登录 | 注册

个人资料

秋风涩

访问： 277227次
积分： 3265
等级： BLOG 5
排名： 第4676名

原创： 65篇 转载： 3篇
译文： 1篇 评论： 5条

文章搜索

文章分类

Android (58)
iOS (62)
HTML5 (6)

文章存档

2014年09月 (1)
2014年08月 (3)
2014年07月 (2)
2012年12月 (3)
2012年08月 (17)

展开

阅读排行

OpenGL ES入门详解 (31874)
苹果开发者账号购买或续 (18785)
Android开发如何正确使 (15613)
iTunes下载的ipa文件的 (12343)
OpenGL ES之glUniformi (10521)
CGBitmapContextCreate (10151)
HTML5plus 移动 App开发 (9322)
Objective-C/C++混编编译 (8765)

CSDN Android客户端发布 扒一扒最NB的开发项目 他们都已提交，就差你了！ CSDN博主维权信息收集 最流行的语言都在这，想学就学！

OpenGL ES入门详解

分类： iOS Android 2012-07-12 10:22 31885人阅读 评论(0) 收藏 举报

1.决定你要支持的O 本。目前，**OpenGL ES**包含**1.1**和**2.0**两个版本，**iPhone 3G+**和**iPad**开始支持**OpenGL ES2.0**。而 之间的差异非常大，不仅仅在编程思想上，**API**之间的差距也很大。因此，如果你想使用**OpenGL ES**开发**3D**程序或游戏，那么首先就要决定使用哪个版本，还是说两个版本都支持。**OpenGL ES**定义了代表

```
enum
{
    kEAGLRenderingAPIOpenGLES1 = 1,    //1.1版
    kEAGLRenderingAPIOpenGLES2 = 2     //2.0版
}typedef NSUInteger EAGLRenderingAPI;
```

以iPhone代码为例，你可以通过以下方式判断用户设备所支持的OpenGL ES版本，如果支持2.0，就使用2.0进行渲染；如果仅支持**1.1**，则使用**1.1**版进行渲染：

```
EAGLRenderingAPI api = kEAGLRenderingAPIOpenGLES2;   //默认优先使用2.0版
m_context = [[EAGLContext alloc] initWithAPI:api];    //使用2.0版初始化EAGLContext
if (!m_context ) {          //使用2.0版初始化EAGLContext失败
    api = kEAGLRenderingAPIOpenGLES1;       //将API设为1.1版
    m_context = [[EAGLContext alloc] initWithAPI:api];//使用1.1版初始化EAGLContext
}
if (!m_context || ![EAGLContext setCurrentContext:m_context]) {    //1.1版初始化失败，则释放内存
    [self release];
    return nil;
}
if (api == kEAGLRenderingAPIOpenGLES1) {
    //使用1.1版开始渲染
}
else {
    //使用2.0版开始渲染
}
```

2.你想让你的**OpenGL ES**程序跨手机平台运行么？**Android**、**iPhone**、**windows phone**手机系统是当前最主流的手机系统，如何才能让我们编写出来的程序可以跨平台运行呢？好消息是，这三个平台都支持**OpenGL ES**，而且都支持**C++**语言。也许你明白了，方法就是通过**C++**来封装**OpenGL ES**，尽量少用特定平台的耦合代码。这样我们就可以将封装的**OpenGL ES**引擎在不同的手机平台的使用，而且仅需要修改特定平台连接**OpenGL ES**的代码即可。

3.入门常见代码解析

3.1创建渲染缓冲区

```
GLuint m_renderbuffer;//创建一个渲染缓冲区对象
```

OpenGL ES之glTexImag (8562)
HTML5 canvas自适应手 (7438)

评论排行

HTML5 canvas自适应手 (2)
HTML5plus 移动 App开发 (1)
苹果开发者账号购买或续 (1)
obj2opengl：转换OBJ 3D (1)
OpenGL ES之glEnable和 (0)
OpenGL ES几何变换和坐 (0)
HBuilder百度地图显示不 (0)
OpenGL ES之glCullFace (0)
OpenGL ES之glDrawEle (0)
OpenGL ES贴图坐标和顶 (0)

推荐文章

* 2015博文大赛精彩文章
*为什么我说Rust是靠谱的编程语言
*Android UI常用实例 如何实现欢迎界面（Splash Screen）
*Android应用层View绘制流程与源码分析
*Android屏幕适配全攻略
*一个多月来的面试总结(阿里, 网

最新评论

HTML5 canvas自适应手机屏幕宽
秋风涩: @wayhb:可以的，我自己写了一个俄罗斯方块程序，用了不少图片和程序编写的图形，没有变形啊。
HTML5 canvas自适应手机屏幕宽
泪奔的蜗牛: height: 100%;width: 100%，这样设置使得图像变形，虽然全屏了，但是也不能用！
HTML5plus 移动 App开发入门
georson: 超赞！！！！
苹果开发者账号购买或续费支付
Oftentap-米点联系人: 一般购买这儿账号提交申请后要等多久苹果才回复？
obj2opengl：转换OBJ 3D模型到
aiyongyyy: 这个确实非常好用啊

glGenRenderbuffers(1, &m_renderbuffer);//创建一个渲染缓冲区对象

glBindRenderbuffer(GL_RENDERBUFFER, m_renderbuffer);//将该渲染缓冲区对象绑定到管线上

3.2创建帧缓冲区

GLuint m_framebuffer;//创建一个帧缓冲区对象

glGenFramebuffers(1, &m_framebuffer);//创建一个帧染缓冲区对象

glBindFramebuffer(GL_FRAMEBUFFER, m_framebuffer);//将该帧染缓冲区对象绑定到管线上

glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_RENDERBUFFER, m_renderbuffer);//将创建的渲染缓冲区绑定到帧缓冲区上，并使用颜色填充

3.3设置视口

glViewport(0, 0, width, height);//定义视口大小，说白了就是OpenGL ES窗口大小

3.4创建着色器

3.4.1创建一个空着色器

GLuint shaderHandle = glCreateShader(shaderType);//shaderType代表着色器的类型，可以是GL_VERTEX_SHADER（顶点着色器）或GL_FRAGMENT_SHADER（片元着色器）

3.4.2指定着色器源代码

glShaderSource(shaderHandle, 1, &source, 0);// source代表要执行的源代码字符串数组，1表示源代码字符串数组的字符串个数是一个，0表示源代码字符串长度数组的个数为0个

3.4.3编译着色器

glCompileShader(shaderHandle);//编译着色器

3.4.4检查编译是否成功

GLint compileSuccess;

glGetShaderiv(shaderHandle, GL_COMPILE_STATUS, &compileSuccess);//查看编译着色器是否成功，可选的查询状态有L_DELETE_STATUS,GL_COMPILE_STATUS, GL_INFO_LOG_LENGTH, GL_SHADER_SOURCE_LENGTH

```
//如果编译出错，则记录出错信息
if (compileSuccess == GL_FALSE) {
    GLchar messages[256];
    glGetShaderInfoLog(shaderHandle, sizeof(messages), 0, &messages[0]);
    std::cout << messages;
    exit(1);
}
```

3.5创建渲染源程序

3.5.1创建一个空源程序

GLuint programHandle = glCreateProgram();//创建一个渲染程序

3.5.2向源程序中添加着色器

glAttachShader(programHandle, shaderHandle);//将着色器添加到程序中

3.5.3链接源程序

glLinkProgram(programHandle);//你可能添加了多个着色器，链接程序

3.5.4检查链接程序是否成功

GLint linkSuccess;

glGetProgramiv(programHandle, GL_LINK_STATUS, &linkSuccess);//查看连接是否成功

//链接失败记录失败信息

```
    if (linkSuccess == GL_FALSE) {
        GLchar message[256];
        glGetProgramInfoLog(programHandle, sizeof(message), 0, &message[0]);
        std::cout << message;
        exit(1);
    }
```

3.6顶点结构体

//此处定义了一个代表顶点的结构体，内部包含一个有两个点（x，y）组成的顶点的位置信息，和一个四个值（r, g, b, a）表示的颜色信息

```
struct Vertex{
    float Position[2];
    float Color[4];
};
//创建顶点数组，里面有6个顶点信息
const Vertex vertices[] = {
    {{-0.5, -0.866}, {0.5, 1, 0.5, 1}},
    {{0.5, -0.866}, {0.2, 0.6, 0.5, 1}},
    {{0, 1}, {0.6, 0.1, 0.8, 1}},
    {{0.5, -0.866}, {0.5, 0.5, 0.5, 1}},
    {{1.5, -0.866}, {0.5, 0.5, 0.5, 1}},
    {{1, 0.4}, {0.5, 0.5, 0.5, 1}}
};
```

3.7着色器

3.7.1顶点着色器

```
#define STRINGIFY(A) #A

const char *SimpleVertexShader = STRINGIFY(
attribute vec4 Position;//位置，vec4说明有4个点组成，attribute表示属性，由程序提供输入值
attribute vec4 SourceColor;//源颜色，RGBA
varying vec4 DestinationColor;//目标颜色，输出值，用来传递到片元着色器，vary表示可变传出变量
uniform mat4 Projection;//投影矩阵，mat4表示一个4*4的矩阵，uniform表示统一的，不变的，每个顶点都是固定的这个值
uniform mat4 Modelview;//模型矩阵
void main(void){
    DestinationColor = SourceColor;//简单的将源颜色赋给目标颜色
    gl_Position = Projection * Modelview * Position;//通过矩阵乘法获得目标位置，gl_Position是OpenGL ES内定的值，必须指定
}
);
```

3.7.2片元着色器

```
const char *SimpleFragmentShader = STRINGIFY(
varying lowp vec4 DestinationColor;//由顶点着色器传入，lowp表示低精度
void main(void){
    gl_FragColor = DestinationColor;//片元颜色，gl_FragColor也是OpenGL ES内定的，必须指定
}
);
```

3.8开始渲染

3.8.1填充（清理）屏幕

glClearColor(0.1f, 0.9f, 0.5f, 1);//指定填充屏幕的RGBA值
glClear(GL_COLOR_BUFFER_BIT);//指定要清除哪些缓冲区，GL_COLOR_BUFFER_BIT表示颜色缓冲区，GL_DEPTH_BUFFER_BIT表示深度缓冲区，GL_STENCIL_BUFFER_BIT表示模板缓冲区

3.8.2从着色器代码中获取属性信息

GLuint m_simpleProgram = programHandle;

GLuint positionSlot = glGetAttribLocation(m_simpleProgram, "Position");//从着色器源程序中的顶点着色器中获取Position属性
GLuint colorSlot = glGetAttribLocation(m_simpleProgram, "SourceColor");//从着色器源程序中的顶点着色器中获取SourceColor属性

3.8.3开启顶点属性数组
glEnableVertexAttribArray(positionSlot);
glEnableVertexAttribArray(colorSlot);

3.8.4为着色器属性赋值
GLsizei stride = sizeof(Vertex);//单个顶点的数据长度
const GLvoid *pCoords = &vertices[0].Position[0];//顶点数组中的位置数组首地址
const GLvoid *pColors = &vertices[0].Color[0];/顶点数组中的颜色数组首地址
glVertexAttribPointer(positionSlot, 2, GL_FLOAT, GL_FALSE, stride, pCoords);//为顶点着色器位置信息赋值，positionSlot表示顶点着色器位置属性（即，Position）；2表示每一个顶点信息由几个值组成，这个值必须位1，2，3或4；GL_FLOAT表示顶点信息的数据类型；GL_FALSE表示不要将数据类型标准化（即fixed-point）；stride表示数组中每个元素的长度；pCoords表示数组的首地址
glVertexAttribPointer(colorSlot, 4, GL_FLOAT, GL_FALSE, stride, pColors);//同上

3.8.5渲染顶点
GLsizei vertexCount = sizeof(vertices)/sizeof(Vertex);//顶点个数
glDrawArrays(GL_TRIANGLES, 0, vertexCount);//将顶点数组使用三角形渲染，GL_TRIANGLES表示三角形，0表示数组第一个值的位置，vertexCount表示数组长度

3.8.6渲染完毕，关闭顶点属性数组
glDisableVertexAttribArray(positionSlot);
glDisableVertexAttribArray(colorSlot);

4.OpenGL 1.1和2.0在编程上的区别

4.1函数命名上的区别

1.1版API函数和宏末尾通常都为加上OES（即OpenglES的缩写），2.0版本基本上都把这个后缀名给去掉了，如：

1.1API函数和宏：glBindRenderbuffer~~OES~~(GL_RENDERBUFFER_~~OES~~, m_renderbuffer);

2.0API函数和宏：glBindRenderbuffer(GL_RENDERBUFFER, m_renderbuffer);

4.2渲染方式不同

1.1版是基于不可编程管线，2.0版是基于可编程管线的，明显的差别是**1.1**不支持着色器而2.0支持着色器，如下：

在**1.1**版将顶点渲染到屏幕上一般这样写：

glMatrixMode(GL_PROJECTION);
const float maxX = 2;
const float maxY = 3;
glOrthof(-maxX, maxX, -maxY, maxY, -1, 1);
glMatrixMode(GL_MODELVIEW);

2.0版支持着色器，就不再这么渲染顶点了。将顶点属性和变换方式全部放在顶点着色器和片元着色器中，然后程序中通过着色器来渲染

4.3编写同样的程序API不同

如，同样是激活和关闭顶点数组，在**1.1**中是glEnableClientState(GL_VERTEX_ARRAY);和glDisableClientState(GL_VERTEX_ARRAY);

但在2.0中，就变成了：glEnableVertexAttribArray(*);和glDisableVertexAttribArray(*);

差别的来源就在于**1.1**不使用着色器，而2.0使用着色器。

4.4编程难易程度不同。

1.1是基于不可编程管线的，因此，管线的各个组件都是写好的，我们仅需要调用即可。而2.0是基于可编程管线的，灵活性大大增加了，但是编写的难度和复杂度也增加了，因为什么功能都得自己写了。

5.着色器的使用流程

刚开始学习OpenGL ES2.0，对其着色器十分不感冒，什么是着色器，着色器该怎么使用，着色器包含哪些内容呢?

着色器就是一段包含着色信息的源代码字符串。通常着色器分为顶点着色器（Vertex Shader）和片元着色器（Fragment Shader），两个着色器分别写在不同的文件中，文件没有固定的后缀名，可以根据你自己的爱好写，但是最好能区别文件中写的是顶点着色器还是片元着色器，不然时间长了自己都不知道哪个文件中写的是什么信息了。如你可以给你的顶点着色器后缀名命名为：vert, ver, v, vsh等，给你的片元着色器后缀名命名为：frag, fra, f, fsh等。

着色器源代码和OpenGL源代码不是一起编译的，所以要特别强调我刚才说的"**着色器是**一段包含着色信息的**源代码字符串**"。所以，OpenGL源代码肯定是和工程一起编译的，但是着色器源代码是在运行期编译的。你可能会问，着色器的源代码是一个字符串怎么编译呢？所以OpenGL ES提供了一套运行期动态编译的流程：

（1）创建着色器：glCreateShader

（2）指定着色器源代码字符串：glShaderSource

（3）编译着色器：glCompileShader

（4）创建着色器可执行程序：glCompileShader

（5）向可执行程序中添加着色器：glAttachShader

（6）链接可执行程序：glLinkProgram

上一篇 Objective-C/C++混编编译器设置
下一篇 OpenGL ES之glUniform函数

主题推荐 opengl es 手机系统 windows phone 跨平台 对象

猜你在找

Android 44 Graphic系统详解4 一个view的绘制之旅
Cocos2d-x 30开发六使用cocoStudio创建一个骨骼动画
Android提高篇之自定义dialog实现processDialog正在
ErrorThe SDK Build Tools revision 1903 is too low
ShaderToy开篇
【精品课程】太空大战游戏实战课程
【精品课程】三维游戏引擎开发-渲染
【精品课程】JavaScript for Qt Quick(QML)
【精品课程】C++语言基础
【精品课程】全能项目经理训练营

准备好了么？跳吧！ 更多职位尽在 CSDN JOB

Linux C/C++ 网络加速软件开发工程师 我要跳槽
北京思朗特科技有限公司 | 15-25K/月
高级JAVA软件工程师 我要跳槽
北京思朗特科技有限公司 | 15-25K/月
高级软件工程师 我要跳槽
北京鑫源联创科技有限责任公司 | 7-8K/月
Android工程师 我要跳槽
北京鑫源联创科技有限责任公司 | 10-14K/月