## 0809702001（虚拟现实）实验四 纹理映射

**主题：** 利用VC集成开发环境，配置OpenGL库函数， 建立基本的OpenGL应用程序，完成实验四的任务，实验报告请于上机一周完成上传至学习通。

**准备**：

* 打开您的VSStudio 并且设置好您的工作目录；
* 下载OpenGL安装包所需文件（<http://d.download.csdn.net/down/2560229/ssagnn23>），主要包括GL.H GLAUX.H GLU.H glut.h   
  GLAUX.LIB GLU32.LIB glut32.lib glut.lib OPENGL32.LIB   
  glaux.dll glu32.dll glut32.dll glut.dll opengl32.dll
* 复制并配置OpenGL库函数到制定的目录，检查复制后是否文件已经存在于指定目录下

**任务1：**建立纹理映射应用程序

* **在**VSStudio 菜单栏上选择**File->New->Project，选择Win32应用程序，输入项目名称TextureProj**
* **点击“OK”后，在后续的对话框中选中Empty project，然后点击Finish。**
* **右击项目名TextureProj，选择属性(property)，再选择链接器(Linker)中的输入选项(Input)，附加依赖项(Additional Dependencies):opengl32.lib glu32.lib glaux.lib**
* **右击项目名TextureProj下的源文件，选择添加(ADD)，再选择新建项 (New Item)，在弹出来的对话框中选择C++ File(.cpp)，输入文件名Texbind.cpp,然后点击添加(ADD)，检查源文件是否建立成功。**

**任务2：在Texbind.cpp中输入代码，人工生成一类棋盘格图像checkImage，读取一个外部图像otherImage, 然后作为纹理贴图贴到两个正方形两个表面。**

#define checkImageWidth 64

#define checkImageHeight 64

static GLubyte checkImage[checkImageHeight][checkImageWidth][4];

static GLuint texName[2];

void makeCheckImages(void)

{

int i, j, c;

for (i = 0; i < checkImageHeight; i++) {

for (j = 0; j < checkImageWidth; j++) {

c = ((((i&0x8)==0)^((j&0x8))==0))\*255;

checkImage[i][j][0] = (GLubyte) c;

checkImage[i][j][1] = (GLubyte) c;

checkImage[i][j][2] = (GLubyte) c;

checkImage[i][j][3] = (GLubyte) 255;

c = ((((i&0x10)==0)^((j&0x10))==0))\*255;

otherImage[i][j][0] = (GLubyte) c;

otherImage[i][j][1] = (GLubyte) 0;

otherImage[i][j][2] = (GLubyte) 0;

otherImage[i][j][3] = (GLubyte) 255;

}

}

}

void init(void)

{

glClearColor (0.0, 0.0, 0.0, 0.0);

glShadeModel(GL\_FLAT);

glEnable(GL\_DEPTH\_TEST);

makeCheckImages();

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

glGenTextures(2, texName);

glBindTexture(GL\_TEXTURE\_2D, texName[0]);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP);

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\_RGBA, checkImageWidth,

checkImageHeight, 0, GL\_RGBA, GL\_UNSIGNED\_BYTE,

checkImage);

glBindTexture(GL\_TEXTURE\_2D, texName[1]);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGBA, checkImageWidth,

checkImageHeight, 0, GL\_RGBA, GL\_UNSIGNED\_BYTE,

otherImage);

glEnable(GL\_TEXTURE\_2D);

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glBindTexture(GL\_TEXTURE\_2D, texName[0]);

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0); glVertex3f(-2.0, -1.0, 0.0);

glTexCoord2f(0.0, 1.0); glVertex3f(-2.0, 1.0, 0.0);

glTexCoord2f(1.0, 1.0); glVertex3f(0.0, 1.0, 0.0);

glTexCoord2f(1.0, 0.0); glVertex3f(0.0, -1.0, 0.0);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, texName[1]);

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0); glVertex3f(1.0, -1.0, 0.0);

glTexCoord2f(0.0, 1.0); glVertex3f(1.0, 1.0, 0.0);

glTexCoord2f(1.0, 1.0); glVertex3f(2.41421, 1.0, -1.41421);

glTexCoord2f(1.0, 0.0); glVertex3f(2.41421, -1.0, -1.41421);

glEnd();

glFlush();

}

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei) w, (GLsizei) h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60.0, (GLfloat) w/(GLfloat) h, 1.0, 30.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(0.0, 0.0, -3.6);

}

void keyboard (unsigned char key, int x, int y)

{

switch (key) {

case 27:

exit(0);

break;

default:

break;

}

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(250, 250);

glutInitWindowPosition(100, 100);

glutCreateWindow(argv[0]);

init();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutMainLoop();

return 0;

}

* 归纳纹理映射的四个步骤，以及总结四个步骤各自相关的对应函数？用框表的形式表达出来。
* 为何glBindTexture函数对每一个纹理对象都用了两次，每次的含义是什么？用两次的作用是什么？
* 尝试用AUX\_RGBImageRec auxDIBImageLoad（LPCTSTR filename）这个函数读入外部文件otherImage来指定纹理对象映射到第二个正方形的表面。
* **尝试解释**void makeCheckImages(void)**函数中每个语句的含义：**
* 用glColor3f（）函数指定蓝色绘制两个正方形， 再用函数void glTexEnv（）设置纹理颜色如何与片元颜色相互作用？把不同的参数展示结果在报告中展现出来。
* **任务2：在Texbind.cpp中输入代码，人工生成一苍蝇图像FlyImage，映射到一个正方形上，并设置**纹理的重复与截取方式。

#include <GL/glew.h>

#include <GL/glut.h>

#include <stdlib.h>

#include <stdio.h>

#define ImageWidth 128

#define ImageHeight 128

static GLubyte FlyImage[ImageHeight][ ImageWidth][4];

static GLfloat borderColor[4] = {1.0,0,0};

static GLuint texName;

//生成苍蝇图像

void makeFlyImage(void)

{

int i, j, c;

/\*\*\*\*\*\*\*\*\*/

}

void init(void)

{

glClearColor (0.0, 0.0, 0.0, 0.0);

glShadeModel(GL\_FLAT);

glEnable(GL\_DEPTH\_TEST);

makeCheckImage();

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

glGenTextures(1, &texName);

glBindTexture(GL\_TEXTURE\_2D, texName);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);

//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\_CLAMP);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T,GL\_CLAMP);

//glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S,GL\_CLAMP\_TO\_EDGE);

//glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T,GL\_CLAMP\_TO\_EDGE);

//glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S,GL\_CLAMP\_TO\_BORDER);

//glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T,GL\_CLAMP\_TO\_BORDER);

//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\_WRAP\_S, GL\_MIRRORED\_REPEAT);

//glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_MIRRORED\_REPEAT);

glTexParameterfv(GL\_TEXTURE\_2D, GL\_TEXTURE\_BORDER\_COLOR,borderColor);

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGBA, ImageWidth, ImageHeight,

0, GL\_RGBA, GL\_UNSIGNED\_BYTE, FlyImage);

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glEnable(GL\_TEXTURE\_2D);

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

glBindTexture(GL\_TEXTURE\_2D, texName);

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0); glVertex2f( -1.0, -1.0);

glTexCoord2f(0.0, 2.0); glVertex2f( -1.0, 1.0);

glTexCoord2f(2.0, 2.0); glVertex2f( 1.0, 1.0);

glTexCoord2f(2.0, 0.0); glVertex2f( 1.0, -1.0);

glEnd();

glFlush();

glDisable(GL\_TEXTURE\_2D);

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(256, 256);

glutInitWindowPosition(0, 0);

glutCreateWindow(argv[0]);

init();

glutDisplayFunc(display);

glutIdleFunc(display);

glutMainLoop();

return 0;

}

* 用范例2.3的苍蝇fly图像来生成flyimage;
* 设置不同的截取（CLAMP）和滤波方式，展示不同的效果？用图或者表的形式展示结果，并总结设置参数对结果的影响？
* 设置不同的重复（REPEAT）和滤波方式，展示不同的效果？用图和表的形式展示结果，并总结设置参数对结果的影响？