|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
| **计算机图形学实验报告**  计算机图形学第九小组   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 姓名 | 刘麦 | 梁增权 | 胡涵哲 | 高健程 | 钟杨琦 | 王增超 | | 学号 | 21301043 | 21301042 | 21301040 | 21301064 | 21301088 | 21301082 | | | |

目录

[1 实验目的 3](#_Toc164202652)

[2 实验环境 3](#_Toc164202653)

[3 实验原理 3](#_Toc164202654)

[4 实验内容 4](#_Toc164202655)

[5 实验结果 4](#_Toc164202656)

[6 实验总结 5](#_Toc164202657)

[7 实验分工 8](#_Toc164202658)

# 1 实验目的

通过本次实验，进一步理解和掌握二维、三维几何变换的基本原理；理解和掌握视图变换、投影变换的基本原理。加深对二维、三维几何变换基本原理的理解，以及对视图变换、投影变换的掌握。通过实践，掌握OpenGL（或其他API）实现二维、三维图形绘制和变换的技能，并且加强团队合作能力。

# 2 实验环境

编程语言：OpenGL（或其他API）开发工具：OpenGL开发环境操作系统：Windows 10

# 3 实验原理

1. 二维图形绘制：原理： 使用OpenGL的绘图函数在屏幕上绘制二维图形，通过指定绘图模式和顶点坐标来确定图形的形状和位置。方案设计： 在OpenGL初始化后，设置视口和投影方式，然后在渲染循环中使用绘图函数绘制所需的二维图形。2. 二维变换：原理： 通过修改OpenGL的变换矩阵来实现二维图形的缩放、反射、切变、旋转、平移等操作。方案设计： 在绘制二维图形前，通过修改OpenGL的模型视图矩阵来实现所需的变换操作，然后再进行绘制。3. 二维组合变换：原理： 将多个二维变换操作按照一定顺序组合起来，对图形进行复合变换。方案设计： 将多个变换操作的变换矩阵相乘，然后应用到绘制的图形上，实现复合变换效果。4. 三维图形绘制：原理： 使用OpenGL的绘图函数绘制三维图形，如立方体、球体等。方案设计： 在OpenGL初始化后，设置视口和投影方式，然后在渲染循环中使用OpenGL的绘图函数绘制所需的三维图形。5. 三维变换：原理： 通过修改OpenGL的变换矩阵来实现三维图形的缩放、平移、绕x/y/z轴旋转等操作。方案设计： 在绘制三维图形前，通过修改OpenGL的模型视图矩阵来实现所需的变换操作，然后再进行绘制。6. 三维组合变换：原理： 将多个三维变换操作按照一定顺序组合起来，对三维图形进行复合变换。方案设计： 将多个变换操作的变换矩阵相乘，然后应用到绘制的三维图形上，实现复合变换效果。7. 键盘控制及投影类型切换：原理： 使用OpenGL的键盘事件处理函数捕获键盘输入，根据输入进行相应的操作。方案设计： 在渲染循环中监听键盘事件，根据按键输入来移动相机或切换投影类型，然后重新渲染场景。

# 4 实验内容

1. 二维图形绘制：初始化OpenGL环境。设置视口和投影方式为二维。在渲染循环中使用OpenGL的绘图函数绘制所需的二维图形，如点、线、多边形等。

2. 二维变换：在绘制二维图形前，通过修改OpenGL的模型视图矩阵来实现所需的变换操作，如缩放、反射、切变、旋转、平移。根据需要选择合适的变换参数，将变换矩阵应用到绘制的图形上。

3. 二维组合变换：将多个二维变换操作的变换矩阵相乘，得到一个复合变换矩阵。将复合变换矩阵应用到绘制的二维图形上，实现多个变换的组合效果。

4. 三维图形绘制：初始化OpenGL环境。设置视口和投影方式为三维。在渲染循环中使用OpenGL的绘图函数绘制所需的三维图形，如立方体、球体等。

5. 三维变换：在绘制三维图形前，通过修改OpenGL的模型视图矩阵来实现所需的变换操作，如缩放、平移、绕x/y/z轴旋转。根据需要选择合适的变换参数，将变换矩阵应用到绘制的图形上。

6. 三维组合变换：将多个三维变换操作的变换矩阵相乘，得到一个复合变换矩阵。将复合变换矩阵应用到绘制的三维图形上，实现多个变换的组合效果。

7. 键盘控制及投影类型切换：在渲染循环中监听键盘事件。根据按键输入来移动相机或切换投影类型。根据不同的键盘输入，修改视图矩阵或投影矩阵，然后重新渲染场景。

# 5 实验结果

在实验中，我们成功实现了以下功能：（1）、二维图形绘制：使用OpenGL绘制了点、线、多边形等二维图形。（2）、二维变换：实现了缩放、反射、切变、旋转、平移等二维变换操作。（3）、二维组合变换：将多个二维变换操作按照一定顺序组合起来，对图形进行复合变换。（4）、三维图形绘制：使用OpenGL绘制了立方体、球体等基本的三维图形。（5）、三维变换：实现了缩放、平移、绕x/y/z轴旋转等三维变换操作。（6）、三维组合变换：将多个三维变换操作按照一定顺序组合起来，对三维图形进行复合变换。（7）、键盘控制及投影类型切换：通过键盘输入实现了相机的移动和透视投影与正交投影的切换。

从中，我们可以观察到以下几点：（1）、变换效果： 通过对二维和三维图形的变换操作，可以明显地看到图形的形状、大小和位置发生了变化。例如，缩放操作会改变图形的大小，旋转操作会使图形绕着指定的轴旋转等。（2）、组合变换： 在实现组合变换时，需要注意变换的顺序对最终效果的影响。不同顺序的变换组合会产生不同的结果，因此需要根据具体需求选择合适的变换顺序。（3）、键盘控制： 键盘控制功能提供了交互性，使用户能够通过键盘输入实时控制场景的展示。这增强了用户体验，使得场景的观察和控制更加灵活和直观。（4）、投影类型切换： 透视投影和正交投影各有其特点，透视投影能够呈现出景深感，而正交投影则能够更准确地显示物体的大小和形状。通过切换投影类型，用户可以根据需要选择合适的投影方式。

总的来说，实验结果符合预期，各功能正常运行并能够达到预期的效果。

# 6 实验总结

刘麦：

在本实验中我使用了CLion进行编程，官网教程以csdn等大多都是基于Visual Studio，所以在配环境时遇到了一些困难

解决方法：

1配置glfw

从官网下载 GLFW 源码并解压。 使用 CLion 不需要下载 CMake，直接打开工程并选择glfw 的根目录。然后需要手动设置编译库文件。依次在设置需要在设置的CMake中更改缓存变量。最后构建项目。

2配置 GLAD 环境

在编译 时无法确定大多数函数的位置，需要在运行时查询并保存函数地址供后续使用，需要在运行时获取函数地址并将其保存在函数指针中。可以使 用 GLAD 库来实现https://glad.dav1d.de选择相应的信息后下载。

3新建OpenGL工程

对于 CLion 来说，需要手动配置外部库和链接关系。新建一个工程，打开其中CMakeLists.txt 文件并做出修改，将路径指向新建的文件夹，文件夹包括include ：glfw 和 glad 的 include，lib： glfw3.dll 和 libglfw3dll.a ，src：glad.c。

在这次实验中，我使用了CLion集成开发环境，并遇到了与Visual Studio不同的配置挑战。虽然我对编译和cmake知识不熟悉，但通过查阅教程和博客，我成功地配置了环境并编译了GLFW源码。在创建OpenGL项目时，我学会了如何配置CMakeLists文件来链接外部库，并解决了运行时的错误，深入理解了项目配置和库链接。OpenGL的学习让我直观地理解了渲染管线和着色器，特别是通过编写自定义着色器，我对着色器的功能有了更深的了解。尽管学习过程中遇到了挑战，但我期待继续学习OpenGL，提升我的图形编程技能。

梁增权：

在这个实验中，我使用OpenGL实现了二维图形的绘制和变换，包括缩放、反射、切变、旋转和平移，以及二维组合变换。这是第一次使用OpenGL库编写代码，在配环境和写代码的过程中遇到了许多困难。

在实现二维变换的过程中，我遇到了一些困难。将变换矩阵定义为三维时，使用glUniformMatrix3fv方法无法实现平移变换，查了许多资料也无法解决，后面将变换矩阵定义为四维，使用glUniformMatrix4fv很好的解决了这个问题。

在这个实验中，必不可缺的就是变换矩阵。变换矩阵是实现二维变换的核心概念。我熟悉了变换矩阵的构造和作用，能够更好地掌握图形的变换过程，并且能够灵活地应用到不同的场景中。

通过这个实验，我不仅学习了OpenGL的基本绘图和变换原理，还锻炼了编写OpenGL程序的能力。同时，实验也展示了二维图形变换的重要性和应用场景，对于理解计算机图形学中的基本概念有很大帮助。

这个实验是一次很好的学习和实践机会，我通过实际编写代码加深了对二维图形和OpenGL的理解，也提高了自己的编程能力。在实验中我收获了很多知识和经验，也提升了自己的能力和信心。我对图形学领域的奥妙有了更深的体会。图形学涉及到数学、物理和计算机科学等多个领域，通过对图形学的学习和实践，我感受到了其中的无穷魅力。

王增超：

本次实验我主要负责三维变换的缩放，三维组合变换和添加键盘对场景的控制。使用了vscode进行代码编写，通过OpenGL提供的函数进行图形操作。

相关环境配置来自csdn和b站

首先我按照教程安装MinGW中的gcc，g++和gdb编译调试器，并添加环境变量。然后配置OpenGL环境时编译器选择32位。最后在网站上生成glad的静态库，在glad路径下生成glad.o与libglad.a文件。将所有文件添加进vscode后进行测试。环境的配置并没有遇到问题。

在三维变换中，OpenGL提供的矩阵函数可以实现大部分需求。在实现三维组合变换时，我将不同的变换操作按照一定顺序组合起来，然后应用到图形上，就实现了对图形的多个操作。最后通过不断的尝试找到一个相对合适的图形旋转和平移速度。

添加键盘对场景的控制时我调用了GLFW库中的一个函数glfwGetKey，它可以用于查询指定的按键当前是否被按下，检测到按键被按下时会执行相关操作。

实际上手操作让我体会到了计算机图形学的魅力，通过代码使图形动起来的过程非常具有吸引力，期待在后续的学习中不断提高自己的能力，能够使的图形更加生动。

高健程：

在实验中，我承担了三维图形绘制和变换的任务，其中缩放变换是我主要负责的部分。这项任务并不轻松，因为确定缩放因子需要考虑到图形的当前状态和所需的变换效果。一开始，我感到困惑和挣扎，因为不同的缩放因子可能会导致完全不同的结果。例如，过大的缩放因子可能导致图形变得过于扁平或过于密集，而过小的缩放因子则可能导致图形消失在视野之外。因此，我花了大量的时间研究数学原理，并进行实践练习，以找到最佳的缩放因子计算方法。

在解决问题的过程中，我还遇到了一些编程上的挑战。特别是，如何将数学原理转化为可执行的代码是一个需要细心思考的问题。我不断地尝试不同的算法和数据结构，寻找最适合实验要求的实现方式。同时，我也和团队的其他成员进行了充分的讨论和合作，共同解决了遇到的问题，并相互学习、取长补短。

通过这次实验，我不仅加深了对三维图形绘制和变换的理解，还提高了我的解决问题和团队协作能力。我学会了如何克服困难，不断地探索和学习，并将理论知识与实践应用相结合。这次经历不仅提升了我的专业技能，也为我未来的学习和工作打下了坚实的基础。

胡涵哲：

本次实验我主要负责三维图形的绘制及旋转平移，使用vscode进行代码的编写，但是在配置环境时遇到了不少问题。

首先在然后安装了MinGW中的gcc、g++以及gdb编译调试器。接着将MinGW文件下的bin文件添加到系统环境变量中。

接着在vscode中，按键Ctrl+Shift+P，输入Create C++ project试图创建一个C++文件工程，发现无法创建，发现是因为vscode中没有安装插件C/C++、C/C++ Project Generator，安装后问题得以解决。

接下来开始配置OpenGL环境，首先下载glfw，将解压包中的glfw-3.3.6.bin.WIN32\glfw-3.3.6.bin.WIN32\include\下的GLFW文件，复制到vscode创建的C++工程下的include文件夹下。

将解压包中的glfw-3.3.6.bin.WIN32\glfw-3.3.6.bin.WIN32\lib-mingw路径下的libglfw3.a与libglfw3dll.a两个文件，复制到vscode创建的C++工程下的lib文件夹下。

将解压包中的glfw-3.3.6.bin.WIN32\glfw-3.3.6.bin.WIN32\lib-mingw路径下的glfw3.dll文件，复制到vscode创建的C++工程下的output文件夹下。

再配置glad，将glad\include路径下的两个文件夹，复制到vscode创建的C++工程下的include文件夹下。将glad路径下生成的libglad.a文件，复制到vscode创建的C++工程下的lib件夹下。环境配置完成。

在绘制三维图形时，我发现绘制出来的立方体没有边框，是纯白色，最后我修改了“render”函数，加了这两行代码：

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

glColor3f(1.0f, 1.0f, 1.0f);

修改绘制模式为线框，然后再次调用 glDrawElements 来绘制边框。这样就可以同时绘制出立方体的实体表面和边框了。

期待在后期的学习中继续提升自己的能力！

钟杨琦：

本次实验我主要负责三维图形的视角移动和投影切换部分。

首先我在vscode上进行了必要的环境配置。通过查阅资料，配置了MinGW中的gcc、g++以及gdb编译调试器.配置的过程，遇到了现有的安装版本与教程中的版本不一致的问题，这导致了配置c\_cpp\_properties.json和Makefile文件的时候，出现了找不到对应的文件，或者其中的代码。通过不断尝试，尝试修改版本号，完成了样例的运行和glad、GLFW环境的配置。

然后在小组其他同学完成的代码基础上，进行了视角移动和投影切换部分的编译。最开始，我另写了一个新的函数尝试完成视角的移动。通过不断的调试和debug，在代码不暴红的情况下，成功运行。但是，理想中按asdw进行的视角移动却没有实现。通过查阅资料和chatgpt，最开始我以为是必须要导入glm库，并且引入getLookAtMatrix方法才能实现。但是，我尝试引入新库的时候，还是无法实现视角移动功能。最后，我将一切推翻重做。应用平移来移动相机 glTranslatef(-cameraX, -cameraY, -cameraZ); 最终实现了相机视角移动。

在完成视角移动的功能后，投影切换功能便是相对容易的实现。通过glfwGetKey方法检测按键的输入，glFrustum 和glOrtho方法实现不同投影的切换。

通过上手实践，我不仅掌握了视角移动和投影切换的实现方法，还提高了自己的编程能力和解决问题的能力。我深刻体会到，学习三维图形渲染需要耐心和细心，同时也需要不断尝试和探索。

# 7 实验分工

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 刘麦 | 梁增权 | 胡涵哲 | 高健程 | 钟杨琦 | 王增超 |
| 学号 | 21301043 | 21301042 | 21301040 | 21301064 | 21301088 | 21301082 |
| 班级 | 2102班 | 2102班 | 2102班 | 2103班 | 2103班 | 2103班 |
| 职务 | 组长 | 组员 | 组员 | 组员 | 组员 | 组员 |
| 工作量 | 16.7% | 16.7% | 16.7% | 16.7% | 16.7% | 16.7% |

梁增权、刘麦：二维图形绘制和二维变换、键盘控制及投影类型切换、制作PPT。

高健程、钟杨琦、王增超、胡涵哲：三维图形绘制和三维变换、键盘控制及投影类型切换、撰写实验报告。