**深 圳 大 学 实 验 报 告**

**课程名称：­ 计算机图形学**

**实验项目名称： 期中大作业 俄罗斯方块**

**学院： 计算机与软件工程学院**

**专业： 计算机科学国际班**

**指导教师： 周虹**

**报告人123 学号123 班级：计算机科学国际班**

**实验时间：2025年 09月22日 -- 2025年 10月26 日**

**实验报告提交时间： 2025.10.25**

**教务部制**

|  |
| --- |
| 实验目的与要求：  1.强化OpenGL的基本绘制方法、键盘等交互事件的响应逻辑，实现更加复杂的绘制操作，完成一个简化版俄罗斯方块游戏。  2.方块/棋盘格的渲染和方块向下移动。创建OpenGL绘制窗口，然后绘制网格线来完成对棋盘格的渲染。随机选择方块并赋上颜色，从窗口最上方中间开始往下自动移动，每次移动一个格子。初始的方块类型和方向也必须随机选择，另外可以通过键盘控制方块向下移动的速度，在方块移动到窗口底部的时候，新的方块出现并重复上述移动过程。  3.方块叠加。不断下落的方块需要能够相互叠加在一起，即不同的方块之间不能相互碰撞和叠加。另外，所有方块移动不能超出窗口的边界。  4.键盘控制方块的移动。通过方向键（上/下/左/右）来控制方块的移动。按“上”键使方块以旋转中心顺（逆）时针旋转，每次旋转90°，按“左”和“右”键分别将方块向左/右方向移动一格，按“下”键加速方块移动。  5.游戏控制。当游戏窗口中的任意一行被方块占满，该行即被消除，所有上面的方块向下移动一格子。当整个窗口被占满而不能再出现新的方块时，游戏结束。通过按下“q”键结束游戏，和按下“r”键重新开始游戏。  6.其他扩展。在以上基本内容的基础上，可以增加更多丰富游戏性的功能，如通过空格键使方块快速下落等。 |
| 实验过程及内容：  **1. 添加全局数组来表示所有方块的方向和颜色**    把这几个方块的二维数组的指针放入一个数组中，用相同的顺序，指定每种方块下落时对应的颜色，以对它们加以区分：  **2. 修改newTile函数，以实现下落方块的随机初始化，着色和结束判断**  如图，设置了初始的方块位置，随机选择了初始的旋转以及形状。  使用tile[i] = (\*allShapes[currentShapeType])[rotation][i];找到当前方块的每个点的坐标  然后使用if (!checkValid(tilepos + tile[i]))来循环判断每个点是否处于有效位置。  这里设置了一个状态机，指定了游戏的不同状态，之后再说。总之，只要发现一个方块的位置不合法，就把游戏状态直接设置为结束即可，会在状态机循环中进行设置。    最后更新当前方块的VBO, 然后进行上色    **3. 修改checkValid()函数，使其同时检查与已放置方块的碰撞。**  并且在使用这个函数的地方也修改其用法    **4. 实现方块自动向下移动**  在 main 函数的主循环中加入时间控制逻辑  先设置上一次刷新的时间以及每次刷新的时间间隔这两个全局变量：  然后在main函数的循环中设置对时间的反应，如果间隔>=fallInterval，就尝试将当前方块向下移动一个格子，然后固定方块和生成新方块。判断是否加分和游戏是否结束的逻辑都在这两个函数中。注意这里因为创建了状态机，只在STATE\_PLAYING中进行下落的逻辑。    **5. 实现行消除**  完成checkFull()和eliminateRow()函数，并且在setTile中调用这两个函数。  先检查当前行是否所有格子都被填满，如果有一个空的格子就直接return false    如果已经满了，就可以消除这一行，于是从当前行开始，把board(是否被占据)和boundColors这两个变量继承上一行的。因为最顶部的一行没有办法继承，这里不管有没有填充，直接修改为空，修改为黑色即可。  最后，我还加入了得分系统，这里如果行被清除，增加一个得分。    然后修改setTile()函数，因为完成了功能的解耦和，直接在其中调用这两个函数即可。注意，如果当前行被消除之后，已经将上一行移动到当前行，所以y\_check也要减1。这里稍作优化，只用从当前方块到达的最低的一行开始向上检查即可。    **6. 实现游戏重启**  实现restart函数，重置游戏的所有参数    **7. 通过状态机控制游戏的各种状态切换，以及显示/输出逻辑**  先设置一个枚举表示游戏的开始，进行，暂停，结束状态    在newTile()函数中，如果初始方块无法生成，就判定为游戏结束，切换当前状态：    在display()渲染函数中，根据不同的状态进行不同的显示。即，只有在游戏进行状态和暂停状态显示下落的当前方块。另外，由于窗口内绘制文字较为复杂，这里只在命令行打印提示字符。注意\r和std::endl的不同效果。    然后是key\_callback函数，在这里根据不同的状态相应不同的按键。在刚才的display()中已经有了命令行提示：  在任意状态都可以按ESC或者q退出游戏。  在游戏开始状态按ENTER键开始游戏    在游戏进行状态，设置上下左右案件的不同响应，直接调用moveTile()函数移动方块。如果新的位置合法，就更新方块的位置和显示，否则返回false    上键是旋转，左右键是左右移动，下键是向下移动，空格键也可以向下移动。新增了一个p键，用于切换到暂停状态。    在暂停状态下，只相应p键，即恢复游戏。在游戏结束状态下，只响应R键重新开始。     1. **新增功能, 使用json文件存储游戏的历史记录**   存档功能是非常实用的一个扩展内容, 我准备引入一个json库(nlohmann-json)来对json文件进行解析.  这里, 我切换成了windows + vcpkg进行包管理, 因为更方便跨平台编译文件  首先使用vcpkg(而不是我之前使用的pacman)安装对应的库, 然后修改cmakelist文件:  在loadAndSave.h中定义了GameRecord类来管理游戏记录(在这里#include <nlohmann/json.hpp> )    首先在loadAndSave.cpp中实现了几个辅助函数. 其中, setDateTime函数实现了记录当前时间,并且对其进行格式化:    然后是加载游戏记录的函数(之后还有几个错误处理分支, 没有截图):  然后创建一个可以调用加载的游戏记录数组, 格式化打印到控制台的函数. 在其中, 我根据用户得分从高到低进行了排序:    然后是存储当前的游戏信息的函数:  在main.cpp中, 需要调用这些函数. 首先是, 在进入渲染循环之前, 先从控制台获取用户id(如果用户留下id), 再打印所有的记录.  这个辅助函数在命令行与用户交互, 获取用户输入的用户名. 如果不愿意输入用户名, 就使用默认用户名Guest  新建了一个辅助函数gameover(), 在newTile和退出状态的时候调用这个函数, 调用loadAndSave中的saveGameRecord()函数, 将游戏状态保存到json文件中    最后的json文件如图所示:   1. 运行截图     如图所示, 我的俄罗斯方块程序主要通过控制台进行交互.  左侧控制台程序展示了游戏交互的信息, 包括这个简易的    这个排行榜可以看到所有用户的用户名, 游戏时间, 游戏时长, 并且以得分从高到低排名    这是会遇到的7种不同方块, 每种方块都固定设置为一种颜色. 按上箭头可以切换方向      固定的方块变为灰色, 和彩色的下落方块以及黑色的背景相区分  反复按p键可以从游戏中暂停, 在暂停中按p键可以恢复游戏:    按q或者ESC键随时可以退出游戏.  在开始界面先确认, 然后设置用户名, 然后按ENTER进入游戏  最后, 如果生成的新方块和已有的方块发生冲突, 就结束游戏:  以上就是我实现的主要功能. 为了方便助教和老师运行文件, 我特意迁移到行amd64+windows+msvc+vcpkg的环境下进行了开发, 也提交了可执行文件 |

深圳大学学生实验报告用纸

|  |
| --- |
| 实验结论：  **一、 实验结论**  本次期中大作业，我基于提供的框架代码，成功实现了一个功能相对完整的俄罗斯方块游戏。我主要完成的工作包括：   1. 实现了所有7种方块, 定义了包括 I, O, S, Z, J, L, T 在内的所有标准方块形状及其旋转状态的坐标数据。 2. 完成了核心游戏逻辑：   实现了方块的随机生成（包括形状和颜色）与自动下落（基于时间）。  完善了碰撞检测逻辑，确保方块能在边界内移动，并能正确堆叠在其他已固定方块上。实现了方块的固定 (setTile) 和行消除 (checkFull, eliminateRow) 机制，包括消除后上方方块的下落。   1. 实现了完整的游戏流程控制：   引入了游戏状态机 (GameState)，管理开始界面、游戏进行、暂停和游戏结束四个状态。实现了相应的键盘交互逻辑（Enter 开始，P 暂停/恢复，R 重新开始，Q/ESC 退出）。   1. 添加了数据持久化:   引入了 nlohmann/json 库，并创建了 loadAndSave 模块。  实现了在游戏开始前获取用户名，并在游戏结束后自动保存包含用户名、得分、时长和时间的游戏记录到 JSON 文件。  实现了在游戏启动时加载并打印历史游戏记录的功能。   1. 解决了跨平台和环境问题:   在 Windows 环境下配置了 vcpkg 来管理 glfw, glm, nlohmann-json 等依赖。  解决了跨平台 API 兼容性问题以及 MSVC 编译错误。  **二、 心得体会**  这次大作业是对开学以来所学知识的一次非常好的综合实践。通过从一个基础框架开始，逐步添加功能，我更深入地理解了如何将 OpenGL 的渲染能力与具体应用（游戏）的逻辑结合起来。  实现俄罗斯方块的核心挑战在于状态管理和逻辑判断。例如，精确的碰撞检测 (checkValid)、方块旋转的有效性判断 (rotate) 以及行消除后的连锁反应 (setTile 调用 checkFullRow) 都需要非常仔细地考虑各种边界情况。调试这些逻辑虽然花了不少时间，但也让我对在二维数组（棋盘格 board）上处理游戏逻辑有了更深的体会。  引入游戏状态机 (GameState) 让整个程序的流程控制清晰了很多，避免了用一堆布尔标志位管理状态可能带来的混乱。同时，学习使用 vcpkg 管理依赖，以及解决 Windows 环境下的编译问题，也提升了我在不同平台上配置和构建 C++ 项目的能力。  最后，实现 JSON 文件读写来保存游戏记录，让我接触到了 C++ 中数据持久化的基本方法。我完成了需求理解、任务分解、代码调试等流程，并最终看到一个完整的游戏能够运行起来，还是非常有成就感的。这次经历让我对用 C++ 和 OpenGL 开发简单游戏的全过程有了更完整的认识。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。