



逻辑教育
Logic education

Hello 视觉全训班

OpenGL

OpenGL ES

GPUImage

Metal



视觉全训班. 渲染基础 之 案例解析

@ CC老师

全力以赴·非同凡“想”

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

使用固定存储着色器渲染多种图元

案例1: 001 — OpenGL 图元绘制



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

GLShaderManager
存储着色器管理工具类。

modelViewMatrix
模型视图矩阵

projectionMatrix
投影矩阵

viewFrustum
设置图元绘制时的投影方式。

**pointBatch/
lineBatch/
lineStripBatch/
lineLoopBatch/
triangleBatch/
triangleStripBatch/
triangleFanBatch**
帮助类/容器类, 绘制7种不同的图元。

GLShaderManager
GLMatrixStack
GLMatrixStack
GLFrame
GLFrame

//投影设置

GLFrustum

//容器类 (7种不同的图元对应7种容器对象)

GLBatch
GLBatch
GLBatch
GLBatch
GLBatch
GLBatch
GLBatch

```
shaderManager;  
modelViewMatrix;  
projectionMatrix;  
cameraFrame;  
objectFrame;
```

```
viewFrustum;  
pointBatch;  
lineBatch;  
lineStripBatch;  
lineLoopBatch;  
triangleBatch;  
triangleStripBatch;  
triangleFanBatch;
```

cameraFrame
设置观察者视图坐标

objectFrame
设置图形环绕时, 视图坐标



课堂Demo [01]

公共全局变量:

transformPipeline
变换管道, 存储投影/视图/投影视图变换矩阵

```
//几何变换的管道  
GLGeometryTransform transformPipeline;
```

```
GLfloat vGreen[] = { 0.0f, 1.0f, 0.0f, 1.0f };  
GLfloat vBlack[] = { 0.0f, 0.0f, 0.0f, 1.0f };
```

```
// 跟踪效果步骤  
int nStep = 0;
```

vGreen/vBlack
定义2种颜色, 绿色/黑色

nStep
记录用户按空格的次数,
用来显示渲染的不同图形.

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 KeyPressFunc(unsigned char key, int x, int y)
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

KeyPressFunc函数:根据空格次
数.切换不同的“窗口名称”

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);
    glutKeyboardFunc(KeyPressFunc);
    glutSpecialFunc(SpecialKeys);
    GLenum status = glewInit();
    if (GLEW_OK != status) {

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

    }
    setupRC(); 设置数据
    glutMainLoop(); 类似于iOS runloop 运行循环
}
```

CC老师
CC老师



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());

    //调用顶部载入单元矩阵
    modelViewMatrix.LoadIdentity();
}
```

`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



逻辑教育
Logic education

课堂Demo [01]

重要的函数 KeyPressFunc 函数

KeyPressFunc 触发条件:

1. 用户敲击“空格”键位

处理业务:

1. 判断用户输入的是“空格”

2. 设置窗口名称

3. 手动触发重新渲染.

```
void KeyPressFunc(unsigned char key, int x, int y){  
    if(key == 32) {  
        nStep++;  
        if(nStep > 6)    nStep = 0;}  
    switch(nStep) {  
        case 0:  
            glutSetWindowTitle("GL_POINTS");  
            break;  
        case 1:  
            glutSetWindowTitle("GL_LINES");  
            break;  
        case 2:  
            glutSetWindowTitle("GL_LINE_STRIP");  
            break;  
        case 3:  
            glutSetWindowTitle("GL_LINE_LOOP");  
            break;  
        case 4:  
            glutSetWindowTitle("GL_TRIANGLES");  
            break;  
        case 5:  
            glutSetWindowTitle("GL_TRIANGLE_STRIP");  
            break;  
        case 6:  
            glutSetWindowTitle("GL_TRIANGLE_FAN");  
            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

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.

```
//特殊键位处理 (上、下、左、右移动)
void SpecialKeys(int key, int x, int y)
{
    if(key == GLUT_KEY_UP)
        //围绕一个指定的X,Y,Z轴旋转。
        objectFrame.RotateWorld(m3dDegToRad(-5.0f), 1.0f, 0.0f, 0.0f);

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

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

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

    glutPostRedisplay();
}
```

课程研发:CC老师

课程授课:CC老师



```
void SetupRC(){
```

```
//1. 初始化准备
```

```
背景颜色
```

```
存储着色器管理器初始化
```

```
开启深度测试
```

```
设置变换管道中模型视图矩阵/投影矩阵
```

```
设置观察者视图坐标位置
```

```
//2. 三角形绘制准备
```

```
定义三角形顶点数据
```

```
使用三角形批次类,使用点/线/线段/线环的方式绘制图形
```

```
//3. 绘制金字塔准备
```

```
定义金字塔顶点数据
```

```
使用三角形批次类,使用GL_TRIANGLES 绘制金字塔
```

```
//4. 绘制六角形准备
```

```
循环定义顶点数据
```

```
使用三角形批次类,使用GL_TRIANGLES_FAN. 传输数据
```

```
//5. 绘制三角形环准备
```

```
循环定义顶点数据
```

```
使用三角形批次类,使用GL_TRIANGLE_STRIP. 传输数据
```

提示: 完整源码请移步于课程案例源码.

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

课程研发:CC老师

课程授课:CC老师

重要的函数 RenderScene 函数

```
void RenderScene(){
```

```
//1. 渲染工作
```

```
清理缓存区
```

```
将mCamera 观察者坐标系压栈
```

```
将mObjectFrame 图形环绕坐标系压栈
```

```
固定管线渲染点/线/线段/线环。
```

```
//2. 修改图形属性
```

```
判断金字塔/六边形/三角形环时,则添加边框。
```

```
目的让图形能够更加清晰。
```

```
//3. 绘制完毕则还原矩阵。
```

```
//4. 交换缓存区。
```

```
}
```

提示:完整源码请移步于课程案例源码.

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

提示: 完整源码请移步于课程案例源码.

```
void DrawWireFramedBatch(){  
  
    //1. 填充图形内容  
    //2. 绘制边框部分  
  
    多边形偏移.  
    颜色混合  
    绘制边框  
  
    //3. 将设置的属性还原  
}
```

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. 为图形之间增加间隔. 防止Z-Fighting

注意:

此部分内容后面课程会详细讲解. 现阶段不作为重点讲解. 大家了解即可.



逻辑教育
Logic education

课堂Demo [01] 总结

请同学简单描述一下每个函数负责的功能

main 函数

SetUp 函数

SpecialKeys 函数

KeyPressFunc 函数

ChangeSize 函数

RenderScene 函数

DrawWireFramedBatch 函数

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