深圳大学考试答题纸

(以论文、报告等形式考核专用)  
二○ 二一 ～二○ 二二 学年度第 一 学期

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程编号 | 1500610003 | | | 课序号 | 01 | 课程名称 | | 计算机图形学 | | 主讲教师 | 周漾、黄惠 | | 评分 |  |
| 学 号 | 2019085071 | | 姓名 | | | 石弋川 | 专业年级 | | 2019级软件工程 | | | | | |
|  | | | | | | | | | | | | | | |
| 教师评语： | | | | | | | | | | | | | | |
| 题目： | | 虚拟场景建模 | | | | | | | | | |  | | |

宋体五号，至少八页，可以从下一页开始写。

**成绩评分栏：**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **评分项** | **俄罗斯方块文档**  （占12分） | **俄罗斯方块代码**  （占24分） | **俄罗斯方块迟交倒扣分**  （占0分） | **虚拟场景建模文档**  （占16分） | **虚拟场景建模代码**  （占38分） | **演示与答辩**  （占10分） | **虚拟场景建模迟交倒扣分**  （占0分） | **大作业总分** |
| **得分** |  |  |  |  |  |  |  |  |
| **评分人** |  |  |  |  |  |  |  |  |

一、实验内容

在屏幕上显示一个包含多个虚拟物体的虚拟场景，并且响应一定的用户交互操作。

二、具体内容

**1、场景设计和显示**

学生可以通过层级建模（实验补充1和2）的方式建立多个虚拟物体，由多个虚拟物体组成一个虚拟场景，要求在程序中显示该虚拟场景，场景可以是室内或者室外场景；场景应包含地面。层级建模的最深层次需要达到至少四层。

**2. 添加纹理**

参考实验4.1，为场景中至少两个主要物体添加纹理贴图 。

**3. 添加光照、材质、阴影效果**

参考实验3.2，实验3.3和实验3.4，实现光照效果、材质、阴影等。

**4. 用户交互实现视角切换完成对场景的任意角度浏览**

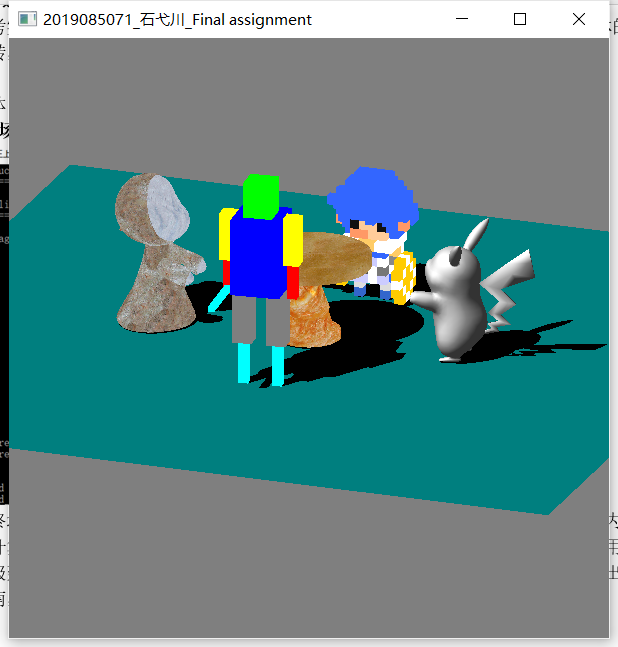
参考实验3.1，完成相机变换。

**5. 通过交互控制物体**

参考实验2.3，实现物体的变换，允许用户通过键盘或者鼠标实现场景中至少两个物体的控制（移动，旋转，缩放等等）。

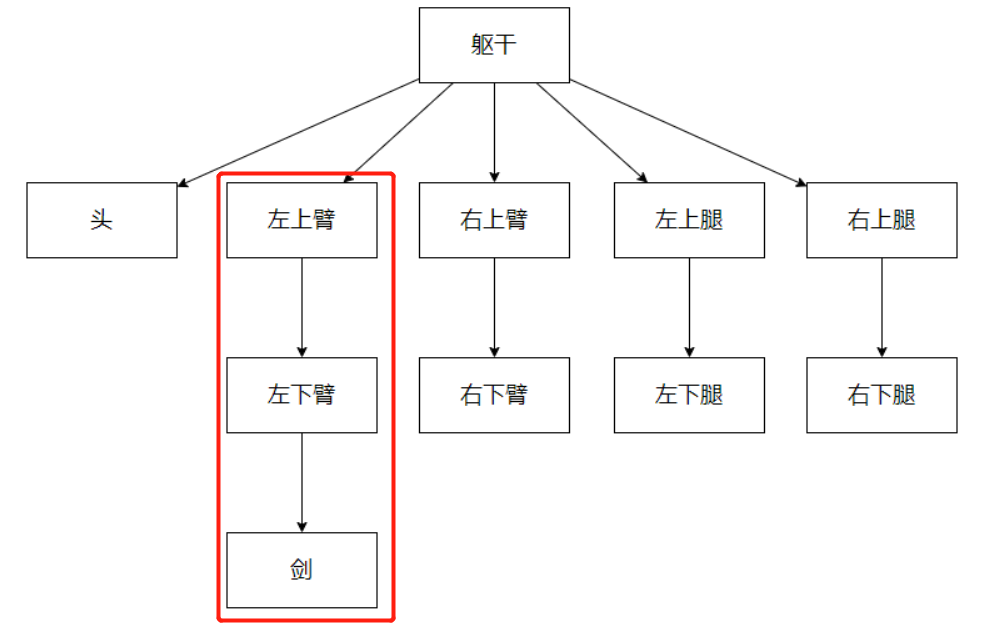
三、具体内容

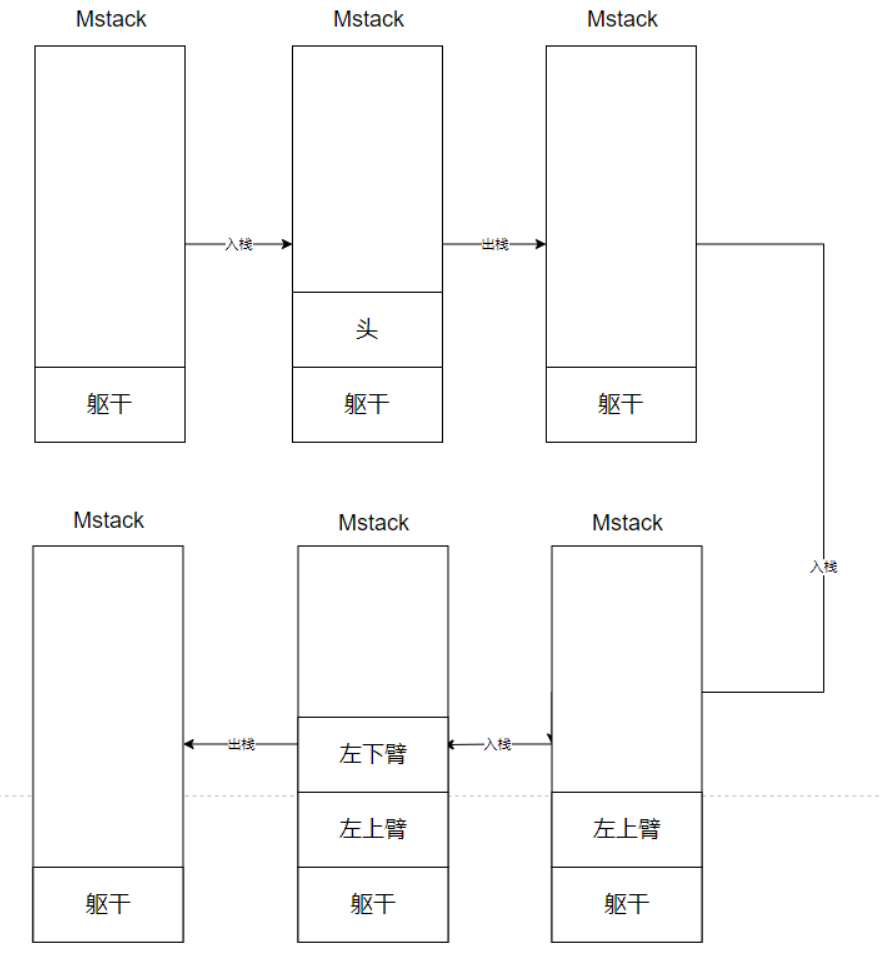
**1、场景设计和显示**



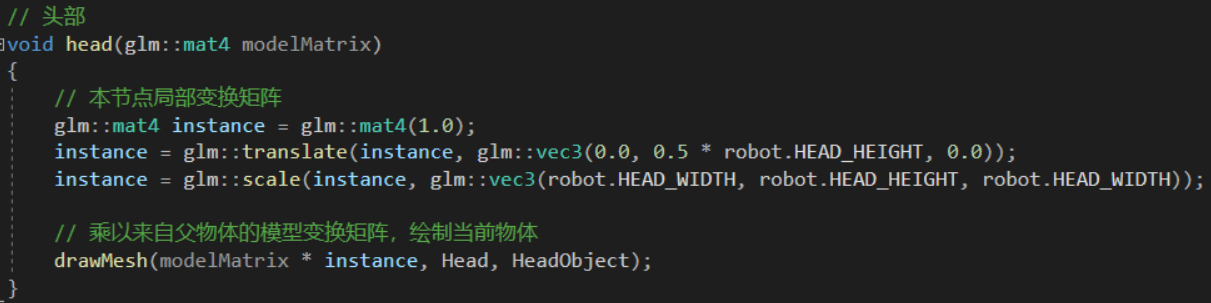
最终场景效果如上图所示。该虚拟场景主要描述了一个包含和多个物体和地面的室内场景，几个通过计算机图形学绘制的小人围在一个桌子周围。在该场景中包含多个虚拟物体，运用了物体绘制，层级建模，纹理贴图，光照效果，阴影效果等多种技术。除此之外还有一系列打印出的详细的操作指南，可以通过键盘进行多种视角切换，物体控制和交互等。

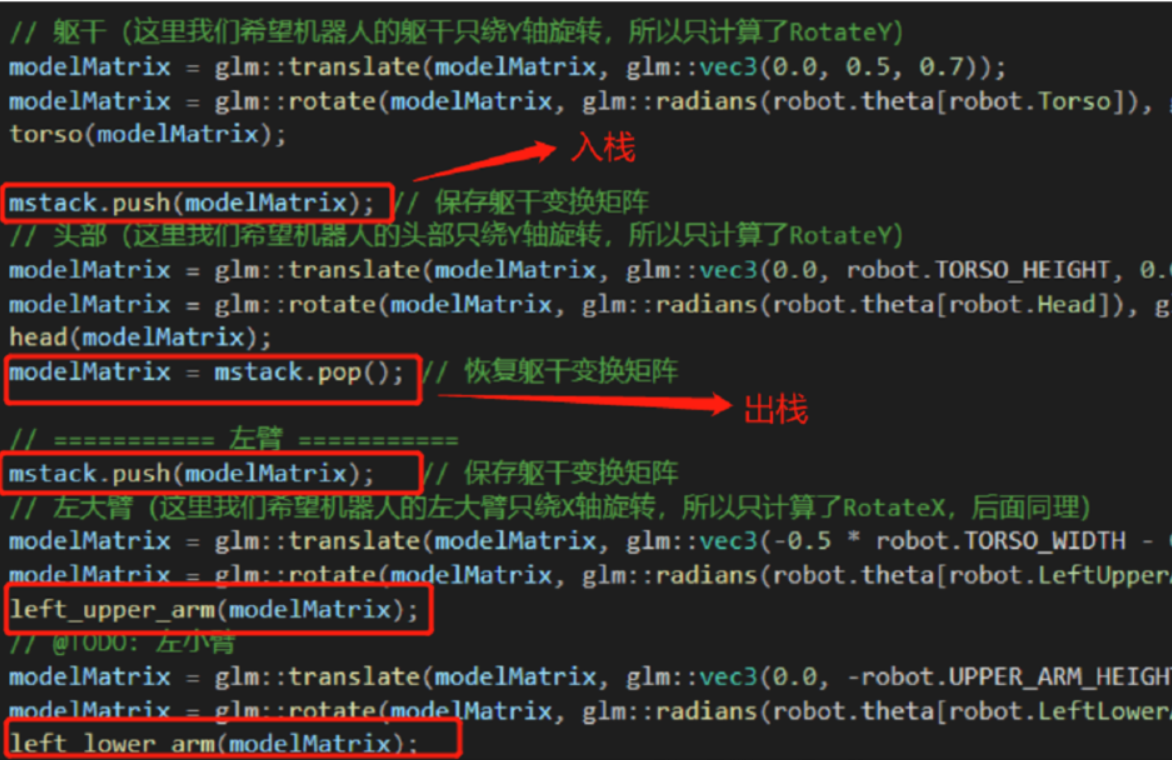
根据实验要求需要进行层级建模，这里主要利用层级树的方法进行遍历绘制多个长方体，最终拼成一个机器人。按深度优先顺序，即 “躯干 -> 头 -> 左上臂 -> 左小臂 -> 剑 -> 右上臂 -> 右下臂 -> 左上腿 -> 左下腿 -> 右上腿 -> 右下腿”的顺序完成层级树的遍历，完成display（）函数。其中躯干->左上臂->左下臂->剑这一局部达到了四层的最深层次。在具体实现时，也可以理解为使用堆栈的方式来保存和恢复节点变换矩阵，如下图所示：





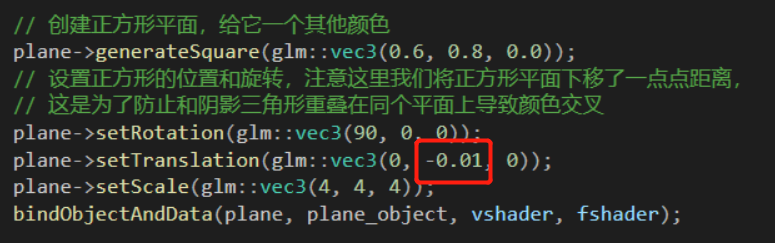
在具体的代码实现时，先分别实现物体各部位的绘制函数，如head(glm::mat4 modelMatrix)等，再根据上图树的思想，利用堆栈进行绘制。





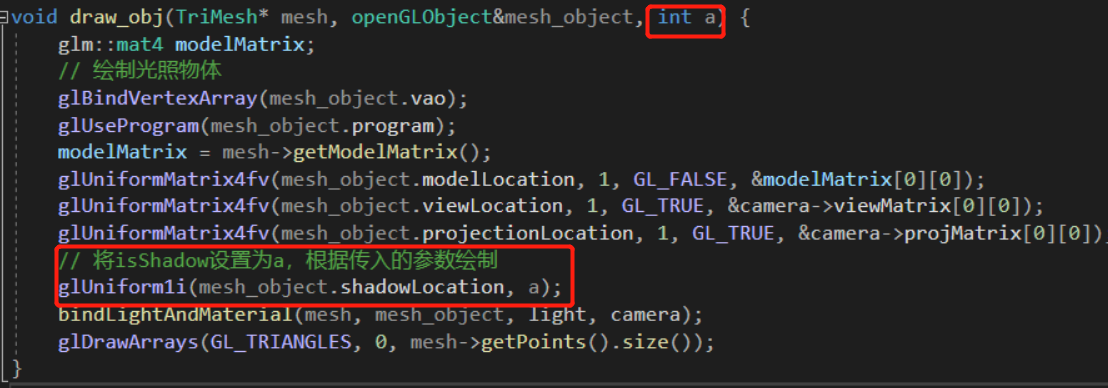
Figure根据级联层间建模

实验还要求应包含地面，这里可以采用调用generateSquare函数的方法生成一个正方形图像，直接绘制作为地面，值得注意的是为了使后面物体的阴影能展现出来，所以设置正方形位置的时候在y故意设为-0.01使其低于阴影平面。



Figure绘制正方形平面

设置完地面图像的基本信息调用bindObjectAndData函数后将数据传给vshader和fshader，再调用自定应的draw\_obj函数，根据传入的参数不同自定义是直接绘制图像还是根据光源绘制或者根据纹理绘制。



Figure绘制简易图形函数draw\_obj

由于需要绘制正方形图形作为地面，所以在调用该函数时传入了1表示根据默认设置直接绘制图像。



**2、添加纹理**

* 读取带纹理的obj文件

为物体添加纹理，主要在于readObj函数。这部分是读取并存储包括顶点坐标vertex\_positions、顶点法线数据vertex\_normals、顶点纹理数据或者说UV坐标vertex\_textures，先分别读取数据再push\_back压入这些向量中。

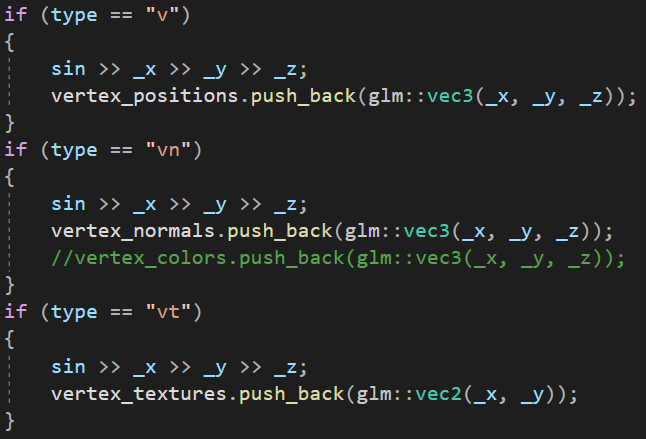


Figure 读取obj文件readObj函数

这部分负责读取并存储以“f”开头的坐标信息，包括片面的顶点索引数据faces、三角面片顶点的法线索引下标normal\_index以及三角面片顶点的纹理坐标索引下标。其中值得注意的是在obj文件中索引下标是从1开始的，所以需要在传入的时候减1。

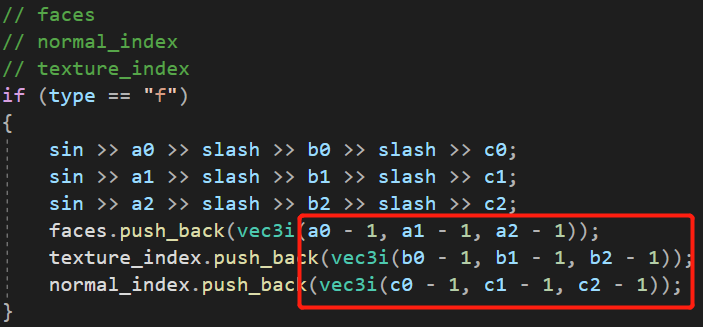
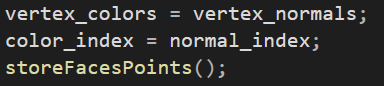


Figure 读取obj文件readObj函数

在读取完obj内的数据后，用法向量的值赋予vertex\_color和color\_index并调用storeFacesPoints函数存储数据。

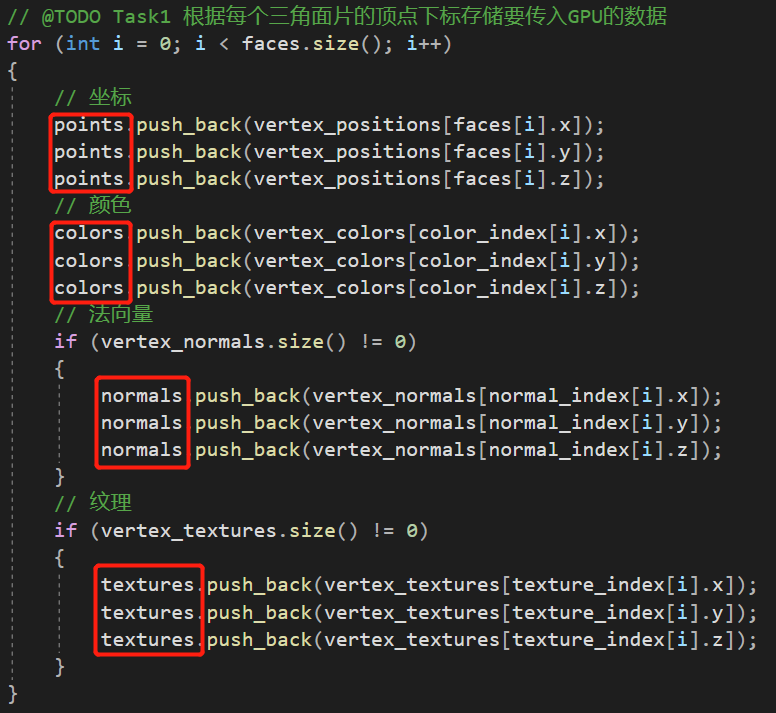


* 完善数据的读取

在storeFacesPoints函数，主要需要完成归一化和四个信息的存储。这里自定义了norm函数用以归一化，而四个信息的存储或者准确说将三角面片的顶点索引传入GPU中，分别是坐标points、颜色colors、法向量normals、纹理textures，由于前面已经将这些信息存储到vertex中，所以这里只需要用for循环遍历，按角标索引以及各个点的信息逐个压入即可。



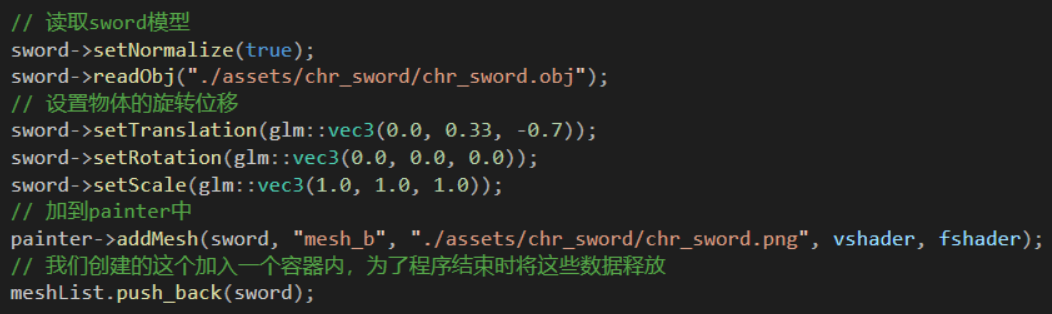
Figure归一化函数norm



Figure存储三角面片的顶点下标函数storeFacePoints

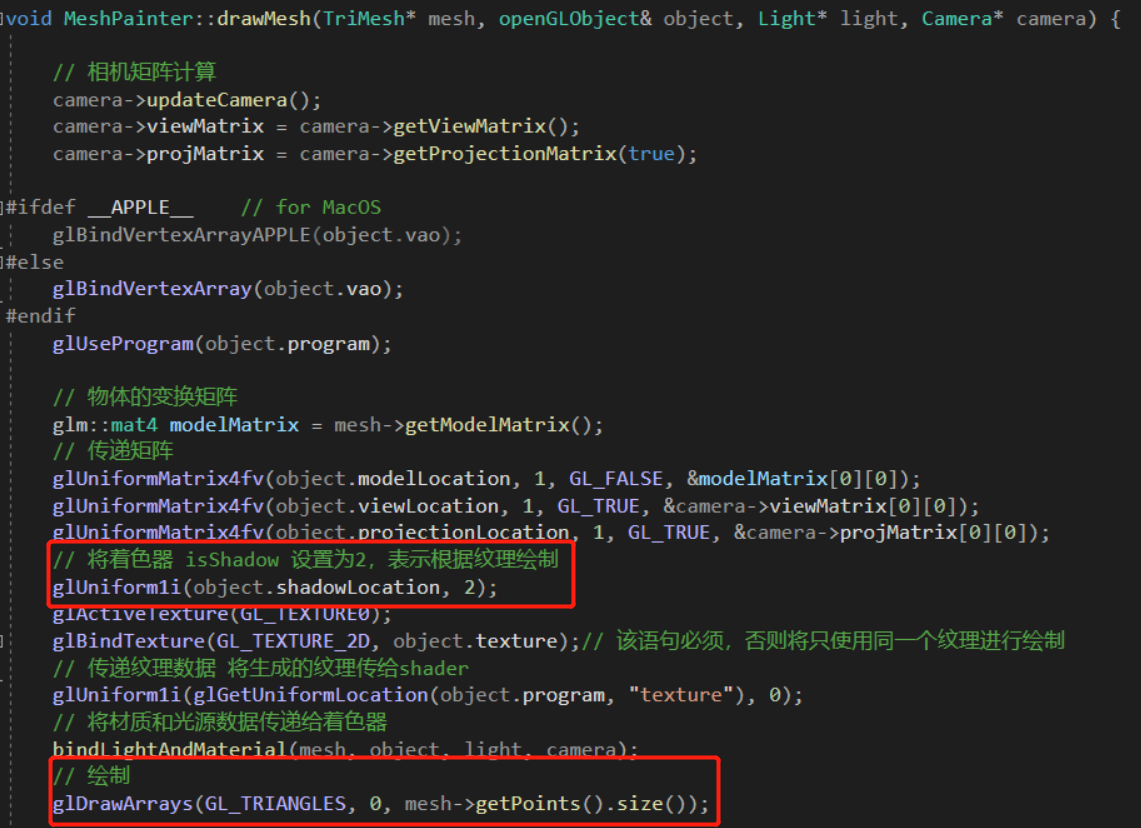
* 绘制图像

这里主要以一个持剑小人sword为例展现具体绘制含纹理物体的过程，先确认归一化，然后调用readObj函数读取chr\_sword.obj文件，设置物体的旋转位移，最后加到选择需要添加的纹理再加入到painter中准备绘制。



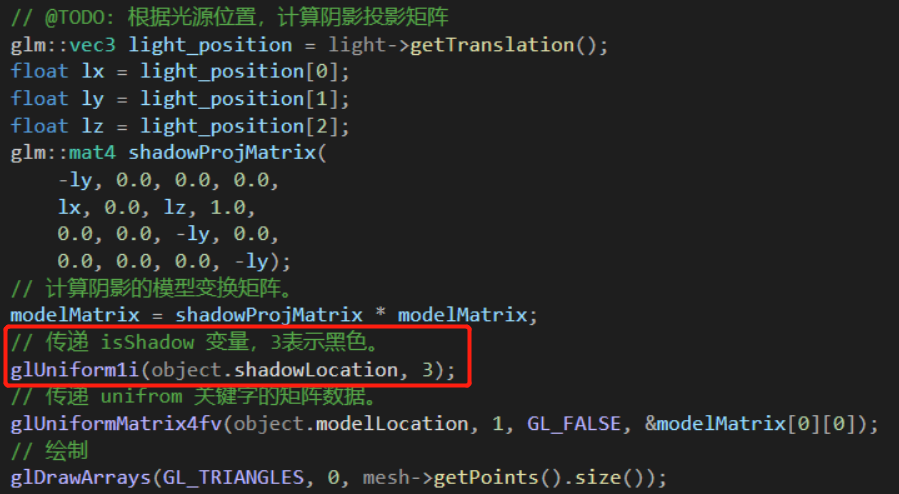
Figure绘制纹理图像

加入到painter后，调用drawMesh函数进行绘制。其中将isShadow设置为2，表示根据纹理绘制。



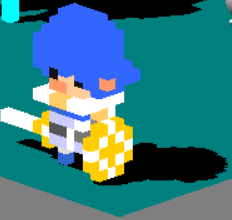
Figure绘制纹理图像函数drawMesh

除此之外，由于还需要根据光照绘制阴影，所以在drawMesh函数后半部分添加了绘制阴影的方法， 具体实现将放到第三个实验要求的部分详细说明，这里暂时截图展示一下。



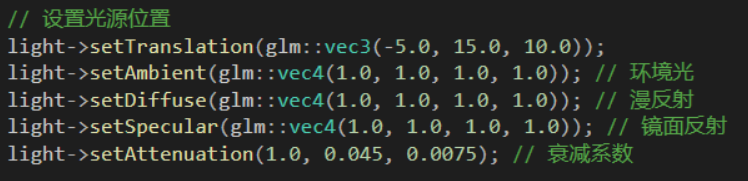
Figure绘制阴影主要部分

最终实现的效果如下图所示：

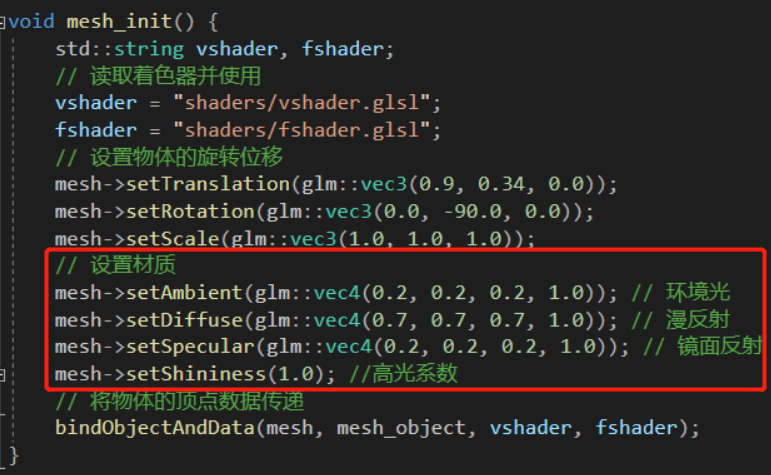


**3、添加光照、材质、阴影效果**

设置光源位置。

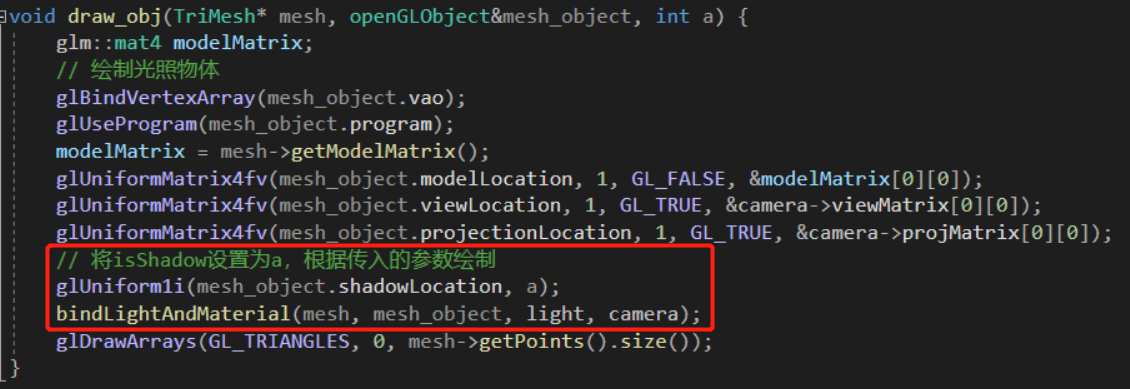


设置mesh物体材质并传递顶点数据。



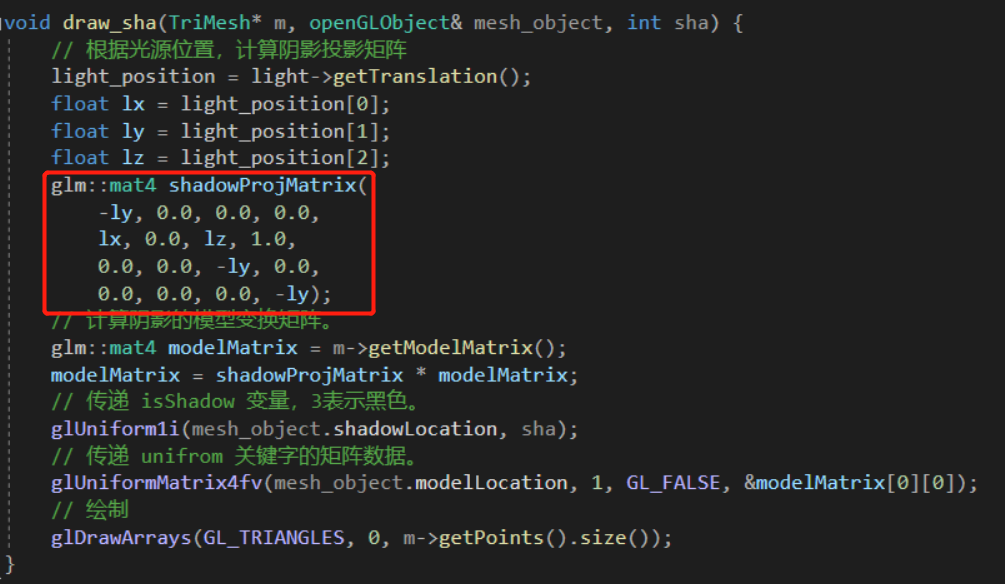
Figure光影物体绘制函数mesh\_init

调用draw­\_obj函数绘制图像，将isShadow设置为0表示根据光照绘制，并传递光照和材质数据。



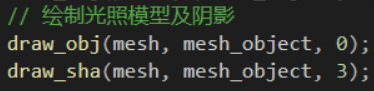
Figure物体绘制函数draw\_obj

调用draw\_sha函数绘制阴影，确定投影矩阵，将isShadow设置为3表示黑色。

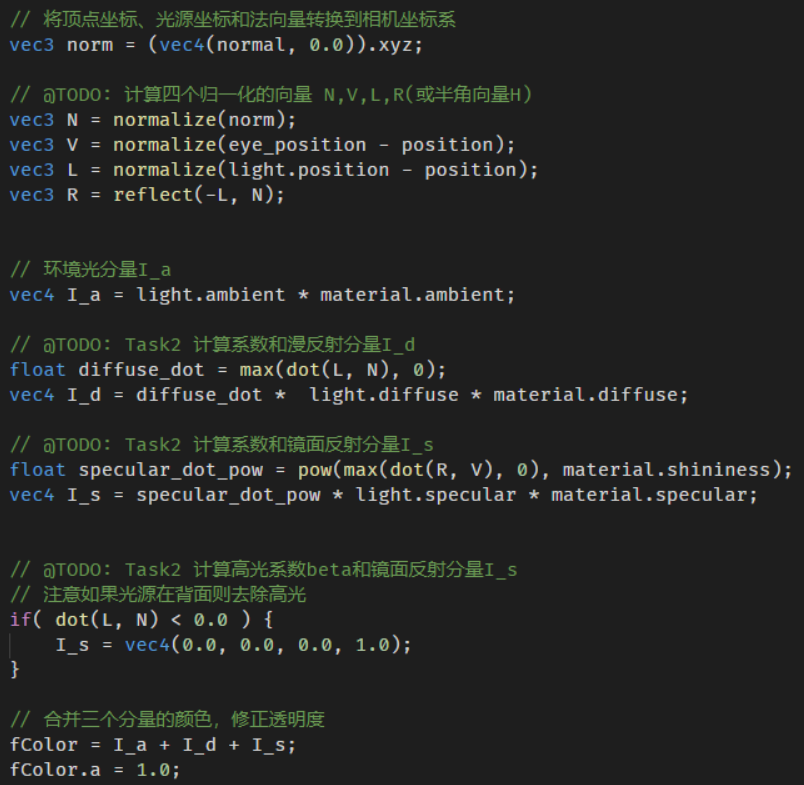


Figure阴影绘制函数draw\_sha

最后在display函数中调用上述两个函数进行绘制。

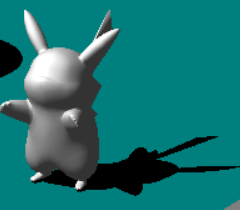


这里对根据光照绘制物体的fshader着色器进行一下补充说明，根据phong光照模型主要有三个分量，分别为环境光分量I\_a，漫反射分量I\_d，镜面反射分量I\_s，将这三个光照分量加在一起得到fColor = I\_a + I\_d + I\_s得到根据光照效果的绘制。



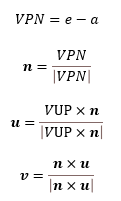
Figure根据光照绘制着色器fshader

最终实现效果如下图所示：



**4、用户交互实现视角切换完成对场景的任意角度浏览**

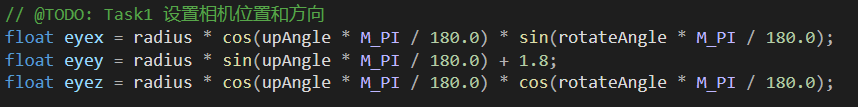
这里主要涉及到Camera.cpp中lookat函数的编写。在该函数中分别定义n，u，n等向量得到viewMatrix矩阵；由于在最开始，我们还需要将相机从坐标原点移动到视点，所以还需要一个平移矩阵并添加平移矩阵T。



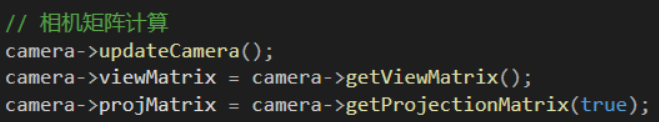


Figure相机视角函数lookAt

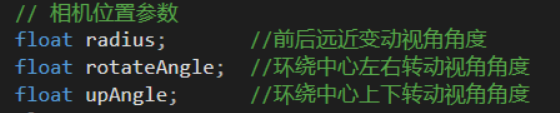
设置相机位置和方向。



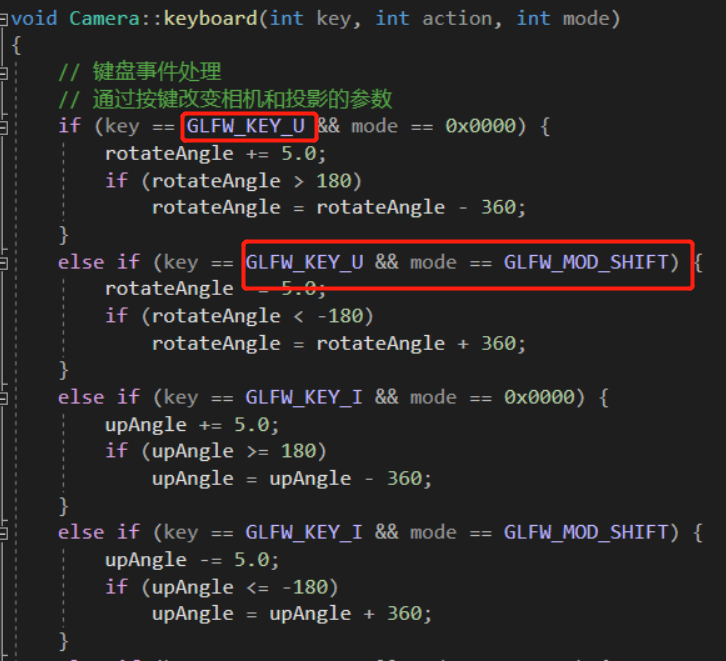
在display函数中调用get函数计算得到相机矩阵viewMatrix和projMatrix。其中在getProjectionMatrix函数中传入true变量表示采用正交矩阵。



在实现任意角度变化前需要先了解相机中的各个参数及实际意义。

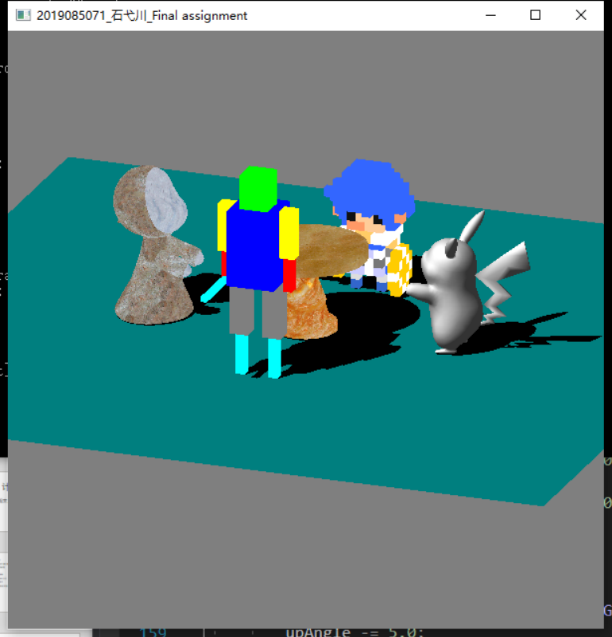
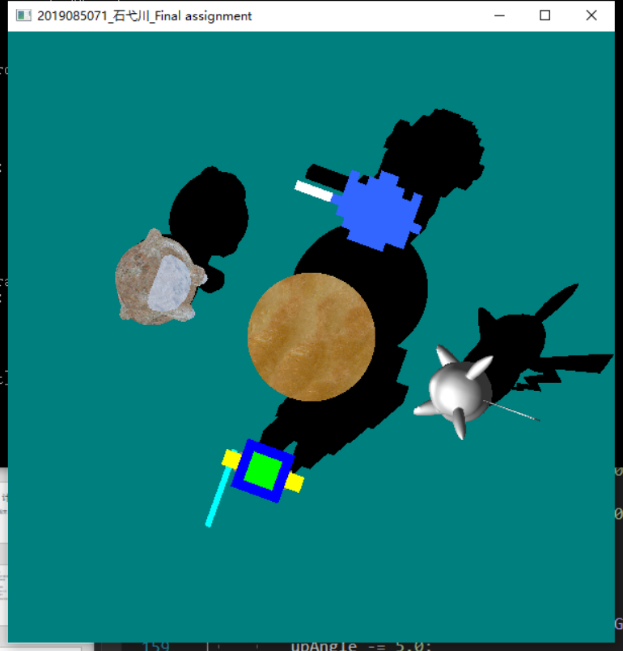


了解各个参数的意义后在在keyboard函数中设置通过按键改变相机参数，实现从任意角度浏览。



Figure键盘控制浏览视角的回调函数keyboard

通过按键U和I控制视角变动，实现从任意角度浏览，最终实现效果如下图所示：

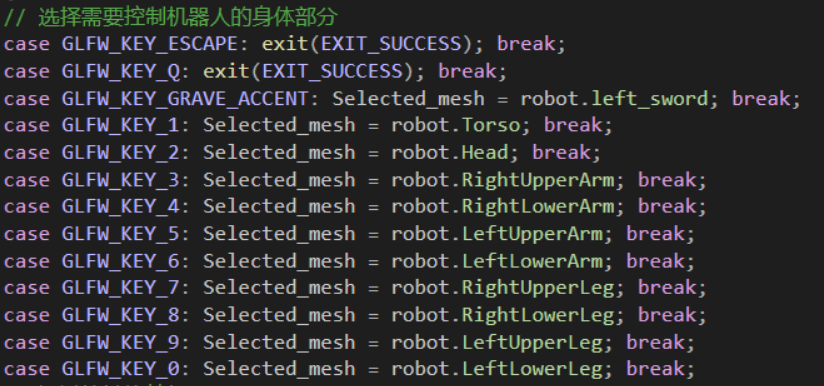
 

Figure键盘控制任意视角浏览

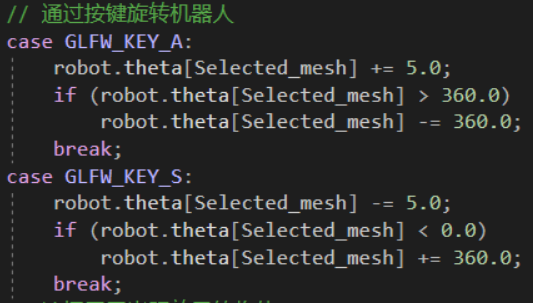
**5、通过交互控制物体**

这部分主要涉及到main.cpp主函数中key\_callback回调函数的编写。实现了若干功能，包括选择需要控制的物体，自由移动，改变光照位置，视角转动，尺寸缩放，旋转平移等。由于场景中有多个物体所以这里分部分说明。

选择需要控制机器人的人体部分。



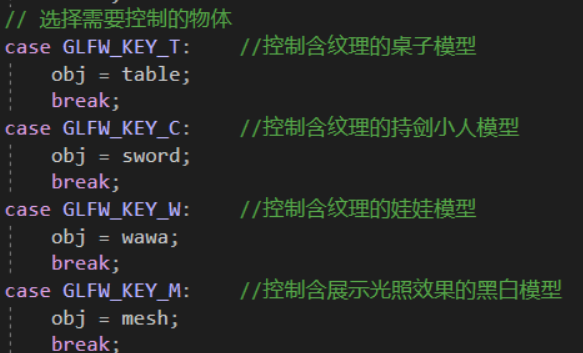
通过按键旋转机器人。



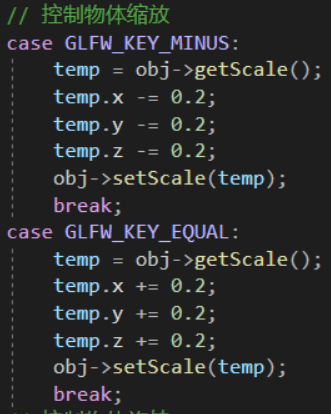
选择展示光照效果的物体。



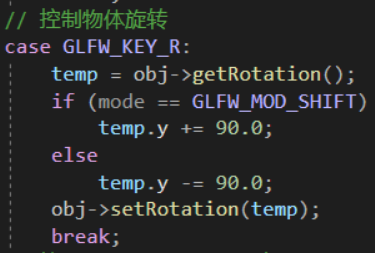
选择需要控制的物体。



控制物体缩放。



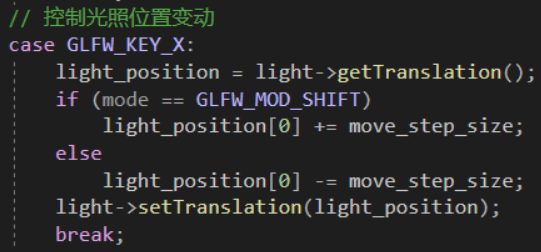
控制物体旋转。



控制物体在平面任意移动。

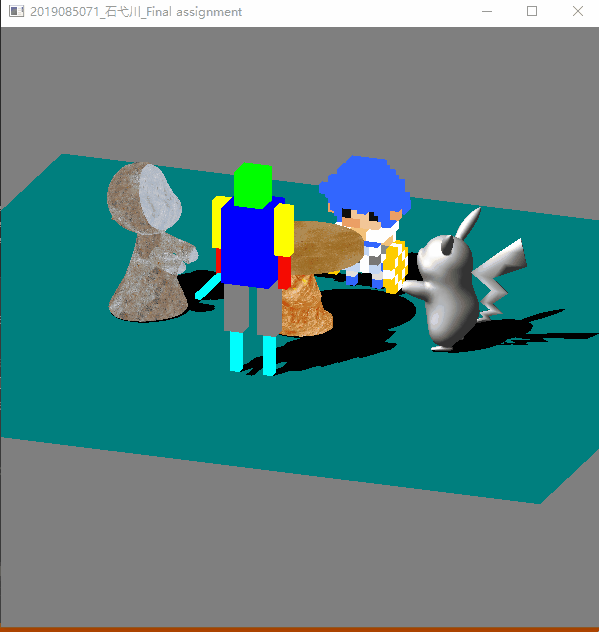


控制光照位置变动。



三、实验效果及结论

一系列操作完成实验后，最终的实验效果如下动态图所示：



在本次实验中，综合了这学期学过的众多知识，包括层级建模，图形绘制，物体绘制，阴影绘制添加纹理，键盘控制，场景交互，相机转动等，所含知识点和实验操作非常多。要想顺利完成整个实验，需要扎实的基础知识，熟悉并实操此前所完成的各种实验并掌握原理，这样才能在自己的场景中灵活运用实现想要的效果。

在我构想的场景中，主要是四个小人围绕在一个桌子周围，看似简单的场景实则蕴含了我们所学的所有知识，是花了大量时间和精力努力学习尝试最终完成的。

通过本次实验，我更加熟悉了此前不太掌握的纹理绘制和层级建模原理，此外最大的收益在于更深入了解了计算机图形学的原理，熟悉了OpenGL下的基本框架，学会该如何利用计算机进行包含众多物体的场景绘制以及交互控制物体。