

**计算机图形学实验**

**--光照、阴影、纹理绘制犹他茶壶**

**学 院：**­­­计算机科学与技术学院

**专 业：**软件工程

**班 级：**云计算2班

**学 号：**1500170434

**学生姓名：**杨美美

学院：计算机科学与技术 专业：软件工程 班级： 云计算2班

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | | 杨美美 | 学号 | 1500170434 | 实验组 |  |
| 实验时间 | | 2017-12-10 | 指导教师 |  | 成绩 |  |
| 实验项目名称 | | OpenGL光照、阴影、纹理绘制犹他茶壶 | | | | |
| 实验要求 | OpenGL光照、阴影、纹理绘制犹他茶壶 | | | | | |
| 实验目的 | 熟悉OpenGL，应用不同光照、阴影、纹理绘制犹他茶壶 | | | | | |
| 实验内容 | 光照  void init(void)  {  //材质反光性设置  GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 }; //镜面反射参数  GLfloat mat\_shininess[] = { 50.0 }; //高光指数  GLfloat light\_position[] = { 1.0, 1.0, 1.0, 0.0 };  GLfloat white\_light[] = { 2.0, 2.0, 2.0, 1.0 }; //灯位置(2,2,2), 最后1-开关  GLfloat Light\_Model\_Ambient[] = { 0.2, 0.2, 0.2, 1.0 }; //环境光参数  glClearColor(0.0, 0.0, 0.0, 0.0); //背景色  glShadeModel(GL\_SMOOTH); //多变性填充模式  //材质属性  glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);  glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);  //灯光设置  glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, white\_light); //散射光属性  glLightfv(GL\_LIGHT0, GL\_SPECULAR, white\_light); //镜面反射光  glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, Light\_Model\_Ambient); //环境光参数  glEnable(GL\_LIGHTING); //开关:使用光  glEnable(GL\_LIGHT0); //打开0#灯  glEnable(GL\_DEPTH\_TEST); //打开深度测试  }  阴影  void draw\_world() // 绘制一个地板  {  glm::vec4 v1(-3, 0,-3, 1), v2(-3, 0, 3, 1), v3( 3, 0, 3, 1), v4( 3, 0,-3, 1);//四个顶点  glm::mat4 m = glm::translate(glm::vec3(0.5f,0.5f,0.5f))  \* glm::scale(glm::vec3(0.5f,0.5f,0.5f)); // 需要将裁剪坐标的[-1,+1]缩放到[0,1]  glm::vec4 t;  glBegin(GL\_POLYGON);  　　glNormal3f(0, 1, 0);  　　t = m\*shadow\_mat\_p\*shadow\_mat\_v\*v1; // 按和生成纹理相同的变换计算纹理坐标  　　glTexCoord4fv(&t[0]); glVertex3fv(&v1[0]);  　　t = m\*shadow\_mat\_p\*shadow\_mat\_v\*v2;  　　glTexCoord4fv(&t[0]); glVertex3fv(&v2[0]);  　　t = m\*shadow\_mat\_p\*shadow\_mat\_v\*v3;  　　glTexCoord4fv(&t[0]); glVertex3fv(&v3[0]);  　　t = m\*shadow\_mat\_p\*shadow\_mat\_v\*v4;  　　glTexCoord4fv(&t[0]); glVertex3fv(&v4[0]);  glEnd();  }  glm::mat4 shadow\_mat\_p; // 光源视角的投影矩阵  glm::mat4 shadow\_mat\_v; // 光源视角的视图矩阵  void tex\_init() // 纹理初始化  {  // 纹理如何影响颜色，和光照计算结果相乘  glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);  // 分配纹理对象，并绑定为当前纹理  glGenTextures(1, &tex\_shadow);  glBindTexture(GL\_TEXTURE\_2D, tex\_shadow);  // 纹理坐标超出[0,1]时如何处理  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP\_TO\_EDGE);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP\_TO\_EDGE);  // 非整数纹理坐标处理方式，线性插值  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);  // 深度纹理，深度值对应亮度  glTexParameteri(GL\_TEXTURE\_2D, GL\_DEPTH\_TEXTURE\_MODE, GL\_LUMINANCE);  }  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_COMPARE\_MODE, GL\_COMPARE\_R\_TO\_TEXTURE);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_COMPARE\_FUNC, GL\_LEQUAL);  纹理  void Draw\_Leg();    // 纹理标示符数组，保存两个纹理的标示符  // 描述: 通过指针，返回filename 指定的bitmap文件中数据。  // 同时也返回bitmap信息头.（不支持-bit位图）      //读纹理图片  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    // 以“二进制+读”模式打开文件filename  filePtr = fopen(filename, "rb");  if (filePtr == NULL) {  printf("file not open\n");  return NULL;  }  // 读入bitmap文件图  fread(&bitmapFileHeader, sizeof(BITMAPFILEHEADER), 1, filePtr);  // 验证是否为bitmap文件  if (bitmapFileHeader.bfType != BITMAP\_ID) {  fprintf(stderr, "Error in LoadBitmapFile: the file is not a bitmap file\n");  return NULL;  }  // 读入bitmap信息头  fread(bitmapInfoHeader, sizeof(BITMAPINFOHEADER), 1, filePtr);  // 将文件指针移至bitmap数据  fseek(filePtr, bitmapFileHeader.bfOffBits, SEEK\_SET);  // 为装载图像数据创建足够的内存  bitmapImage = new unsigned char[bitmapInfoHeader->biSizeImage];  // 验证内存是否创建成功  if (!bitmapImage) {  fprintf(stderr, "Error in LoadBitmapFile: memory error\n");  return NULL;  }    // 读入bitmap图像数据  fread(bitmapImage, 1, bitmapInfoHeader->biSizeImage, filePtr);  // 确认读入成功  if (bitmapImage == NULL) {  fprintf(stderr, "Error in LoadBitmapFile: memory error\n");  return NULL;  }  //由于bitmap中保存的格式是BGR，下面交换R和B的值，得到RGB格式  for (imageIdx = 0;imageIdx < bitmapInfoHeader->biSizeImage; imageIdx += 3) {  tempRGB = bitmapImage[imageIdx];  bitmapImage[imageIdx] = bitmapImage[imageIdx + 2];  bitmapImage[imageIdx + 2] = tempRGB;  }  // 关闭bitmap图像文件  fclose(filePtr);  return bitmapImage;  }  void generateTex()  {  //生成红黑相间的图像  for (int i = 0; i < Height; i++) {  for (int j = 0; j < Width; j++) {  int x = ((i & 4 ) ^(j & 4 )) \* 255;  image[i][j][0] = (GLubyte)x;  image[i][j][1] = 0;  image[i][j][2] = 0;  }  }  }  void Draw\_Triangle() // 纹理操作函数  {  glEnable(GL\_TEXTURE\_2D);  glBindTexture(GL\_TEXTURE\_2D, texture[status]); //选择纹理texture[status]  glPushMatrix();  glTranslatef(0, 0, 4+1);  glRotatef(90, 1, 0, 0);  glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);//设置纹理受光照影响  glutSolidTeapot(1);  glPopMatrix();  glDisable(GL\_TEXTURE\_2D); //关闭纹理texture[status]    glEnable(GL\_TEXTURE\_2D);  glBindTexture(GL\_TEXTURE\_2D, texture[status2]); //选择纹理texture[status2]    glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);//设置纹理不受光照影响  glPushMatrix();  glTranslatef(0, 0, 3.5);  glScalef(5, 4, 1);  drawCube();  glPopMatrix();    glPushMatrix();  glTranslatef(1.5, 1, 1.5);  Draw\_Leg();  glPopMatrix();    glPushMatrix();  glTranslatef(-1.5, 1, 1.5);  Draw\_Leg();  glPopMatrix();    glPushMatrix();  glTranslatef(1.5, -1, 1.5);  Draw\_Leg();  glPopMatrix();    glPushMatrix();  glTranslatef(-1.5, -1, 1.5);  Draw\_Leg();  glPopMatrix();  glDisable(GL\_TEXTURE\_2D); //关闭纹理texture[status2]  } | | | | | |
| 实验结果 | 光照截图： 阴影截图：  光照犹他茶壶 阴影2  纹理截图：  纹理5 | | | | | |
| 实验总结 | 签名： 杨美美 2017年12月10日 | | | | | |