和处理用户输入。

****问题解决能力****：面对各种技术难题时，学会了如何查找资料、阅读文档以及向社区求助，提高了独立解决问题的能力。

****团队协作精神****：在一个团队中工作，每个人都有不同的专长，通过分工合作可以更高效地完成任务。同时，在沟通协调过程中也增强了彼此的理解和支持。

****时间管理经验****：合理安排时间对于项目的顺利完成至关重要。我们学会了制定详细的计划表，并根据实际情况灵活调整进度，确保各个阶段的任务按时完成。

## **6.3 团队合作经验分享**

在整个项目开发过程中，团队合作发挥了至关重要的作用。以下是我们关于团队合作的一些经验和教训：

****明确分工****：项目初期就明确了每位成员的责任范围，如有人负责前端界面设计，有人专注于后端逻辑实现，还有人关注整体架构规划。这有助于避免重复劳动，提高工作效率。

****定期会议****：每周至少召开一次全体会议，汇报各自的工作进展，讨论遇到的问题，并共同商讨解决方案。这种机制保证了信息流通顺畅，及时发现潜在风险。

****有效沟通****：保持开放透明的沟通渠道，鼓励大家随时提出想法和建议。遇到分歧时，尽量站在对方角度思考问题，寻找折衷方案，而不是一味坚持己见。

****互相帮助****：当某个成员遇到困难时，其他成员会主动伸出援手，提供技术支持或情感上的鼓励。这种互助氛围让整个团队更加团结一致。

## 6.4 对未来的展望

尽管当前版本的2048游戏已经达到了预期目标，但我们仍然看到了许多可以继续探索和改进的方向：

****跨平台支持****：考虑将游戏移植到其他操作系统（如Linux、macOS），甚至开发移动端应用，以扩大受众群体。

****功能扩展****：引入更多有趣的游戏模式，如限时挑战、多人对战等，或者添加成就系统来激励玩家不断尝试更高的分数。

****性能优化****：进一步优化程序性能，特别是在大规模网格或复杂动画情况下，确保游戏始终保持流畅运行。

****社区建设****：如果条件允许，可以建立一个小型社区，收集玩家反馈，定期发布更新版本，形成良性循环的发展模式。

总之，通过这次实训，我们不仅收获了一款自己亲手打造的游戏作品，更重要的是积累了宝贵的知识和技术经验。希望在未来的学习和工作中，能够继续发扬这些优点，创造出更多有价值的产品。