学号	姓名	论文规范性（10）	问题分析与调研（30）	方案创新性（20）	实验结果分析与讨论（40）	结课论文总成绩（100）
21301061	陈方舟	6	20	15	32	73

纹理加载功能的实现

摘要：早期计算机生成的三维图像看起来往往像是发亮的塑料，虽然这在当时也是比较先进的，但是它们缺乏各种纹路——如磨损、裂痕、指纹和污渍等，而这些纹路会增加三维物体的真实感。近年来，纹理已经在开发人员中得到普及并作为增强计算机生成的三维图像的真实感的工具。
词语“纹理”在日常使用中表示物体的光滑度或粗糙度，但是在计算机图形学中，纹理指的是一张表示物体表面细节的位图。

引言：

纹理可以映射到平面上，也可以映射到曲面上，并且还可以多层映射。多层映射使用的是多重纹理。

使用纹理映射时，首先要定义一个纹理，然后控制纹理的滤波，说明映射的方式，在绘制顶点时同时给出其纹理坐标即可。

理论上来说，只要是图形文件都可以作为纹理贴图，不过最常用的还是 BMP、JPEG、TGA 文件。BMP 文件处理方便，JPEG 文件压缩率高，而 TGA 文件一贯作为纹理贴图文件用于三维动画设计软件中，如 3DS MAX 等。QUAKE III 是用的是 JPEG 和 TGA 文件。

为了实现纹理贴图我们需要做三件事:将一张贴图加载到OpenGL中，提供纹理坐标和顶点（将纹理对应匹配到顶点上），并使用纹理坐标从纹理中进行取样操作取得像素颜色。由于三角形会被缩放、旋转、平移变换导致最后会以不同的结果投影显示到屏幕上，而且由于 camera 的不同操作看上去也会很不一样。GPU 要做的就是让纹理紧跟

三角形图元顶点的移动使其看上去真实（如果纹理看上去明显游离在三角形上产生错位就不真实了）。为实现这个效果开发者需要为每个顶点提供一系列纹理坐标。在 GPU 光栅化三角形阶段，会对纹理坐标进行插值计算并覆盖到整个三角形面上，并且在片段着色器中开发者要将这些坐标跟纹理进行匹配。这个操作叫做‘取样’，取样的结果叫做‘纹素’（纹理中的一个像素）。纹素通常包含一个颜色值用于画屏幕上对应的一个像素。

OpenGL 支持几种不同类型的纹理：1D,2D,3D,立方体等等，可以应用于不同的技术中。

相关工作描述：

该实验是基于 OpenGL 的卧室仿真设计，房间内包含一张床，一个衣柜，一个房间的顶灯，一个床头柜，一盏台灯以及墙上的一幅画组成。房间的墙壁和地板以及墙上的挂画采用贴图实现并且为其添加仿真纹理。

方法描述：

执行纹理贴图的步骤可以概括为：定义纹理贴图；控制纹理；说明纹理贴图方式；激活纹理映射；定义纹理坐标等。

本实验调用 glBindTexture()函数，将纹理名“绑定”到纹理数据上，当第一次调用此函数时，将会生成一个新的纹理对象，其具有

默认的纹理图像和纹理属性。在初始化绑定之后所调用的任何OPENGL纹理函数都会改变默认值并将新值存储到该纹理对象中。当一个纹理对象被绑定到其数据上之后，可以再次调用 glBindTexture()函数来将此纹理对象设置为当前的纹理状态。

实验设置：

使用 OPENGL 的纹理参数函数，可以设置放大和缩小过滤器。这两种过滤器的参数名分别为 GL_TECTURE_MAG_FILTER 和 GL_TEXTURE_MIN_FILTER。纹理坐标总是根据纹理单元进行计算和测量。无论纹理坐标落入哪个纹理单元，这个纹理单元的颜色就作为这个片段的纹理颜色。

本实验调用下面函数为放大和缩小过滤器设置纹理过滤器：

```
void Texture(GLuint textur)
{
    glBindTexture(GL_TEXTURE_2D, textur);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR_MIPMAP_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_NEAREST);
    glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_MODULATE);
}
```

OPENGL 允许通过纹理模式来指定如何对纹理图颜色进行处理。对于每个纹理，可以调用 glTexEnvf()函数来从 4 个纹理模式中进行选择：

Void glTexEnvf(GLenum target, GLenum pname, GLenum param).

Target 参数必须等于 GL_TEXTURE_ENV，同时也必须将 pname 参数值设为 GL_TEXTURE_ENV_MODE，告诉 OPEBGL 将指定纹理如何与帧缓存

中的颜色进行混合。Param参数可以设为下表中所列出的任何一个值。

模式	说明
GL_BLEND	纹理颜色与像素颜色相乘并与一个常数颜色值混合
GL_DECAL	纹理取代已有的像素
GL_MODULATE	纹理颜色与像素颜色相乘

本程序选择GL_MODULATE参数。

当绘制场景时，必须为每一个顶点指定相应的纹理坐标。纹理坐标用于确定纹理图的每一个texel对应于物体的哪个部分。纹理坐标被指定为浮点值，范围从0.0至1.0.纹理坐标被命名为s、t、r和q，支持从一维至三维的纹理坐标。Q坐标对应于几何图形的w坐标，是一个缩放因子，作用于其余纹理坐标。使用glTexCoord()函数指定一个纹理坐标，有三种形式：

```
Void glTexCoord1f(GLfloat s)
Void glTexCoord2f(GLfloat s, GLfloat t)
Void glTexCoord3f(GLfloat s, GLfloat t, GLfloat r)
```

纹理坐标使用上述函数应用到几何物体的每个顶点，然后OENGL根据需要对纹理进行扩大或收缩，将纹理贴图到几何图形上。

本程序均调用glTexCoord2f()函数，如：

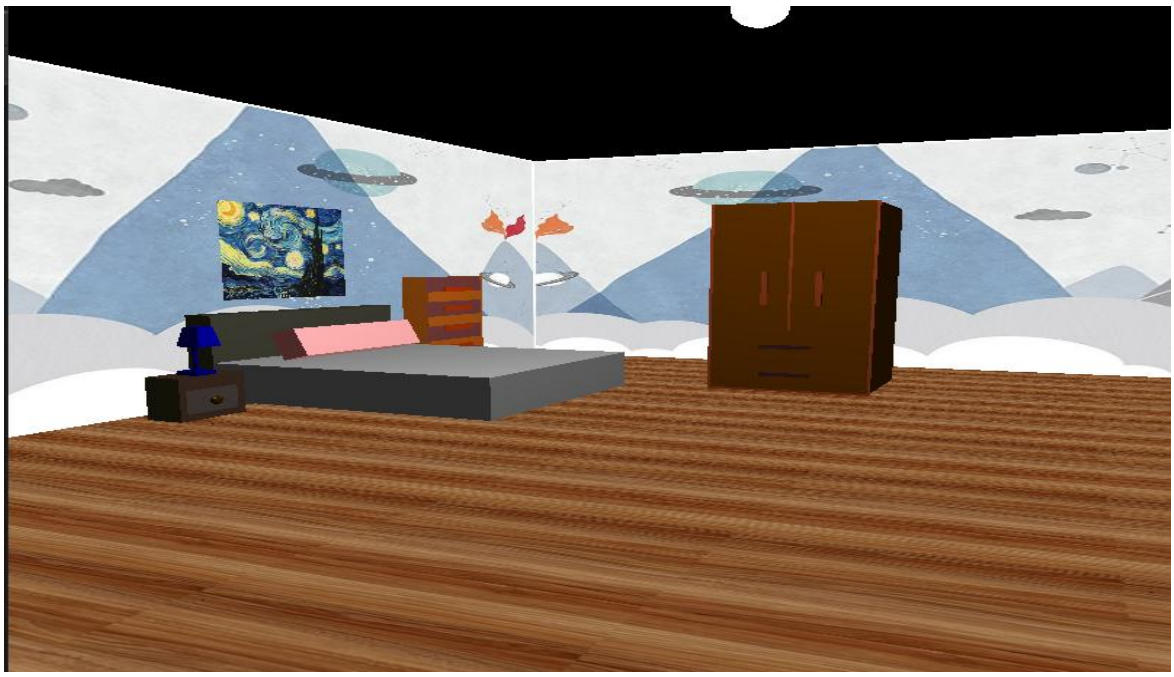
```
glPushMatrix();
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, textureVec[1]);
glBegin(GL_QUADS);
glNormal3f(0, 0, -1);
glTexCoord2f(0, 0); glVertex3f(13, -0.5, 15);
glTexCoord2f(1, 0); glVertex3f(-2, -0.5, 15);
glTexCoord2f(1, 1); glVertex3f(-2, 5, 15);
glTexCoord2f(0, 1); glVertex3f(13, 5, 15);
glEnd();
glDisable(GL_TEXTURE_2D);
glPopMatrix();
```

墙上挂画的纹理映射：

```
void Pic()
{

    glPushMatrix();

    GLfloat no_mat[] = { 0.1, 0.1, 0.1, 1.0 };
    GLfloat mat_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_shininess[] = { 32 };

    //设置材料属性
    glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);    // 材质的环境颜色
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);    // 材质的散射颜色
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);    // 材质的镜面反射颜色
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);    // 镜面反射指数

    glColor3f(1.0, 1.0, 1.0);
    glPushMatrix();
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, textureVec[2]);
    glBegin(GL_QUADS);
    glNormal3f(1, 0, 0);
    glTexCoord2f(0, 0); glVertex3f(-0.99, 1.4, 5);
    glTexCoord2f(1, 0); glVertex3f(-0.99, 1.4, 7);
    glTexCoord2f(1, 1); glVertex3f(-0.99, 3.2, 7);
    glTexCoord2f(0, 1); glVertex3f(-0.99, 3.2, 5);
    glEnd();
    glDisable(GL_TEXTURE_2D);
    glPopMatrix();
    glPopMatrix();
}
```

实验结果与分析：

实验结果：



纹理加载成功！通过 loadGLTexture 函数，我们成功加载了 BMP 文件格式的纹理，并将其应用到 OpenGL 场景中。加载和显示纹理的性能良好，显示效果平滑，未出现卡顿或性能问题。纹理映射准确无误，正确显示了 BMP 文件中的图像内容。通过上述步骤，我们成功实现了一个基本的纹理加载功能，并通过实验验证了其在 OpenGL 场景中的应用效果。

实验分析：

本次实验使用 stb_image.h 库加载 BMP 文件。这个库支持多种图像格式，使用起来非常方便。在实验中，我们使用 stbi_load 函数将图像文件读取到内存中，并获取图像的宽度、高度和通道数。随后使用 glGenTextures 函数生成一个纹理对象，并使用 glBindTexture 函数将其绑定到当前的 OpenGL 上下文中。

在实际应用中，纹理映射不仅需要考虑图像本身的质量，还需要

细致地处理纹理坐标的映射。通过设置合适的环绕方式和过滤方式，可以有效提高纹理的显示效果。

而纹理参数我们是使用 glTexParameteri 设置纹理参数，如环绕方式和过滤方式。实验中设置了环绕方式为 GL_REPEAT，过滤方式为 GL_LINEAR。其次使用 glTexImage2D 函数将图像数据传递给 OpenGL，建立纹理。

最后需要使用 stbi_image_free 释放图像数据占用的内存。

实验结论：

本实验基于 OpenGL 的卧室仿真设计，通过加载和应用纹理来增强计算机生成图像的真实感。在计算机图形学中，纹理指的是一张表示物体表面细节的位图，通过映射不同的纹理，可以显著提升三维图像的视觉效果。实验中，我们在卧室场景中的墙壁、地板和墙上的挂画上使用了纹理贴图。

通过这次实验，我深入理解了纹理在计算机图形学中的重要性。纹理不仅仅是增加图像的细节和真实感，更是使得三维物体看起来更为自然和生动的重要手段。在实验过程中，我学习了如何使用 stb_image.h 库加载不同格式的图像，并将这些图像作为纹理应用到 OpenGL 场景中。

我也了解到了在实际应用中，纹理映射不仅需要考虑图像本身的质量，还需要细致地处理纹理坐标的映射。通过设置合适的环绕方式

和过滤方式，可以有效提高纹理的显示效果。而且我本次实验只尝试使用 BMP 形式实现纹理加载，在日后的学习中，我将会不断扩展可选择的图像格式，并且尝试使用更多的纹理去渲染场景，从而实现更复杂的效果。其中在实验过程中，通过互联网查询资料以及翻阅文献书籍也使我受益匪浅，通过学习和参考相关文献和教程（如 LearnOpenGL 和 OpenGL4 Tutorials），我不仅掌握了纹理加载的基本步骤，还了解了纹理映射的高级应用。这些资源提供了详尽的示例和解释，帮助我更好地理解和实现纹理映射，让我对计算机图形学的纹理相关知识有了更深入的了解，我在未来会更加勤奋地去学习纹理技术，提升自己的在计算机图形学领域的学术水平，取得一番自己的成就。

参考文献：CSDN OpenGL 入门学习[十一 03]bmp 纹理
CSDN 纹理（讲得比较详细的文章）
博客：https://blog.51cto.com/u_6725876/5134239
朱亚平, 白建军, 边晓东,等. OpenGL 编程实例[M]. 人民邮电出版社, 1999.
F.S.HILL JR. 计算机图形学:用 OpenGL 实现[M]. 清华大学出版社, 2006.
石琼, 沈春林, 谭皓. 基于 OpenGL 的三维建模实现方法[J]. 计算机工程与应用, 2004, 40(18):122-124.