# **第1章 入门**

图形编程是基于着色器的——也就是说，有些程序是用诸如C++或Java等标准编程语言编写的，并运行在CPU上；而另一些是用专用的着色器语言编写的，并直接运行在显卡（GPU）上。

## 1.1 语言和库

现代图形编程使用图形库完成，也就是说，程序员编写代码时， 调用一个预先定义的库（或者一系列库）中的函数，由这个库来提供 对底层图形操作的支持。现在有很多图形库，但常见的平台无关图形编程库叫作OpenGL（Open Graphics Library，开放图形库）。本书将会介绍如何在C++中使用OpenGL进行3D图形编程。

在C++中使用OpenGL需要配置多个库。总的来说，你需要以下这些语言和库：

C++开发环境；

OpenGL / GLSL；

窗口管理库；

扩展库；

数学库；

纹理管理库

### 1.1.1 C++

C++是一种通用编程语言，是需要高性能的系统的优秀选择，比如3D图形计算。C++的另一个优点是 OpenGL调用库是基于C语言开发的。在阅读本书时，如果读者使用PC（Windows操作系统），我们推荐使用Microsoft Visual Studio[VS17]；如果在苹果计算机上，我们推荐Xcode [XC18]。附录中也介绍了各个平台下的安装和配置。

### 1.1.2 OpenGL/GLSL

OpenGL的1.0版本出现在1992年，是一种对供应商特定的计算机图形应用编程接口（API）的“开放性”替代。

这本书假定用户的机器有一个支持至少4.3版本OpenGL的显卡。如果你不确定你的GPU支持哪个版本的OpenGL，网上有免费的应用程序GLView可以用来找出答案。

### 1.1.3 窗口管理库GLFW

OpenGL实际上并不是把图像直接绘制到计算机屏幕上，而是渲染到一个帧缓冲区，然后需要由这台机器来负责把帧缓冲区的内容绘制到屏幕上的一个窗口中。有不少库都可以支持这一部分工作。

GLFW是最流行的选择之一，也是我们这本书中选择使用的。它内置了对Windows、macOS、Linux和其他操作系统[GF17]的支持

### 1.1.4 扩展库GLEW

现代版本的OpenGL，比如我们在本书中使用的4以上版本，需要识别GPU上可用的扩展。OpenGL核心中有一些内置的命令用来支持这些，但是为了使用每个现代命令，需要执行很多相当复杂的代码行。在本书中，我们会持续不断地使用这些命令。所以使用一个扩展库来处理这些细节已经成了标准做法，这样能让程序员可以直接使用现代OpenGL命令。常用的是GLEW，它支持各种操作系统，包括 Windows、Macintosh和Linux [GE17]。

### 1.1.5 数学库GLM

3D图形编程大量使用向量和矩阵代数。因此，配合一个支持常见数学计算任务的函数库或者类包，能极大地方便OpenGL的使用。本书中使用的，是OpenGL Mathematics，一般称作GLM。它是一个只有头文件的C++库，兼容 Windows、macOS和Linux [GM17]。

GLM提供与图形概念相关的类和基本数学函数，例如矢量、矩阵和四元数。它还包含各种工具类，用于创建和使用常见的3D图形结构， 例如透视和视角矩阵。

### 1.1.6 纹理管理库SOIL2

从第5章开始，我们将使用图像文件来向我们图形场景中的对象添 加“纹理”。这意味着我们会需要频繁加载这些图像文件到我们的C++/ OpenGL代码中。

本书中使用的纹理图像加载库是SOIL2——SOIL的一个更新的分叉版本。像我们之前选择的库一样，SOIL2兼容各种平台[SO17]。附录中给出了详细的安装和配置说明。

### 1.1.7 可选库

读者可能希望利用很多其他有用的库。例如，在本书中，我们将展示如何从零开始实现一个简单的“OBJ”模型加载器。然而，正如我们将看到的，它没有处理OBJ标准中可用的很多选项。有一些更复杂的现成的OBJ加载器可供选择，比如Assimp和tinyobjloader。在我们的例子中，我们会只用在本书中介绍和实现的简单模型加载器。

## 1.2 安装和配置

[参考之前记录的博客](https://blog.csdn.net/weixin_44848751/article/details/124830818?spm=1001.2014.3001.5501)