浙江大學

魔方模拟器个人分报告

授课教师：	袁昕
姓　　名：	王伟杰
组　　别：	魔方模拟组
日　　期：	2023-7-10

1 分工任务

我作为本次项目的组长，负责统筹组员的工作，并确保项目顺利进行。

在我们课程的第二天，我们确定了开发一个魔方模拟器的项目，并选择使用Xmake作为项目构建工具。随后，我们开始采用MVVM模型构建整个项目的框架。

在第一轮迭代中，我承担了App层和Common层的实现任务。我负责确保应用层与底层的协调和顺畅交互，以及实现通用功能模块的开发。通过我们的工作，我们成功实现了魔方的正常显示、视角移动、魔方旋转和透明度改变等功能。

在第二轮迭代中，我转变了角色，负责了View层的开发工作以及文档撰写。我设计了更好的用户界面，确保用户能够直观地操作魔方模拟器，并编写了清晰详细的中期文档，方便团队成员之间的沟通和交流。

在第三轮迭代中，我负责ViewModel层和Model层的优化和添加新功能的工作。我致力于提升魔方模拟器的性能和用户体验，优化数据处理和逻辑处理模块，并使得魔方可以完成解魔方的任务，并对传统魔方算法进行优化。

通过我的分工和协调，每个迭代阶段的任务得到了明确的分配，使得我们能够高效地推进项目的开发。在整个过程中，我与团队成员紧密合作，利用GitHub的版本控制工具确保代码的协同开发和顺利合并。

2 设计思路

2.1 第一次迭代

第一次迭代的任务是App层和Common层，主要是对整体通讯机制的梳理和构建。

2.1.1 MVVM设计思路

在Model，View，ViewModel三个模块之间，View与ViewModel之间的数据通过App层进行**双向绑定**进行联系，View与Model之间不产生联系，ViewModel操作Model进行数据处理。

2.1.2 消息机制

在View层进行操作后，会触发相应的槽函数。槽函数会准备好参数，并将其传递给对应的Command。然后，执行该Command的exec()方法，该方法会解析参数并将其传递给ViewModel层。ViewModel层调用Model层中相应的方法进行数据操作。操作完成后，Model会通知ViewModel更新显示数据，而ViewModel会通知View刷新显示。

2.1.3 命令虚类构建

为了实现命令模式，在Viewmodel层中构建了一个命令基类。这个命令基类提供了一些通用的方法和属性，供具体的命令类继承和实现。

2.1.4 宏编写

为了简化程序编写，编写了一些宏。这些宏可以帮助减少重复的代码，提高代码的可读性和可维护性，可以简化代码，用于后续序列化、反序列化等操作。例如：

```
#define GET_FRONT(x) ((CubeColor)(((x)&0xF00000)>>20))
#define SET_FRONT(x) (((x)&0xF)<<20)
#define MAKE_CUBE(f, b, l, r, u, d)
(SET_FRONT(f)|SET_BACK(b)|SET_LEFT(l)|SET_RIGHT(r)|SET_UP(u)|SET_DOWN(d))
#define ROTATE_LEFT(x)  MAKE_CUBE(GET_RIGHT(x), GET_LEFT(x), GET_FRONT(x), GET_BACK(x),
GET_UP(x), GET_DOWN(x))
```

2.2 第二次迭代

第二次迭代的任务是View层，主要是命令展示。

2.2.1 命令展示

由于OpenGL库中没有对应的字体渲染函数，在不引入第三方库的前提下，考虑到每次输入的命令较短，我选择直接将命令展示在标题上，在渲染的时候重新设置标题。

2.2.2 添加新命令

对于新的random、load、save和reset命令，我在EventCallback类中添加了新的识别函数。

2.3 第三次迭代

第三次迭代的任务是ViewModel层和Model层，主要是魔方复原算法的实现和复原算法的优化。

2.3.1 复原算法

参考资料：https://rubiks.com/en-US/solve-it

参考网上找到的公式，我将其写成了CubeSolver，使之可以按照公式解决魔方问题，其中有7个状态，根据公式层层递进，一步步推演就可以得到最后复原的方案。这部分需要对Model层中的数据进行修改，我在之前的基础上增加了check函数用于检测现在魔方的状态，并在Viewmodel中进行魔方状态的转换。

2.3.2 算法优化

上一步推演出的算法在一些步骤上存在冗余的情况，如果作为参考可能会影响观看体验，所以在此基础上，我构建了两个过滤器用于缩短复原步骤。

- NoXYZFilter

  通过使用一个名为`XYZMapTables`的映射表来存储旋转方法之间的映射关系。

  `XYZMapTables`是一个嵌套的`std::map`，它的键是旋转方法（`CubeRotateMethod`），值是另一个`std::map`，用于将当前旋转方法映射到下一个旋转方法。

  `Filter`方法接受一个魔方还原步骤（`CubeSteps`），然后使用`XYZMapTables`和当前映射表对每个步骤进行过滤。如果当前步骤是一个**整体旋转**（`isWholeRotate`），则将当前映射表与相应的整体旋转映射表合并，并将该步骤设置为无操作（`ROTATE_NONE`）。否则，将当前步骤替换为映射表中的下一个步骤。

  最后，将过滤后的步骤复制到一个新的步骤列表中并返回。

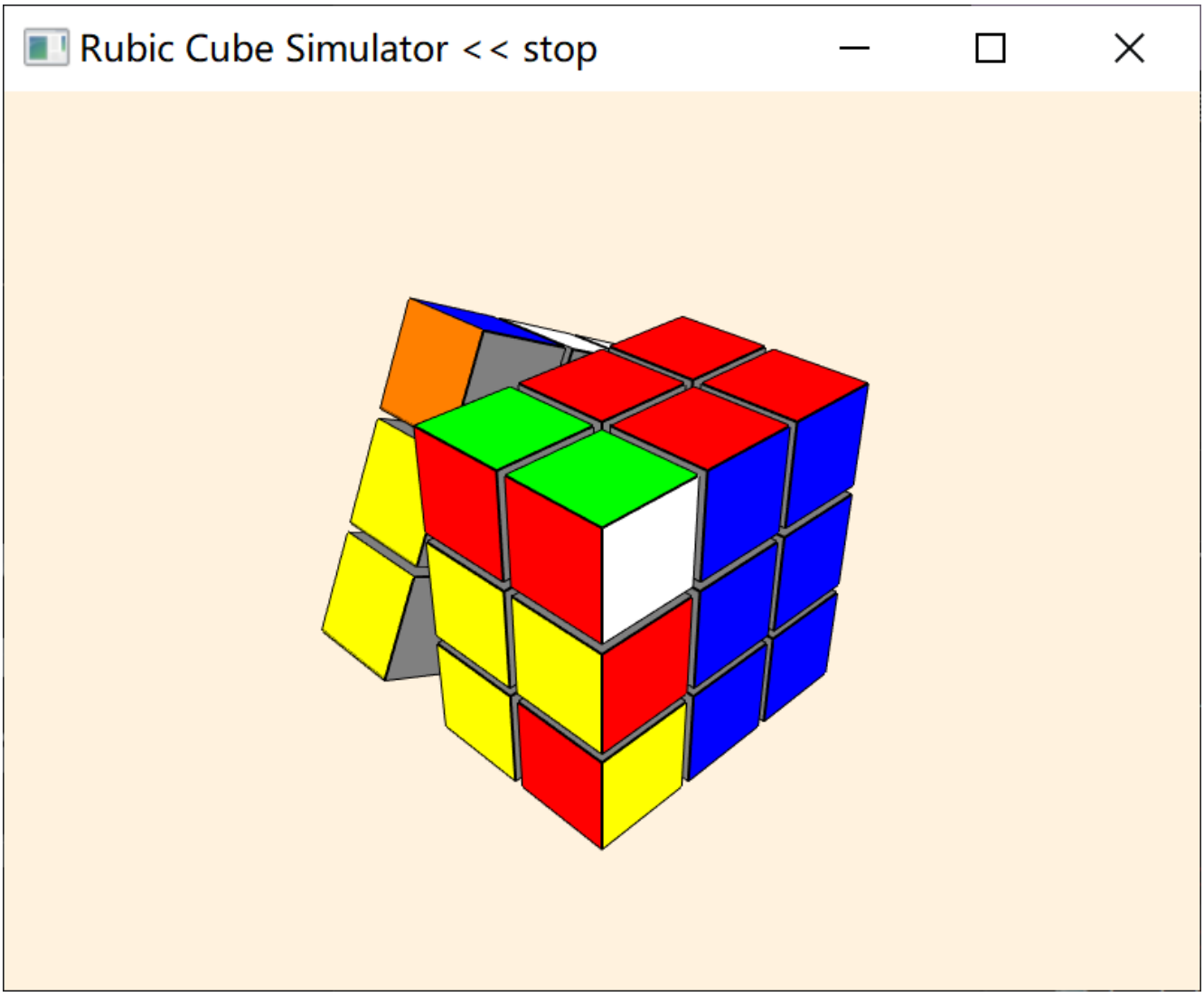
- ReduceFilter

这个过滤器通过检测**连续相同操作**和**互为逆操作**的步骤，并将其缩减为无操作，从而进一步优化魔方的还原步骤。`Filter`方法接受一个魔方还原步骤（`CubeSteps`），并使用两个辅助方法`ReduceContinuous`和`ReduceInverse`来不断缩减步骤，直到无法再缩减为止。在每次迭代中，它将复制当前步骤列表到一个新的列表中，并使用新的列表进行下一轮迭代。

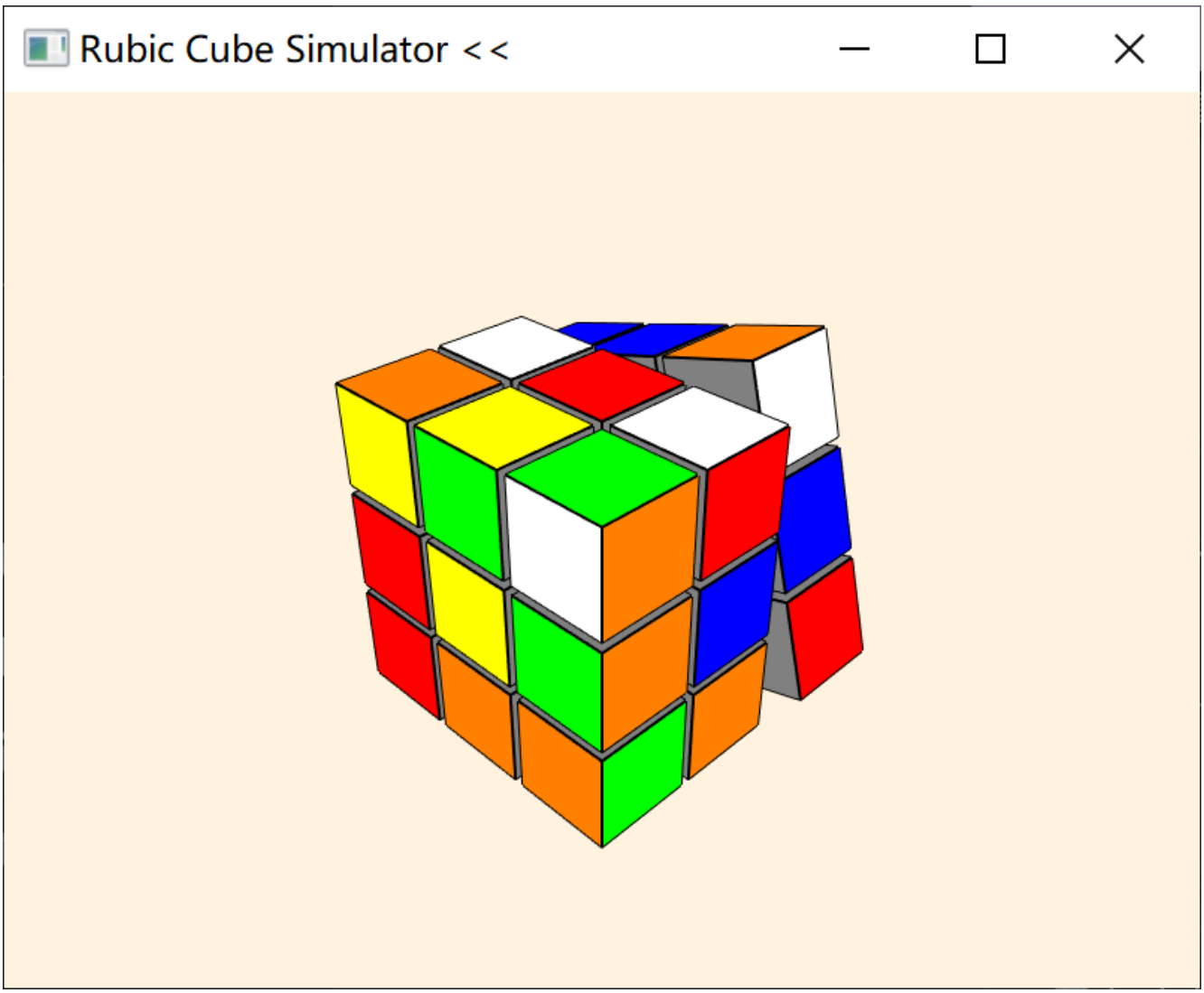
- `ReduceContinuous`方法用于缩减连续相同操作的步骤。它遍历步骤列表中的元素，检查是否有连续的三个相同的步骤。如果找到了这样的连续步骤，它将将第一个步骤替换为其逆操作（`inverse`），并将第二个和第三个步骤设置为无操作（`ROTATE_NONE`）。然后将`found`标记为`true`，表示找到了可缩减的步骤。
- `ReduceInverse`方法用于缩减互为逆操作的步骤。它遍历步骤列表中的元素，检查相邻的两个步骤是否是互为逆操作。如果是，它将将这两个步骤都设置为无操作（`ROTATE_NONE`）。然后将`found`标记为`true`，表示找到了可缩减的步骤。

3 运行效果图

展示命令：



正在自动复原的魔方：



4　心得体会

学习这个课程之前，我对C++的项目管理以及各种软件架构模式没有任何了解，之前在其他课程上和组员合作完成项目时也体会到了很多合作开发的困难。在学习这门课时，我不仅在课堂上学到了很多团队开发和程序框架相关的知识，还亲身体验了团队项目工程的合作开发。在我们完成MVVM项目框架的时候，我深刻体会到了这个模式的便捷之处，每个人都只专注于自己的模块，不需要等待其他人的代码，而且在多次迭代的过程中，项目成员也可以对整个项目的代码进行一遍完整的开发，让大家都能学到很多东西。

在第一轮迭代中，我负责了Common层和App层以及xmake项目管理的任务。通过这个阶段的工作，我对C++和MVVM模式有了更深入的了解。在第二轮迭代中，我负责了View层的开发以及文档的撰写。这使我能够进一步加深对用户界面和文档编写的理解。通过GitHub的版本控制和分支管理，我能够与其他组员无缝协作，确保代码的顺利合并和项目的进展。