



逻辑教育
Logic education

Hello 视觉全训班

OpenGL

OpenGL ES

GPUImage

Metal



视觉全训班.第1节课案例解析

@ 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三角形/正方形,并能通过键盘移动

案例1: 001 — 三角形渲染



课程研发:CC老师
课程授课:CC老师



课堂Demo [01]

工具类:

```
#include "GLShaderManager.h"  
#include "GLTools.h"  
#include <GLUT/GLUT.h>
```

`#include<GLShaderManager.h>` 移入了 GLTool 着色器管理器 (shader Manager) 类。没有着色器，我们就不能在OpenGL (核心框架) 进行着色。着色器管理器不仅允许我们创建并管理着色器，还提供一组“存储着色器”，他们能够进行一些初步和基本的渲染操作。

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

``#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



逻辑教育
Logic education

课堂Demo [01]

帮助类:

```
//定义一个，着色管理器  
GLShaderManager shaderManager;  
//简单的批次容器，是GLTools的一个简单的容器类。  
GLBatch triangleBatch;
```

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[])
```

changeSize 函数:自定义函数.通过
glutReshapeFunc(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

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

准备工作

```
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);
    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
}
```

课程研发:CC老师
课程授课: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);
}
```

`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.



课堂Demo [01]

重要的函数 **setupRC** 函数

```
void setupRC()
{
    glClearColor(0.98f, 0.40f, 0.7f, 1);
    shaderManager.InitializeStockShaders();
    GLfloat vVerts[] = {
        -0.5f, 0.0f, 0.0f,
        0.5f, 0.0f, 0.0f,
        0.0f, 0.5f, 0.0f
    };
    triangleBatch.Begin(GL_TRIANGLES, 3);
    triangleBatch.CopyVertexData3f(vVerts);
    triangleBatch.End();
}
```

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



课堂Demo [01]

重要的函数 RenderScene 函数

```
void RenderScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);
    GLfloat vRed[] = {1.0, 0.0, 0.0, 1.0f};
    shaderManager.UseStockShader(GLT_SHADER_IDENTITY, vRed);
    triangleBatch.Draw();
    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 [02] 解析

使用固定存储着色器渲染三角形/正方形,并能通过键盘移动

案例2: 002—图形移动



课程研发:CC老师
课程授课:CC老师

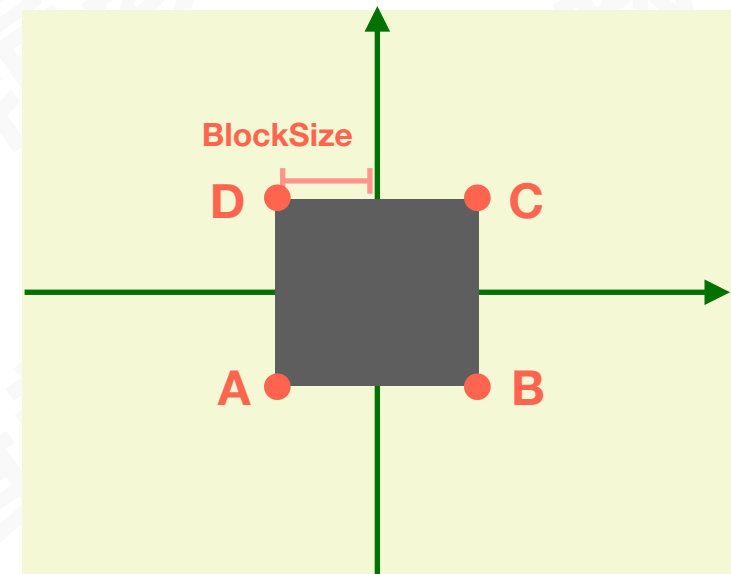


课堂Demo [02]

基于 Demo1 做案例调整

```
//blockSize 边长  
GLfloat blockSize = 0.1f;  
  
//正方形的4个点坐标  
GLfloat vVerts[] = {  
    -blockSize, -blockSize, 0.0f,  
    blockSize, -blockSize, 0.0f,  
    blockSize, blockSize, 0.0f,  
    -blockSize, blockSize, 0.0f  
};
```

图形顶点到圆心距离.
 $\text{BlockSize} = \text{边长} / 2$



正方形的顶点(ABCD)

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

准备工作

```
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); 注册特殊键位响应函数。【当用户使用特殊键位则会调用该函数】
    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
}
```

课程研发:CC老师
课程授课:CC老师



重要的函数 SpecialKeys 函数

```
void SpecialKeys(int key, int x, int y){
```

```
    GLfloat stepSize = 0.025f;  
    GLfloat blockX = vVerts[0];  
    GLfloat blockY = vVerts[10];
```

```
    if (key == GLUT_KEY_UP)      blockY += stepSize;  
    if (key == GLUT_KEY_DOWN)    blockY -= stepSize;  
    if (key == GLUT_KEY_LEFT)    blockX -= stepSize;  
    if (key == GLUT_KEY_RIGHT)   blockX += stepSize;
```

```
    //更新顶点坐标ABCD
```

```
    vVerts[0] = blockX;  
    vVerts[1] = blockY - blockSize*2;
```

```
    vVerts[3] = blockX + blockSize*2;  
    vVerts[4] = blockY - blockSize*2;
```

```
    vVerts[6] = blockX + blockSize*2;  
    vVerts[7] = blockY;
```

```
    vVerts[9] = blockX;  
    vVerts[10] = blockY;
```

```
    triangleBatch.CopyVertexData3f(vVerts);  
    //手动触发重新渲染  
    glutPostRedisplay();}
```

stepSize:移动步长

blockX/blockY: 相对移动顶点D

根据键盘标记判断,此时正方形移动方向(上下左右);然后根据方向,调整相对移动坐标(blockX/blockY)值

根据相对顶点D,计算出ABCD每个顶点坐标值,并更新顶点坐标数组

将顶点数组通过GLBatch帮助类将顶点传输到存储着色器中,并手动出发渲染函数。

课程研发:CC老师
课程授课:CC老师



逻辑教育
Logic education

课堂Demo [02]

思考题: 案例这么实现真的已经没有问题吗?

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



课堂Demo [02]

思考题: 案例这么实现真的已经没有问题吗?

1. 边界问题

2. 如果此时顶点数 >100 ,并且没有规律该怎么办了?



课堂Demo [02]

边界问题:

```
//触碰到边界（4个边界）的处理
//当正方形移动超过最左边的时候
if (blockX < -1.0f)                blockX = -1.0f;

//当正方形移动到最右边时
//1.0 - blockSize * 2 = 总边长 - 正方形的边长 = 最左边点的位置
if (blockX > (1.0 - blockSize * 2)) blockX = 1.0f - blockSize * 2;

//当正方形移动到最下面时
//-1.0 - blockSize * 2 = Y（负轴边界） - 正方形边长 = 最下面点的位置
if (blockY < -1.0f + blockSize * 2) blockY = -1.0f + blockSize * 2;

//当正方形移动到最上面时
if (blockY > 1.0f)                blockY = 1.0f;
```



逻辑教育
Logic education

课堂Demo [02]

数据量问题:

提示: 利用平移矩阵.

课程研发:CC老师
课程授课:CC老师



逻辑教育
Logic education



GPUImage

Metal

OpenGL ES

OpenGL

see you next time ~

@ 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

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