|  |
| --- |
|  |
| C:\Users\ADMINI~1\AppData\Local\Temp\ksohtml3088\wps1.jpg |  |

硕 士 研 究 生 读 书 报 告



题目Prototype implementation of

the OpenGL ES 2.0 shading

language offline compiler

作者姓名 邵晗琦

作者学号 21951119

指导教师 李启雷

学科专业 软件工程

所在学院 软件学院

提交日期 二○一九 年12月

摘要

本文重点讨论了针对OpenGL着色语言离线编译器的设计方案。在最新的3D图形库（包括OpenGL和OpenGLES）中，提供了可编译的渲染管线，要使用这些管线，自然需要专门的编程语言，这种语言在OpenGL中被称为着色器语言。本文在现有的OpenGL着色语言的基础上，设计并实现了一种针对着色器语言的离线编译器设计方案，用于对离线编译器原型的实现。

**关键词**：OpenGL ES， 着色语言，离线编译器

Abstract

The paper focuses on the design of an offline compiler for the OpenGL shading language.Compilable rendering pipelines are provided in the latest 3D graphics libraries(including OpenGL and OpenGL ES).To use these pipelines, a specialized programming language is naturally required, which is called Shading language. Based on the existing OpenGL shading language, this paper designs and implements an offline compiler design solution for the shader language, which is used to implement the offline compiler prototype.

**Keywords：**OenGL ES Shading language Offline compiler

1引言

OpenGL(Open Graphics Library)是用于渲染2D、3D矢量图形的一种跨语言、跨平台的应用程序编程接口，通过调用各种函数，绘制从简单的图像比特到复杂的三维景象，是一种功能强大且调用方便的底层图形库。针对手机、平板电脑等嵌入式设备，OpenGL还提供了OpenGL ES版本，它是OpenGL的一个子集，在移动设备上进行图形图像处理时，它的优势更能体现出来。

OpenGL Shading Language（GLSL）是OpenGL的着色器语言，它是一种高级的图形编程语言，在OpenGL2.0版本开始成为其底核，具有重要的作用。GLSL源自于应用广泛的C语言，并吸收了包括RendeMan等着色语言的优良特性，完美支持对向量和矩阵的各种操作，通过类型限定符管理输入与输出。同时，它也提供了多种API函数，帮助顶点着色器和片段着色器完成丰富的功能，因此也被称作在线着色器编译语言。

在这篇论文中，作者提出了一种基于OpenGL ES 2.0以上版本着色语言的离线编译器实现方案，这种离线编辑器符合OpenGL ES语言的完整规范，为其中的默写GPU生成执行代码，实现离线编译器的功能。

2 背景分析

2.1着色器API函数

在OpenGL的着色语言代码的执行过程中，使用专门的API进行编译操作，因此在实现离线着色语言编译器之前，需要详细理解包括API函数，脱机编译原理和二进制格式支持模式等具体内容。

在OpenGL2.0及OpenGL ES 2.0规范中，对着色器的使用是通过图形渲染管线实现的，如图2-1所示。图形渲染管线在接收一组3D坐标后，可以通过一系列的处理将它转化为屏幕上显示的2D有色像素输出。管线系统分为两个主要部分：第一部分可以将3D坐标信息转化为对应的2D坐标，称为顶点着色器；第二部分则将2D坐标转换为实际的、有颜色的像素。在处理的过程中通过数个特定的、高度专门化的函数，并行执行处理操作，并通过GPU进行执行。而这些着色器程序正是使用着色器语言编写的。

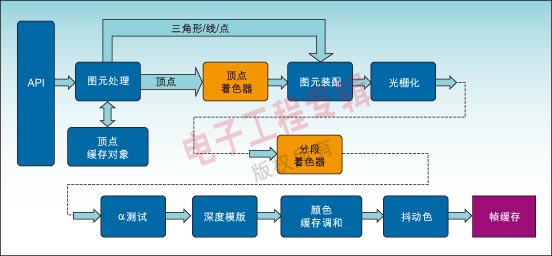


图2-1 OpenGL 2.0图形渲染管线流程图

由于OpenGL2.0规范下的两个着色器对象：顶点着色器和片段着色器的对象有诸多共同的特征，因此通常使用着色器语言来处理两者的些许差别，并通过API函数管理两种着色器对象。

首先通过glCreateShader函数创建一个着色器对象，并选择vertex或fragment作为要实现的着色器类型，接下来通过一下函数提供需要实现的着色器代码，其中shader代表对象的标示符：

void glShaderSource( GLuint shader,GLsizei count, const GLchar\*\* string,

const GLint\* length );

在得到源代码后，即可通过glCompileShader()函数将给定的着色器对象源代码进行编译，并将其链接到需要使用的函数中。以如上方法分别实现顶点和片段着色器的编译和链接后，即可通过glUseProgram()将连接的着色器对象在当前的程序中执行渲染工作，并最终实现渲染功能。

2.2二进制格式支持

在OpenGL ES 2.0及更高的版本中，系统提供了着色语言的在线编译器帮助程序进行编译。同时，作为其他情况的替代选项，OpenGL也提供了对应的API函数作为对二进制图像的接口，这是本文实现着色语言离线编译器的重要基础。OpenGL通过函数如下函数实现二进制图像的处理：

void glProgramBinary( GLuint program,GLenum binaryFormat, const void\* binary,GLsizei length );

其中，program参数指定加载的程序中二进制文件对象的名称，识别需要处理的文件；binaryFormat参数则通过二进制的形式，指定数据的格式；binary参数包含二进制素组存放的地址信息；length参数则控制了该二进制文件所包含的全部字节数量。通过如上接口，即可实现程序对二进制图像的读取，并在内部将其转化为顶点着色器和片段着色器程序的形式，并最终将其转化为GPU上的可执行程序在GPU上实现执行。

通过如上对API及二进制格式处理的分析，即可得出本文实现着色器语言离线编译系统的具体思路：首先将初始的数据通过处理转化为二进制信息，并将编译得到的结果输入程序中，转化为对应的顶点着色器和片段着色器形式，使之可以最终在GPU上进行执行，输出渲染结果。

3着色器语言编译器的设计和实现

3.1总体设计

要实现基于OpenGL ES2.0的着色器语言功能，就需要一个支持离线编译的着色语言编译器，在本文中，作者使用的是一些开源的语言编译器，通过现有的这些编译器系统，可以有效的生成出已编译的二进制图像、所有统一的变量位置和其他重要的数据信息。接下来，他们使用自己的特定二进制图像格式作为编译结果，输入运行核心中，通过glProgramBinary()函数解码这些信息，以实现GPU和状态变量的使用。通过如上的操作思路，即可逐步构建出计划中的着色语言编译系统，并在调试的过程中排除错误，提升准确性。

3.2 具体实施策略

在具体的实现过程中，作者首先参考了线程的在线编译器模块。对于现有的大部分在线着色语言编译器，通常是在源代码中使用在线的编译器和OpenGL自带函数获取目标二进制图像，并将其存储在主存储器中，并使用同一程序进行执行，通过这种方法，可以最简单高效的进行可行性测试。接下来，将重新使用这些被保存的二进制图像，调用glGetProgramBinary函数，将编译完成的二进制图像存储便于后续检索，同时，这些图像将会在稍后被另一个程序使用glProgramBinary函数加载，基于此测试，可以验证这些编译完成的二进制图像的可行性。

对于前端的实现，本文通过将顶点着色器和片段着色器的源代码转换为Tungsten代码，实现可用的着色语言编译器前端，虽然这些操作并不符合OpenGL规范，但在一些包括Mesa 3D图形库的商业实现中，允许采用这些中间代码图像，至此完成了离线编译器的前端系统。

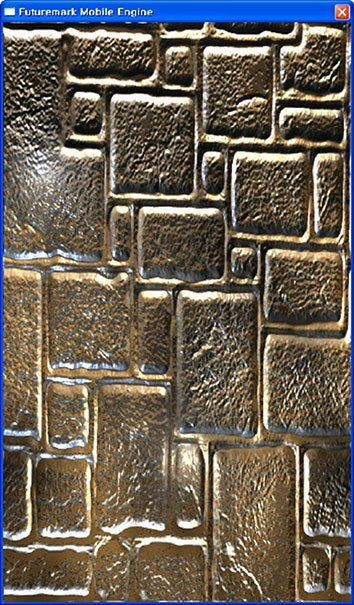
在最后的步骤中，通过实现后端GPU代码生成器，与之前完成的前端相结合，即可得到用于OpenGL着色语言的全尺寸离线编译器。在这个离线编译器中，可以实现对每个GPU型号对应的代码生成，从而共同构建出一整套完整的OpenGL着色语言离线编译系统。当然，也可以在此基础上进行代码优化，提升整体的性能。

根据以上的分布实施策略，即可从低到高逐层构建完整的编译系统，并通过对每个阶段的全面测试与调试，提升性能，排除错误。

4结果及展望

本文的作者团队通过上述步骤的操作，为基于OpenGL的OpenGL ES着色器语言（ESSL）实现了离线编译的功能，通过这些预处理操作，可以处理包括#define，#praga，#version在内的所有ELSL预处理指令，还能针对2.0的全部标准规范，对经过预处理操作的着色器语言代码进行语法和语义的检查，实现对着色器语言语句的完整编译和纠错。

除此之外，该团队还对于Ada-like风格的ESSL增加了专门的语义检查程序，在这个过程中使用的是OpenGL ES 2.0一致性测试套件CTS。这些操作的实施最终得到的编译成功率达到约97%，在对离线编译器的测试中，得到了一些结果如下图3-1所示。

(a) (b) (c)

图3-1(a)“Taeji”测试程序的屏幕截图，资料来源Rightware

(b)“Hoverjet”测试程序的屏幕截图，资料来源Rightware

(c)“Advanced World”测试程序的屏幕截图，资料来源Rightware

得到的离线编译器目前已经实现与现有的应用程序共同使用的功能。

在本文中，作者团队旨在实现OpenGL着色语言的离线编译器。目前市面上已经有一些较为成熟的在线编译系统，但是在很多情况下依然需要使用基于OpenGL SC 2.0标准的离线编译器系统，帮助用户更方便、更高效的在编程和编译过程中OpenGL着色语言的编辑与检查，本文得到的离线编译一同设计步骤，可以切实有效的帮助这种系统的推进与优化，应用在当前的OpenGL编程与设计之中。同时该项目也存在诸多缺陷：例如对一些较复杂的着色语言程序的编译仍存在一定的错漏，稳定性较成熟的在线编译器仍有一定距离；同时，截止发文，团队并未完成完整的离线编译器实现，距离推广应用仍有需要努力的空间。