**《计算机图形学实验》综合实验报告**

**题目 基于OpenGL的三维图形纹理绘制**

**学 号 20201050318**

**姓 名 王志元**

**指导教师 钱文华**

**日 期 2022.6.8**

# 摘要

真实感图形生成，使用计算机生成具有连续色调、真实表面纹理和光照效果的景物真实感图形的过程和技术。**[1]**又称真实感图形合成。

计算机真实感图形是一种光栅图形。光栅图形显示器的屏幕由一系列显示单元组成，每一个显示单元称为一个像素。生成一幅真实感图形时，必须逐个像素地计算画面上相应景物表面区域的颜色。这种计算基于一定的光学物理模型，称为光照明模型，分为局部光照模型和整体光照模型。

局部光照模型仅考虑直接来自光源的入射光在景物表面产生的光照效果，整体光照模型还须计入由于场景中景物表面间的相互反射、折射在景物表面形成的光照效果。基于景物几何模型和光照明模型生成一幅真实感图形的过程称为绘制。

本实验实现了借助**图片**给三维茶壶添加**纹理**效果。程序会根据所给图片的不同给茶壶添加不同的纹理效果。同时结合之间的**交互控制**，实现了使用鼠标对图形的拖拽、旋转等功能。

**关键词：纹理、图片、交互控制**

**目录**

[1 实验背景与内容 4](#_Toc106538835)

[2 开发工具与实验目的 4](#_Toc106538836)

[3 关键代码 5](#_Toc106538841)

[4 实验结果 6](#_Toc106538842)

[5 总结 7](#_Toc106538844)

[6 参考文献 7](#_Toc106538845)

[7 附录 7](#_Toc106538845)

1. **实验背景与内容：**

**实验背景：**

真实感图形绘制是计算机图形学的一个重要组成部分，它综合利用数学、物理学、计算机科学和其它科学知识在计算机图形设备上生成象彩色照片那样的具有真实感的图形。一般说来，用计算机在图形设备上生成真实感图形必须完成以下四个步骤：一是用建模，即用一定的数学方法建立所需三维场景的几何描述，场景的几何描述直接影响图形的复杂性和图形绘制的计算耗费；二是将三维几何模型经过一定变换转为二维平面透视投影图；三是确定场景中所有可见面，运用隐藏面消隐算法将视域外或被遮挡住的不可见面消去；四是计算场景中可见面的颜色，即根据基于光学物理的光照模型计算可见面投射到观察者眼中的光亮度大小和颜色分量，并将它转换成适合图形设备的颜色值，从而确定投影画面上每一象素的颜色，最终生成图形。

**实验内容：**

实现三维茶壶的绘制，在绘制完成后，根据工程目录下所给图片teapot1.BMP对茶壶进行纹理绘制。同时渲染过程中添加色彩、光照、阴影等效果。可以采用光线追踪、光照明模型等算法。

**2、开发工具与实验目的：**

开发工具：Codeblock、OpenGL

实验目的：实现对三维茶壶的光照、阴影、纹理等效果。

**3、关键代码：**

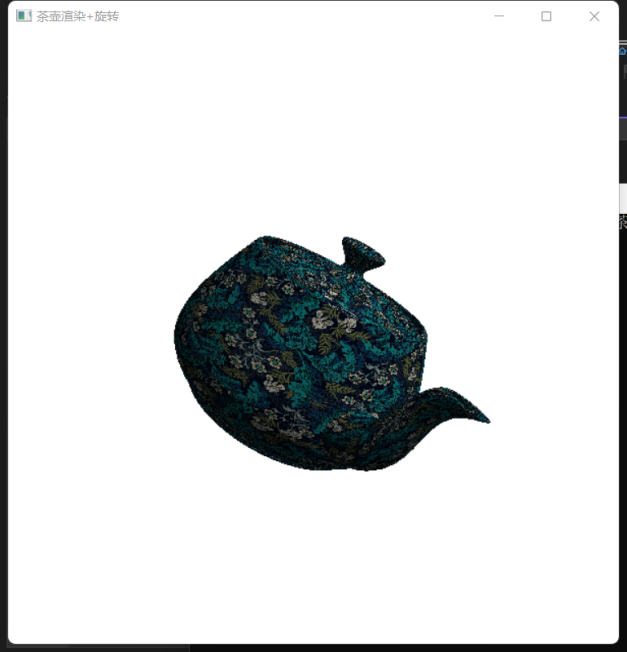
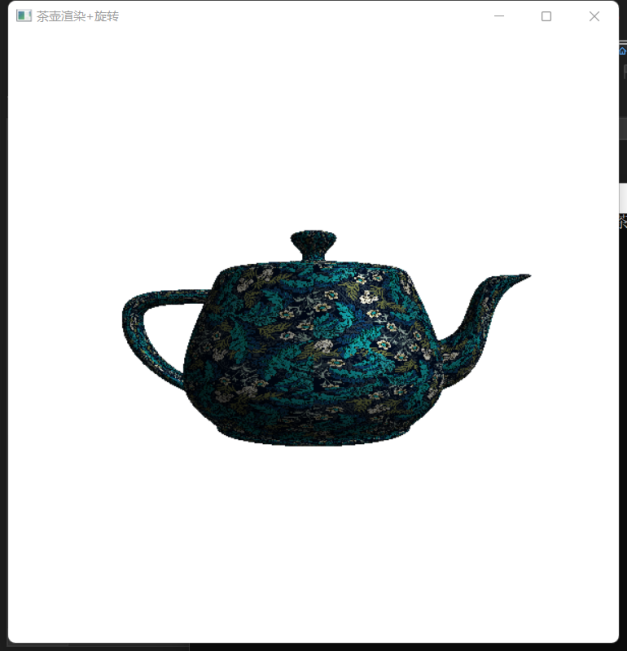
|  |
| --- |
| //读纹理函数 包含文件指针、图像位置索引等参数，可以获取到图片的相关数据并加载到程序中  unsigned char\* LoadBitmapFile(char\* filename, BITMAPINFOHEADER\* bitmapInfoHeader)  //指定当前纹理的放大/缩小过滤方式  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);  //绘制茶壶  GLint GenTeapotList()  //定义光源 给茶壶添加光照效果  GLfloat light\_position[] = { 1.0, 1.0, 1.0, 0.0 };//光的位置  GLfloat color\_light[] = { 1.0,1.0,1.0,1.0 };//光的颜色  GLfloat Light\_Model\_Ambient[] = { 1,1,1,1 };//环境光 |

**4、实验结果：**

添加的纹理图片：

****

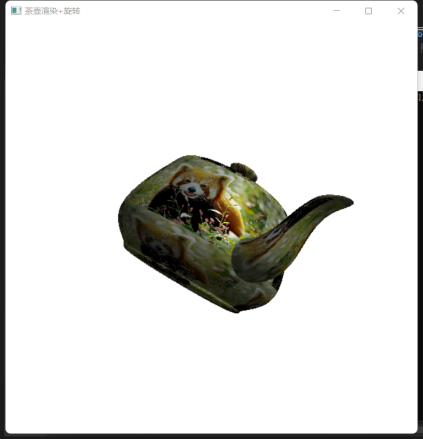
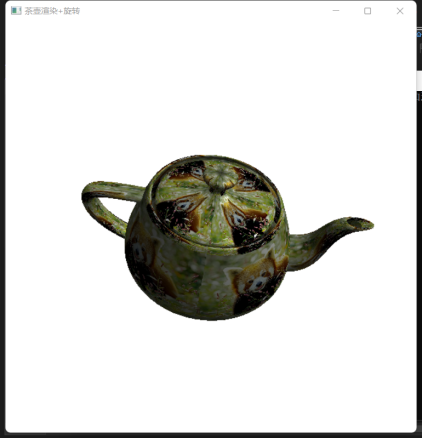
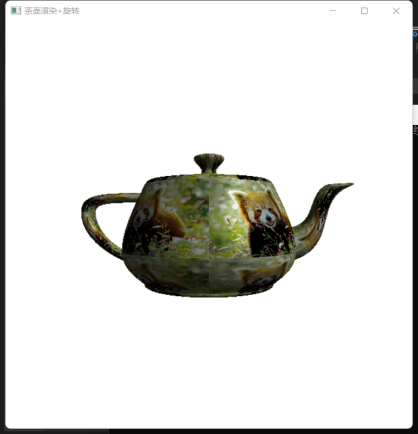
效果：



添加的纹理图片：



效果：



**改进思路：**

光照效果不够明显，缺乏阴影、色彩。在很多动画中，我们可以看到每一束光线的直线部分的位置和方向总是在不断变化，因此每一条光线都要用一个数学方程式来表示，定义光线的空间路径为时间的函数，根据光线在到达屏幕前经过的场景中的目标的色素或颜色来分配给每一束光线一种颜色。这样就可以做到让屏幕上的每一个像素符合每一时刻可以回溯到源头的每条光线。这样绘制出来的图形更加贴近我们现实生活。

**5、总结：**

使用OpenGL实现三维图形的渲染，借助所给图片完成对茶壶的纹理效果。同时调用光照函数模拟光照的生成让图像更为真实。真实感图形的绘制让我对计算机图形学的认知更深刻了一步，目前很多游戏的场景设计离不开真实感图形的生成，虽然目前我技术水平有限，但相信随着不断的学习，自己也可以做出那种精美的图形，让更多人去感受到计算机图形学的魅力。

**6、参考文献：**

1 真实感图形生成，中国大百科全书

2 肖泽群. 基于OpenGL的实时三维人体动画展示方法[J]. 科学与信息化,2021(20):25-26

3 伏玉琛,周洞汝主编. 计算机图形学原理、方法与应用[M] 武汉:华中科技大学出版社,2003.10.第304页

附录：

|  |
| --- |
| #include <stdio.h>  #include <windows.h>  #include <GL/glut.h>  #include <iostream>  #define BITMAP\_ID 0x4D42//读纹理函数  #pragma comment(lib,"glut32.lib")  using namespace std;  GLfloat roate = 0.0;//旋转速率  GLfloat rote = 0.0;//旋转角度  GLfloat anglex = 0.0;//X轴旋转  GLfloat angley = 0.0;//Y轴旋转  GLfloat anglez = 0.0;//Z轴旋转  GLint WinW = 400;  GLint WinH = 400;  GLfloat oldx;//当左键按下时记录鼠标坐标  GLfloat oldy;  GLuint Texture[1];  char TextureName[] = "teapot1.BMP";  // 读纹理函数  unsigned char\* LoadBitmapFile(char\* filename, BITMAPINFOHEADER\* bitmapInfoHeader)  {  FILE\* filePtr;// 文件指针  BITMAPFILEHEADER bitmapFileHeader;// bitmap文件头  unsigned char\* bitmapImage;// bitmap图像数据  int imageIdx = 0;// 图像位置索引  unsigned char tempRGB;// 交换变量  filePtr = fopen(filename, "rb");// 以“二进制+读”模式打开文件filename  if (filePtr == NULL) return NULL;  fread(&bitmapFileHeader, sizeof(BITMAPFILEHEADER), 1, filePtr);// 读入bitmap文件图  if (bitmapFileHeader.bfType != BITMAP\_ID) {// 验证是否为bitmap文件  fprintf(stderr, "Error in LoadBitmapFile: the file is not a bitmap file\n");  return NULL;  }  fread(bitmapInfoHeader, sizeof(BITMAPINFOHEADER), 1, filePtr);// 读入bitmap信息头  fseek(filePtr, bitmapFileHeader.bfOffBits, SEEK\_SET);// 将文件指针移至bitmap数据  bitmapImage = new unsigned char[bitmapInfoHeader->biSizeImage];// 为装载图像数据创建足够的内存  if (!bitmapImage) {// 验证内存是否创建成功  fprintf(stderr, "Error in LoadBitmapFile: memory error\n");  return NULL;  }  fread(bitmapImage, 1, bitmapInfoHeader->biSizeImage, filePtr);// 读入bitmap图像数据  if (bitmapImage == NULL) {// 确认读入成功  fprintf(stderr, "Error in LoadBitmapFile: memory error\n");  return NULL;  }  for (imageIdx = 0; imageIdx < (bitmapInfoHeader->biSizeImage); imageIdx += 3) {//由于bitmap中保存的格式是BGR，于是交换R和B的值，得到RGB格式  tempRGB = bitmapImage[imageIdx];  bitmapImage[imageIdx] = bitmapImage[imageIdx + 2];  bitmapImage[imageIdx + 2] = tempRGB;  }  fclose(filePtr);// 关闭bitmap图像文件  return bitmapImage;  }  //加载纹理  void texload(int i, char\* filename)  {  BITMAPINFOHEADER bitmapInfoHeader;// bitmap信息头  unsigned char\* bitmapData;// 纹理数据  bitmapData = LoadBitmapFile(filename, &bitmapInfoHeader);  glBindTexture(GL\_TEXTURE\_2D, Texture[i]);  // 指定当前纹理的放大/缩小过滤方式  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);  glTexImage2D(GL\_TEXTURE\_2D,0,GL\_RGB,bitmapInfoHeader.biWidth,bitmapInfoHeader.biHeight,0,GL\_RGB,GL\_UNSIGNED\_BYTE,bitmapData);//该纹理有红、绿、蓝数据，无边框  }  //绘制茶壶  GLint GenTeapotList()  {  GLint lid = glGenLists(1);  glNewList(lid, GL\_COMPILE);  GLfloat mat\_diffuse[] = { 0.55, 0.55, 0.55, 1.0 };  GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };  GLfloat mat\_shininess[] = { 90.0 };  glClearColor(1.0, 1.0, 1.0, 1.0);  glShadeModel(GL\_SMOOTH);  glColorMaterial(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE);  glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);  glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);  glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);  glEndList();  return lid;  }  //定义纹理  void init(void)  {  cout << "点击鼠标左键并拖动实现茶壶拖动旋转，点击鼠标右键实现茶壶绕Y轴加速旋转。" << endl;  glEnable(GL\_DEPTH\_TEST);//打开深度测试  //定义光源  GLfloat light\_position[] = { 1.0, 1.0, 1.0, 0.0 };//光的位置  GLfloat color\_light[] = { 1.0,1.0,1.0,1.0 };//光的颜色  GLfloat Light\_Model\_Ambient[] = { 1,1,1,1 };//环境光  glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, color\_light);  glLightfv(GL\_LIGHT0, GL\_SPECULAR, color\_light);  glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, Light\_Model\_Ambient);  glEnable(GL\_LIGHTING);  glEnable(GL\_LIGHT0);  //定义纹理  glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);  glGenTextures(1, Texture);  texload(0, TextureName);  glBindTexture(GL\_TEXTURE\_2D, Texture[0]);  //设置像素存储模式控制所读取的图像数据的行对齐方式  glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);  glDisable(GL\_TEXTURE\_2D);  }  //显示  void display(void)  {  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  glEnable(GL\_TEXTURE\_2D);  glBindTexture(GL\_TEXTURE\_2D, Texture[0]); //选择纹理图案  glCallList(GenTeapotList());  glClear(GL\_COLOR\_BUFFER\_BIT);  glLoadIdentity(); //加载单位矩阵  gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);  glRotatef(rote, 0.0f, 1.0f, 0.0f);  glRotatef(anglex, 1.0, 0.0, 0.0);  glRotatef(angley, 0.0, 1.0, 0.0);  glRotatef(anglez, 0.0, 0.0, 1.0);  glutSolidTeapot(1.2);  rote += roate;  glutSwapBuffers();  glFlush();  }  //重塑  void reshape(GLsizei w, GLsizei h)  {  glViewport(0, 0, (GLsizei)w, (GLsizei)h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluPerspective(60.0, (GLfloat)w / (GLfloat)h, 1.0, 20.0);  glMatrixMode(GL\_MODELVIEW);  glLoadIdentity();  gluLookAt(0.0, 0.0, 4.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);  }  //鼠标响应  void mouse(int button, int state, int x, int y)  {  if (button == GLUT\_LEFT\_BUTTON){  if (state == GLUT\_DOWN){  roate = 0;  rote = 0;  oldx = x;//当左键按下时记录鼠标坐标  oldy = y;  }  }  if (button == GLUT\_RIGHT\_BUTTON){  if (state == GLUT\_DOWN){  roate += 1.0f;  }  }  }  //重绘图像  void motion(int x, int y)  {  GLint deltax = oldx - x;  GLint deltay = oldy - y;  anglex += 360 \* (GLfloat)deltax / (GLfloat)WinW;  angley += 360 \* (GLfloat)deltay / (GLfloat)WinH;  anglez += 360 \* (GLfloat)deltay / (GLfloat)WinH;//根据屏幕上鼠标滑动的距离来设置旋转的角度  oldx = x;  oldy = y;//更新鼠标坐标，使滑动旋转可控  glutPostRedisplay();  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);//设置显示模式  glutInitWindowPosition(0, 0);//设置窗口在屏幕上的位置  glutInitWindowSize(600, 600);//设置窗口的大小  glutCreateWindow("茶壶渲染+旋转");  init();  glutDisplayFunc(display);//调用绘图函数  glutReshapeFunc(reshape);  glutMouseFunc(mouse);  glutMotionFunc(motion);  glutIdleFunc(display);  glutMainLoop(); //进入循环  return 0;  } |