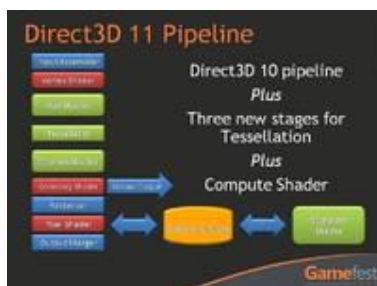


# DirectX 11 新技术预览

## 1, Direct3D11 渲染管线:



Direct3D11 管线

看上去, DirectX 11 比 DirectX 10 更酷。DirectX 11 的很多提升意味着更高的特性性能, 而这些特性很少能在 DX10 中看到。DirectX 11 和 DirectX 10 两者最大的不同之处在于管线, 可以说 DirectX 11 的渲染管线标志着绘图硬件以及软件功能革命性一步。DirectX 11 加入了对 Tessellation (镶嵌) 的支持。Tessellation 由外壳着色器 (Hull Shader)、镶嵌单元 (tessellator) 以及域着色器 (Domain Shader) 组成。同时还加入了计算着色器 (Compute Shader), 计算着色器与 DX10 中引入的 GS 不同, 它并不是渲染管线的一部分, CS 也是 DirectX 11 的重要改进之一, 可以很大程度上协助开发人员弥补现实与虚幻之间的差别。

## 2, Tessellation 镶嵌技术:

在此之前, 关于 DirectX 11 的报道可谓铺天盖地。事实上, 自 R600 发布时, DirectX 11 这个字眼才开始越来越多的出现在网络上。尽管 R6xx 和 R7xx 硬件都具有 tessellator 单元, 但是由于 tessellator 属于专有实现方案 (proprietary implementation), 所以 R6xx 和 R7xx 硬件是不能直接兼容 DirectX 11, 更何况 DirectX 11 采用了极其精密老练的设置过程。事实上, DX11 tessellator 单元本身不具备可编程性, DX11 向 tessellator (TS) 输入或者从中输出的过程是通过两个传统的管线阶段完成的: Hull Shader (HS, 外壳着色器) 和 Domain Shader (DS, 域着色器)。

tessellator 可以把一些较大的图元 (primitive) 分成很多更小的图元, 并将这些小图元组合到一起, 形成一种有序的几何图形, 这种几何图形更复杂, 当然也更接近现实。这个过程也被称作细分曲面 (Subdivision Surfaces)。举例来说, tessellator 可以让一个立方体, 通过处理看起来像是个球形, 这样的话无疑节省了空间。此外, 图形的质量、性能以及可控性也达到了一定的促进。

Hull Shader 负责接收一种由全四边形网格 (quad mash) 计算得到的图元数据 (称作 patches), 并计算控制点 (control points) 的各种变换以及输入的图元各个边的镶嵌配置 (tessellation factors), 从而进行镶嵌。其中 Control points 用来定义想要得到的图形 (比如说一个曲面或者其他) 的图形参数。如果您经常用 Photoshop 绘图软件的话, 不妨把 Control points 理解为 PS 的钢笔工具: 用平面代替线的贝塞尔曲线功能。Hull Shader 采用 control points 来决定如何安排 tessellator 处理数

据，利用 Tessellator 生成大批量的新的图元，然后将这些图元以及控制点传送给 Domain Shader，Domain Shader 将这些数据计算转换成 3D 处理中的顶点，最后 GPU 生成曲线以及多边形。

### 3, 多线程的支持:



DX11 多线程处理

由于 DX11 所新增的特性甚至可以应用到 DX10 硬件中，所以我们对于 DX11 的快速应用都非常期待和乐观。DX11 特性还包括很重要一点：支持多线程（multi-threading）。没错，无论是 DX10 还是 DX11，所有的色彩信息最终都将被光栅化并显示在电脑显示屏上（无论是通过线性的方式还是同步的），但是 DX11 新增了对多线程技术的支持，得益于此，应用程序可以同步创造有用资源或者管理状态，并从所有专用线程中发送提取命令，这样做无疑效率更高。DX11 的这种多线程技术可能并不能加速绘图的子系统（特别是当我们的 GPU 资源受限时），但是这样却可以提升线程启动游戏的效率，并且可以利用台式 CPU 核心数量不断提高所带来的潜力。

对于场景中的人像和三个镜像，DX11 会启动四个单独线程进行并行处理，效率自然要比现在依次进行的做法高很多。

搭载 8 颗以及 16 颗逻辑核心的 CPU 系统已经离我们越来越近，现在游戏开发商们也该赶紧行动起来，是时候解决有些游戏在双核心系统中运行缓慢的问题了。但是开发一款能够很大程度上促进双核以上系统普及的游戏，所能够获得的利润以及需要的付出目前来讲还很不乐观，所以这一进程进展缓慢。对于大多数游戏而言，充分利用四核心以及超过四核心的多线程优势还非常困难。尽管如此，通过多线程技术让简单的平行运算资源产生并显示出来，确实为采用平行运算代码的游戏提供了走红的机会，这些游戏代码也可以以单线程编码的方式存在。由于 DX11 系统中并不是采用一条线程处理所有 DX state change 以及 draw call（或者说大量同步线程共同负责某一任务）的方式，所以游戏开发者可以很自然的创造出线程处理某个场景的某一类或者某一群的客体对象，并为将来所有客体对象或者实体为各自的线程处理打下基础（如果逻辑核心最终达到数百颗之后，这种线程处理方式对于提取硬件性能尤为重要）。

此外，DX10 硬件也能够运行 DX11 游戏时支持多线程，微软的这一计划相当令人兴奋，不过值得一提的是，AMD 以及 NVIDIA 必须为各自的 DX10 硬件开发出相应的驱动软件才能达到这一效果（因为如果没有相应的驱动支持的话，DX10 硬件即

便可以运行 DX11 游戏，对于玩家而言并不会看到真正应有的效果）。当然了，我们希望 NVIDIA，特别是 AMD（因为他同时也是一家可以生产多核心 CPU 的厂商）能够对此感兴趣。而且，如果 A/N 这么做到话，无疑会为游戏开发商们开发 DX11 游戏提供诱因，即便是 A/N 的 DX11 硬件还在襁褓之中。

#### 4, 计算着色器 Compute Shader:

很多游戏开发者都对 DX11 新增的 Compute Shader（通常简称为 CS）特性啧啧称赞。CS 的这一渲染管线能够进行更多的通用目的运算。我们既能在某种可以用来被执行数据的操作中看到这种特性，又能在某种可以用来操作的数据中看到这种特性。

在 DirectX11 以及 CS 的帮助下，游戏开发者便可以使用更为复杂的数据结构，并在这些数据结构中运行更多的通用算法。与其他完整的可编程的 DX10 和 DX11 管线阶段一样，CS 将会共享一套物质资源（也就是着色处理器）。

相应的硬件需要在运行 CS 代码时更灵活些，这些 CS 代码必须支持随机读写、不规则列阵（而不是简单的流体或者固定大小的 2D 列阵）、多重输出、可根据程序员的需要直接调用个别或多线程的应用、32k 大小的共享寄存空间和线程组管理系统、原子数据指令集、同步建构以及可执行无序 IO 运算的能力。

与此同时，CS 也将会随之失去一些特性。因为单个线程已经不再被看成是一个像素，所以线程将会丧失几何集合功能。这就意味着，尽管 CS 程序依然可以利用纹理取样功能，但是自动三线 LOD 过滤计算将会丧失自动功能（LOD 必须被指定）。此外，一些并不重要的普通数据的深度剔除（depth culling）、反锯齿（anti-aliasing）、alpha 混合（alpha blending）以及其他运算不能在一个 CS 程序中被执行。

除了某些特殊应用的渲染，游戏开发者可能同时也希望做一些诸如 IK(inverse kinematics, 反向运动学)、物理、人工智能以及其他在 GPU 上执行的传统的 CPU 任务之类的运算。用 CS 算法在 GPU 上执行这些数据意味着这些数据将会更快的被渲染，而且一些算法可能在 GPU 上的执行速度更快。如果某些总是产生同样结果的算法既可以出现在 CPU 上又可以出现在 GPU 上的话，诸如 AI 以及物理等运算甚至可以同时在 CPU 和 GPU 上运行（这种运算实际上也可以代替带宽）。

即便是这些运算代码在相同的硬件（CPU 或者 GPU）上运行，PS 以及 CS 代码的执行也是两个截然不同的过程，这主要取决于被执行的算法。有趣的是，暴露数据以及柱状数据经常被用作 HDR 渲染。用 PS 代码计算这些数据的话就需要几条通道和几种技巧，以便提取所有像素，从而集中或者平分这些数据。尽管共享数据将会或多或少的减缓处理速度，但是共享数据的方式要比在多通道中计算速度更快，而且这样可以使 CS 成为这些算法的理想处理阶段。

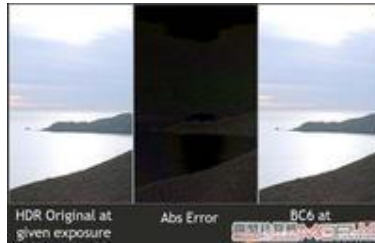
#### 5, Shader Model 5.0:

DirectX 10 的 Shader Model 4.0（Shader Model 以下简称“SM”）带来了整数运算和位运算的功能，DirectX 10.1 的 SM 4.1 加入了对 MSAA 的直接采样和控制。而 DirectX 11 包含的 SM 5.0，采用面向对象的概念，并且完全可以支持双精度数据。随着 SM 5.0 的发布，微软也会将 HLSL 语言更新至最新版本，其中包含了诸如动态

着色、动态分支和更多的对象等。总之，面向专业开发人员的 SM 5.0，依旧是以降低编程的难度和复杂为目的。

为了解决 Shader 灵活性与弹性不足的问题，微软在 HLSL5.0 中带来解决之道。HLSL5.0 提出 shader 子程序的概念，即允许程序员将各种小段、简单或为个别需要而特制的 shader 程序链接起来，再根据实际需要动态调用，这样既能够提高硬件兼容性，同时减少“巨型 shader”对寄存器空间的占用，有效提升性能。

#### 6, 改进的纹理压缩:



BC6 纹理压缩

精细的纹理对视觉效果的增益是显而易见的。目前的 3D 游戏越来越倾向于使用更大、更为精细的纹理，但是过大的纹理严重占用显存和带宽。由于目前纹理压缩仍然不支持 HDR 图像，因此 DirectX 11 提出了更为出色的纹理压缩算法——BC6 和 BC7。BC6 是为 HDR 图像设计的压缩算法，压缩比为 6 : 1；而 BC7 是为低动态范围纹理设计的压缩模式，压缩比为 3 : 1。两种压缩算法在高压比下画质损失更少，效果更出色。

这是一幅对 HDR 文件的压缩示意图，BC6 的压缩相对于原图来说，仅仅损失了极小的画质，却获得了非常出色的效果。

纹理质量对画面效果起着至关重要的作用。比如我们运行 3D 游戏时，画面内同样一个物体，观察距离较远时，纹理锐利而清晰，但当你拉近视角，近距离细看时，纹理就非常粗糙了。更不用说在某些游戏中还有类似放大镜、望远镜等道具，启用这些道具后，只能看到更为粗糙和不真实的纹理。出现这种问题，一方面是纹理压缩率损失严重，细腻的纹理压缩存放后，损失大量细节；另一方面是大纹理难以保证保证游戏运行速度和软件体积，如果在游戏中大面积采用分辨率高达 4000dpi 的纹理贴图，那么显卡的运算资源和显存容量很快就会告罄。因此，DirectX 11 最快速和最直观的改变就是再次改进了纹理的压缩算法，将纹理体积和纹理质量控制在一个相当优秀的范围之内。