16 实例化与视椎体剔除

实例化技术常用于对场景中同一对象反复绘制多次的情形。视椎体剔除技术则是通过简单的测试将位于视椎体外的整组三角形从后续的处理流程中剔除出去。

16.1 硬件实例化

实例化技术常用于同一对象在场景中被绘制多次的情形，而每次绘制时，该物体的位置，朝向，缩放大小，材质乃至纹理可能都各不相同。

要令每个实例都各自维护一套顶点数据与索引数据将极大地耗费系统资源。因此，我们以存储一份相对于其局部空间的几何体副本的方法来加以取代。这样一来，在多次绘制同一对象的过程中，每次只要按具体需求使用不同的世界矩阵与材质即可。

Direct3D实例化API使我们可以通过一次绘制调用构造出一个对象的多个实例。再者，有了动态索引的辅助，实例化技术将比Direct3D11时期更具灵活性。

16.1.1 绘制实例数据

在前面各章的演示程序中，我们其实一直都在绘制实例数据。然而，实例的数量却为1，即DrawIndexedInstanced方法的第二个参数。

照此方法一次性绘制出10个对象仍不满足我们的需求，因为这些物体将拥有相同的材质与纹理，且处于相同的位置。

16.1.2 实例数据

在创建输入布局时，可以通过枚举项D3D12\_INPUT\_CLASSIFICATION\_PER\_INSTANCE\_DATA替代D3D12\_INPUT\_CLASSIFICATION\_PER\_VERTEX\_DATA来指定输入的数据为逐实例数据流，而非逐顶点数据流。随后，再将第二个顶点缓冲区与含有实例数据的输入流相绑定。Direct3D 12仍然支持这种向流水线传递实例数据的方式，但是我们要另择一种更为现代化的方法。

这种所谓的现代方法就是为所有实例都创建一个存有其实例数据的结构化缓冲区。例如，若要将某个对象实例化100次，就应当创建一个具有100个实例数据元素的结构化缓冲区。接着把此结构化缓冲区资源绑定到渲染流水线上，并根据要绘制的实例在顶点着色器中索引相应的数据。那么怎样才能在顶点着色器中确定要绘制的实例呢？为此，Direct3D提供了系统值标识符SV\_InstanceID，可供用户在顶点着色器中方便实现上述目的。

(例子)

16.1.3 创建实例缓冲区

16.2 包围体与视椎体

为了实现视椎体剔除，我们要熟知视椎体与各种包围体的数学描述。包围体即近乎于目标物体体积的基本几何对象。

16.2.1 DirectXMath碰撞检测库

DirectXCollision.h工具库，它是DirectXMath库的一个组成部分。此库提供了一份常见几何图元相交测试的快速实现，例如有射线与三角形相交检测，射线与包围盒相交检测，盒盒相交检测，盒与平面相交检测，盒与视椎体相交检测以及球体与视椎体相交检测等。

16.2.2 包围盒

网格的轴对齐包围盒是一种将目标网格紧密包围，且各面皆平行于坐标主轴的长方体。我们可以通过最小点与最大点来描述AABB。或者以另外一种方式来表示AABB：将盒的中心记作c，扩展向量记为e，后者存储的是由包围盒中心沿坐标轴至各盒面的距离。

(例子)

在某坐标系中的轴对齐包围盒，却没有与其他不同坐标系中的坐标轴相对齐。特别是在局部空间计算出的目标网格的AABB，需要将它进行变换才能得到世界空间的定向包围盒(OBB)。但在实际工作的过程中，我们通常总是先将网格变换到局部空间里，再以局部空间内的轴对齐包围盒进行碰撞检测。还有一种方法，即放弃轴对齐包围盒，仅采用定向包围盒。此时，我们只需保存好定向包围盒相对于世界空间的朝向即可。(结构体表示)

通过一个给定的点集并借助DirectX碰撞检测库中的下列静态成员函数，我们就能构建出所需的AABB与OBB(例子见书本)。

16.2.3 包围球

网格的包围是一种紧密围绕目标网格的球体。我们可以通过包围球的球心与半径来描述它。一种计算网格包围球的方法是先计算其AABB。接下来再求取AABB的中心，以此作为该包围球的球心。包围球的半径事实上是球心c至网格上的任意顶点p之间的最大距离。

假设我们已经计算出了位于局部空间中网格的包围球。经世界变换之后，由于缩放操作的影响，包围球可能已不再紧密围绕与目标网格。这样一来，我们还要对其半径在此进行相应的缩放处理。为了弥补世界变换过程中非等比缩放所带来的影响，我们一定要将包围球的半径按最大缩放量进行缩放。

16.2.4 视椎体

可以在数学上用左右顶底远近这6个相交的平面来指定视椎体。

16.2.4.1 构建视椎体的众平面

构造视椎体各平面的一种简便方法是：在观察空间中，采取以原点为中心，沿z轴正方向俯瞰视椎体的标准形式。此时，可根据在Z轴上至原点的距离来方便地确定近平面与远平面，而左平面与右平面，顶平面与底平面这两组平面既对称又经过原点。因此，我们在表达观察空间中的视椎体时，就不必存储所有的平面方程，只需简单地记录顶，底，左，右4个平面的斜率，以及近平面与远平面在Z轴上至原点的距离即可。(结构体代码)

(例子)

16.2.4.2 视椎体与球体的相交检测

如果一个球体完全位于视椎体中则记作相交，这是因为我们将视椎体看作是一种体积，而非边界。由于视椎体是由6个内向平面围成的空间范围，所以它与球体相交的测试过程可以按下列方式执行：如果存在一视椎体平面L，且球体位于L的负半空间之内，那么我们就判定该球体完全位于视椎体之外。若不存在这样的平面，则此球体与视椎体相交。

因此，视椎体与球体的相交检测就化简为球体与平面的6次相交检测。通过球心到平面的带符号距离可以判断球体与平面相交，在平面的前侧或者后侧。BoundingFrustum类提供了下列成员Contains来测试球体与视椎体是否相交。

16.2.4.3 视椎体与轴对齐包围盒的相交检测

视椎体与AABB的相交检测和视椎体与球体相交检测所采用的的策略相同。如果存在一视椎体平面L，且有包围盒位于L的负半空间，就可推断此盒完全在视椎体之外，若不存在这一平面，我们则判定相交。

(检测代码)

16.3 视椎体剔除

硬件会在裁剪阶段自动丢弃位于视椎体以外的三角形。但是，当我们拥有数以百万计的三角形时，仍需先通过绘制调用将它们提交至渲染流水线，这会产生API开销，而后再传至顶点着色器，很可能还要经过曲面细分阶段以及几何着色器，直到这些三角形进入裁剪环节才能执行丢弃处理。很明显，该流程的效率是及其低下的。

视椎体裁剪的思路是：利用应用程序代码，在高于以三角形为基本单元的层级中，按组剔除三角形。先来构建包围体，包围盒或包围球都可以，用它们来包围场景中的每一个物体。如果包围体与视椎体不相交，就无须将对应的物体交由Direct3D绘制。这样一来，利用不大的CPU测试即可节省GPU资源，使它不必在不可见的几何图形上浪费计算时间。

(例子程序)