

**实验目的**

本实验旨在通过使用OpenGL图形库，深入理解和掌握计算机图形学的基本概念和技术，并具体实现以下任务：

1.实现二维图形绘制：通过OpenGL库实现基本的二维图形绘制，包括点、线、多边形等基本图形的绘制。

2.进行二维变换：掌握OpenGL中的二维变换函数，包括缩放、反射、切变、旋转、平移等操作，能够对二维图形进行各种变换操作。

3.实现二维组合变换：将多个二维变换操作组合在一起，实现对二维图形的复杂变换效果。

4.实现三维图形绘制：使用OpenGL库实现基本的三维图形绘制，包括立方体、球体等基本的三维几何体的绘制。

5.进行三维变换：掌握OpenGL中的三维变换函数，包括缩放、平移、绕x/y/z轴旋转等操作，能够对三维图形进行各种变换操作。

6.实现三维组合变换：将多个三维变换操作组合在一起，实现对三维图形的复杂变换效果。

7.添加键盘控制：实现通过键盘控制场景的功能，包括移动观察相机、在透视投影和正交投影间切换，提高交互性和用户体验。

通过完成以上任务，学生将能够全面了解和掌握OpenGL图形库的基本功能和使用方法，深入理解计算机图形学中的二维和三维图形表示与变换原理，为进一步学习和应用计算机图形学提供坚实的基础。

**实验环境**

Windows10系统

Visual Studio2022+OpenGL

Python+OpenGL

C++用到的库：GLFW,GLAD,GLUT,CMATH

Python用到的库：Pygame,MATH

**实验原理**

二维：建立图形变换的参考坐标系，以便于描述对象几何信息的数值，对于一个二维图形作平移、旋转、放缩变换，可以转换为在二维坐标系中图形的所有点分别可以对应到在x，y轴方向分别平移tx，ty（平移）、绕一点旋转固定的角（旋转）、在x，y轴方向分别放缩sx，sy倍，将原图形的点通过极坐标或者相加、相乘，结合二维矩阵实现二维变换

三维：定义立方体的顶点坐标和索引，定义旋转角度，旋转速度，相机位置，平移速度，

对于一个三维图形作平移、放缩变换、xyz轴旋转，通过调用glScalef，glTranslatef，glRotatef函数来实现放缩，平移，旋转，通过glFrustum设置透视投影的视景体，glOrtho设置正交投影

**实验内容**

**二维：**

1. 初始化窗口和OpenGL环境

调用pygame.init()初始化Pygame，使用pygame.display.set\_mode()创建窗口，gluPerspective()设置透视投影矩阵，glTranslatef()移动摄像机位置

2.绘制坐标轴和原始图形

通过draw\_axes()函数绘制X轴和Y轴，并在坐标轴上添加了标刻度。使用draw\_square()函数绘制了一个原始大小的绿色正方形

3.绘制变换图形

使用 draw\_scaled\_square(scale\_factor) 函数绘制蓝色正方形，并进行缩放操作。然后定义反射后的顶点，沿Y轴和X轴进行反射。接下来，利用 draw\_rotated\_square(angle\_degrees) 函数计算旋转后的顶点坐标。最后，利用 draw\_translated\_square(translation\_x, translation\_y) 函数进行平移操作。

**三维：**

1. 创建并设置窗口

创建一个800x600的窗口，并将其设置为当前的OpenGL上下文，然后初始化GLAD来加载OpenGL函数指针。最后，设置视口大小为整个窗口大小(800x600)。这样可以确保后续的OpenGL渲染能够正确显示在窗口中。

1. 设置透视投影矩阵

创建一个透视投影矩阵(projMatrix)。用于将场景中的点从视锥体空间变换到裁剪空间，在裁剪空间中进行透视除法，最终得到规范化设备坐标(NDC)，从而进行后续的顶点处理和光栅化操作。

基于透视投影的原理，通过计算透视投影矩阵的各个元素，来实现将3D场景投影到2D平面上。

1. 循环渲染

启用深度测试，以便在渲染时根据深度值决定哪些像素应该被绘制。

设置深度测试函数，将深度测试函数设置为小于等于（GL\_LEQUAL），意味着如果像素的深度值小于等于当前深度缓冲区中的深度值，则通过深度测试。

在循环中，不断更新角度（可能是基于时间的旋转），然后调用render(window)函数进行渲染。

1. 相机移动

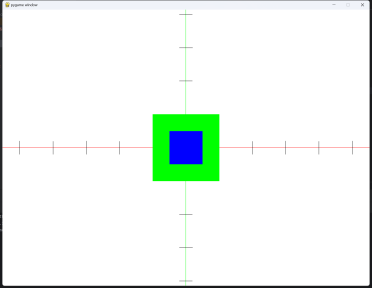
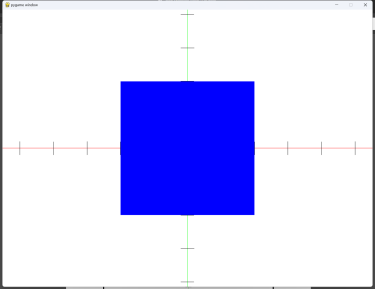
绘制立方体：

Cube()根据预先定义的顶点、面和颜色绘制立方体的各个面。设置透视投影，利用OpenGL函数gluPerspective()设定视角、宽高比以及近裁剪面和远裁剪面。设置正交投影，使用OpenGL函数glOrtho()设定平行视角的六个面的位置

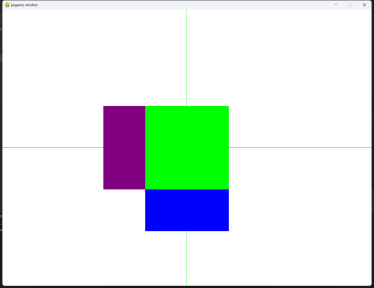
**实验结果**

关于二维的变换结果：

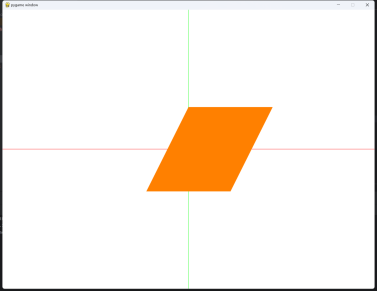
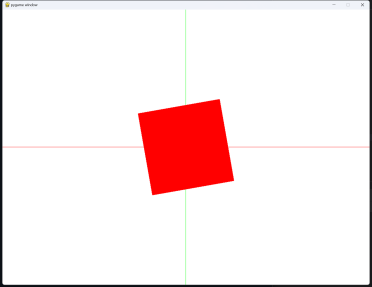
二维图缩放

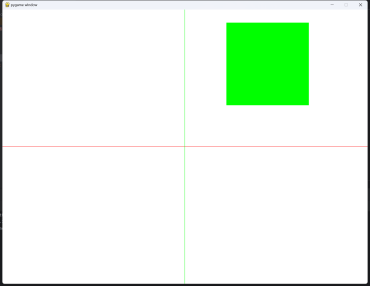
反射



切变 旋转

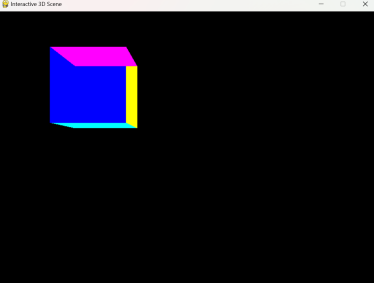
 

平移

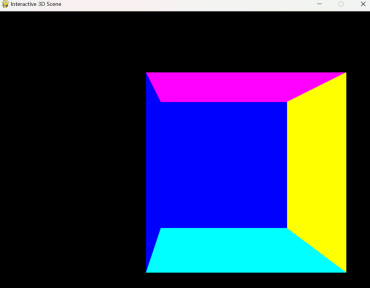
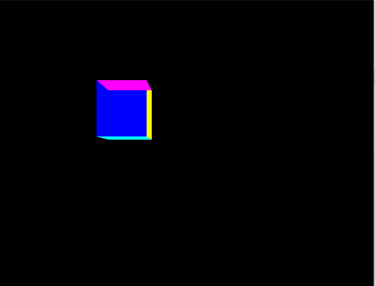


关于三维的变换结果：

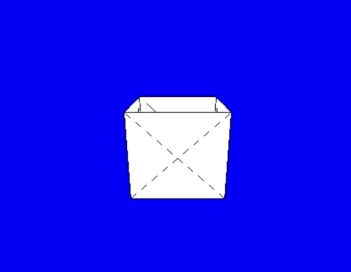
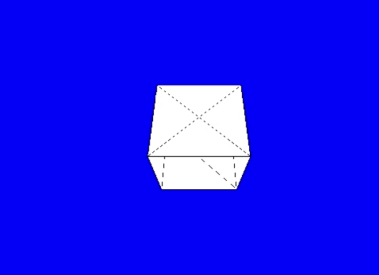
平移：



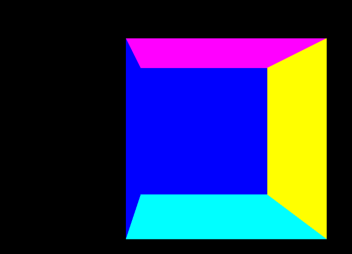
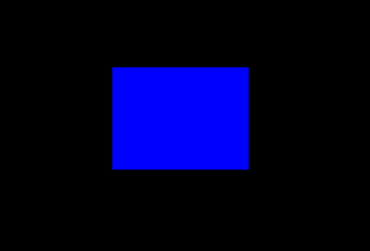
缩放：



关于三维的旋转结果：



关于三维的投影变换结果：



思考与讨论：

1.实验结果的准确性：是否符合预期的变换效果，是否与OpenGL的文档描述一致。

2.实验结果的可靠性：是否在不同环境下都能稳定地实现相同的效果，是否存在潜在的误差来源。

3.不同的变换效果进行评估和比较：例如缩放、旋转、平移等各自对图形的影响程度，是否符合三维图像的实际变化，存不存在形变等

4.如何进一步拓展实验内容，例如添加更复杂的图形变换、优化性能等方面。

5.过程中遇到的主要挑战：OpenGL API的复杂性、对图形变换算法的难以理解等。

**实验总结**

贾东原：

在使用OpenGL和Python进行二维与三维图形绘制的基本变换中， 应当注意纹理加载与应用： 使用Pillow库加载图像，并进行预处理，如翻转和格式转换，以适配OpenGL的纹理处理要求。 glGenTextures和glBindTexture函数用于生成和绑定纹理。通过glTexImage2D将图像数据上传到GPU。 设置纹理参数（如glTexParameteri），以定义纹理的过滤方式和环绕方式，影响纹理的显示效果。 基础图形绘制： 使用glBegin和glEnd来定义一个图形的绘制开始和结束。图形类型（如GL\_QUADS）指定了顶点如何组合成图形。 顶点函数（如glVertex3f）用于指定每个顶点的位置，而纹理坐标（glTexCoord2f）则关联纹理图像的相应部分。 图形变换： 缩放：通过调整顶点坐标的比例，可以实现图形的缩放。例如，draw\_scaled\_square函数通过乘以一个缩放因子来调整正方形的大小。 反射：改变顶点坐标的符号可以实现图形的反射。 旋转与位移：通常通过变换矩阵（glRotate, glTranslate等）来实现。 坐标系绘制： 绘制坐标轴有助于理解当前的视图和图形变换的效果。通过不同颜色标记X轴和Y轴，可以更直观地观察图形的方向和位置变化。 在编写函数时，一定要注意OpenGL的状态机特性，即当前的状态（如绑定的纹理、激活的纹理单元、设置的颜色等）会影响后续的绘制命令。正确管理这些状态对于实现复杂的图形效果至关重要。

陈昱鑫：

在使用OpenGL和PyCharm进行二维图像组合变换的实验中，我深刻体会到了图形处理技术的魅力和复杂性。通过编写代码实现图像的平移、旋转、缩放等变换操作，我不仅深入了解了OpenGL的基本原理和使用方法，还学会了如何利用PyCharm这一强大的集成开发环境来提高编程效率。

在实验过程中，我遇到了许多挑战和问题，比如坐标系的转换、变换矩阵的计算等。但通过查阅资料、尝试调试，我逐渐找到了解决方案，并成功实现了图像的各种组合变换效果。这让我对图形处理技术有了更深入的理解，也增强了我在编程方面的自信心。

通过这次实验，我不仅提升了对OpenGL和PyCharm的熟练程度，还培养了解决问题的能力和团队协作精神。我意识到图形处理领域是一个充满挑战和机遇的领域，我会继续努力学习，不断提升自己的技能。这次实验让我收获颇丰，也让我对未来充满了期待和信心。

王思栋：

学习使用 C++ OpenGL 库绘制三维立体图形是一项富有挑战性但又非常有趣的任务。在这个过程中，我深刻体会到了以下几点：

首先，我发现了 OpenGL 的强大之处。通过 OpenGL，我可以直接控制图形硬件，实现高性能的图形渲染。OpenGL 提供了丰富的函数和接口，使得我可以以编程的方式创建和操作三维对象，控制光照、阴影和材质等属性，从而实现各种各样的视觉效果。

其次，学习 OpenGL 还加深了我对计算机图形学的理解。通过编写 OpenGL 程序，我深入了解了图形管线的工作原理，包括顶点处理、几何处理、光栅化和像素处理等阶段。这不仅让我对图形学的理论知识有了更深入的认识，还提升了我的编程能力和对计算机图形学技术的应用能力。

另外，学习使用 OpenGL 绘制三维立体图形也锻炼了我的问题解决能力。在编写 OpenGL 程序的过程中，我常常会遇到各种各样的问题，例如图形显示不正确、性能不佳或者功能实现困难等。通过查阅文档、阅读源代码、搜索网络资源以及与其他开发者交流，我逐渐学会了如何解决这些问题，提升了自己的调试和排错能力。

最后，学习 OpenGL 还让我感受到了编程的乐趣。通过不断地调整参数、优化算法和实现新的功能，我可以看到立体图形逐渐呈现出我想要的效果，这种成就感是非常有满足感的。同时，OpenGL 的开放性和灵活性也给了我无限的创造空间，让我可以尽情地发挥自己的想象力和创造力，创造出更加精彩和令人惊叹的图形效果。

综上所述，学习使用 C++ OpenGL 库绘制三维立体图形是一次充满挑战和乐趣的过程，不仅加深了我的计算机图形学知识，还提升了我的编程能力和问题解决能力，让我对计算机图形学技术有了更深入的理解和认识。

武纪伟：

在本次实验中，最让我有所感悟的是其中涉及到透视投影和正交投影的设置，这是3D图形显示中非常关键的部分。透视投影能够模拟人眼实际看到的场景，而正交投影则忽略了视点的远近，用于工程图纸或地图等需要精确尺寸的场合。通过实验，我更加深入理解了两种投影方式的应用场景和数学原理。在本次实验中我设置了立方体的顶点和面，并在循环中进行变换（平移、旋转等）。这一过程加深了我对OpenGL坐标系统和变换矩阵的理解，尤其是如何通过变换矩阵控制对象在三维空间中的位置和方向。

在使用 Pygame 库处理键盘事件，实现交互式的视图控制的过程中。通过响应键盘事件来调整摄像机的位置（前后左右移动），我体验到了事件驱动编程在实时图形应用中的重要性。在主循环中，每次更新图像前都要清空颜色和深度缓存，然后再绘制新的图形。这种方法确保了画面的正确更新，也让我意识到在复杂场景中性能优化的必要性，比如合理使用缓存和减少不必要的计算。

通过编写和运行这段代码，我不仅仅是在重复教科书上的理论，而是将其付诸实践，看到了理论在实际应用中的表现。它加深了我的理解与对图形编程深入学习的兴趣。

席正莅：

通过这次实验，我深入了解了二维和三维图形的绘制与变换技术，以及如何利用OpenGL实现这些操作。实践中，我逐步掌握了缩放、旋转、平移等基本变换的原理和实现方法，对图形处理技术的理解有了显著提升。

在实验过程中，我与同组同学密切合作，共同讨论解决问题的方案，提高了团队合作和沟通协作能力。通过相互交流、分享经验，我们不断完善实验方案，解决了许多技术难题，取得了令人满意的成果。

实验中遇到了许多挑战，例如对OpenGL API的不熟悉、理解图形变换算法的困难等。

但通过不懈的努力和团队的支持，我一步步克服了这些困难，最终实现了预期的功能，收获颇丰。

这次实验不仅加深了我对图形处理技术的理解，还为我今后的学习和实践提供了良好的基础。我也意识到图形处理技术在计算机图形学、游戏开发等领域的重要性，激发了我进一步深入学习和应用的兴趣。

总的来说，这次实验是我在图形处理领域的一次重要实践，不仅提升了我的专业能力，也培养了我解决问题的能力和耐心。我相信通过不断学习和实践，我将会在图形处理领域取得更大的成就，为未来的发展打下坚实的基础。

饶毓：

在进行C++和OpenGL的三维图形变换实验时，我收获颇丰。以下是我对此次实验的心得感想：

首先，C++和OpenGL的结合为我提供了一个强大的工具集，使我能够在三维空间中创建和操作图形。通过OpenGL的API，我可以控制图形的渲染、光照、纹理等各个方面，这使得我能够创建出丰富多样的三维场景。

在实验过程中，我深入了解了三维图形变换的基本原理和技巧。通过平移、旋转和缩放等操作，我可以改变图形在三维空间中的位置和形态。这些变换操作不仅有助于我创建出更加逼真的三维场景，也让我对计算机图形学有了更深入的理解。

同时，我也意识到在进行三维图形编程时，性能优化是一个非常重要的方面。通过合理地管理内存、优化渲染流程和使用高效的算法，我可以提高程序的运行效率，减少渲染时间，从而为用户带来更好的体验。

此外，我还学会了如何调试和优化OpenGL程序。由于OpenGL的错误信息通常比较隐晦，因此我需要仔细分析代码，逐步排查错误。通过不断地尝试和调整，我逐渐掌握了解决OpenGL问题的方法和技巧。

总的来说，这次C++和OpenGL的三维图形变换实验让我受益匪浅。我不仅掌握了相关的编程技巧，也提高了自己的问题解决能力。我相信在未来的学习和工作中，这些经验和技能将会对我产生积极的影响。

**实验分工**

陈昱鑫：二维图形组合变换；撰写PPT和实验报告

贾东原：二维图形基本变换；撰写实验报告

席正莅：实现三维相机功能；撰写PPT和实验报告

王思栋：三维图形基本变换；撰写PPT和实验报告

武纪伟：实现相机功能；撰写相关PPT和实验报告

饶毓：三维图形组合变换；撰写实验报告