

**实验报告**

一、实验目的

1. 实现利用OpenGL绘制二维和三维图形的基本原理和方法。
2. 理解二维、三维几何变换的基本原理，包括平移、旋转、缩放等操作，并能够在OpenGL中实现这些变换。
3. 掌握视图变换和投影变换的基本原理，包括透视投影和正交投影的区别与应用，并能够在OpenGL中实现视图和投影的变换。
4. 实践通过鼠标和键盘控制视角的变化，包括观察者的位置和方向的改变，以及在透视投影和正交投影间的切换，从而加深对视图和投影变换的理解和应用。

二、实验环境

开发环境：PyCharm

开发工具：PyOpenGL

开发语言：Python 3.11

三、实验原理

平移（Translation）：平移是指沿着指定的方向将对象移动到新的位置。在三维空间中，平移可以通过变换矩阵修改对象的坐标来实现。

旋转（Rotation）：旋转是指围绕坐标轴或其他某一点将对象转动一定角度，通过旋转变换的变换矩阵实现。

缩放（Scaling）：缩放是指按照指定的比例调整对象的大小。在三维空间中，通过缩放变换矩阵进行缩放

切变（Shearing）：切变是指在几何变换中，通过斜向移动对象的各个部分来改变其形状。在三维空间中，切变可以沿着一个或多个坐标轴方向进行。切变变换矩阵用于描述对象在各个坐标轴方向上的移动程度。

镜像（Mirroring）：镜像是指通过某个平面、直线或点对对象进行对称变换，使得对象在这个镜像面上的每一点到镜像后的对应点的距离都相等，且这两点之间的直线与镜像面垂直。

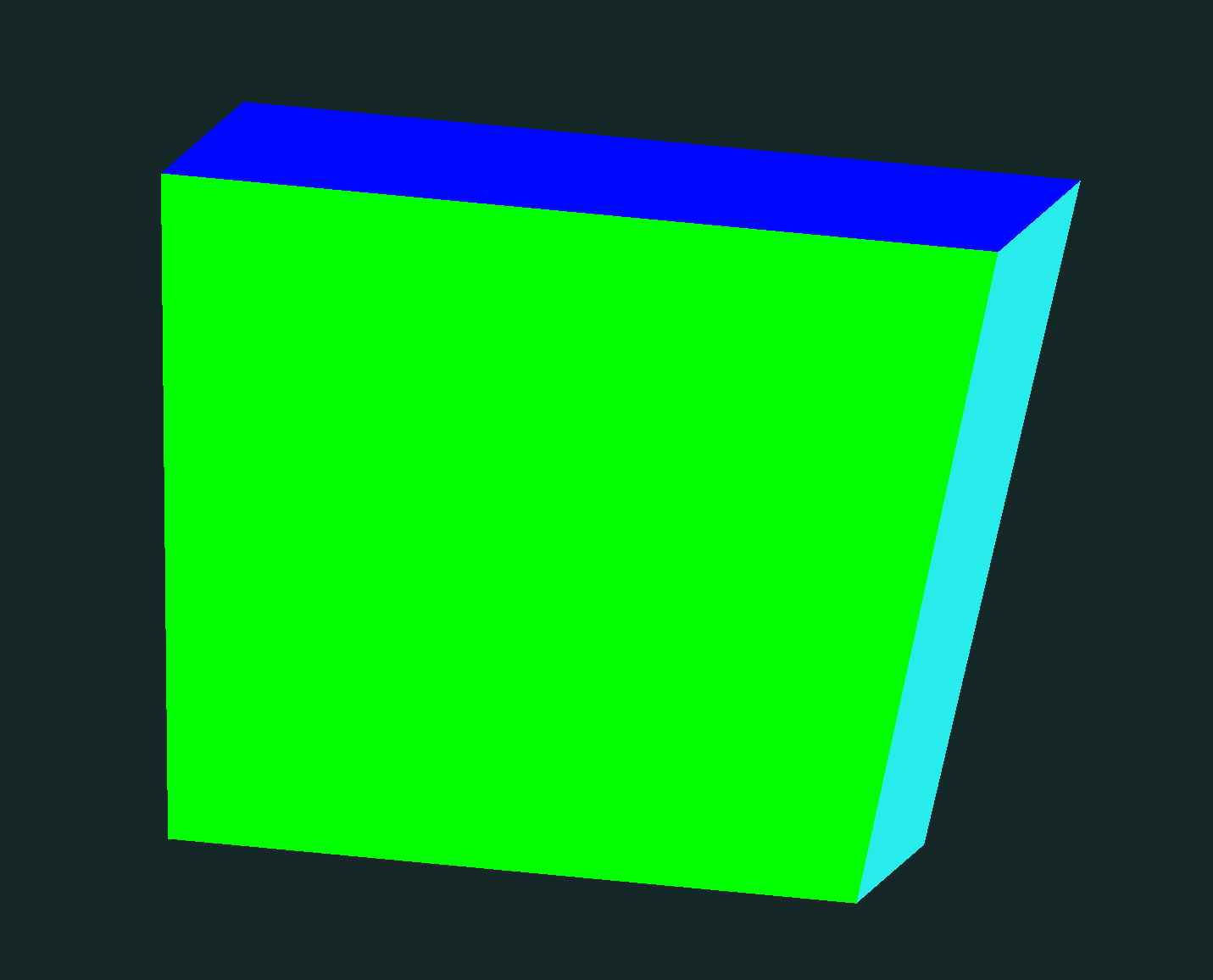
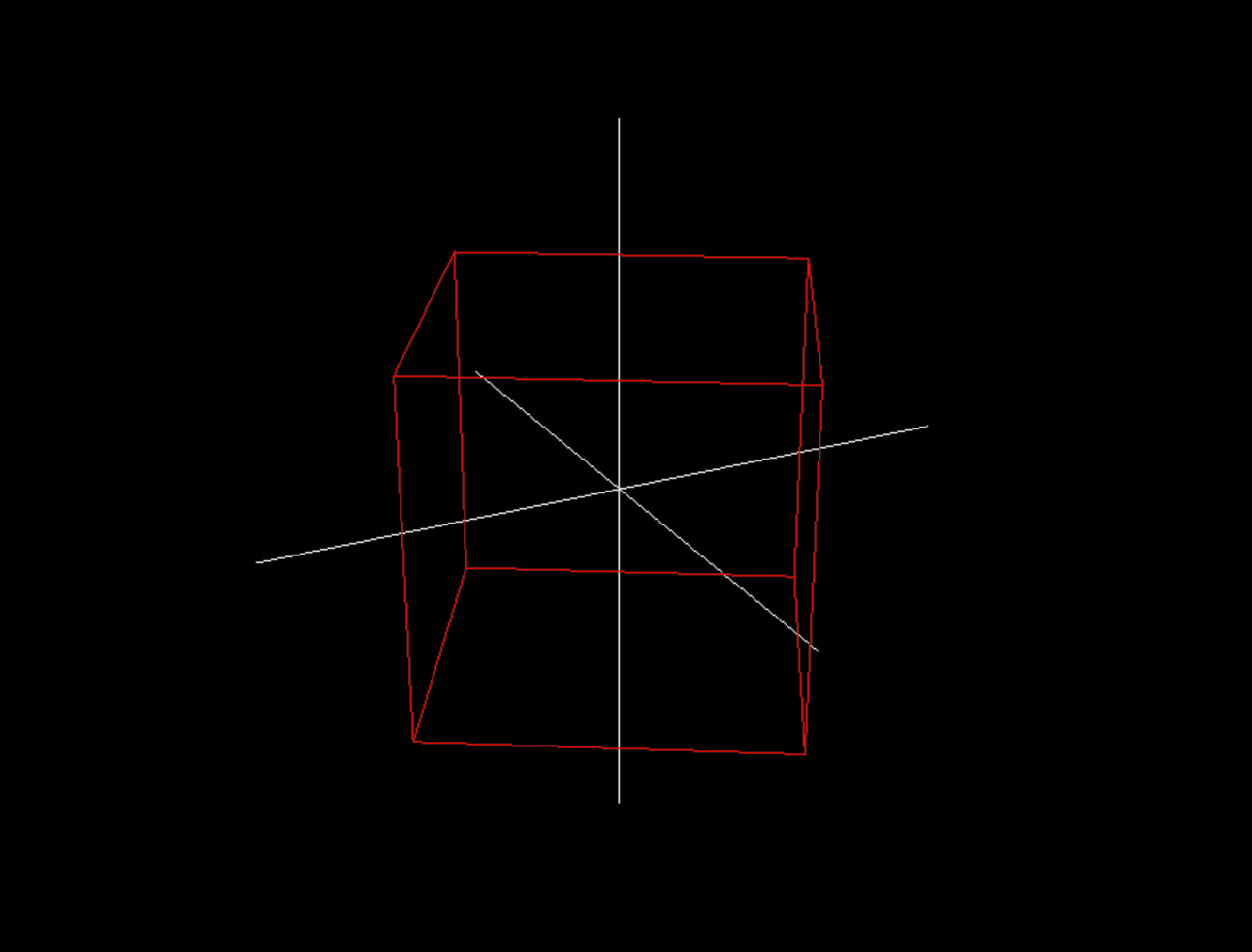
通过xpos，ypos，last\_x和last\_y保留当前鼠标位置和上一次鼠标的位置，从而计算鼠标位置偏移量，得到偏航角和俯仰角，实现视角的改变。

通过glfw.get\_key(window,glfw.KEY\_W)==glfw.PRESS判定键盘是否被按下，从而完成对应操作达到对视角的控制和投影方式的切换。

四、实验内容

1. **初始化环境**：在OpenGL中以及PyCharm，初始化环境包括创建窗口、设置窗口大小、创建OpenGL上下文等。
2. **设置投影**：根据实验要求选择透视投影或正交投影。透视投影用于产生透视效果，使远处的对象看起来较小；而正交投影则没有远近关系，远近的对象大小相同。
3. **设置视角和相机位置**：确定观察者的位置和方向，以及相机的位置和朝向。使用OpenGL以及PyOpenGL的视图矩阵函数来设置视图矩阵，确定观察者的位置和朝向。
4. **绘制对象**：根据实验要求绘制二维图形和三维图形。使用OpenGL以及PyOpenGL的基本绘图函数以及顶点数组来绘制各种几何图形，也可以加载外部模型文件进行绘制。
5. **应用变换**：使用OpenGL以及PyOpenGL提供的变换函数对对象进行平移、旋转和缩放。通过调用函数对模型进行变换操作，根据需要连续调用这些函数来组合多个变换。
6. **渲染更新显示**：在每一帧中，清除颜色缓冲区和深度缓冲区，然后重新渲染场景。通过调用**glClear()**函数来清除缓冲区，然后绘制场景中的对象，并最终交换前后缓冲区来显示渲染结果。
7. **键盘控制**：添加键盘事件处理函数，根据按键操作来控制场景。使用OpenGL以及PyOpenGL的键盘事件处理函数来注册键盘事件回调函数，在回调函数中根据按键操作来改变相机位置、切换投影类型等。

五、实验结果



六、实验总结

**梁鑫垚：**我通过绘制二维图形来学习基本的图形绘制技术和OpenGL/Pygame的使用，然后，学习了各种二维变换，如缩放、反射、切变、旋转和平移。通过手动实现这些变换，我能够更深入地理解它们对图形的影响，接着，学习了如何将这些基本变换组合起来，以创建更复杂的变换效果，这进一步加深了我的理解。在这之后，我转向了三维图形的绘制技术，从平面世界进入了立体世界。

**张佳硕：**在实验中，我完成了通过使用 pyOpenGL 实现的二维和三维图形绘制以及相应的几何变换。具体来说，我成功实现了以下部分：

绘制了二维图形，并进行了二维变换，包括缩放、反射、切变、旋转和平移。

绘制了三维图形，并进行了三维变换，包括缩放、平移和绕 x/y/z 轴旋转。在代码中，特别是在处理键盘和鼠标事件时，我花费了一些时间来理解如何正确捕获用户输入并作出相应的变换。通过阅读 Pygame 和 OpenGL 的文档，并参考一些示例代码，我顺利解决了这些问题，并实现了对场景的交互控制。

总的来说，通过这次实验，我加深了对二维、三维几何变换原理的理解，并学会了如何使用 Pygame 和 OpenGL 来实现图形绘制和变换。

**杨临悠：**本次实验我尝试使用OpenGL和glfw对二维图型和三维图型进行绘制，并且尝试了对二维图形的变换，在二维图型种实现了通过键盘来控制旋转（R），平移（T），缩放（S）等。

其中，对三维图形，我对其每个面都染上不同的颜色，以便观察我们的三维图形，然而我在对三维图形的视图观察方向的设置上遇到困难，这使得我对组合变换后三维图形的观察方面遇到困难，造成我的组合变换看起来比较抽象，组合变换中，我让立方体按照时间绕X轴和Z轴旋转，并且在Y轴沿着-1到1平移，这使得我的图形在旋转到视角之后我没办法处理而出现穿模，希望以后能解决这个问题，并且更好的学习图形学。

总的来说：这次试验更加使我理解了通过更改矩阵的方式来使得图形发生变换，为二维和三维图形的原理有了更直观更可视化的理解。

**佘剑雄**：在进行了OpenGL实现二维和三维图形的绘制与变换的实验后，我深感图形编程的魅力和挑战。这次实验不仅提升了我的编程技能，还加深了我对计算机图形学的理解。

在实验过程中，我首先学习了OpenGL的基本概念和操作，包括如何设置视口、投影、模型视图矩阵等。

在三维图形的绘制阶段，我感受到了很大的挑战。我首先学习了如何设置相机位置和方向，以便从不同的视角观察三维场景。接下来，我通过指定三维顶点和面来创建简单的三维物体，如立方体、球体等。

在掌握三维图形的绘制方法后，我开始尝试进行更复杂的变换操作。通过使用模型视图矩阵和投影矩阵，我实现了三维图形的平移、旋转和缩放，以及视角的变换。这些操作让我对三维图形的空间关系和视觉效果有了更直观的认识。

总的来说，这次实验对我而言是一次宝贵的学习经历。我不仅掌握了OpenGL的基本用法，还深入了解了二维和三维图形的绘制与变换原理。同时，我也意识到自己在计算机图形学方面还有很多需要学习和提高的地方。

**陈方舟：**一开始，我对OpenGL的认识仅限于它是一个强大的图形库，用于渲染2D和3D矢量图形。然而，随着我深入学习，我意识到掌握OpenGL需要对计算机图形学有较深的理解，包括顶点处理、光照模型、视图和投影等概念。刚开始接触时，我觉得有些挑战，因为需要同时处理几何学、线性代数和编程技能。通过实际编写代码来创建一个3D立方体，我对OpenGL的理解有了质的飞跃。实现从定义顶点数据、索引到使用VAO、VBO、EBO管理这些数据，我逐步了解了OpenGL流水线的工作方式。特别是在配置顶点属性指针时，我开始明白如何将顶点信息有效地传递给图形管线，以及如何用着色器来实现颜色和光照效果。

着色器编程是我发现最有趣也是最具挑战性的部分。学习如何编写顶点着色器和片段着色器，以及如何通过它们来控制顶点处理和像素渲染，是一次充满启发性的经历。同时，处理视图和投影矩阵加深了我对3D世界与屏幕坐标之间关系的理解。

任何编程学习之旅都免不了遇到错误和调试，OpenGL不例外。其中，调试视觉效果比通常的软件开发复杂，因为可能需要调整的地方非常多，从顶点数据错位到着色器逻辑错误。然而，每次成功解决问题后，我都会有极大的成就感。

通过这次实验，我不仅学会了构建并渲染一个3D立方体，还提高了我的OpenGL技能，对计算机图形学有了更深的认识和兴趣。我期待将来能够将这些知识应用到更为复杂的项目中，如游戏开发或者虚拟现实等。而且，这一过程激励我继续探索和学习先进的图形处理技术，以解锁更多的可能性。

**梁昊臻：**在制作二维图形和三维图形的过程中，我通过pyOpenGL实现了图形的平移、旋转、缩放等简单变化。对于二维图形，我利用pyOpenGL的绘制函数，成功绘制出了各种形状，如矩形、圆形等，并通过改变其顶点的坐标实现了图形的平移和缩放。对于三维图形，我则利用pyOpenGL的矩阵变换功能，实现了图形的旋转和缩放，使得三维图形能够呈现出更丰富的视觉效果。

为了增强图形的交互性，我还添加了鼠标键盘控制视角的功能。这一功能的实现主要依赖于pyOpenGL的事件处理机制。通过监听鼠标和键盘的输入事件，我能够实时地改变视角的位置和方向，使得用户能够自由地观察图形。这种交互方式不仅提高了用户的体验，也使得图形展示更加生动和直观。

在制作过程中，我也遇到了一些挑战和困难。例如，在实现三维图形的旋转时，我最初遇到了旋转轴和旋转角度的问题，导致图形旋转的效果并不理想。此外，在添加鼠标键盘控制视角功能时，我也遇到了一些技术难题，但通过不断尝试和改进，我最终成功地实现了这一功能。

总之，这次学习pyOpenGL的经历让我收获颇丰。我不仅掌握了二维图形和三维图形的制作方法，还学会了如何添加交互功能来提升用户的体验。

七、实验分工

梁昊臻：实验1,2,3,4,7以及PPT的撰写

陈方舟：实验1,2,3,4,7

张佳硕：实验1,2,3,4,5

梁鑫垚：实验1,2,3,4以及文档的撰写

杨临悠：实验1,2,3,4,6

佘剑雄：实验1,2,3,4,6