# 介绍

# 数学基础

# 第三章 渲染器设计

我们如何才能干净地管理 OpenGL 状态？我们如何确保每个人都在使用 OpenGL 最佳实践？甚至，我们如何同时支持 OpenGL 和 Direct3D？

回答这些问题的第一步是抽象，更具体地说，使用使应用程序的大部分代码与 API 无关的接口抽象底层呈现 API，例如 OpenGL 或 Direct3D。我们将此类接口及其实现称为渲染器。本章介绍 OpenGlobe 中渲染器背后的设计。首先，我们务实地考虑渲染器的动机，然后我们看一下渲染器的主要组件：状态管理、着色器、顶点数据、纹理和帧缓冲区。最后，我们看一个简单的例子，它使用我们的渲染器渲染一个三角形。

如果您有使用渲染器的经验，您可能只想略读本章并继续了解虚拟地球渲染的内容。后面章节中的示例基于我们的渲染器构建，因此需要对它有一定的了解。

本章不是关于 OpenGL 或 Direct3D 的教程，因此您需要了解一个 API 的背景知识。这也不是关于如何将每个 OpenGL 调用包装在面向对象的包装器中的描述。我们所做的不仅仅是包装功能;我们正在提高抽象水平。

我们的渲染器包含相当多的代码。为了保持讨论的重点，我们只在这些页面中包含最重要和最相关的代码片段。有关完整实现，请参阅 OpenGlobe.Renderer 项目中的代码。在本章中，我们将重点介绍公共接口的组织以及其中的设计权衡;我们不关心微小的实施细节。

在本章中，当我们提到 GL 时，我们特指 OpenGL 3.3 核心配置文件。同样，当我们提到 D3D 时，我们指的是 Direct3D