华南理工大学

C++实训作业报告

设计题目: 使用C++及EasyX库实现俄罗斯方块及其图形化

姓名: 林浩 学号: 202230100271

班级: 软件工程2班

指导教师: 郭芬

|  |
| --- |
| **设计报告** |
| 【问题描述】  在本次C++实训 中，要求不使用游戏引擎，不参考他人俄罗斯方块游戏实现，独立自写C++程序，并使用相关图形库实现一个游戏。  本人选择使用Visual Studio及EasyX图形库，制作一个俄罗斯方块游戏，实现俄罗斯方块基本游戏功能，以及加入音乐，和部分游戏设置。  其中根据题目要求，如果制作俄罗斯方块游戏，需要用到多态  【系统设计说明】  （1）数据结构说明。  整个项目中使用的数据结构是实现程序功能的基础。以下是对项目中关键数据结构的说明：  1. Button 类  Button 类用于创建和管理用户界面按钮。它使用以下数据结构：  - 位置和尺寸：存储按钮的 x 和 y 坐标（左上角），以及 width 和 height。  - 文本：按钮上显示的文本，类型为 std::string。  - 回调函数：当按钮被点击时调用的函数，使用 std::function<void()> 存储。  - 状态标志：例如 isMouseOver 表示鼠标是否悬停在按钮上。  - 缩放比例：scale 用于实现鼠标悬停时的缩放效果。  2. Block 类  Block 类表示俄罗斯方块游戏中的单个方块或一组小方块。数据结构包括：  - 方块类型：blockType 表示方块的种类。  - 小方块数组：smallBlocks 数组，存储方块的四个构成部分的位置。  - 图像资源：img 指向当前方块图像的指针。  - 静态图像数组：imgs 存储所有不同方块类型的图像资源。  - 尺寸：size 表示每个小方块的尺寸。  3. Tetris 类  Tetris 类管理整个俄罗斯方块游戏的逻辑。关键数据结构有：  - 游戏地图：map，一个二维 std::vector，存储游戏区域中每个方块的状态。  - 分数和等级：score 当前分数，level 当前等级。  - 行消除计数：linecount 表示消除的行数。  - 历史最高分：highscore 存储玩家的历史最高分数。  - 游戏状态标志：例如 gameOver 表示游戏是否结束，stop 控制游戏循环的停止。  - 方块实例：curBlock 当前活动方块，nextBlock 下一个方块。  - 按钮实例：例如 button\_back 用于返回菜单的按钮。  4. Page 类  Page 类负责游戏的用户界面页面管理。数据结构包括：  - 页面数组：pages 存储不同页面的背景图像。  - 按钮数组：buttons 二维数组，每个页面包含多个按钮。  - 当前页面索引：currentIndex 表示当前显示的页面。  - 玩家排名数组：在 ranking 函数中使用，存储玩家的名称和分数。  5. music 类  music 类及其子类管理游戏中的音乐和音效。数据结构有：  - 音乐类型：musictype 表示当前要播放的音乐类型。  - 子类：gg 和 winwin 分别用于播放游戏结束和胜利的音乐。  6. 其他数据结构  - 玩家信息：在 Page 类的 ranking 函数中使用，通常包括 name（玩家名称）和 score（玩家分数）。  - 点和尺寸：在 Block 类中使用 Point 结构来表示小方块的位置，以及使用尺寸 size 来表示小方块的尺寸。  这些数据结构共同构成了俄罗斯方块游戏的逻辑基础，使得游戏的各种功能得以实现。  （2）程序功能说明。包括：  各模块功能；  算法描述；  接口说明；  关键变量说明。  **一、Block类**  Block 类功能说明  各模块功能  1. 构造函数 (Block()): 初始化方块的类型和位置，加载并分割图像资源以供不同类型的方块使用。    2. 下落操作 (drop()): 将方块向下移动一个单元格，模拟方块的下落。  3. 移动操作 (moveLeftRight(int offset)): 根据提供的参数，将方块水平移动指定的单元格数。  4. 旋转操作 (rotate()): 旋转方块，变换其形状，但不改变位置。  5. 绘制操作 (draw(int leftMargin, int topMargin)): 在屏幕上绘制方块，包括其位置和图像。  6. 图像资源获取 (getImage()): 提供对所有方块类型图像资源的访问。  7. 边界检查 (blockInMap()): 检查方块是否在游戏地图的边界内，确保方块不会移出游戏区域。  8. 固化操作 (solidify()): 将方块的位置固化到游戏地图中，表示方块已经到达最终位置。  9. 类型获取 (getBlockType()): 获取方块的类型标识。  10. 赋值操作符 (operator=): 允许对 Block 对象进行赋值操作，复制方块的状态。  算法描述  - 下落算法：每次调用 drop() 时，将方块的每个小方块的行坐标增加一，然后调用 blockInMap() 检查新位置是否有效。  - 移动算法：调用 moveLeftRight() 时，根据提供的偏移量更新方块每个小方块的列坐标，然后同样进行边界检查。  - 旋转算法：rotate() 通过交换每个小方块的行列坐标实现旋转效果，同样需要进行边界检查。  接口说明  - 构造函数：Block() 初始化新方块。  - 下落方法：void drop()  - 移动方法：void moveLeftRight(int offset)  - 旋转方法：void rotate()  - 绘制方法：void draw(int leftMargin, int topMargin)  - 图像资源获取：static IMAGE getImage()  - 边界检查：bool blockInMap(const std::vector<std::vector<int>>& map)  - 固化方块：void solidify(std::vector<std::vector<int>>& map)  - 获取方块类型：int getBlockType()  - 赋值操作符：Block& operator=(const Block& other)  关键变量说明  - blockType：整型，表示方块的类型标识。  - smallBlocks：包含四个 Point 结构的数组，表示方块的四个小方块在游戏地图中的行列坐标。  - img：指向 IMAGE 的指针，表示当前方块的图像资源。  - imgs：静态数组，包含所有方块类型的图像资源。  - size：静态变量，表示每个小方块的大小。  这些功能和变量共同构成了 Block 类，为俄罗斯方块游戏中方块的行为和视觉表现提供了完整的实现。  **二、Tetris类**  Tetris 类功能说明  各模块功能  1. 构造函数 (Tetris(int rows, int cols, int left, int top, int blockSize)): 初始化游戏的基本参数，包括行数、列数、左边距、上边距和方块大小。  2. 初始化方法 (init()): 设置游戏的初始状态，包括加载图像资源、初始化地图、设置分数和最高分等。  3. 游戏循环 (play()): 游戏的主循环，处理用户输入、方块下落、行消除、游戏结束等逻辑。  4. 按键事件处理 (keyEvent()): 响应用户的键盘输入，实现方块的旋转、加速下落、左右移动等功能。  5. 窗口更新 (updateWindow()): 重新绘制游戏窗口，包括背景、方块、分数等元素。  6. 方块下落 (drop()): 控制当前活动方块向下移动，并在到达底部或碰到其他方块时停止。  7. 行消除 (clearLine()): 检查并消除满行，更新分数和关卡等级。  8. 游戏结束检查 (checkOver()): 判断当前方块是否已填满游戏区域的顶部，从而判断游戏是否结束。  9. 分数保存 (saveScore()): 游戏结束时保存当前分数，并更新历史最高分。  10. 游戏结束界面显示 (displayOver()): 显示游戏结束或胜利的界面。  11. 鼠标事件处理 (mouseClick(int mousex, int mousey) 和 mouseMove(int mousex, int mousey)): 处理鼠标点击和移动事件，用于按钮交互和游戏控制。  算法描述  - 方块下落算法：通过 drop() 方法控制，方块逐行下移，直到碰到其他方块或到达底部。使用一个计时器控制下落速度。  - 行消除算法：clearLine() 方法从下至上检查每一行，如果行满则消除该行，并给予玩家相应的分数。  - 游戏结束判断算法：checkOver() 方法检查方块是否填满了游戏区域的顶部，如果是，则游戏结束。  接口说明  - 构造函数：Tetris(int rows, int cols, int left, int top, int blockSize)  - 初始化方法：void init()  - 游戏循环：void play()  - 按键事件处理：void keyEvent()  - 窗口更新：void updateWindow()  - 方块下落：void drop()  - 行消除：void clearLine()  - 游戏结束检查：void checkOver()  - 分数保存：void saveScore()  - 游戏结束界面显示：void displayOver()  - 鼠标点击事件处理：void mouseClick(int mousex, int mousey)  - 鼠标移动事件处理：void mouseMove(int mousex, int mousey)  关键变量说明  - rows 和 cols：游戏地图的行数和列数。  - leftMargin 和 topMargin：游戏区域在窗口中的左边距和上边距。  - blockSize：单个方块的尺寸。  - map：二维数组，表示游戏地图的状态。  - score、level 和 linecount：分别表示当前分数、当前等级和已消除的行数。  - highscore：历史最高分。  - gameOver 和 stop：分别表示游戏是否结束和是否停止游戏循环。  - curBlock 和 nextBlock：分别表示当前活动方块和下一个预览方块。  - imgBg、imgOver、imgWin 和 imgback：游戏界面相关的图像资源。  - button\_back、button\_continue、button\_again 和 button\_bk：游戏内按钮，用于控制游戏流程。  Tetris 类通过这些功能和变量实现了俄罗斯方块游戏的核心逻辑和界面展示。  **三、Button类**  Button 类功能说明  各模块功能  1. 构造函数 (Button(int x, int y, int width, int height, const std::string& text, const std::function<void()>& onClick)): 初始化按钮的属性，包括位置、尺寸、文本以及点击时的回调函数。  2. 绘制方法 (draw()): 在屏幕上绘制按钮的外观，包括边框、填充颜色和文本。  3. 鼠标悬停检查 (checkMouseOver(int mousex, int mousey)): 检查鼠标是否位于按钮上方，并根据情况改变按钮的缩放比例以实现视觉效果。  4. 点击检查 (bool checkClick(int mousex, int mousey)): 检查鼠标在按钮上方时是否发生了点击事件，并执行绑定的回调函数。  5. 获取宽度 (int getWidth()): 返回按钮的宽度。  算法描述  - 绘制算法：根据按钮的状态（如是否被鼠标悬停）来设置不同的颜色和缩放比例，然后绘制边框、填充颜色和文本。  - 鼠标悬停检测算法：通过比较鼠标的位置与按钮的位置及尺寸来判断鼠标是否悬停在按钮上。  - 点击事件检测算法：在鼠标悬停的基础上，进一步检测鼠标左键的点击动作，触发按钮的回调函数。  接口说明  - 构造函数：Button(int x, int y, int width, int height, const std::string& text, const std::function<void()>& onClick)  - 绘制方法：void draw()  - 鼠标悬停检查：void checkMouseOver(int mousex, int mousey)  - 点击检查：bool checkClick(int mousex, int mousey)  - 获取宽度：int getWidth()  关键变量说明  - x 和 y：按钮左上角的坐标。  - width 和 height：按钮的宽度和高度。  - scale：按钮的缩放比例，用于实现鼠标悬停时的视觉效果。  - isMouseOver：布尔值，表示鼠标是否在按钮上方。  - text：按钮上显示的文本。  - onClick：当按钮被点击时执行的回调函数，类型为 std::function<void()>。  Button 类通过封装按钮的行为和外观，使得在应用程序中创建和使用按钮变得简单和一致。通过回调函数，按钮可以很容易地与其他功能或事件处理逻辑集成。  **四、Page类**  Page 类功能说明  各模块功能  1. 构造函数 (Page(int width, int height)): 初始化页面的基础属性，包括页面的宽度、高度以及当前页面索引。  2. 初始化方法 (initlize()): 进行页面的初始化操作，如加载图形库、加载背景图像、创建按钮等。  3. 运行方法 (run()): 启动页面的事件循环，处理鼠标点击、鼠标移动等事件，并绘制页面。  4. 添加页面 (addPage(IMAGE page)): 向页面管理中添加新的页面图像。  5. 添加按钮 (addButton(int index, Button button)): 在指定页面上添加按钮。  6. 设置当前索引 (setCurrentIndex(int index)): 更改当前显示页面的索引。  7. 鼠标点击事件处理 (mouseClick(int mousex, int mousey)): 处理鼠标点击事件，如按钮点击。  8. 鼠标移动事件处理 (mouseMove(int mousex, int mousey)): 处理鼠标移动事件，如按钮悬停效果。  9. 绘制页面内容 (draw()): 绘制当前页面的内容，包括背景和所有按钮。  10. 排名方法 (ranking(player t[])): 用于显示玩家排名信息。  算法描述  - 页面绘制算法：根据当前页面索引，绘制对应的背景和按钮。  - 事件处理算法：通过消息循环，响应用户的鼠标点击和移动事件，并调用相应的按钮事件处理方法。  - 排名排序算法：对玩家排名信息进行排序，通常是基于分数的降序排列。  接口说明  - 构造函数：Page(int width, int height)  - 初始化方法：void initlize()  - 运行方法：void run()  - 添加页面：void addPage(IMAGE page)  - 添加按钮：void addButton(int index, Button button)  - 设置当前索引：void setCurrentIndex(int index)  - 鼠标点击事件处理：void mouseClick(int mousex, int mousey)  - 鼠标移动事件处理：void mouseMove(int mousex, int mousey)  - 绘制页面内容：void draw()  - 排名方法：void ranking(player t[])  关键变量说明  - width 和 height：页面的宽度和高度。  - currentIndex：当前显示页面的索引。  - pages：存储所有页面图像的向量。  - buttons：二维向量，存储每个页面上的所有按钮。  - settings：存储页面设置的字符数组。  - background0 和 background1：不同类型的背景图像。  - set 和 rank：设置和排名页面的图像。  - key：静态成员变量，用于存储页面的按键状态。  Page 类作为游戏的页面管理器，负责维护游戏的多个页面状态，处理用户交互，并提供页面间的导航功能。通过管理背景和按钮，Page 类为游戏提供了用户界面的基础架构。  **五、music类**  music 类功能说明  各模块功能  1. 构造函数 (music()): 初始化音乐播放器的基本属性。  2. 音乐播放函数 (virtual void musicplay1()): 根据从文件中读取的音乐类型播放相应的音乐。  3. 子类音乐播放函数 (virtual void musicplay2()): 在派生类中重写，用于播放特定的音乐，如游戏结束或胜利的音乐。  算法描述  - 音乐选择算法：musicplay1() 函数通过从文本文件中读取一个整数值来决定播放哪一首音乐。  - 音乐播放控制：使用 mciSendString 函数来控制媒体控制接口播放音乐，包括打开音乐文件、播放和关闭音乐。  接口说明  - 构造函数：music()  - 音乐播放函数：virtual void musicplay1()  - 派生类音乐播放函数：virtual void musicplay2()  关键变量说明  - musictype：整型，表示音乐的类型或编号，用于决定播放哪一首音乐。  子类功能说明  1. gg 类 (class gg : public music): 用于播放游戏结束的音乐。  2. winwin 类 (class winwin : public music): 用于播放游戏胜利的音乐。  子类接口说明  - 游戏结束音乐播放函数：void musicplay2() 在 gg 类中重写，用于播放结束音乐。  - 游戏胜利音乐播放函数：void musicplay2() 在 winwin 类中重写，用于播放胜利音乐。  子类关键变量说明  - 无特定于子类的成员变量，但 musicplay2() 函数在每个子类中使用不同的音乐文件路径。  music 类及其子类通过封装音乐播放的逻辑，使得在游戏的不同阶段播放相应的音乐变得简单和一致。通过读取配置文件和调用媒体控制接口，实现了音乐的动态选择和播放控制。  **六、初始主界面Main函数**  作为主界面主函数，根据用户输入进入设置。  【实现环境】   1. 设计平台要求   Visual Studio 2022   1. 运行环境要求   Win10及以上 |
| 【设计实现及分析】   程序设计和调试过程出现的问题及解决方法，程序测试方法和数据等     1、EasyX没有使用过，不熟悉。  解决方法：上b站看EasyX的教程。  2、画面结构设计  解决方法：根据个人喜好反复实验，敲定大体框架。  3、EasyX的字符串指令报错  解决方法：将字符集设置设置为多字节字符集。  4、生成窗口使用像素大小与设置不符，导致没办法根据像素相对位置直接计算所需的数据  解决方法：上网查找资料，了解是系统 DPI 设置导致，会导致画面根据设置自动放缩。//设定操作系统版本为Win10（需要放到 include <graphics.h> 前面）  //用来使进程不受系统 DPI 设置影响，部分电脑可能有设置画面放缩 125% 或 150% 等的设置  //利用该方法使创建的窗口大小直接对于应电脑像素， 免除了在计算窗口画面像素时不同设备带来的偏差  define WINVER 0x0A00  define \_WIN32\_INNT 0x0A00  include<ShellScalingApi.h>  pragma comment(lib, "Shcore.lib")  加入上述头文件和静态库，并在程序中调用，  //设置该进程不受系统 DPI 设置影响  SetProcessDpiAwareness(PROCESS\_PER\_MONITOR\_DPI\_AWARE);  5、对俄罗斯方块的基本规则不了解  解决方法：网络查找，根据灰机wiki上的俄罗斯方块中文维基，  明确俄罗斯方块的基本方块形态，旋转规则，下降规则等。  6、方块表达方法及其旋转实现  解决方法：在设计之初，没有了解俄罗斯方块的旋转规则，想要模仿现实物理规则  来考虑旋转，即旋转时需要考虑旋转路径上是否有阻挡，思路是对每一个  方块的每一个状态都特别地记录转动时要检测的额外会阻挡的方块，可以  通过多态和虚函数实现。  为了找到更加直接的资料，在维基上找到了关于俄罗斯方块的旋转准则，  发现在经典版本中只有逆时针旋转，以及旋转时只要求下一个状态的位置  为空，根据这个思路，大大地简化我的方块旋转部分制作，同时不同的方块  由于都可以看作有四种状态，且每种状态都是4个格子，使得可以使用统一  的旋转和检测方法，即通过状态转移并直接判断矩阵中是否有空位使旋转  合法。于是也不需要再使用虚函数，只需要写一套使用方块规则即可通用。这种判断方法的思想也同样应用到了移动和下落的的编写中。    对于表达方块，最初想要用一个二维数组去表达某个状态下的方块，  这种方式非常直观，相当于直接在一个4×4的矩阵中把方块画出来。  但很快在实际制作的时候发觉这种方式不仅不便利而且开销浪费大，  于是又想到直接记录以左上角为基准，记录方块格子与其相对位置的坐标，  这种方式不仅简洁，而且在访问时也更加快速直接。  7、生成的方块等常常要发生在局部的函数里  使用new关键字和指针生成对象，将对象生成在堆区，并且用完及时释放内存。  8、使用\_getch()非阻塞获取用户输入函数时，选中游戏运行框无法进行操作，只有  只有选中控制才能输入完成操作，不符合基本游戏逻辑，而且很不方便。  解决方法：上网查询学习使用EasyX自带的ExMessage数据类型，以及相关的  如peekmessage()  等无阻塞获取用户输入函数，实现在选中游戏运行框时能正常输入用户  信息。  9、使用EasyX自带的ExMessage数据类型输入用户信息后，按一次按键，方块移动了  两次。  解决方法：通过调试加入cin来获取并输出自己的输入，发现再cout变量只输出了一次  在调试停住后，再继续程序，方块只移动了一次。  再尝试回原来的程序，  由于怀疑是否是按下按键停留太快的问题，尝试特别快速地按下放开按键，  发现方块依然移动两次。  怀疑是否是按下和抬起按键各记一次，尝试在一次按键中适当放缓抬起速度，使之不会连续输入而只输入一次，发现在按下和抬起时方块各移动一次。  基本确定方块的两次移动是由于程序在获取输入时把按键按下和抬起各视作了一次输入。  最后通过增加一个判断VK\_KEYDOWM，判断是否是按键按下，排除了按键抬起的影响。在一次按下抬起中仅输入一次操作。  10、方块作为一类对象，拥有在矩阵中画图的功能函数，导致该组方块类与该矩阵数组强绑定，为想法中进一步开发双人模式等带来不便。  解决方法：迁移功能函数入Tetrix类，方块类仅保留记录自己形态状态的功能，具体如何被使用不放入方块类中。  11、在音乐板块，需要经常用到字符串作为指令输入，如何优化该部分的信息输入  解决方法：使用sprintf函数，公式化地生成指令。  于是只需要记录关键信息，如音量、歌曲名等，就可以在对音乐部分修改音量、换歌曲等时，通过给出关键信息自主生成对应字符串指令执行。  【讨论】   程序的评价，包括程序结构、时空效率、调试方法等    程序评价是一个全面分析程序各方面性能的过程，以下是对提供的俄罗斯方块游戏程序的评价，包括程序结构、时空效率、调试方法等方面：  程序结构  1. 模块化：程序通过类和对象的方式实现了模块化，如 Button、Block、Tetris 和 Page 等类分别封装了不同的功能，易于理解和维护。  2. 可读性：代码结构相对清晰，函数和变量命名具有一定的描述性，有利于阅读和后续开发。  3. 扩展性：通过面向对象的设计，程序便于扩展新功能，如添加新的游戏模式或改进用户界面。  时空效率  1. 时间效率：程序中的主要循环 Tetris::play() 使用了 Sleep 函数来控制游戏节奏，但可能存在更高效的时间控制方法。  2. 空间效率：使用二维数组存储游戏地图，空间复杂度相对较高，但考虑到游戏规模，这是合理的。  3. 优化：程序中可能存在一些冗余操作，如重复的图像加载和销毁，可以通过优化减少资源消耗。  调试方法  1. 日志输出：程序中使用了 std::cout 进行简单的日志输出，有助于跟踪程序执行流程和发现问题。  2. 断点调试：由于是C++程序，可以使用IDE提供的断点调试功能，逐行执行代码，观察变量状态和程序流程。  用户界面与交互  1. 用户体验：程序提供了基本的用户界面和交互功能，但可能需要进一步的用户测试来评估和改进用户体验。  2. 响应性：程序的响应性取决于事件处理的效率，需要确保在用户操作时能够快速响应。  可维护性与文档  1. 代码注释：代码中的注释较少，建议增加必要的注释来提高代码的可维护性。  【系统使用说明】  使用安装了EasyX库的Visual Statio打开工程文件，运行程序  根据游戏提示使用程序。 |
| **工作小结** |
| 在这次历时五周的C++实训中，我经历了从初步构思到最终项目交付的完整过程，获得了宝贵的编程经验和个人成长。  第一周：项目启动与开题报告  在实训的初期，我主要进行了项目的构思和规划。这一阶段，我深入思考了游戏的设计和功能需求，并撰写了开题报告。虽然进度相对缓慢，但确立了项目的方向和目标，为后续开发奠定了基础。  第二周：核心功能开发  这一周，我专注于实现俄罗斯方块游戏的基本功能。周一，我完成了用户界面的基本布局设计。随后，我着手编码，实现了方块的下落、移动、旋转和消除等核心功能。这些功能的实现，依赖于Block类和Tetris类中的相关方法，如drop()、moveLeftRight()、rotate()和clearLine()。到了周末，游戏已经能够进行基本的方块操作和行消除。  第三周：功能完善与测试  在这一周，尽管其他工作较为繁忙，我依然努力推进项目进度。我为游戏添加了预显示下一个方块的功能，并通过Page类的addButton()方法增加了操作提示。此外，我还将音乐集成到游戏中，使用了music类及其子类gg和winwin来控制不同游戏阶段的音乐播放。尽管尝试了美术方面的改进，但最终由于效果不佳而放弃。这一阶段，我还把程序打包给朋友进行了测试，收集了反馈。  第四周：设置功能与优化  这一周，我专注于完善游戏的设置功能。通过Page类的initlize()和draw()方法，我添加了对音乐和游戏模式等的调整选项，使玩家可以根据自己的喜好定制游戏体验。同时，我对游戏进行了进一步的优化，提升了程序的性能和用户体验。  第五周：实训报告与总结  随着期末考试的临近，我在第五周完成了实训报告的撰写。在报告中，我总结了整个实训过程中的学习成果和经验教训，包括程序设计、代码实现、问题解决和创新实践等方面。我还回顾了Button类和Page类在用户界面设计中的作用，以及Tetris类在游戏逻辑控制中的重要性。  在整个实训过程中，我没有参考网络上现有的游戏代码，而是独立完成了从构思到编程再到实现的全过程。这不仅锻炼了我的编程能力，还提升了我的工程能力、网络信息检索能力和独立学习能力。通过接触程序图形化，我对游戏开发有了更深入的理解。  此外，我还通过撰写博客和制作视频，记录并分享了自己的制作经历。这不仅提升了我的表达和沟通能力，也让我体会到了分享知识的乐趣。  总之，这次C++实训是一次充满挑战和收获的学习之旅。它不仅让我对代码编程有了更加深入的了解，也为我未来的学习和职业生涯奠定了坚实的基础。 |