# 《计算机图形学》课程报告

**课题名称： 机器人3D 图形建模和渲染**

课题负责人名（学号）： 宋俐潼2017141223035

同组成员名单（角色）：

宋俐潼 2017141223035 代码和报告编写

王艺瑾 2017141463228 代码和报告编写

指导教师： 陈蓉

**评阅成绩：**

评阅意见：

提交报告时间：2019 年 12 月 25日

**机器人3D 图形建模和渲染**

软件工程 专业

**学生** 宋俐潼 王艺瑾 **指导老师** 陈蓉

**[摘要]** 本次项目运用了计算机图形学所覆盖的大部分知识，例如物体建模、建模变换、漫游、投影、光照、贴图、透明以及雾化等等，然后通过WEBGL完成对于一个机器人3D 图形建模和渲染。而在此项目中，我们可以完成对于机器人整体以及部件的运动、缩放、旋转、平移，进行视角的切换、实现贴图、雾化以及透明的效果，从而加深对于图形学知识的理解。

**关键词**：贴图 透明 雾化 纹理 图形学 WebGL 建模

1. 概述（阐述问题特点，及本文采用的解决问题的关键技术）

随着对于计算机图形学学习的深入，我们想要使用已学习的知识完成一个综合性项目的实现，其功能主要含有：层级建模、视角变换、光照、透明以及雾化的效果等。通过完成这些功能，可以构造出一个更具有现实意义以及真实性的3D世界，与此同时还加深了我们对于图形学知识的认识与理解。而在实现项目效果的过程中，它我们使用到了WebGL这样一种JavaScript API，它可以帮我们更快速、便捷地完成效果的展现。

1. 设计（分析问题，设计解决问题的总体框图和思路）

在设计中有几个关键问题，其一是监听放缩键J/L和控制机器人变形的键来正确进行层次建模，其二是监听视点和投影变换键，修改视图投影矩阵viewProjMatrix来正确在显示器上成像，其三是正确实现光照，并正确将纹理映射到机器人上，其四是利用混合功能来实现半透明和雾化效果。

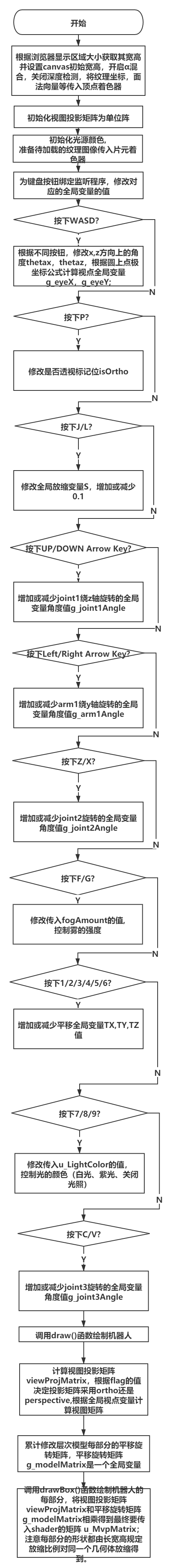
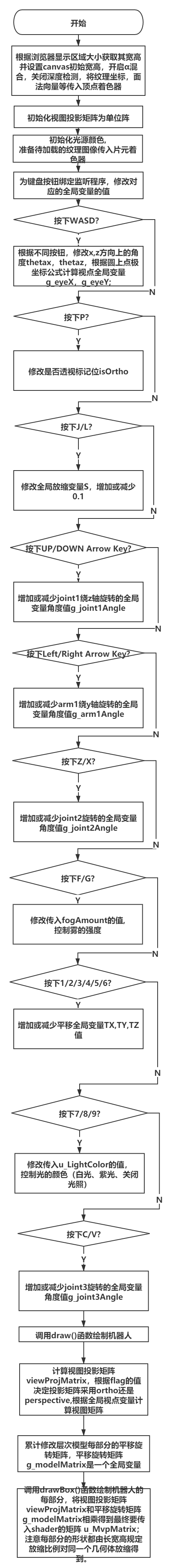
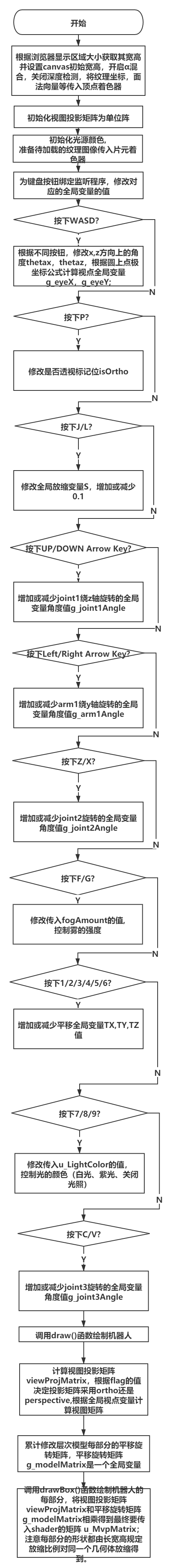
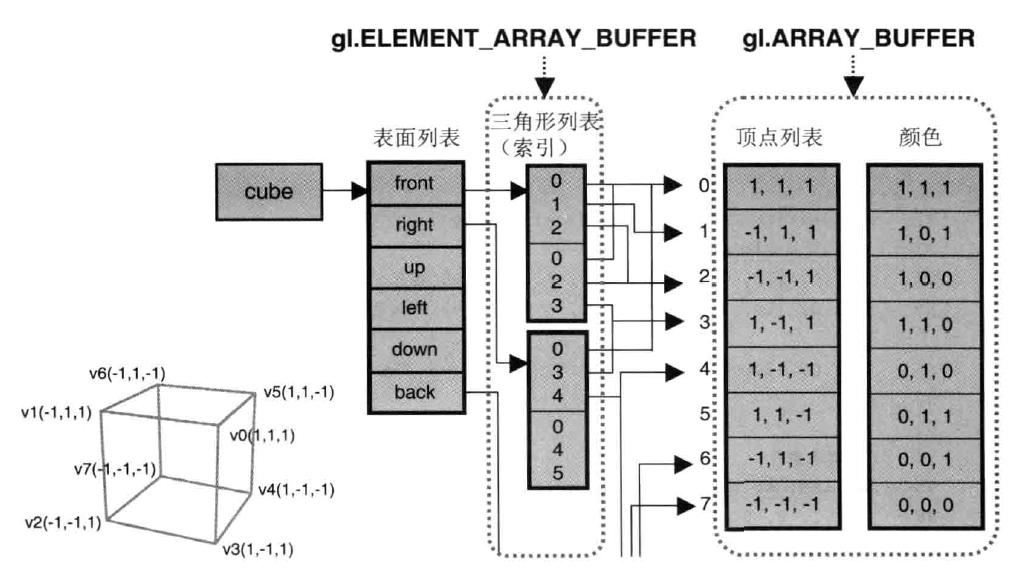
程序的总设计思路如下图所示（由于图片过大，拆分成了多张）。

图2.1流程图（1）  图 2.2流程图（2）

 图 2.3流程图（3）

1. 实现（阐述基于设计而实现的平台，关键技术和主要代码）
2. 层次建模&平移功能



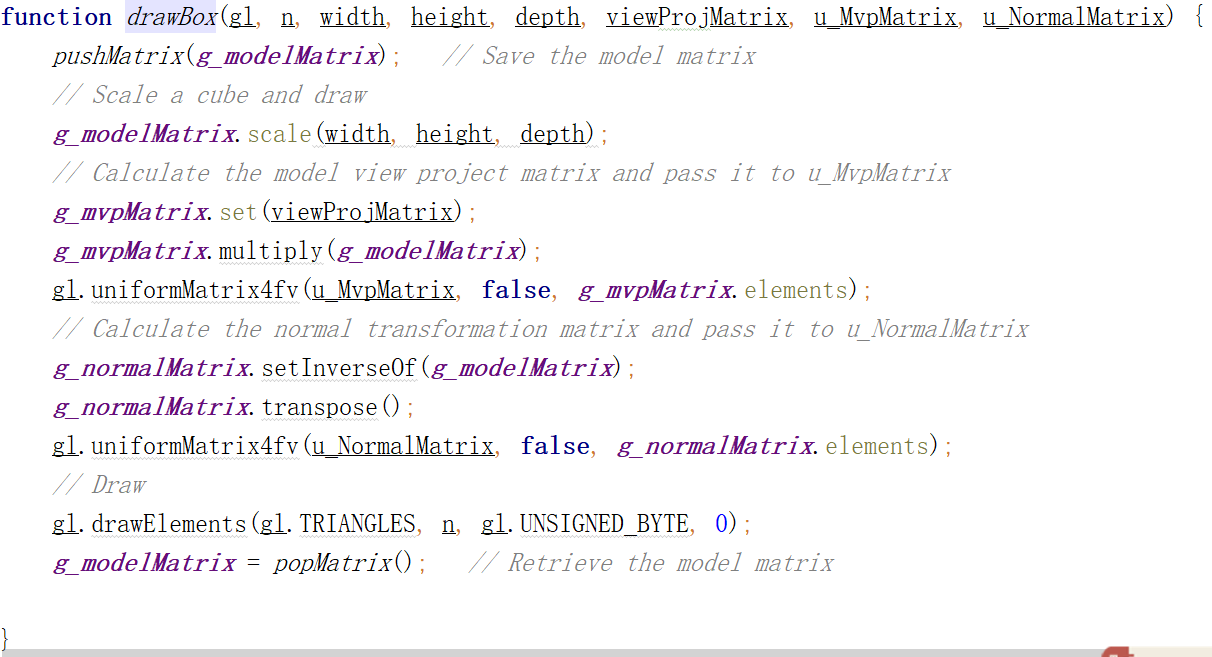
图 3.1原始BOX绘制数据结构示意图[1]

图 3.2 drawBox()函数

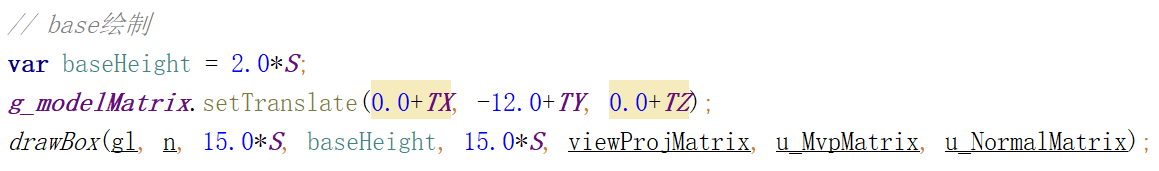
drawBox()是绘制机器人每一部分的函数,所有机器人部件共享同一个模型变换矩阵g\_modelMatrix, 父部件先修改这一矩阵，子部件在父部件修改的基础上修改这一矩阵，就实现了父部件带动子部件变化而子部件变化不影响父部件的效果。

图 3.3 平移base部件&修改放缩倍数代码

根据层级建模原理，优先级最高的根部件平移则带动所有部件平移。对数字键1,2,3,4,5,6监听，使得他们分别用于增加或减小TX,TY,TZ的值。具体监听设置与(四)中描述类似，在此不再赘述。

1. J/L放缩功能

由图3.2可知，所有机器人的部件都由原始BOX放缩并平移到对应位置得到，所以图3.3修改了drawBox()的width,height,depth参数就修改了原始BOX的放缩倍数，平移之后的部件自然放大或者缩小。

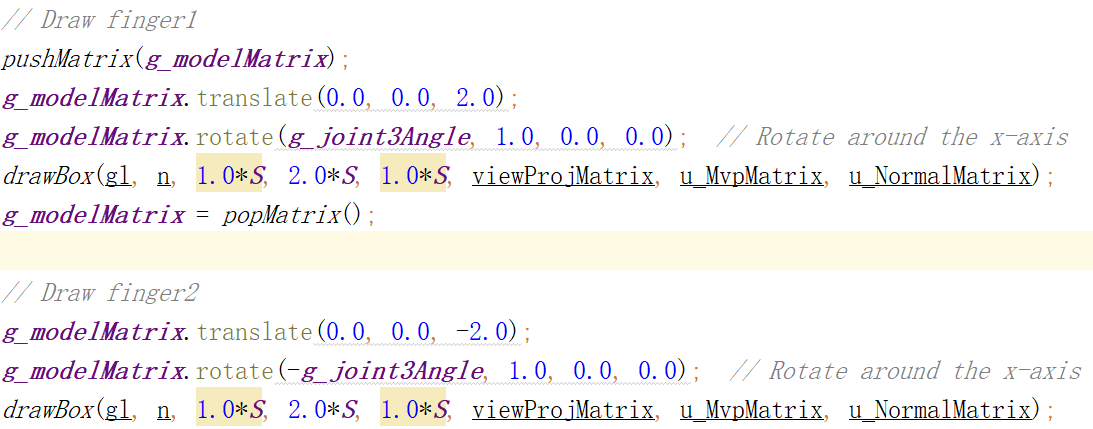
1. 层次建模中对同级别部件的特殊处理

图3.4绘制机器人两个手指代码

由于一个手指的运动不应该引向另一个手指，所以手指1对模型变换矩阵g\_modelMatrix的修改不应该影响到手指2，为此，我们采用了一个堆栈，在修改手指1之前将矩阵压入堆栈，修改之后再将其弹出恢复到矩阵未被修改时候的状态。

1. 使用W/S以及A/D视点变换

首先，声明表示视点坐标的角度全局变量, **var *theta***=45;

**var *phi***=45;

viewProjMatrix.lookAt(***ra***\***Math**.cos(***phi***)\***Math**.cos(***theta***),***ra***\***Math**.cos(***phi***)\***Math**.sin(***theta***),***ra***\***Math**.sin(***phi***), 0.0, 0.0, 0.0, 0.0, 0.0, **Math**.cos(***phi***));*//(0,1,0)*

图3.5设置lookat函数

之后在监听函数中增加W/S/A/D这四个键的监听。 四个按键对应的ASKII码分别是：

|  |  |
| --- | --- |
| 按键 | ASCII值 |
| A | 65 |
| S | 68 |
| W | 87 |
| S | 83 |

表3.1按键-ASCII值

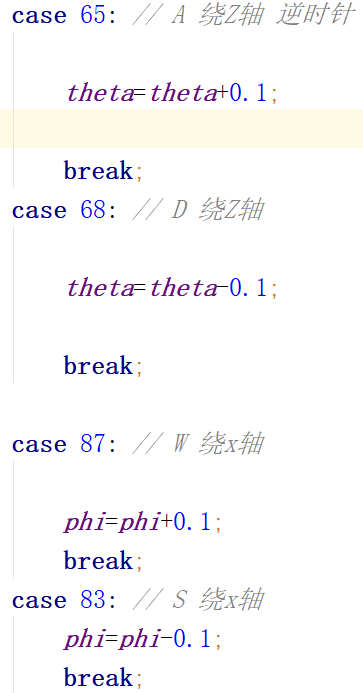
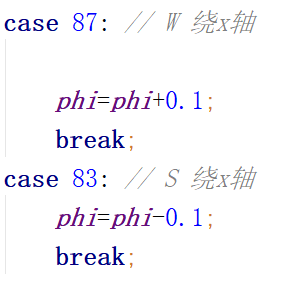
因此增加四个case语句，针对A按键（作用是使视点绕Z/轴逆时针旋转），实现代码如下：

图3.6使视点绕Z轴旋转 图3.7使视点绕X轴旋转

每按下一次A按键，视点沿着视线与Z轴的夹角减小ANGLE\_STEP。然后使用现在与Z轴的夹角计算出当前视点的坐标。其他三个按键的实现同理，不再赘述。

1. 使用P进行投影方式的切换

图3.8 声明isOrtho变量

首先使用变量isOrtho来表示当前投影的方式，如果为isOrtho为-1，当前投影方式为perspective，如果为1,则当前投影方式为ortho。

然后给P键加监听。由于P键对应ASKII为80，故代码如下：

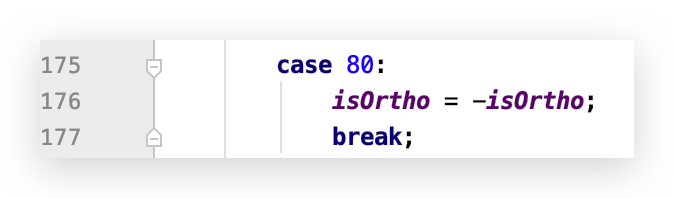


图3.9 监听P键

然后在draw()函数中对应isOrtho进行判断，如果isOrtho为1则使用setOrtho()函数将投影方式转化为ortho。如果isOrtho为-1，则使用setPerspective()函数将投影方式转化为perspective。

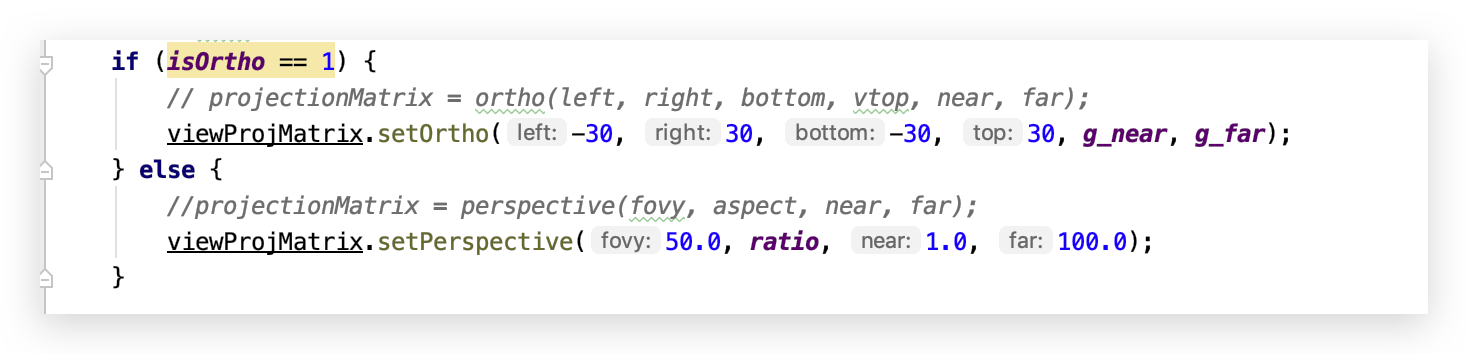


图3.10 转化投影方式

（六）光照的实现

可以通过如下公式实现漫反射光[2]：

也就是说，计算漫反射光颜色需要:a.入射光颜色b.表面基底色c.入射光方向，d.表面法线方向。在WEBGL中的实现方法如下：

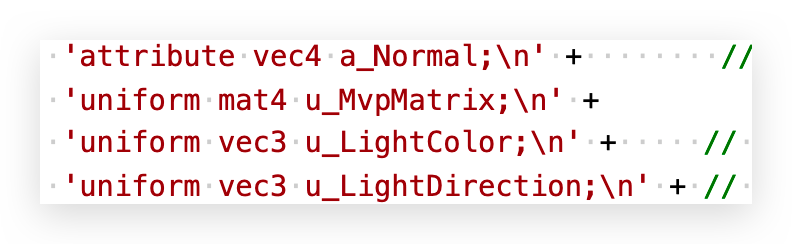
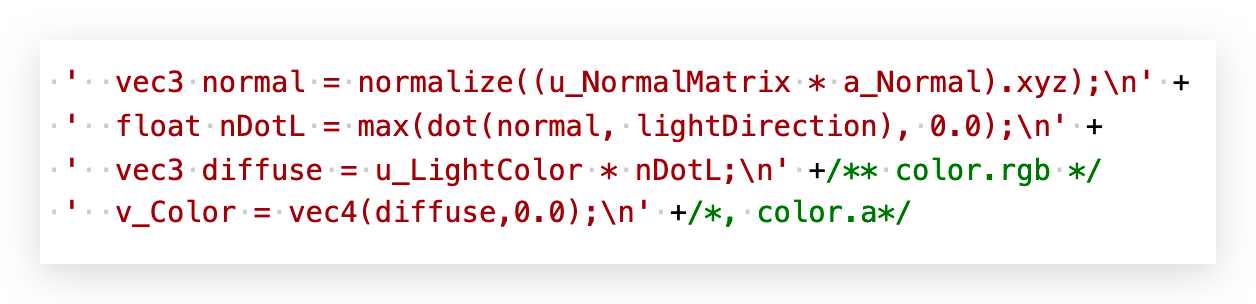
首先是顶点着色器中：

图 3.11 光照的实现 顶点着色器代码

然后是表面法线方向

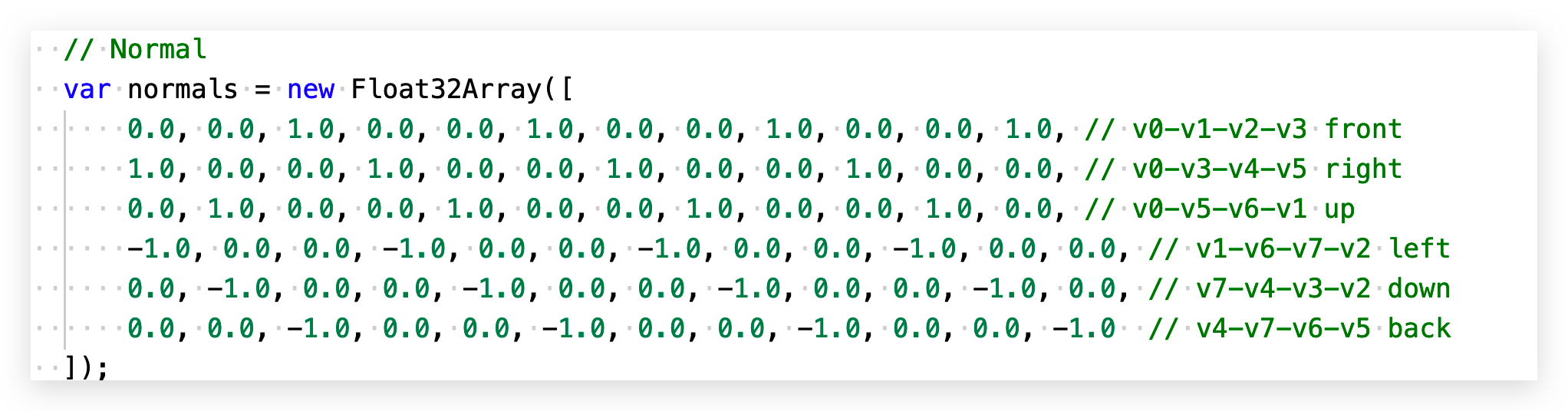


图 3.12 表面法向量

然后在片元着色器中，将纹理与光照进行线性叠加。

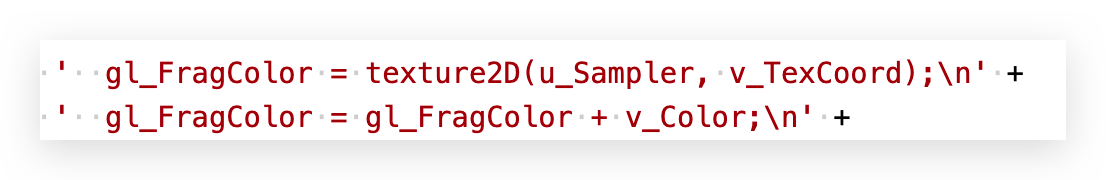
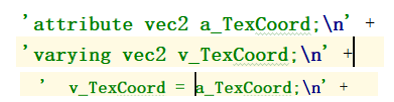


图 3.13 线性叠加

（七）纹理的实现

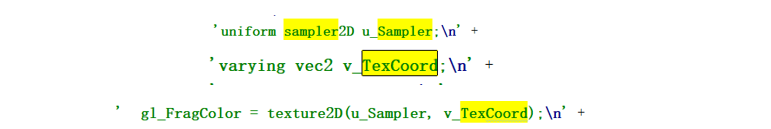
图3.14顶点着色器中和纹理有关的部分

图3.15 片元着色器中和纹理有关的部分

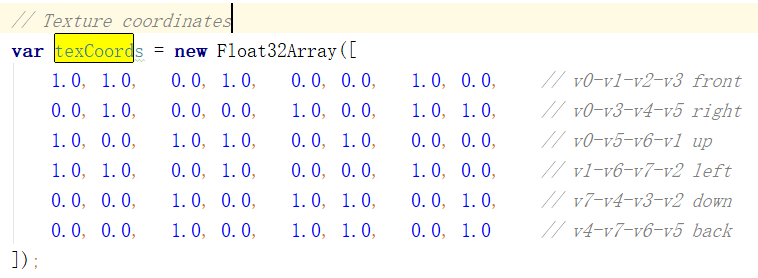


图 3.16 纹理坐标

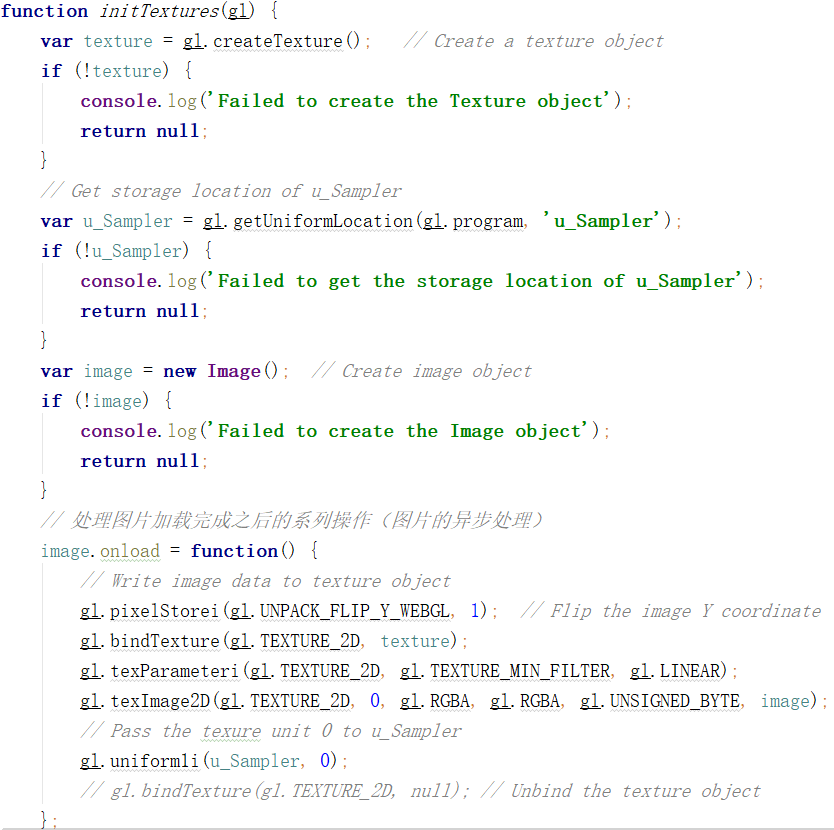
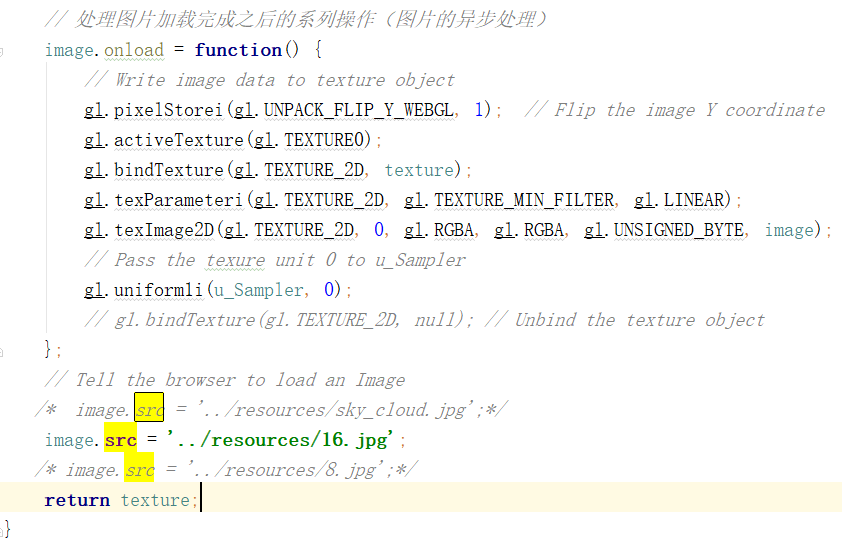


图3.17 initTextures()函数

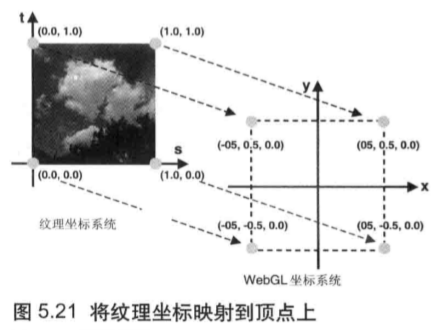


图3.18 将纹理坐标映射到顶点上[3]

我们在initTexture()函数中准备待加载的纹理图像，令浏览器读取它，监听纹理图像的加载事件，一旦加载完成，就在WebGL系统中使用纹理。具体来说，加载完成后的动作包括对纹理图像进行y轴反转，开启0号纹理单元，向target绑定纹理对象，配置纹理参数，配置纹理图像，最后将0号纹理传递给着色器的过程。

由图3.16和图3.18我们可以了解到我们是如何将纹理坐标映射到原始正方体的某个点上。

观察顶点及片元着色器可知，我们将纹理坐标赋值给了顶点着色器中的varying变量v\_TexCoord,它的值被传给片元着色器中的同名、同类型变量，更准确地说，顶点着色器中的v\_TexCoord变量在传入片元着色器之前经历了内插过程，如图3.19所示。

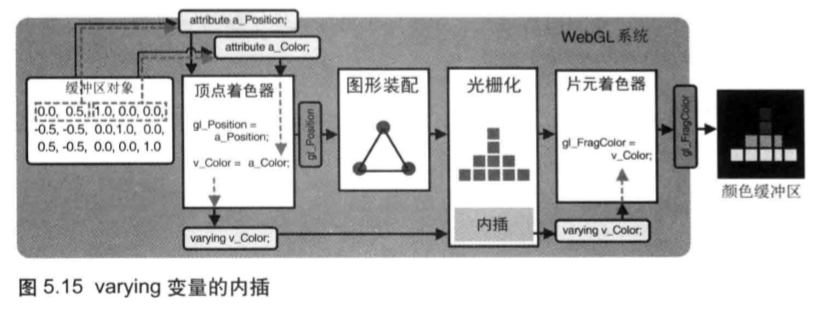
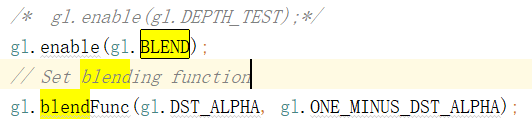


图3.19 varying变量的内插[4]

这样，纹理坐标光栅化后就传入了片元着色器，片元着色器根据片元的纹理坐标，使用texture2D()从纹理图像中抽取出纹素颜色，赋给当前片元。

（八）半透明的实现

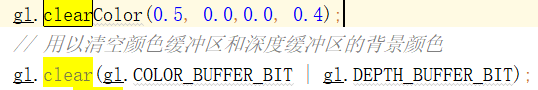
图3.20开启混合&关闭深度检测

图3.21设置canvas背景色

如图3.20所示，为了实现三维透明效果，我们开启了混合功能，并设置混合函数的参数为gl.DST\_ALPHA, gl.ONE\_MINUS\_DST\_ALPHA，使得src\_factor=(α,α,α) dst\_factor=(1-α,1-α,1-α)根据公式<混合后颜色>=源颜色x src\_factor+目标颜色 x dst\_factor.在这里源颜色为机器人颜色，目标颜色为canvas颜色。

canvas颜色的设置分两步：首先，gl.clearColor()来指定颜色。接着，gl.clear()来使用指定颜色清空颜色缓冲区。

另外，为了保证三维图形半透明的正确效果，又因为本程序中只有透明图形，所以采用了直接关闭深度检测的方案。

1. 雾化的实现

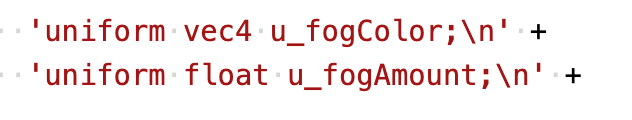
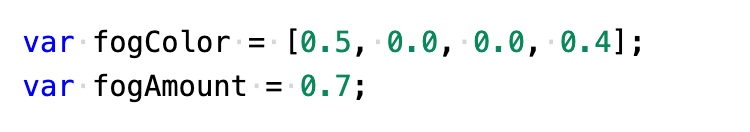
首先，在片元着色器中声明如下两个变量，代表雾的颜色与雾化因子。

图3.22 在片元着色器中声明变量

然后，设定雾的最初颜色和默认雾化因子。

图 3.23 设定雾的颜色和因子

将雾化因子设定在[0,1]范围

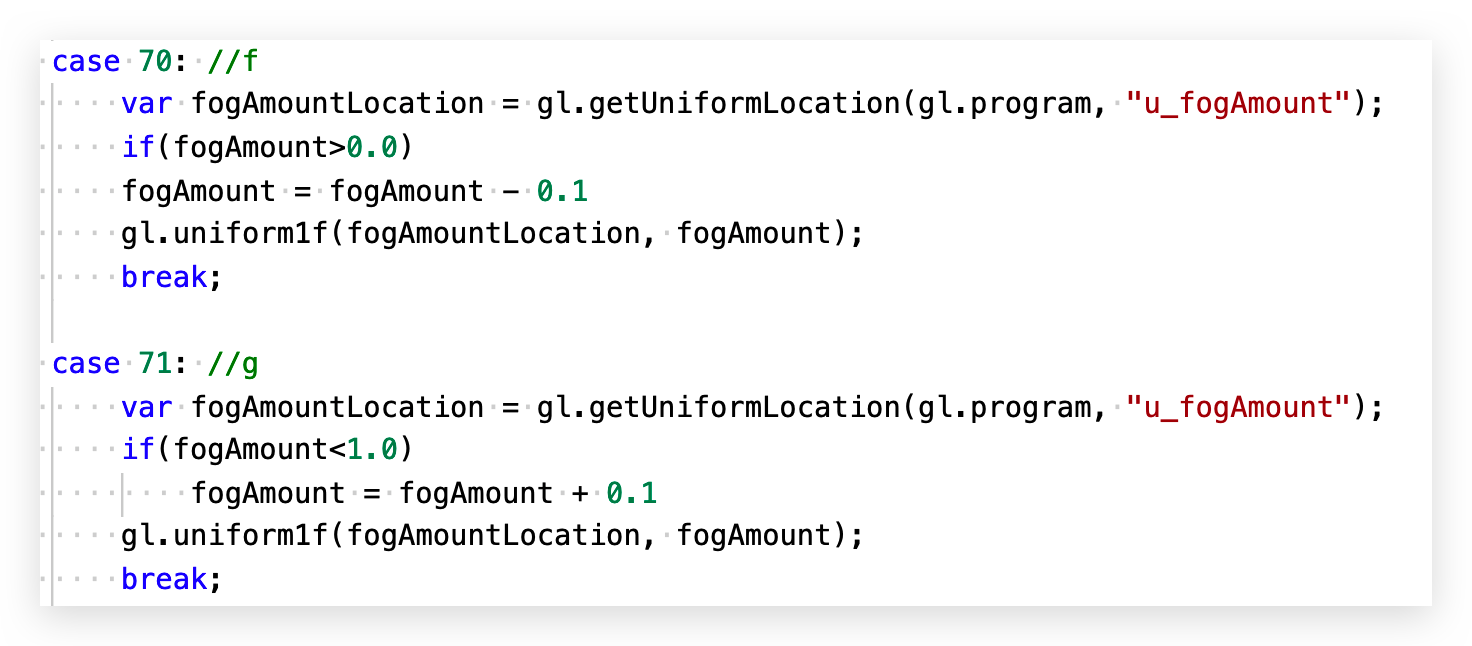


图 3.24 控制雾化因子的范围

最后，根据公式,在片元着色器中计算最终的片元颜色 。

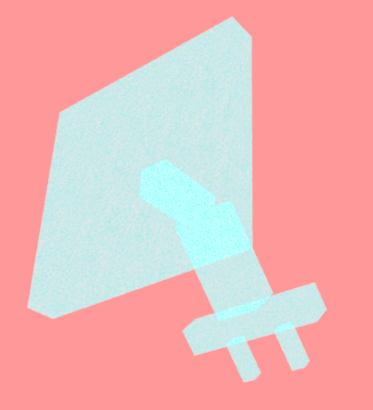
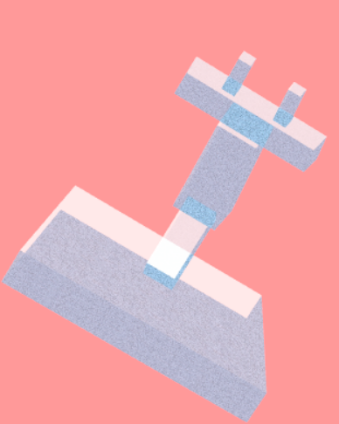
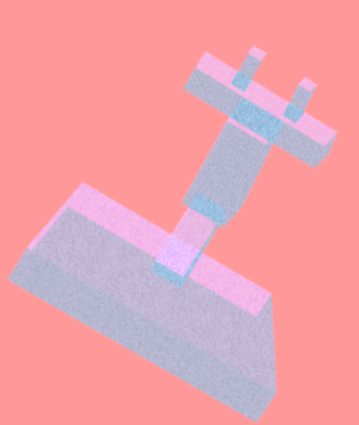
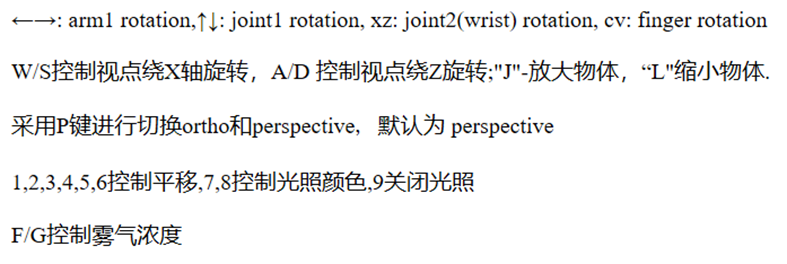
图 3.25 根据公式计算最终的片元颜色

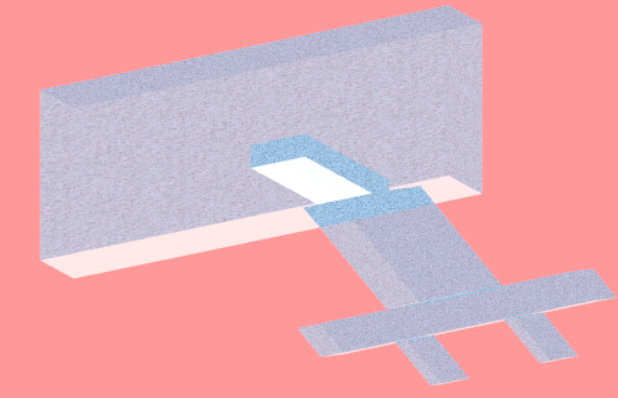
1. 总结（总结现有的优点和不足，个人的心得体会和改进方向）

本项目功能完备，灵活运用了计算机图形学知识，完整地将课堂中的知识用于实践。通过本门课程的学习，我们掌握了基本的计算机图形学知识，为日后的深入学习打下了坚实基础。

下面说明各种操作及展示运行效果。

图 4.1 页面上的操作说明

图 4.2 半透明及纹理效果展示 图4.3 光照效果展示

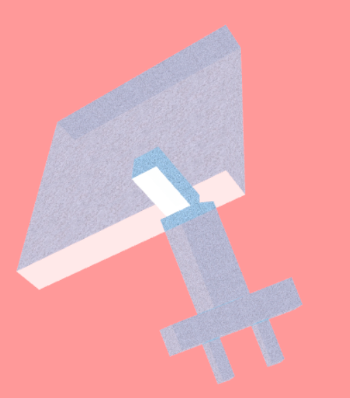
图

图4.4投影方式变化效果截图

如图4.4，人的视点没有改变，但是由于投影方式变化，机器人形状发生了变化。

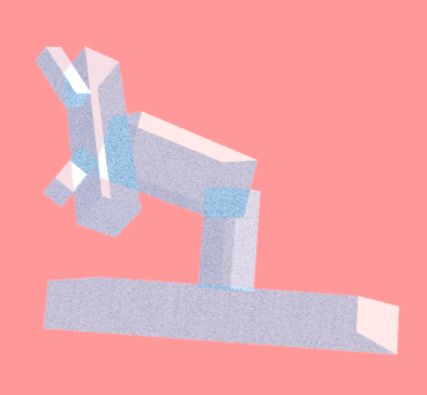


图 4.5机器人变形效果截图

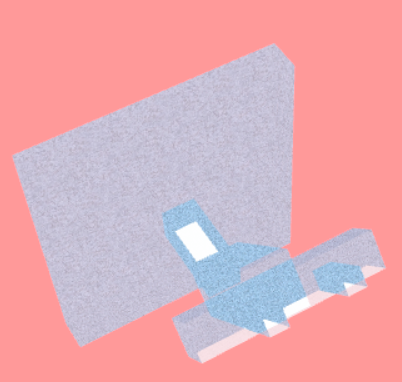
 

图4.6 雾化效果展示

平移、放缩、漫游效果请亲自运行程序验证，截图难以展示其效果.

# 参考文献

[1] KouichiMatsuda,RodgerLea.《WebGL编程指南》p322

[2] KouichiMatsuda,RodgerLea.《WebGL编程指南》ch8 p281

[3] KouichiMatsuda,RodgerLea.《WebGL编程指南》p156

[4] KouichiMatsuda,RodgerLea.《WebGL编程指南》p151