浙江大学实验报告

课程名称：____计算机图形学____ 指导老师：_____成绩：_____
实验名称：___OpenGL 显示加速技术_____ 实验类型：_____基础实验_____同组学生姓名：_____

装

订

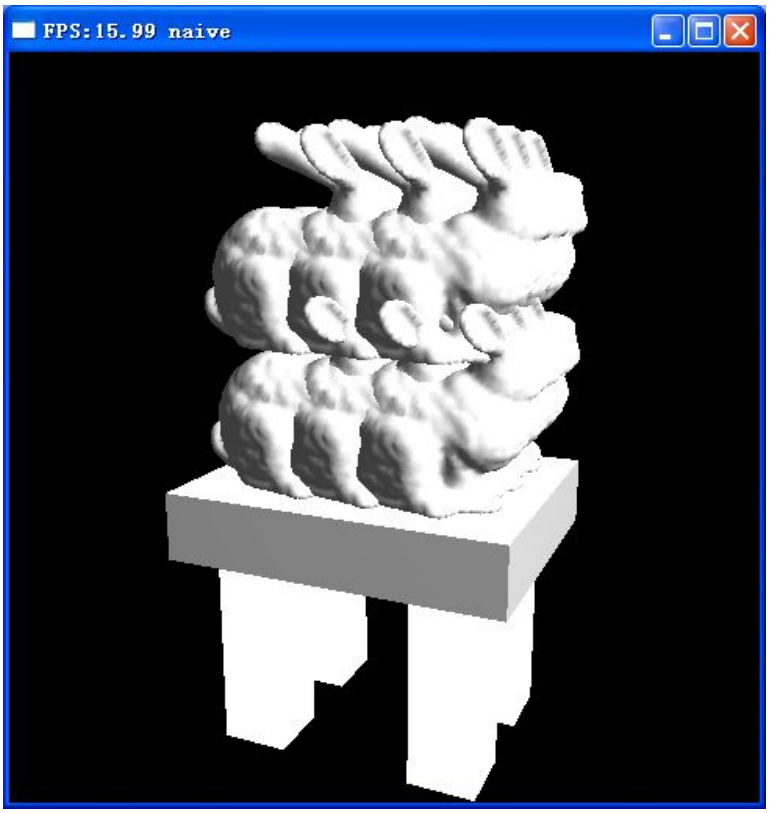
线

一、实验目的和要求

通过实现实验内容，掌握 OpenGL 中顶点数组和显示列表的使用，并验证课程中关于 OpenGL 显示加速技术的内容。

二、实验内容和原理

使用 Visual Studio C++编译已有项目工程。



要求修改代码达到以下要求：

1. 补充完成函数 drawVA()，实现使用顶点数组绘制场景：

```
void drawVA()
{
…
}
```

2. 补充完成函数 Gen3DObjectList ()，实现显示列表的生成：

```
GLint Gen3DObjectList()
{
…
};
```

3. 分析对比使用三种方法得到的 fps。
4. 添加拾取功能，对于鼠标点中的 Bunny 或桌子，改变显示颜色。

三、主要仪器设备

Visual Studio C++
glut-3.7.6-bin.zip
模板工程

四、操作方法和实验步骤

1.补充完成函数 drawVA()，实现使用顶点数组绘制场景：

在模板工程中的 bunny.cpp 中已经定义了 bunny 模型的顶点数据：

```
// 每个面的三个顶点的指标
short face_indicies[16301][3] = {
......
}
// 每个点的法向
GLfloat normals [8146][3] = {
......
}
// 每个顶点的位置坐标
GLfloat vertices [8146][3] = {
......
}
```

将这些数据传输给 opengl 后便可以利用顶点数组进行加速绘制：

```
void drawVA()
{
    glEnableClientState(GL_NORMAL_ARRAY);  // 激活顶点法向量数组
    glEnableClientState(GL_VERTEX_ARRAY);  // 激活顶点空间位置数组
    glNormalPointer(GL_FLOAT, 0, normals);  // 指定顶点法向量数组
    glVertexPointer(3, GL_FLOAT, 0, vertices);  // 指定顶点空间位置数组

    // 使用顶点数组进行绘制
    glDrawElements(GL_TRIANGLES, 16301*3, GL_UNSIGNED_SHORT, face_indicies);

    // 关闭顶点数组
    glDisableClientState(GL_VERTEX_ARRAY);
    glDisableClientState(GL_NORMAL_ARRAY);
```

```
}
```

2.补充完成函数 Gen3DObjectList ()，实现显示列表的生成：

使用显示列表绘制图形先要创建显示列表，首先需要为每一个显示列表指定一个唯一的标识符：

```
// 显示列表标识
GLint dl = 0;
```

接着通过使用函数 glNewList（）和 glEndList（）创建显示列表，并返回显示列表标识：

```
GLint Gen3DObjectList()
{
// #pragma MSG("请完成, by 唐敏")
    GLint lid = glGenLists(1);
    glNewList(lid, GL_COMPILE);  // 编译，但不立即显示
    // 下面是绘制指令，同不使用显示列表方式相同
    drawNaive();

    glEndList();  // 该显示列表被创建
    return lid;   // 返回标识
}
```

在创建显示列表之后，通过 glCallList(...)来调用显示列表实现重绘图形：

```
void drawDL()
{
    glCallList(dl);
}
```

3.添加拾取功能，对于鼠标点中的 Bunny 或桌子，改变显示颜色

OpenGL 中通过选择模式实现了物体的拾取，主要步骤如下：

定义鼠标点击响应函数，进入选择模式：

```
// 鼠标点击响应函数
void MouseClick(int button,int action,int xMouse,int yMouse)
{
    if(button == GLUT_LEFT_BUTTON && action == GLUT_DOWN)  // 鼠标左键按下
    {
        GLuint selectBuf[BUFSIZE];  // 选择缓存区，保存击中记录的信息
        GLint hits;   // 保存击中记录
        GLint viewport[4];  // 视点信息
        glGetIntegerv (GL_VIEWPORT, viewport); // 获得当前视点信息
        glSelectBuffer (BUFSIZE, selectBuf);  //设置选择缓冲区的大小

        // 进入选择模式
        glRenderMode(GL_SELECT);                    // Enter the SELECT render mode
        glInitNames();    // 初始化名字堆栈
```

```
        glPushName(-1);  // 压入一个不会使用的标识

        glMatrixMode(GL_PROJECTION);
        glPushMatrix();
        glLoadIdentity();

        //定义一个2×2的选择区域
        gluPickMatrix((GLdouble) xMouse, (GLdouble) (viewport[3] - yMouse), 2.0, 2.0,
viewport);
        gluPerspective(45.0f, whRatio,0.1f,100.0f);
        glMatrixMode(GL_MODELVIEW);

        // 在选择模式下绘图，Draw_Triangle传入GL_SELECT参数会根据击中记录设置物体材质
        Draw_Triangle(GL_SELECT);

        glPopMatrix ();
        glFlush ();

        // 恢复投影变换
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glViewport(0,0,wWidth,wHeight);
        gluPerspective(45.0f, whRatio,0.1f,100.0f);

        // 获得击中记录并处理
        hits = glRenderMode (GL_RENDER);

        ProcessPicks(hits, selectBuf);
        updateView(wHeight, wWidth);
    }
}
```

通过处理击中记录判断当前哪一个物体被选中（选择选择缓冲区中最小深度值最小的物体）：

```
//  处理击中记录的函数
void ProcessPicks(GLint nPicks, GLuint pickBuffer[])
{
    if (nPicks <= 0)  // 无击中记录则不处理
    {
        return;
    }
    GLint i;
    GLuint name, *ptr, minZ, *ptrNames, numberOfNames;
    minZ = 0xffffffff;
    printf("选中的数目为%d个\n",nPicks);
```

```
    ptr=pickBuffer;

    for(i = 0; i < nPicks; i++)
    {
        name = *ptr;    // 击中记录的名字数
        ptr++;  // 最小深度记录
        if (*ptr < minZ)  // 找到最小深度的物体名字
        {
            numberOfNames = name;
            minZ = *ptr;
            ptrNames = ptr+2;  // 当前ptr指向最小深度记录，ptr+1指向最大深度记录，ptr+2指向
击中记录中的第一个名字
        }
        ptr += name + 2; // ptr 移动到下一个击中记录
    }
    if (*ptrNames == 1)
    {
        printf("你选择了Bunny\n");
        bSelectBunny = !bSelectBunny;
    }
    if(*ptrNames == 2)
    {
        printf("你选择了Desk\n");
        bSelectDesk = !bSelectDesk;
    }
    printf("\n\n");
}
```

还要注意当在选择模式中绘制时要把想要选中的物体名字（数字标识）压入名字堆栈中：

```
void Draw_Triangle(GLenum mode) // This function draws a triangle with RGB colors
{
    if(mode == GL_SELECT) glLoadName(1); // 将Bunny的名字标识压入名字堆栈

    glPushMatrix();
    glTranslatef(-1, -1, 5.5);
    drawBunny();
    glPopMatrix();

    ..... //

    if(mode == GL_SELECT) glLoadName(2); // 将Desk的名字标识压入名字堆栈
    Draw_Desk();
}
```

通过设置物体材质改变物体的颜色：

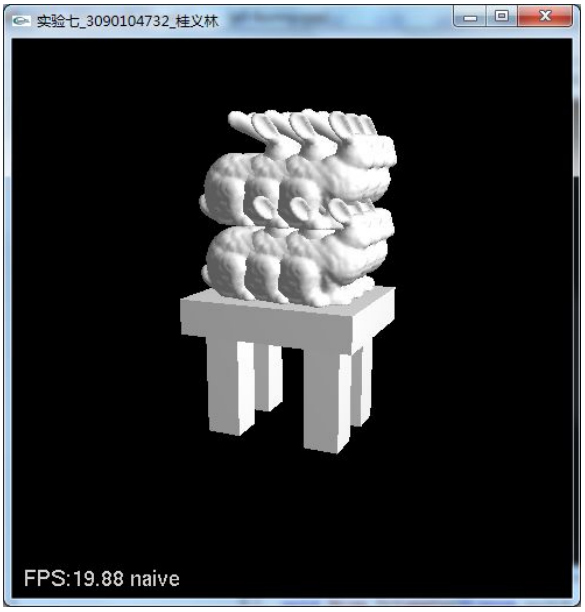
```
void Draw_Desk()
{
    // 根据物体的选中状态设置材质以改变颜色
    if (bSelectDesk)
    {
        glMaterialfv(GL_FRONT, GL_AMBIENT, mat_desk2);
    }
    else
    {
        glMaterialfv(GL_FRONT, GL_AMBIENT, mat_desk1);
    }
    glPushMatrix();
    glTranslatef(0, 0, 3.5);
    glScalef(5, 4, 1);
    glutSolidCube(1.0);
    glPopMatrix();
    ......
}
void drawBunny()
{
    // 根据物体的选中状态设置材质以改变颜色
    if (bSelectBunny)
    {
        glMaterialfv(GL_FRONT, GL_AMBIENT, mat_bunny2);
    }
    else
    {
        glMaterialfv(GL_FRONT, GL_AMBIENT, mat_bunny1);
    }
    glRotatef(90, 1, 0, 0);
    glScalef(3, 3, 3);
    if (drawMode == 0)
        drawNaive();
    else if (drawMode == 1)
        drawVA();
    else
        drawDL();
}
```

五、实验数据记录和处理

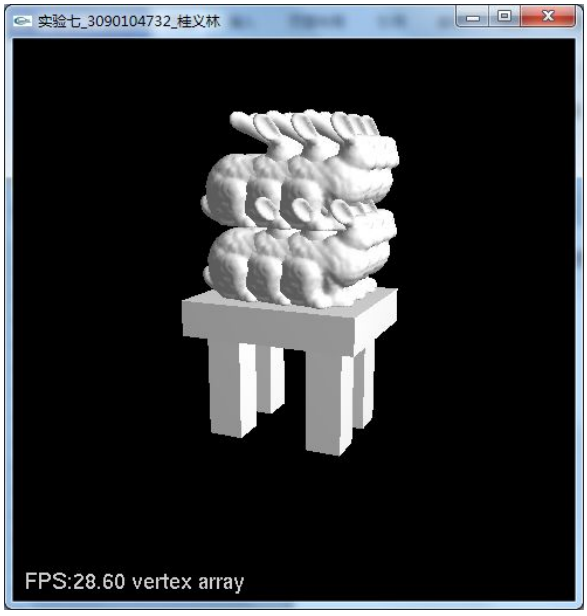
无

六、实验结果与分析

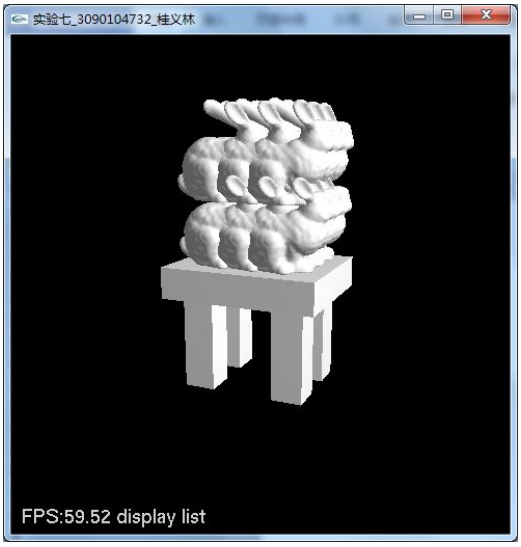
1.三种绘制方式 FPS 的比较



直接绘制结果



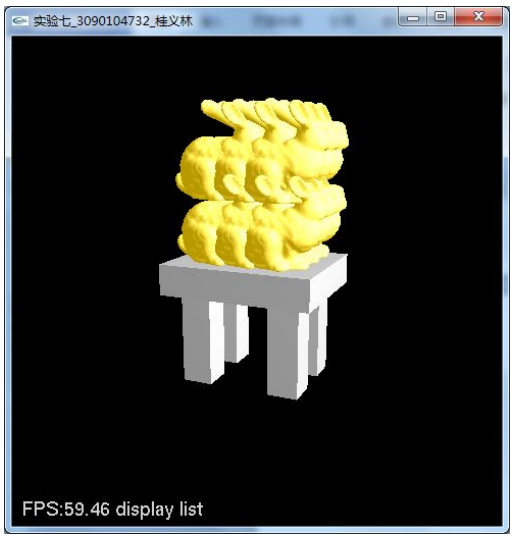
顶点数组绘制结果

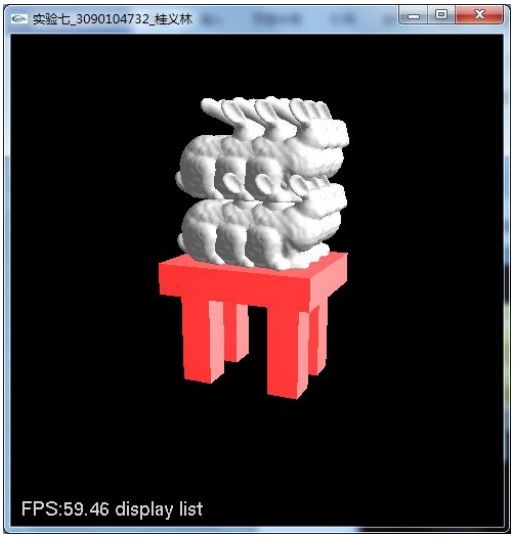


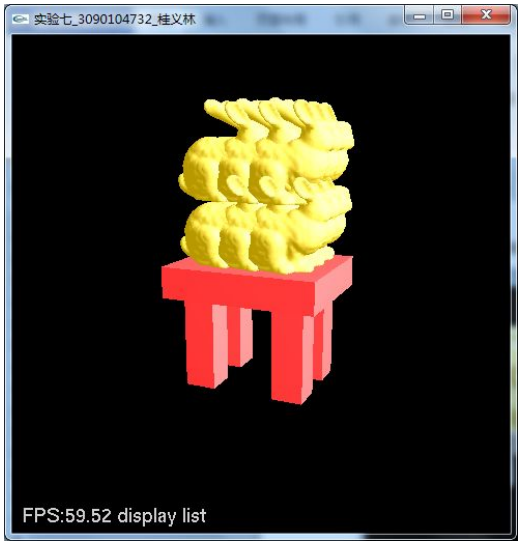
显示列表绘制结果

从上面图中的结果可以看出，使用顶点数组和显示列表都比直接绘制的 FPS 要高，但使用显示列表方式 FPS 提高的更加明显。

2. 鼠标拾取改变 bunny 和 desk 的颜色







七、讨论、心得

通过本次实验我了解并掌握了 OpenGL 中顶点数组和显示列表的使用，通过这两种方式在处理大量绘图数据时可以起到加速绘制的效果。由于本次实验中有 18 个 Stanford_bunny 需要绘制，绘制数据十分庞大，因此使用显示列表加速的效果比顶点数组绘制的效果更为明显，应为顶点数组只是把顶点数据交给 OpenGL 处理减少面之间的共享顶点，从而减少数据冗余，而显示列表方式是预先处理绘制数据，在显示时直接调用，因此效果更好。

在添加拾取功能的过程中我了解了 OpenGL 是通过选择模式和名字堆栈来实现物体的选取，通过自己编写处理击中记录的函数选择鼠标左键点击处深度最小的物体并改变其颜色。在实验中我发现在开启光照的模式下，通过 glColor()函数是无法改变物体的颜色的，必须为物体设置相应的材质才能够正确的显示物体的颜色。