9.1 OpenGL初步

- OpenGL软件包
- OpenGL的绘制流程
- OpenGL的基本语法
- OpenGL环境配置
- OpenGL的程序实例

OpenGL软件包

- OpenGL是SGI（Silicon Graphics Inc.）公司对IRIS GL进行改进，扩展可移植性，形成的一个跨平台开放式图形编程接口。
- OpenGL标准由1992年成立的独立财团OpenGL Architecture Review Board（ARB）以投票方式产生，并制成规范文档公布。
- 2006年，Khronos集团，技术联合体

OpenGL图形软件包

- OpenGL 1.x
  - 1992年，1.0，即时渲染模式
  - 1997年，1.1，纹理对象和顶点数组等
  - 1998年，1.2，3D纹理，多重纹理等
  - 2001年，1.3，多重纹理等
  - 2002年，1.4，纹理环境，深度纹理比较等
  - 2003年，1.5，增加对缓冲区对象的支持

OpenGL图形软件包

- OpenGL ES 1.x
  - 2003，精简版本，去除即时模式并采用定点数代替浮点数，用于移动平台和低功耗平台
  - 2004年，1.1，增加缓冲区对象、多重纹理等

OpenGL图形软件包

- 一切都是可编程的：OpenGL 2.x
  - 2004，可编程的，基于着色器的API
  - 2006，2.1，像素缓冲区对象
- OpenGL ES 2.x
  - 2007，着色器

OpenGL图形软件包

- 几何和顶点处理的演变：OpenGL 3.x
  - 2008年，纹理格式，纹理阵列，变换反馈
  - 2009年3月，3.1，实例化渲染等
  - 2009年8月，3.2，核心和兼容性配置，增加几何着色器。

OpenGL图形软件包

- 最新版本：**OpenGL 4.x**
  - **2010年，曲面细分（Tessellation），增加曲面细分控制和曲面细分评估着色器**
  - **4.1 （2010年7月）；4.2（2011年8月）；4.3和4.4（2013年）；4.5 （2014年）；4.6（2017年）**

OpenGL的绘制流程——工作方式

- 一个完整的窗口系统的OpenGL图形处理系统的结构为：最底层为图形硬件，第二层为操作系统，第三层为窗口系统，第四层为OpenGL，最上面的层为应用软件。

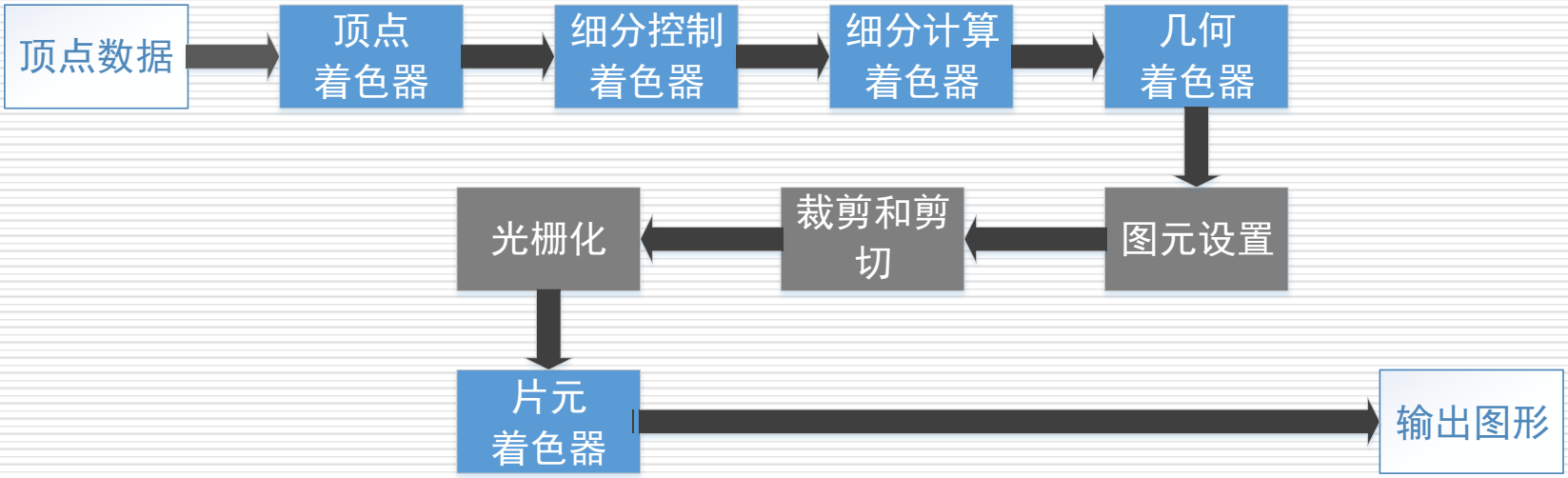
OpenGL的绘制流程——主要操作

- 从OpenGL的几何图元中设置数据，用于构建形状；
- 使用不同的着色器（Shader）对输入的图元进行计算操作，判断其位置，颜色以及其他渲染属性
- 将输入图元的数学描述转换为与屏幕位置对应的像素片元（fragment），即光栅化；

OpenGL的绘制流程——主要操作

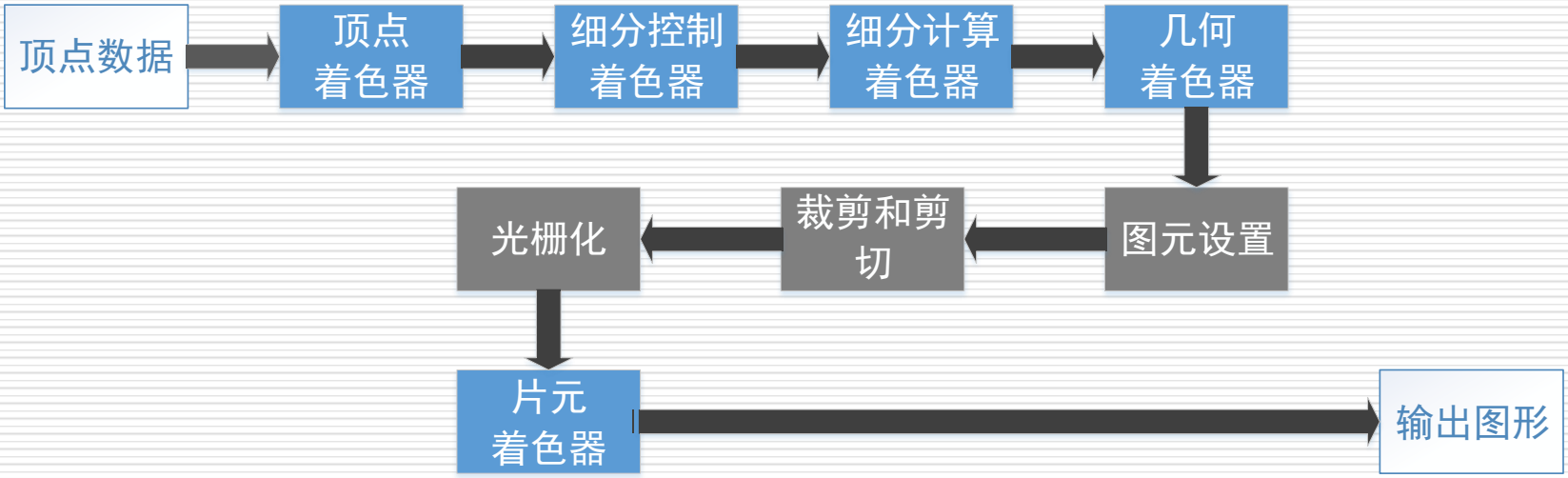
- 针对光栅化过程产生的片元，执行片元着色器，决定其最终的颜色和位置。
- 针对片元的其他操作，如可见性判断，融合等等。

OpenGL的绘制流程——渲染管线



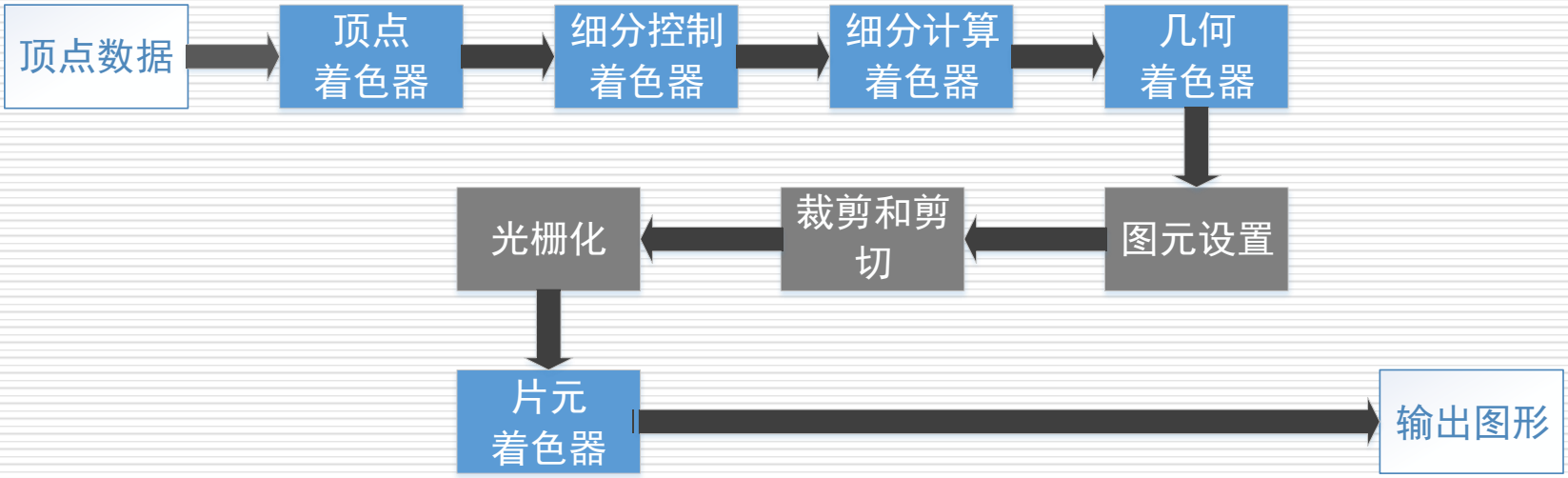
- 顶点数据：将顶点数据保存在缓存对象中，OpenGL通过绘制命令传输顶点数据。

OpenGL的绘制流程——渲染管线



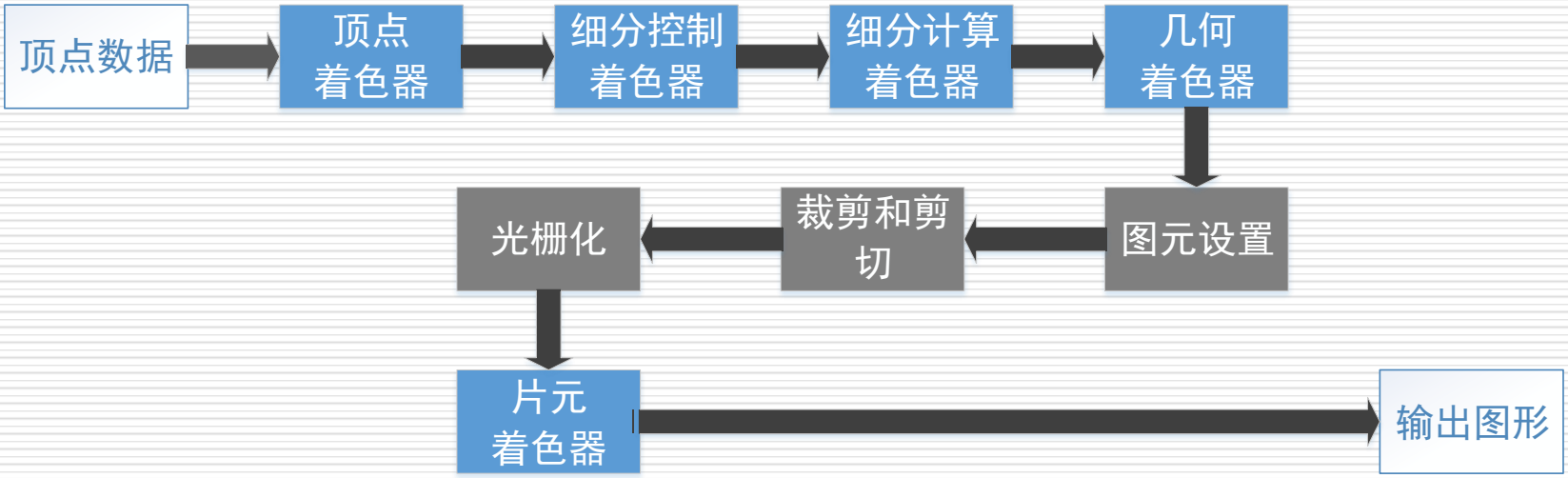
- 顶点着色器：对顶点数据进行处理，包括变换，顶点着色（材质属性）

OpenGL的绘制流程——渲染管线



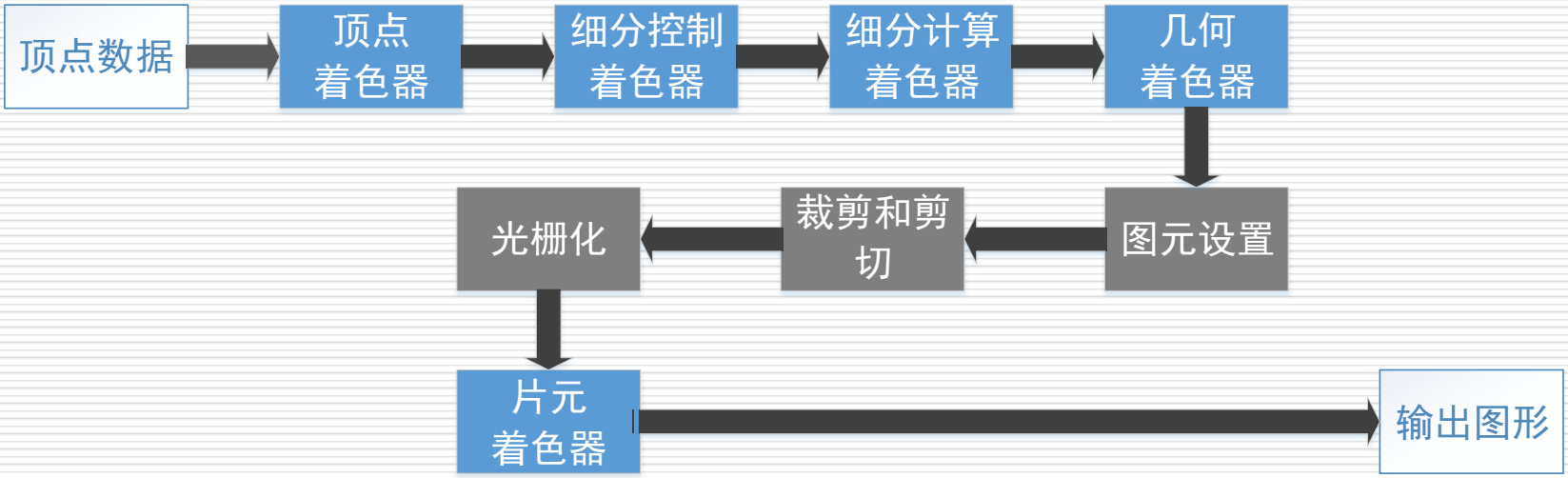
- 细分着色器：使用面片（patch）描述物体的形状，细分可以使模型外观变平滑，但会增加点的数量。

OpenGL的绘制流程——渲染管线



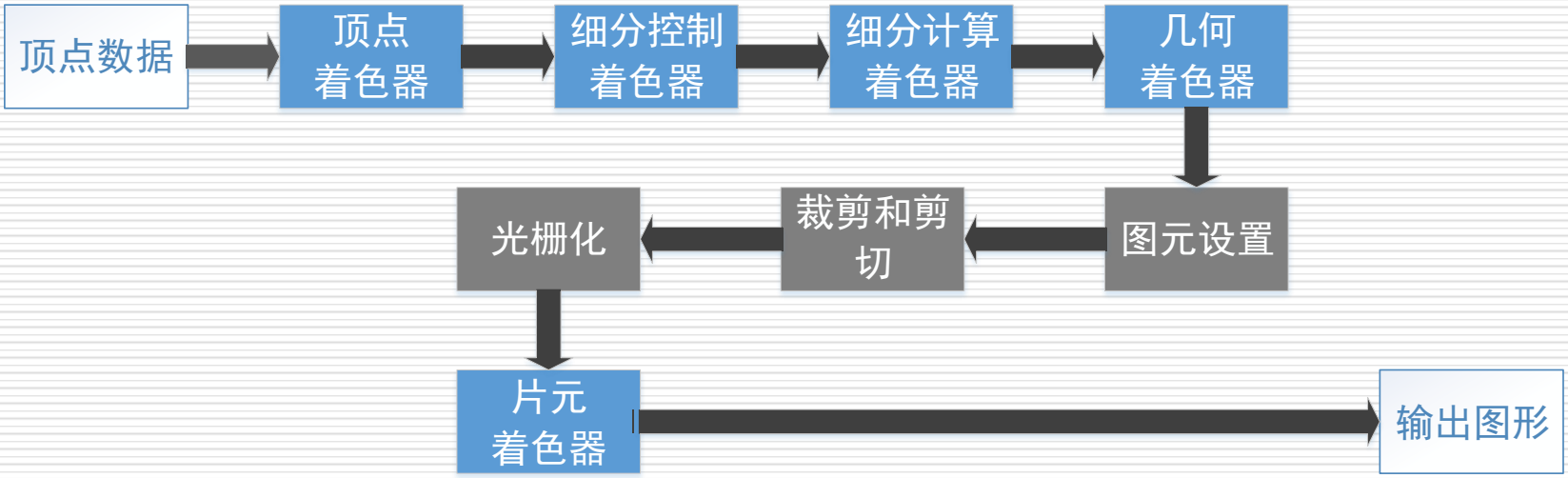
- 几何着色：在光栅化之前对每个几何图元进行更进一步的处理。

OpenGL的绘制流程——渲染管线



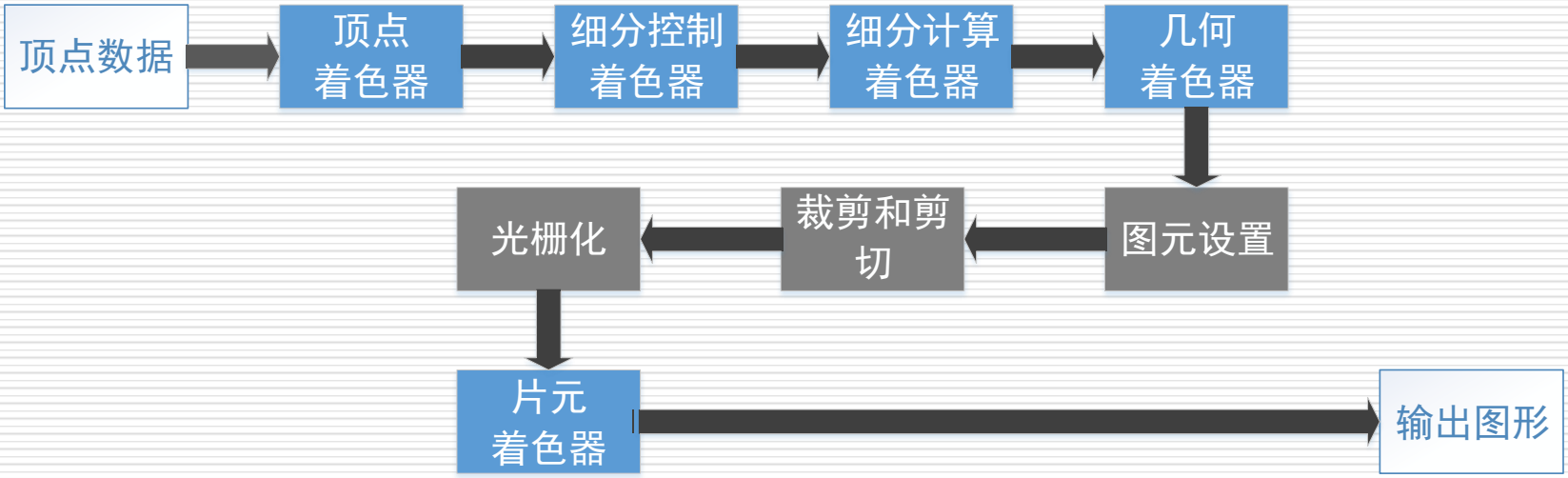
- 图元装配：将顶点与相关的几何图元组织起来，准备下一步操作。

OpenGL的绘制流程——渲染管线



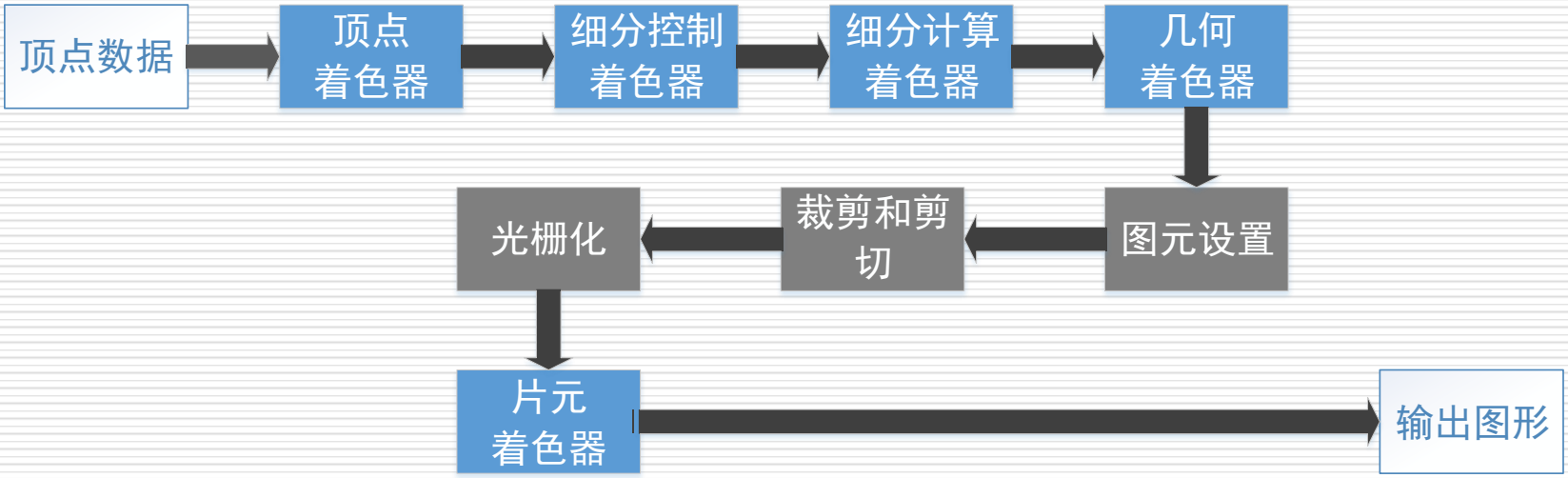
- 裁剪和剪切：OpenGL根据指定的方式自动完成裁剪过程。

OpenGL的绘制流程——渲染管线



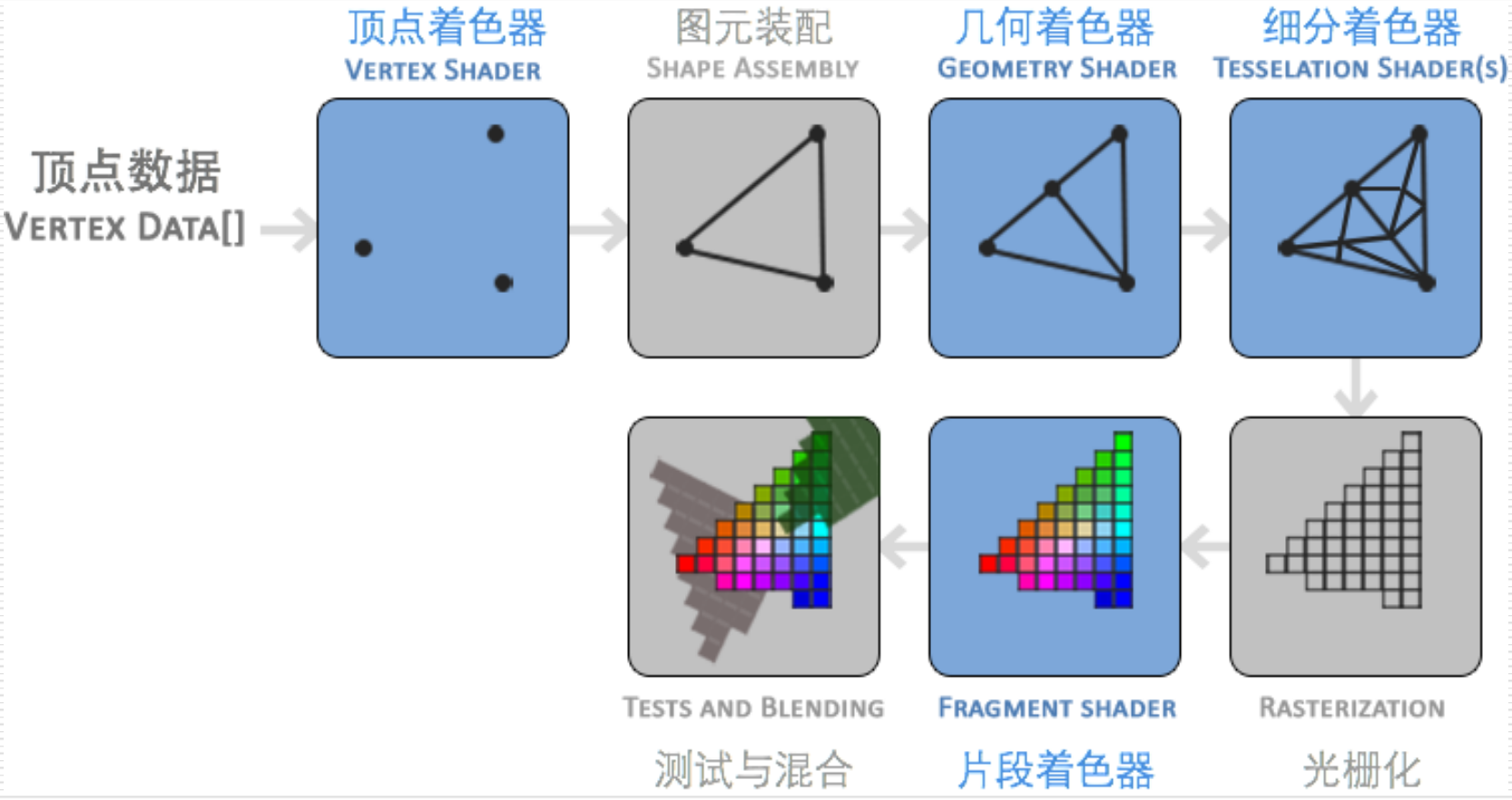
- 光栅化：根据图元顶点坐标，生成片元。片元是候选的像素点。

OpenGL的绘制流程——渲染管线



- 片元着色器：可以决定片元最终的颜色，是否显示或者终止片元的处理。

OpenGL的绘制流程——渲染管线



OpenGL的基本语法——相关库

- 核心库：gl
- 实用程序库： glu
- 编程辅助库：aux
- 实用程序工具包：glut
- 实用工具库：glfw
- 跨平台扩展库：glew

OpenGL的基本语法——相关库

- X窗口系统扩展：glx
- Apple窗口系统扩展： agl
- Windows窗口系统扩展：wgl
- 第三方库：glad

OpenGL的基本语法——命名规则

- OpenGL函数都遵循一个命名约定，即采用以下格式：

<库前缀><根命令><可选的参数个数><可选的参数类型>

函数glUniform2f(...)，glUniform2fv(...)

OpenGL的基本语法——数据类型

表2.2 OpenGL的数据结构

OpenGL数据类型	内部表示法	定义为C类型	C字面值后缀
GLbyte	8位整数	signed char	B
GLshort	16位整数	short	S
GLint，GLsizei	32位整数	long	L
GLfloat，GLclampf	32位浮点数	float	F
GLdouble，GLclampd	64位浮点数	double	D
GLubyte，GLboolean	8位无符号整数	unsigned char	Ub
GLshort	16位无符号整数	unsigned short	Us
GLuint，GLenum，GLbitfield	32位无符号整数	unsigned long	Ui

OpenGL的环境配置

- GLFW的环境
  - 使用CMake工程生成工具，编译GLFW库,http://www.glfw.org/download.html
  - 用Visual Studio的插件“**NuGet**”

OpenGL的环境配置

- GLFW的环境
  - 访问https://www.glfw.org/download.html下载glfw源码；
  - 下载Cmake(https://cmake.org/download)，建议下载**win32-x86**版本
  - 用Cmake编译源码，得到include和lib。

OpenGL的环境配置

- GLAD是一个开源的库，配置也与其他的库有些不同，GLAD使用了在线服务。

  https://glad.dav1d.de/

- 选择3.3以上的OpenGL(gl)版本，Profile设置为Core，选中生成加载器(Generate a loader)，忽略拓展(Extensions)中的内容，生成库文件。

OpenGL的程序实例

- 用GLFW实现窗口处理
  - **glfwCreateWindow**
  - **glfwMakeContextCurrent()**
  - **glfwWindowShouldClose()**
  - **glfwSwapBuffers()**
  - **glfwPollEvents()**

OpenGL的程序实例

- 加载**GLAD**库
  - **gladLoadGLLoader((GLADloadproc)glfwGetProcAddress)**

OpenGL的程序实例

- 顶点数组对象（**Vertex Arrary Object，VAO**）
  - **VAO**保存了所有顶点数据的引用。
  - **VAO**把顶点存储在一个对象中，每次绘制模型时，只需要绑定这个**VAO**对象就可以了。

OpenGL的程序实例

- 顶点数组对象（**Vertex Arrary Object，VAO**）
  - **glGenVertexArrays(1, &vertex_array_object);**
  - **glBindVertexArray(vertex_array_object);**

OpenGL的程序实例

- 顶点缓冲对象（**Vertex Buffer Objects，VBO**）
  - **VBO**在显存中开辟出的一块内存缓存区，用于存储顶点的各类属性信息
  - 在渲染时，直接从**VBO**中取出顶点的各类属性数据，不需要从**CPU**传输数据，处理效率更高

OpenGL的程序实例

- 顶点缓冲对象（**Vertex Buffer Objects, VBO**）
  - **glBindBuffer(GL_ARRAY_BUFFER, vboId);**
  - **glBufferData(GL_ARRAY_BUFFER , sizeof(vertices), vertices, GL_STATIC_DRAW);**

OpenGL的程序实例

- 顶点缓冲对象（**Vertex Buffer Objects, VBO**）
  - **glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);**
  - **glEnableVertexAttribArray(0);**

OpenGL的程序实例

- 索引缓冲对象（**Element Buffer Object，EBO**）
  - **EBO**跟**VBO**类似，也是在显存中的一块内存缓冲器，只不过**EBO**保存的是顶点的索引。
  - 解决同一个顶点多次重复调用的问题，减少内存空间浪费，提高执行效率。

OpenGL的程序实例

- 着色器管理
  - 着色器定义：**GLSL**语言
  - 分配对象：**glCreateShader**
  - 关联代码：**glShaderSource**
  - 编译：**glCompileShader**
  - 检查编译结果：**glGetShaderiv**

OpenGL的程序实例

- 着色器管理
  - 编译错误：**glGetShaderInfoLog**
  - 创建着色器程序：**glCreateProgram**
  - 关联着色器对象：**glAttachShader**
  - 生产完整角色器：**glLinkProgram**
  - 操作结果：**glGetProgramiv**

OpenGL的程序实例

- 着色器
  - 运行着色器：**glUseProgram**
  - 删除着色器对象：**glDeleteShader**
  - 删除着色器：**glDeleteProgram**

OpenGL的程序实例

- 绘制图形
  - 清除原有的所有图形数据
  - 绑定**VAO**
  - 绘制图形
- 窗口大小改变
  - 指定视区：**glViewport**