



逻辑教育
Logic education

Hello CC

OpenGL 主题I3I

视觉班—OpenGL 渲染技巧 课堂案例解析

课程研发:CC老师
课程授课:CC老师

转载需注明出处,不得用于商业用途.已申请版权保护



逻辑教育
Logic education

课堂Demo [01] 解析

感受OpenGL 下的多种渲染技巧

案例1: 001 — OpenGL 深度测试



001--深度测试

课程研发:CC老师
课程授课:CC老师



课堂Demo [01]

``#include<GLTools.h>`` GLTool.h头文件包含了大部分GLTool中类似C语言的独立函数

``#include<GLMatrixStack.h>`` 矩阵的工具类.可以利于GLMatrixStack 加载单元矩阵/矩阵/矩阵相乘/压栈/出栈/缩放/平移/旋转

``#include<GLFrustum.h>`` 矩阵工具类,用来快速设置正/透视投影矩阵.完成坐标从3D->2D映射过程.

工具类:

```
#include "GLTools.h"
#include "GLMatrixStack.h"
#include "GLFrame.h"
#include "GLFrustum.h"
#include "GLBatch.h"
#include "GLGeometryTransform.h"

#include <math.h>
#ifdef __APPLE__
#include <glut/glut.h>
#else
#define FREEGLUT_STATIC
#include <GL/glut.h>
#endif
```

``#include<GLFrame.h>`` 矩阵工具类,表示位置.通过设置vOrigin, vForward ,vUp

``#include<GLGeometryTransform.h>`` 变换管道类,用来快速在代码中传输视图矩阵/投影矩阵/视图投影变换矩阵等.

``#include<GLBatch.h>`` 三角形批次类,帮助类,利用它可以传输顶点/光照/纹理/颜色数据到存储着色器中.

`#include <math.h>` 数学库

在Mac 系统下, ``#include<glut/glut.h>``
在Windows 和 Linux上, 我们使用freeglut的静态库版本并且需要添加一个宏

课程研发:CC老师
课程授课:CC老师

torusBatch
帮助类/容器类

modelViewMatrix
模型视图矩阵

transformPipeline
变换管道. 存储模型视图
/投影/模型视图投影矩
阵.

GLShaderManager
存储着色器管理工具类.

```
////设置角色帧, 作为相机
GLFrame
viewFrame;
//使用GLFrustum类来设置透视投影
GLFrustum
viewFrustum;
GLTriangleBatch
torusBatch;
GLMatrixStack
modelViewMatix;
GLMatrixStack
projectionMatrix;
GLGeometryTransform
transformPipeline;
GLShaderManager
shaderManager;
```

viewFrame
设置观察者视图坐标

viewFrustum
设置图元绘制时的投影
方式.

projectionMatrix
投影矩阵

```
//标记: 背面剔除、深度测试
int iCull = 0;
int iDepth = 0;
```

iCull/iDepth
是否开启背面剔除/深度
测试

课程研发:CC老师
课程授课:CC老师



课堂Demo [01]

重要的函数

```
void changeSize(int w ,int h)
void RenderScene(void)
void setupRC()
int main(int argc ,char *argv[])
void ProcessMenu(int value)
void SpecialKeys(int key, int x, int y)
```

changeSize 函数:自定义函数.通过
glutReshaperFunc(函数名)注册为重塑函
数.当屏幕大小发生变化/或者第一次创建窗
口时,会调用该函数调整窗口大小/视口大小.

RenderScene 函数:自定义函数.通过
glutDisplayFunc(函数名)注册为显示渲
染函数.当屏幕发生变化/或者开发者主动渲
染会调用此函数,用来实现数据->渲染过程

setupRC 函数: 自定义函
数,设置你需要渲染的图形的
相关顶点数据/颜色数据等数
据装备工作

ProcessMenu函数:右击菜单栏.

SpecialKeys函数:特殊键位处理
(上、下、左、右移动)

main 函数: 程序入
口.OpenGL 是面向
过程编程.所以你会发
现利用OpenGL处理图
形/图像都是链式形
式.以及基于OpenGL
封装的图像处理框架
也是链式编程

课程研发:CC老师
课程授课:CC老师



课堂Demo [01]

重要的函数 main 函数

```
int main(int argc ,char *argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH | GLUT_STENCIL);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Triangle");
    glutReshapeFunc(changeSize);
    glutDisplayFunc(RenderScene);
    glutSpecialFunc(SpecialKeys);
    glutCreateMenu(ProcessMenu);
    glutAddMenuEntry("Toggle depth test",1);
    glutAddMenuEntry("Toggle cull backface",2);
    glutAddMenuEntry("Set Fill Mode", 3);
    glutAddMenuEntry("Set Line Mode", 4);
    glutAddMenuEntry("Set Point Mode", 5);
    glutAttachMenu(GLUT_RIGHT_BUTTON);

    GLenum status = glewInit();
    if (GLEW_OK != status) {

        printf("GLEW Error:%s\n",glewGetErrorString(status));
        return 1;

    }
    setupRC();
    glutMainLoop();
}
```




逻辑教育
Logic education

课堂Demo [01]

重要的函数 ProcessMenu 函数

KeyPressFunc 触发条件:

1. 用户右击选择

处理业务:

1. 判断用户选择的功能
2. 打开/关闭背面剔除
3. 打开/关闭深度测试
4. 使用点/线/面的方式填充图形
5. 重新渲染(手动触发)

```
//右键菜单栏选项
void ProcessMenu(int value)
{
    switch(value)
    {
        case 1:
            iDepth = !iDepth;
            break;

        case 2:
            iCull = !iCull;
            break;

        case 3:
            glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
            break;

        case 4:
            glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
            break;

        case 5:
            glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);
            break;
    }

    glutPostRedisplay();
}
```

课程研发:CC老师

课程授课:CC老师



课堂Demo [01]

重要的函数 SpecialKeys 函数

```
//键位设置, 通过不同的键位对其进行设置
//控制Camera的移动, 从而改变视口
void SpecialKeys(int key, int x, int y)
{
    if(key == GLUT_KEY_UP)
        viewFrame.RotateWorld(m3dDegToRad(-5.0), 1.0f, 0.0f, 0.0f);

    if(key == GLUT_KEY_DOWN)
        viewFrame.RotateWorld(m3dDegToRad(5.0), 1.0f, 0.0f, 0.0f);

    if(key == GLUT_KEY_LEFT)
        viewFrame.RotateWorld(m3dDegToRad(-5.0), 0.0f, 1.0f, 0.0f);

    if(key == GLUT_KEY_RIGHT)
        viewFrame.RotateWorld(m3dDegToRad(5.0), 0.0f, 1.0f, 0.0f);

    //重新刷新window
    glutPostRedisplay();
}
```

SpecialKeys 触发条件:

1. 用户使用“上/下/左/右”键位

处理业务:

1. 计算出围绕X/Y坐标轴上下左右旋转的视觉坐标系.



课堂Demo [01]

重要的函数 `changeSize` 函数

```
void changeSize(int w ,int h)
{
    glViewport(0, 0, w, h);
    //创建投影矩阵, 并将它载入投影矩阵堆栈中
    viewFrustum.SetPerspective(35.0f, float(w) / float(h), 1.0f, 500.0f);
    projectionMatrix.LoadMatrix(viewFrustum.GetProjectionMatrix());

    // 初始化渲染管线
    transformPipeline.SetMatrixStacks(modelViewMatix, projectionMatrix);
}
```

`changeSize` 触发条件:

1. 新建窗口
2. 窗口尺寸发生调整

处理业务:

1. 设置OpenGL 视口
2. 设置OpenGL 投影方式等.

课程研发:CC老师
课程授课:CC老师



setupRC 触发条件:

1. 手动main函数触发

处理业务:

1. 设置窗口背景颜色

2. 初始化存储着色器shaderManager

3. 设置图形顶点数据

4. 利用GLBatch 三角形批次类, 将数据传递到着色器

```
void SetupRC()  
{  
    // 设置背景颜色  
    glClearColor(0.3f, 0.3f, 0.3f, 1.0f );  
  
    //初始化着色器管理器  
    shaderManager.InitializeStockShaders();  
  
    //将相机向后移动7个单元: 肉眼到物体之间的距离  
    viewFrame.MoveForward(7.0);  
  
    //创建一个甜甜圈  
    //void gltMakeTorus(GLTriangleBatch& torusBatch, GLfloat majorRadius, GLfloat  
minorRadius, GLint numMajor, GLint numMinor);  
    //参数1: GLTriangleBatch 容器帮助类  
    //参数2: 外边缘半径  
    //参数3: 内边缘半径  
    //参数4、5: 主半径和从半径的细分单元数量  
  
    gltMakeTorus(torusBatch, 1.0f, 0.3f, 52, 26);  
    //点的大小  
    glPointSize(4.0f);  
}
```



```
void RenderScene(void)
{
    //清除窗口和深度缓冲区
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    //根据设置iCull标记来判断是否开启背面剔除
    if(iCull)
    {
        glEnable(GL_CULL_FACE);
        glFrontFace(GL_CCW);
        glCullFace(GL_BACK);
    }
    else
        glDisable(GL_CULL_FACE);
    //根据设置iDepth标记来判断是否开启深度测试
    if(iDepth)
        glEnable(GL_DEPTH_TEST);
    else
        glDisable(GL_DEPTH_TEST);
    //把摄像机矩阵压入模型矩阵中
    modelViewMatix.PushMatrix(viewFrame);
    GLfloat vRed[] = { 1.0f, 0.0f, 0.0f, 1.0f };
    shaderManager.UseStockShader(GLT_SHADER_DEFAULT_LIGHT,
transformPipeline.GetModelViewMatrix(), transformPipeline.GetProjectionMatrix(), vRed);
    torusBatch.Draw();
    modelViewMatix.PopMatrix();
    glutSwapBuffers();
}
```

RenderScene 触发条件:

1. 系统自动触发
2. 开发者手动调用函数触发.

处理业务:

1. 清理缓存区(颜色,深度,模板缓存区等)
2. 使用存储着色器
3. 绘制图形.



逻辑教育
Logic education

课堂Demo [02] 解析

熟悉OpenGL裁剪API

案例2: 002—OpenGL 裁剪



002--OpenGL 裁
剪

课程研发:CC老师
课程授课:CC老师



课堂Demo [02]

重要的函数 main 函数

```
int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(800,600);
    glutCreateWindow("OpenGL Scissor");
    glutReshapeFunc(ChangeSize);
    glutDisplayFunc(RenderScene);
    glutMainLoop();

    return 0;
}
```



课堂Demo [02]

重要的函数 `changeSize` 函数

```
void ChangeSize(int w, int h)
{
    //保证高度不能为0
    if(h == 0)
        h = 1;

    // 将视口设置为窗口尺寸
    glViewport(0, 0, w, h);
}
```

`changeSize` 触发条件:

1. 新建窗口
2. 窗口尺寸发生调整

处理业务:

1. 设置OpenGL 视口
2. 设置OpenGL 投影方式等.

课程研发:CC老师
课程授课:CC老师



重要的函数 RenderScene 函数

```
//召唤场景
void RenderScene(void)
{
    //设置清屏颜色为蓝色
    glClearColor(0.0f, 0.0f, 1.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT);

    //1.现在剪成小红色分区
    //(1)设置裁剪区颜色为红色
    glClearColor(1.0f, 0.0f, 0.0f, 0.0f);
    //(2)设置裁剪尺寸
    glScissor(100, 100, 600, 400);
    //(3)开启裁剪测试
    glEnable(GL_SCISSOR_TEST);
    //(4)开启清屏, 执行裁剪
    glClear(GL_COLOR_BUFFER_BIT);

    // 2.裁剪一个绿色的小矩形
    //(1).设置清屏颜色为绿色
    glClearColor(0.0f, 1.0f, 0.0f, 0.0f);
    //(2).设置裁剪尺寸
    glScissor(200, 200, 400, 200);
    //(3).开始清屏执行裁剪
    glClear(GL_COLOR_BUFFER_BIT);

    //关闭裁剪测试
    glDisable(GL_SCISSOR_TEST);

    //强制执行缓存区
    glutSwapBuffers();
}
```

RenderScene 触发条件:

1. 系统自动触发
2. 开发者手动调用函数触发.

处理业务:

1. 清理缓存区(颜色, 深度, 模板缓存区等)
2. 使用存储着色器
3. 绘制图形.

课程研发:CC老师

课程授课:CC老师



逻辑教育
Logic education

课堂Demo [03] 解析

OpenGL 混合—颜色混合

案例3: 003—OpenGL 混合



案例基于第一节课正方形渲染.

课程研发:CC老师
课程授课:CC老师



重要的函数 **RenderScene** 函数

完整源码请查看 课程**Demo**

```
void RenderScene(void){  
  
    此处省略代码 .. ..  
    //组合核心代码  
    //1.开启混合  
    glEnable(GL_BLEND);  
    //2.开启组合函数 计算混合颜色因子  
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);  
    //3.使用着色器管理器  
    // *使用 单位着色器  
    //参数1: 简单的使用默认笛卡尔坐标系 (-1, 1), 所有片段都应用一种颜色。GLT_SHADER_IDENTITY  
    //参数2: 着色器颜色  
    shaderManager.UseStockShader(GLT_SHADER_IDENTITY, vRed);  
    //4.容器类开始绘制  
    squareBatch.Draw();  
    //5.关闭混合功能  
    glDisable(GL_BLEND);  
    //同步绘制命令  
    glutSwapBuffers();  
  
}
```

课程研发:CC老师
课程授课:CC老师



逻辑教育
Logic education

Hello Coder

学习,是一件开心的事

知识,是一个值得分享的东西

献给,我可爱的开发者们.

课程研发:CC老师
课程授课:CC老师

转载需注明出处,不得用于商业用途.已申请版权保护