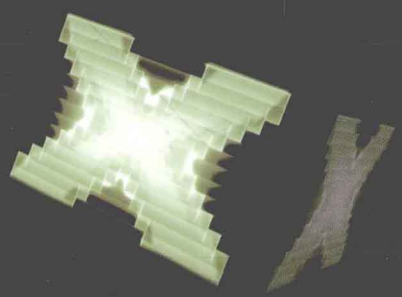
DirectX 11
高级图形开发技术实战

韩元利　王汉东　编著



- 详解DirectX 11的.NET高级开发技术，全面释放GPU编程能力
- 面向Shader Model 5.0 图形硬件编程，深入剖析各类着色器特性
- 基于Windows 8与DXGI 1.2图形框架，彻底突破工程实践瓶颈

科学出版社

DirectX 11 高级图形开发技术实战

韩元利　王汉东　编著

科学出版社

北京

内 容 简 介

本书主要围绕新一代可编程图形管道流水线技术，介绍了 DirectX 图形应用中各个环节的开发知识。从图形硬件的可编程图形管道流水线认识开始，到基于 GPU 命令集与 Shader Model 图形编程接口标准、高级着色语言(HLSL)所构成的着色程序开发知识，全面地介绍了各种不同着色器的功能编程过程，再到以 DXGI、Direct2D、Direct3D 整合的 DirectX 11 图形开发，最后给出了具体的项目实践过程，构成了新一代图形应用开发完备的知识链。全书以 Direct3D 11 的.NET 图形应用开发为主线，基于最新的 Windows 8、DXGI 1.2 高级图形技术，面向具体的工程实践，通过与 Direct3D 9 的开发对比，论述了 Direct3D 11 进行二维图形开发与三维图形应用程序开发的技术与实践过程。

本书内容体系完整，软硬件知识结合，可供从事三维图形程序设计、可视化系统设计、GPU 高性能计算及其他图形应用程序设计的开发人员阅读参考，可作为大中专院校相关专业的教材，也可供游戏开发培训机构作为 Direct3D 相关课程的培训教材。

图书在版编目(CIP)数据

DirectX 11 高级图形开发技术实战 / 韩元利，王汉东编著. —北京：科学出版社，2013.11

ISBN 978-7-03-038857-5

Ⅰ. ①D… Ⅱ. ①韩… ②王… Ⅲ. ①多媒体-软件工程 Ⅳ. ①TP311.56

中国版本图书馆 CIP 数据核字(2013)第 243795 号

责任编辑：张 濮 陈 静 / 责任校对：钟 洋

责任印制：张 倩 / 封面设计：迷底书装

科 学 出 版 社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

骏走印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2013 年 11 月第 一 版 开本：787×1 092 1/16

2013 年 11 月第一次印刷 印张：17 1/4

字数：480 000

定价：68.00 元

(如有印装质量问题，我社负责调换)

前　言

微软公司的 DirectX 9 发布至今已经十多年了，最初的三维图形开发领域呈现 Direct3D 与 OpenGL（Open Graphics Library）平分天下的双强形势，承载了绝大部分三维应用程序的开发。然而近几年，由于强大的微软平台体系支持与商业推动，Direct3D 的发展突飞猛进，全面超越了 OpenGL，成为引领图形学发展的主导力量。

Direct3D 是微软公司提供的用于三维图形开发的支持库，大部分三维图形开发应用与游戏开发都依赖于它的环境，以实现应用软件与图形硬件之间的桥梁作用，从而承载大量复杂的图形处理运算与表达。

Direct3D 9 到 Direct3D 11 的图形编程，在理念、方法与实现上都有很大的转变，这使得基于 Direct3D 9 及其以前版本开发的图形应用程序遇到了更新障碍，Direct3D 图形开发方面相关的书籍也出现了断代的局面，广泛结合图形硬件编程开发成为图形技术发展的重要方向，而这种直接面向硬件的编程使设计者无法继续选择 Direct3D 9 进行程序开发。

绝大部分 Direct3D 图形开发书籍主要着眼于 Direct3D 9 的介绍，固定流水线模式与图形处理知识已不能满足当前三维图形开发的需要。传统的三维图形开发过于注重开发的整体过程，而如今的图形开发（包括二维图形开发）均是结合图形硬件运算的多场所开发，图形建模、图形表达与渲染呈现出独立发展、高效协同完成图形作品加工表达的新趋势。受传统图形开发的影响，没有 Direct3D 11 开发实践经验，是不可能编写出真正的 Direct3D 11 普及教程的。本书通过具体工程实践首次全面地介绍了以 DirectX 11 为代表的最新图形开发技术，从图形硬件的渲染编程到三维图形的整体应用开发，详细而全面地描述了现代计算机图形开发的知识。

最重要的是，DirectX 的图形开发不是越来越复杂，而是越来越简单，因为 DirectX 的发展趋势就是将更多的图形事务交给图形硬件，通过简单的程序开发，就可以创建丰富的、逼真的图形效果。这必然会引导更多的开发人员积极使用这一技术。DirectX 11 发展已有几年时间了，技术发展成熟稳定，相信在不久的将来，该图形技术必将成为业界应用最广泛、最持久和用户群体最庞大的技术体系，希望本书能够引导广大开发人员学习并使用新图形开发技术。

DirectX 11 只能部署在 Windows 7 及以上版本的 Windows 操作系统，本书的知识架构是直接面向 Windows 8 的 DirectX 11.1 和 DXGI 1.2 之上的，突出了可编程 GPU 流水线与传统流水线结构的不同之处；并且面向最新的 Shader Model 5.0 图形硬件开发，及时跟进当前二维、三维一体化表达技术的发展，全面地介绍了各类着色器的功能、开发过程与处理机制。

为了帮助开发人员更好地学习和了解本书的知识架构，全面掌握各章节内容，下面给出本书的内容组织结构图，如图 1 所示。通过对组织结构图的学习，能够将全书各个章节的内容有机地组织起来，形成一个高效而严密的知识体系。开发人员也可以根据组织结构图，对薄弱环节进行重点学习。

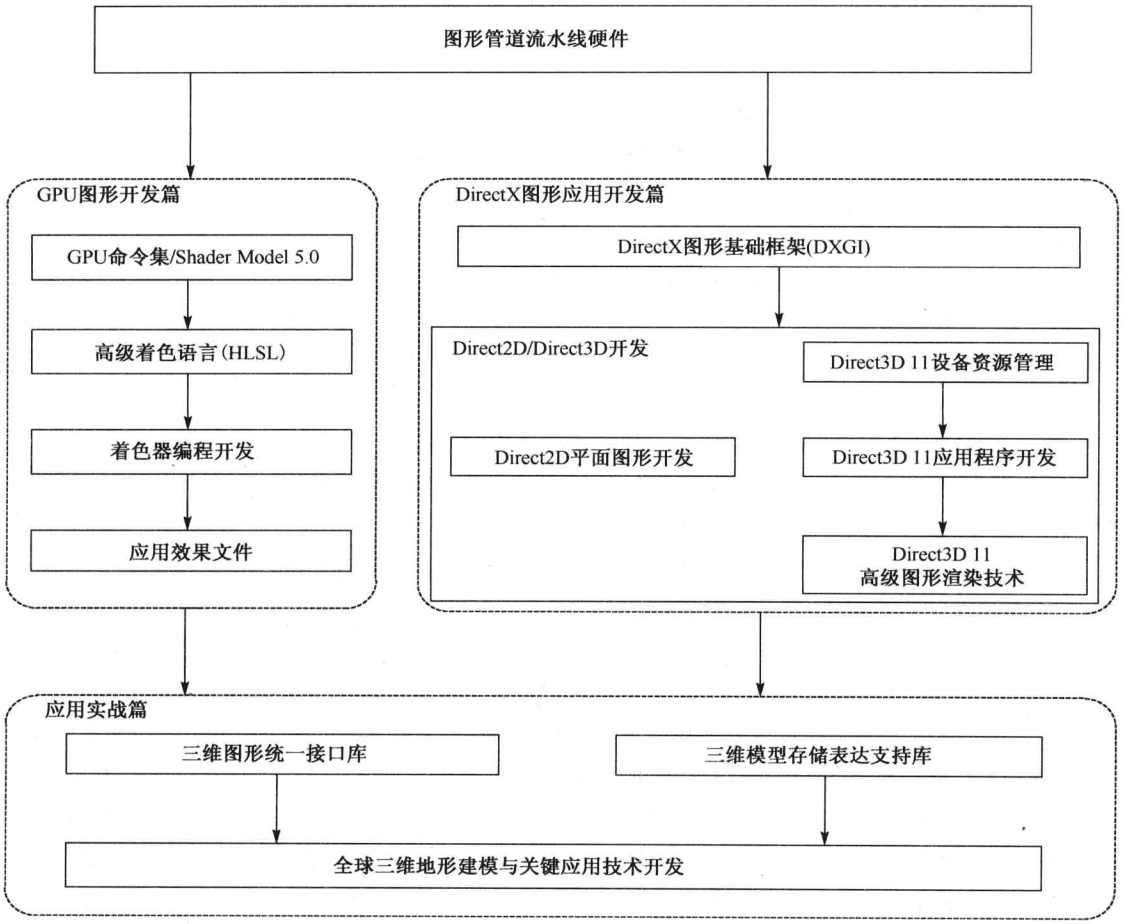


图 1　本书内容组织结构图

结合各章内容，对组织结构图说明如下。

第 1 章简要地介绍了 DirectX 的发展历程、体系构成、针对.NET 的三维图形开发技术路线和三维程序开发过程中的 DirectX 开发调试工具。

对于三维图形学最重要的承载主体，第 2 章介绍了现代图形管道流水线技术，这是本书所有后续内容的基础。无论针对 GPU 的渲染开发编程，还是针对 DirectX 的三维应用开发，什么样的图形管道流水线硬件决定能够开发什么样的着色程序和 DirectX 应用。读者对第 2 章内容的掌握，尤其是对流水线结构与众多专业名词概念的理解，对后续章节内容的理解非常重要。

第 3 章针对新一代先进的 GPU 图形硬件开发，介绍了图形流水线硬件设备各阶段支持的命令集。无论 GPU 的命令集，还是微软公司的 Shader Model，提供的都只是一些没有编程逻辑与语法结构的指令集合。要对 GPU 进行渲染编程，同应用程序开发一样，开发人员也期望有一种与算法语言类似的 GPU 编程语言，这就是着色语言的由来。着色语言种类很多，如 NVIDIA 的 Cg、OpenGL 支持的 GLSL 等。

第4章主要针对DirectX所使用的微软公司的高级着色语言(High Level Shader Language，HLSL)，介绍其图形编程的基础知识。

第 5 章具体介绍图形管道流水线各阶段着色功能的开发实现，包括对当前应用热点技术——GPU 高性能并行通用计算的具体开发介绍，在图形表达之外更全面地揭示了图形硬件的新应用。

第 6 章论述着色程序语言通过效果文件实现加载到图形硬件的调用方法。

第 2～6 章实际上围绕 GPU 的图形编程进行介绍，除了第 6 章中结合 DirectX 的效果调用，其他内容都与 DirectX 无关，对 GPU 编程比较熟悉的开发人员可以跳过相应章节的内容，直接学习 DirectX 的开发内容。

第 7～11 章主要围绕 Direct2D、Direct3D 的应用程序开发进行介绍。绝大部分三维应用程序都会涉及二维图形、文字的绘制，自 Direct2D 推出以来，微软公司不断致力于推动基于 Direct2D/Direct3D 整合的二维、三维一体化绘制技术。因此，了解 Direct2D 的架构及其开发技术，对 Direct3D 高级开发的开发人员来说是非常有必要的。第 7 章介绍了 Direct2D 平面图形开发。第 8 章从总体上介绍了 DirectX 图形架构，它是 Direct3D 各版本及其他图形技术的共同基础，是一个总领式的图形铺垫。第 9 章论述了 Direct3D 11 资源管理。第 10 章论述了使用 SlimDX 开发 Direct3D 11 应用程序的基本流程与方法。

Direct3D 11 具备很多新的图形渲染绘制技术，第 11 章论述了 Direct3D 11 的部分高级图形渲染技术，它的许多概念与实现已经超出了当前的主流配置环境，但可以对开发人员起到引导作用，并促进他们对新技术的关注与学习。

后面的实现部分是在前面两大组织部分的基础上，根据作者多年来的工程开发实践经验，组织编写的一些应用开发实践内容。第 12 章针对大部分三维应用开发过程中硬件兼容性及更新升级带来的维护困难问题，提出了一个完整的解决方案，建立了针对 DirectX 各版本的三维图形统一接口库，实现了多种硬件体系的自动化兼容，以及相对独立的、集中的、更易于维护与更新的三维、二维整合表达方案，这是本书面向应用推出的核心应用实践环节。

第 13 章介绍了三维应用开发中另一个重要的实现环节，内容包括三维模型的导入、导出、渲染表达等，该部分内容建立在 DirectX 统一图形接口库的基础上，实现了独立模型的功能业务封装，达到了简化应用程序开发的目的。

第 14 章具体针对全球三维地理信息系统（Geographic Information System，GIS）平台的开发，运用图形统一接口库进行应用实践开发，并着重论述结合先进的 GPU 编程对海量数据模型的高效实现，对保障三维渲染效率起到了很大的改进作用。同时，针对全球三维地形表达中关键的技术突破进行了论述，帮助开发人员推进全球化应用开发技术的发展，并充分认识新一代 GPU 技术的发展，有助于突破传统难以解决的技术问题。

本书紧紧围绕新一代三维图形开发技术进行介绍，从 GPU 开发、Direct3D 11 开发到应用实践结合开发，体系完整，内容集中，有助于广大爱好三维图形编程的读者进行 Direct3D 11 的.NET 开发。对于了解三维图形学、三维空间数据基础知识的读者，可以引导其进入 Direct3D/2D 的新技术开发领域；对于相关基础知识不太了解的读者，在简单了解相关概念后容易上手。对是否需要掌握 Direct3D 9 的开发编程没有要求，因为 Direct3D 11 与 Direct3D 9 在技术实现上差异相当大。当然了解 Direct3D 9 或以前版本的图形开发，对更好地掌握 Direct3D 11 的开发，理解现代图形学发展趋势会有很大的帮助。

本书由韩元利博士完成初稿撰写，王汉东博士完成校稿，李小宁、陈燕平、薛向阳等参

加了本书部分代码和文稿的编写、调试、整理、资料翻译与校对工作，在此表示衷心的感谢。同时，还要感谢单位领导、同事对本研究课题的支持、关心与帮助，感谢家人在生活上给予的理解与关照。

由于时间仓促，作者水平有限，书中难免存在不足之处，敬请广大读者批评指正。

联系方式：goldenhyl@gmail.com。

作　者

2013 年 3 月 22 日于武汉

专 业 名 词

为了帮助读者更好地理解本书的内容，下面对主要的专业名词按其所属类别作出解释。

1. 基础图形学类

(1)模型(model)：用以呈现的自然几何形体要素，它是自然世界的实体概念。一个窗口可以是一个模型，一个人也可以是一个模型。这里的模型可以是二维的，也可以是三维的，可以是静态的，也可以是动态的。

(2)图元(primitive)：一个模型通常由多个图元对其精细表面进行模拟表达。基本的图元类型包括点图元、线图元、三角形图元、四边形图元等。

(3)顶点数据与索引数据：描述图元的顶点与组成图元顶点的索引组合的数据存储结构。通常为节约模型的数据，各个图元之间的顶点独立出来实现图元间的广泛共享，而仅以顶点的索引组合来描述图元的构成，如第 m、n、q 个顶点构成一个三角形图元，这样就形成了模型的索引数据。图元与顶点是计算机图形管道流水线上最基本的处理单元，而像素是图元光栅化后的表达单位。

(4)拓扑结构(topo architecture)：概念上指一个模型中所有图元的组织结构。具体地讲，是指定模型光栅化前几何形体与存储结构之间的约定关系。常见的拓扑类型有 Trilist、Tristripe、Linelist、Linestripe、Point 等类型。以 Trilist 为例，它表明模型的索引存储中，每 3 个索引值确定一个三角形图元，则指定 n 个三角形图元需要 $3n$ 个索引值。充分运用拓扑结构有助于减少数据量的存储与传送，提高图形处理与运算的效率。

(5)面片(patch)：多种相同类型的图元放在一起构成的几何区间单元，它有一致的拓扑结构，表明多个图元构成面片集合的几何构造形式。一个模型可以由一个或多个几何面片构成。

(6)三角网(TriNet)：一种特殊的限定为三角形图元的几何面片，通常用于对区域地形和复杂模型进行表达。图形建模与空间分析中广泛地应用了三角网的知识，使它独立于面片，更能代表模型的概念。

(7)规则网格(grid)：一种特殊的限定为四边形图元的几何面片，通常用于大型区域地形的动态层级表达。在大多数规则网格模型中，使用规则网格主要是出于模型建构与组织的便利性，最终将一个规则矩形拆分为两个三角形进行表达。

(8)画面(image)：纹理资源的数据形式表达。如果缓冲区资源中存放的是图像数据，也是画面，那么本书中的画面可以是表达呈现出来的窗口内容，也可以是流水线上光栅化后的数据流。

(9)帧(frame)：流水线上画面传送的计量单位，传送或绘制一个画面称为一帧画面，也代指画面。

(10)世界空间(world space)：建立在工程场景和项目级体系之上，具有统一尺度、方位的坐标空间，用于将构成项目、场景等各方面的数据有机地统一到一个整体空间中形成有序的表达。

(11)模型空间(model space)：也称为用户空间，是建立在局部坐标系基础上的空间子系统，便于保持模型对象的独立性需求。

(12)数字地球空间(digital earth space)：一种沿标准球面向外拓展的空间，与大地椭球空间(也称地学空间)相似，属于不同类型的地球空间但是基准面形状不一样。这种空间可以通过极坐标与地心坐标系统进行空间描述。

(13)视空间(view space)：一种以视点为锥顶，由相机视向和视角大小构成的锥体空间，空间内部的模型对视点可见。

(14)画面空间(image space)：也称为像素空间，是指二维平面资源中像素的度量体系。

(15)存储空间(storage space)：用于存储数据资源的场所与位置，从分布位置来看，可以分为主存储器(主存)空间和显示存储器空间，包括存储器、缓冲池、寄存器等多种形式。

2. 硬件开发类

(1)图形设备接口(Graphics Device Interface，GDI)：包括传统的二维图形接口 GDI、GDI+和 Direct2D，泛指应用程序中访问图形硬件设备实现图形图像处理与表达的功能接口库，也包括 Direct3D 等三维图形设备接口，通常是指由操作系统或图形硬件厂方提供的图形访问功能库。

(2)缓冲区(buffer)：在显示存储器上分配的一块用于存储模型顶点数据、索引数据、纹理数据等的存储区，用于存储图形表达的相关处理数据。模型中的图元面片均以顶点数据、索引数据、纹理数据等数据形式存放于缓冲区内，并以资源的形式供图形管道流水线进行处理转换。

(3)图形适配器(adapter)：也叫图形显卡，是计算机硬件系统主要部件之一。

(4)图形管道流水线设备(graphics pipeline device)：简称图形管道设备或图形设备，是图形适配器的一种虚拟对象，直接体现图形适配器的功能，也是访问和调用图形功能的主接口。

(5)图形终端(output)：也称显示设备、显示终端，用于呈现系统输出窗口画面的终端设备，最常规的有显示器、图形打印输出设备等。

(6)图形管道逻辑流水线(logical pipeline)：与图形管道流水线设备相对应，是针对统一渲染架构建立起来的保持图形图像处理基本业务流程不变的逻辑结构。

(7)统一渲染架构(unified shader architecture)：图形在管道逻辑流水线的概念下，打破流水线上功能硬件部署的次序与位置，最大化地实现硬件资源共享，如各种着色器运算部件的统一，能够有效地降低硬件的部署成本。

(8)图形运算器(Graphics Process Unit，GPU)：狭义的图形运算器是指图形硬件中进行着色运算与普通计算的特殊功能部件，与中央处理器(Central Process Unit，CPU)不同的是，它着重于通过集中部署大量的运算单元实现高性能的并行运算，因而适合大批量的诸如图形处理与海量数据的高效并行计算。广义的 GPU 也指代图形管道流水线设备。

(9)流输出端(Stream Out)设备：用于从流水线上截取经过前端功能设备处理后的中间数据到应用程序中，对通用计算等需要利用图形功能部件进行运算的应用非常有用。

(10)流输入端(Input Assembler)设备：与流输出端设备对应，是在应用程序中将模型数据(顶点数据、索引数据)注入图形管道的接口设备，也是图形管道流水线上部署的众多功能性部件中最前沿的功能部件。

(11)着色器(shader)：图形管道上部署的可编程功能部件，用以完成一定的图形处理与运算功能。目前，Direct3D 11 部署的着色器部件有顶点着色器(Vertex Shader，VS)、外壳着色器(Hull Shader，HS)、域着色器(Domain Shader，DS)、几何着色器(Geometry Shader，GS)、像素着色器(Pixel Shader，PS)、计算着色器(Compute Shader，CS)6 种可编程功能部件。

(12)着色程序：由开发人员通过着色语言开发，经编译后运行于着色器内的一段程序代码。

(13)着色语言：也叫渲染语言，开发着色程序的语言，常用的有 HLSL、GLSL 等。

(14)着色效果(Effect)：简称效果，是效果文件中的着色程序经过 GPU 编译后生成渲染命令集合的可执行实例，它由图形管道流水线上的各个着色器加载并执行，从而对图形绘制产生渲染效果。

(15)着色模式(shader model)：用于统一定义图形硬件开发功能的行业标准，同时为着色语言的建立与着色程序的开发提供接口支持。

3. DirectX 概念类

(1)DirectX 基础图形框架(Direct X Graphics Infrastructure，DXGI)：微软公司随着 DirectX 10 推出的一种独立于 Direct3D 与 Direct2D 的、跨应用程序编程接口(Application Programming Interface，API)程序库的、服务于总体图形处理与应用的基础运行环境，其提出的目标主要是为各种图形 API 创建一个统一的、稳定的图形技术体系基础。微软公司提出的整合各种图形 API 的基础图形框架，将先前各种图形 API 的成熟的基础概念、共性功能整合出来形成稳定的基础，便于在多种图形 API 之间实现功能互操作，同时简化各版本特性功能的维护与更新工作。

(2)资源视图(resource view)：对缓冲区数据进行解析与读/写访问的一种手段与途径，也可以认为是对缓冲区进行解析访问的一种工具对象。存放在缓冲器中的数据(无论什么类型)都是以二进制字节统一存放的。当用户需要读取这些内容的时候，就需要指定数据的解析形式。同一缓冲区可以有不同的资源视图，以进行不同的数据读/写访问。

(3)表面接口(surface interface)：DXGI 中访问 GPU 资源的一种公共接口与工具，主要用于 GPU 资源与 CPU 资源的交换读取与回写。

(4)渲染目标视图(render target view)：它首先是一种资源视图，在显示存储器中占据一块存储区域，存储的内容是画面。它可以认为是应用程序中承载绘制行为的画布，是图形设备暴露给应用程序直接访问绘制成果画面的一种途径。

(5)深度模板视图(depth stencil view)：它与渲染目标视图一样，只不过存储的不是画面数据，而是画面的深度数据，对应于传统的深度缓冲区的功能定义。

(6)顶点布局(vertex layout)结构：流水线上定义顶点输入数据分布构成的结构体，相当于指定流水线众多股数据中每一股流经数据的类型与用途定义。

(7)效果文件(effect file)：管理与组织着色程序的源代码文件，便于在应用程序中编译后加载到图形管道流水线设备中，并使用着色程序进行图形渲染。

(8)交换链路(swap chain)：是将图形管道流水线硬件绘制出来的画面以高效的方法送到图形显示终端，是衔接流水线末端与显示终端之间的逻辑功能设施。

(9)后台缓冲区(back buffer)：建立在交换链路末端与图形呈现终端之间的存放从图形管道流水线处理得到的图形画面的缓冲区队列。

(10) 翻转传送模式(flip mode)：通过与桌面窗口管理器共享后台缓冲区画面资源，实现画面快速呈现传送的一种模式。

(11) 复制传送模式(bitblt mode)：其机制是将后台缓冲区中的画面资源复制到桌面窗口管理器中，实现画面窗口表达的一种模式。

(12) 功能级别(function level)：Direct3D 中用于衡量、区分图形硬件设备性能的分类标准，通常每一版本的 Direct3D 会根据升级次数推出两三个功能级别的定义，每个级别具体指定了它具备的图形运算功能列表，设备创建时，可以最大限度地利用功能层级创建适合该硬件功能的图形设备，供应用程序充分使用它的性能。

(13) 命令列表(command list)：延迟渲染技术中用于记录渲染行为命令与渲染状态的列表，在延迟渲染时，通过执行命令列表实现渲染过程的回放。

(14) 图形管道子设备(child device)：部署在图形管道流水线上的承担一定图形处理与数据接入、输出、转换及状态设置的功能部件设备，也称为功能子设备，包括各种着色器、流输入/输出端和各种状态设置设备。

(15) 共享表面(shared surface)：在多个图形 API 和窗口桌面管理器之间实现画面资源共享的一种方式，为功能库之间的画面资源共享提供了一种有效途径。

(16) 设备环境(device context)：将设备的运行环境、图形绘制功能从具体的设备管理中分割出来的一种独立的配备环境与功能绘制接口。

(17) 立即设备环境(immediate device context)：在场景绘制与渲染过程中，实现立即绘制行为的一种设备环境，图形管道中只能存在一个立即设备环境。

(18) 延迟设备环境(deferred device context)：为延迟渲染技术提供的一种设备环境，可以根据需要创建多个延迟设备环境，并由命令列表中的状态命令进行延迟设备环境的配置。

4. 图形处理类

(1) 绘制(draw)：应用程序中的图形模型经图形管道流水线向帧缓冲区中进行表达的业务过程，通常调用设备的 draw 等方法实现。画图的动作行为决定要画什么形状的图形，确立最终实体画面的“模”，通常把这种动作行为称为绘制。

(2) 渲染(render)：画图时的设备环境决定了画出来的物体的质量、效果，确立了最终实体画面的“样”，通常把这种效果表达称为渲染。

(3) 呈现(present)：帧缓冲区中的画面数据提交到显示终端上的过程，通过交换链路的 present 方法来调用。

(4) 曲面细分建模(Tessellation)：一种对曲面面片进行更多图元细分优化表达的建模技术，通过更多的图元来拟合曲面，能够更逼真细致地表达复杂模型表面。

(5) 曲面细分器(Tessellator)：DirectX 11 中图形管道上部署的一种能够自动进行曲面细分优化建模的功能部件，它与外壳着色器、域着色器协同工作，完成曲面细分建模的处理。

(6) 延迟渲染(deferred shading)技术：通俗地讲，就是将原来需要放在图形管道流水线前端的图形处理环节尽量地放在后端，针对窗口画面有限区域进行局部渲染，避免了前期大量的顶点运算、图元处理等中间成果被窗口裁剪掉而产生的无效运算负荷，从而在不影响渲染效率的情况下，便于开发更复杂的高级渲染效果或直接提高场景的渲染效率。

(7) 多过程渲染(multi-pass render)：也称为多通道渲染，是指对同一画面的多道工序的渲染合成，以更好地模拟艺术家的绘画效果。

(8) 多线程渲染(multi-thread render)：Direct3D 11 中增加的高级渲染特性在延迟渲染环境中，通过对整体场景绘制行为的分割形成多个行为命令子集，并通过多个线程实现各个行为命令子集的同步并行渲染。

5. 并行计算类

(1) 核(kernel)：一段需要被并行执行的程序代码，执行一种运算或数据处理，也就是计算着色器的实现代码。

(2) 进程(process)：并行计算中的进程是指核的一次执行运算行为。

(3) 核心(core)：硬件上用于执行核运算和进程的场所，也就是执行核代码的运算器设施。

(4) 异构计算(heterogeneous computing)：对所有能够使用的计算核心进行集中分配调用。例如，把 CPU 与 GPU 上的计算核心放在一起，供并行算法分配相应的进程。

目　录

第 1 章　DirectX 概况

DirectX（Direct Extension）是微软公司创建的多媒体编程接口，采用 C++语言实现，遵循 COM 结构标准。DirectX 被广泛应用于 Microsoft Windows、Microsoft Xbox、Microsoft Xbox 360 等电子游戏开发。目前，DirectX 的最新版本为 DirectX 11，创建在最新的 Windows 7/8 和 Windows Vista 操作系统上。

本章简要介绍 DirectX 的发展历史、DirectX 的主要构成，明晰 DirectX 与 Direct3D、Direct2D 等图形开发功能库之间的关系，同时针对.NET 下的图形开发技术路线进行了研究对比，确立了基于 SlimDX 进行新一代图形应用开发的技术优势。

1.1　DirectX 的版本发展

在 DirectX 出现之前，游戏开发人员都在努力解决因硬件不兼容而产生的问题，由于存在硬件配置的差别，每个人不可能都能玩相同的游戏，这迫切需要图形表达的标准化，于是微软公司推出了 Windows Game SDK for Windows 95，这就是 DirectX 1。DirectX 给游戏开发商提供了一套单一的应用程序接口（API），从而保证了图形程序在不同硬件配置情况下的兼容性。此后各版本的 DirectX 相继发布，在 Windows 下运行的游戏种类大量增加，目前，这种发展态势依旧非常迅猛。

1.1.1　DirectX 版本的历史

1994 年末，微软公司推出了第二代图形多媒体交互操作系统 Windows 95。由于之前大部分开发人员更依赖于在 DOS 系统下编写程序，所以决定新操作系统的最终价值在于其能否展现图形应用程序的优势。如果不能提供灵活统一的图形多媒体应用程序开发模式，那么新的操作系统获得成功的可能性就不大。

DOS 系统能够通过中断技术直接访问显卡、键盘、鼠标、声卡和其他系统硬件设备。而 Windows 95 系统出于保护存储系统的目的，限制了其对硬件的直接访问，取而代之的是一套更为规范、标准的访问方法，因而需要一种接口使开发人员在 Windows 95 中也能编写高效的图形和多媒体应用程序。DirectX 作为微软公司的初步解决方案被独立于操作系统提了出来，用于解决应用程序与相关多媒体硬件之间的通信与调用接口问题。

DirectX 的第一个版本 Windows Games SDK 发布于 1995 年 9 月，它作为 Windows API 的一部分，用以替换 Windows 3.1 中的设备控制接口（Device Control Interface，DCI）和 Windows 图像 API（Windows Graphics API，WinGAPI）。在 DirectX 出现之前，微软已将 OpenGL 包括在 Windows NT 系统中，只是 OpenGL 对硬件要求严苛，限制了一些工程师和 CAD 用户的开发能力。在这种背景下，Direct3D 作为 OpenGL 的代替品被提出来并加入 DirectX 体系中。随着硬件技术的发展，OpenGL 逐渐成为行业标准，从而导致 Direct3D 的支持者和 OpenGL