**directx3d渲染管线说明**

2016-07-31 09:28:12 [zw\_James](https://me.csdn.net/guiiu) 阅读数 1309更多

分类专栏： [Directx](https://blog.csdn.net/guiiu/article/category/6326531)

D3D渲染管线的意义是什么呢？说白了就是将3D图像转换成2D图像到屏幕的过程。

       那如何操作呢？这便是咱接下来需要解决的！

       此篇理论东西多点，随着下一章节的学习会好点，所以，大家伙就跟我一块重温学习下吧。

        在基于D3D的图元绘制均是以三角形网格的搭建为基础的，这个过程称之为建模。那如何在程序中实现一个三角形的绘制呢？你要记得一个关键点就是顺时针和逆时针的顺序，我个人比较喜欢顺时针。下边会讲解顺时针和逆时针的区别！本章的重点是讲述D3D的固定渲染管线，所以涉及的其他知识点，会在接下来的学习中，补充！

         一、顶点格式 创建一个顶点，并不是单纯数学意义上的只包含位置就完事了，D3D为我 们扩展了很多，咱举个例子，每个顶点，都有位置Postion、法线Noraml、纹 理TexTure1、颜色Colour等。那拥有这么多的附加属性，我们怎么弄呢？还记 得我们在C++中学过的结构体吗，对，就是它，用它来实现定义。举个例子：

             Struct Vertex

                                {

                                  float x, y , z; //位置

                                    float nx, ny; //法线

                                    float \_u, \_v; //纹理

                                    DWORD Colour; //颜色

                                }

            这里需要一个注意的地方，就是当你再定义灵活顶点格式(FVF)的时候，顺 序应与你定义的结构体的顺序一致。例如：

        #define FVF\_VERTEX(D3DFVF\_XYZ | D3DFVF\_NORML | D3DFVF\_TEX1 | D3DFVF\_DIFFUSE)

         二、三角形的组织 这里先不说了，到下一章实战操作的时候再整！记得顺序、普通组织、 索引组织这几个概念。

        三、简要介绍下摄像机 摄像机的作用就是确定哪些可见的3D物体转换成2D图形。 可视体：由远近裁剪面组成的平截台体。只显示在可视体内部的图形，其他的 一律剔除，这个过程叫做裁剪。 投影窗口：是这个可视体内的3D几何图形投影生成的3D物体的2D图像的2D区域。 详细的说明会专门拿出一章来学习D3D的摄像机。

        四、固定渲染管线<本章的核心，前面仅需了解即可> 在我们的图形几何学上有了3D场景及摄像机，最后我们还需要将它们转换成2D 图形来显示在屏幕上。这就是固定渲染管线所作的工作。 实际上D3D的渲染管线大部分工作都是在做图形变换。D3提供了一个简便的函数 g\_pDevice->SetTransform(变换类型，变换矩阵)。 下边我们一起回顾或者学习一下D3D的渲染流水线的理论细节：

               1、本地空间 说白了就是建模空间(美工建模，或者你在本地组织三角形的组织方式)

                2、世界空间 通过平移旋转缩放将那些本地空间的物体转换成世界空间内的物体，以 实现场景的组织。D3D提供了一个位置变换的函数D3DXMaterxTranslation； 旋转函数D3DXMaterxRoationX/Y/Z/Axis;缩放函数D3DXMaterxScalling。具体 请你参阅DXAPI。

               3、视图空间(即摄像机空间) 世界空间中的几何图和摄像机的位置是相对于世界坐标而定义的。但是有 一个缺陷是如果摄像机的位置在世界空间中的位置不是确定的或者死板的，那 通过该摄像机去观察物体那是非常难于控制和低效的。在这种尴尬局面下，诞 生了视图空间这个概念！我们这样操作摄像机，首先将摄像机移至世界空间的 原点，然后旋转摄像机，让它的正方向与世界空间的Z方向一致。这样当你再移 动或者旋转摄像机的时候，世界空间中的几何图就会随着摄像机的变化而做相 同的变化。使其变得顺其自然，而不是被动的啦！ D3D同样为我们提供了一个简便的函数D3DXMatriLookAtLH。

              4、背面拣选 D3D通过背面拣选方式剔除(即删除多余的处理过程)无用的背面多边形，来 提高程序的执行效率。 D3D默认的顶点是以顺时针组织的多边形为该多边形的正面。 D3D同样给我们提供了一个函数来做这项工作： g\_Device->SetRendState(D3DRS\_CULLMODE，Value)。默认的是D3DCULL\_CCW。

              5、光照 光照的定义是在世界空间中定义的，并不是在视图空间定义的。但是视图空间 将这些点弄的非常真实。

              6、裁剪 超出凭借台体的部分几何图将被无情裁剪，借此提高程序执行的效率跟性能。

               7、投影 投影的作用就是将3D场景转换成2D图像，然后转到投影窗口上去。 这种从n维到n-1维的过程就叫投影。游戏开发中用到的投影是透视投影。 因为透视投影可以使远离摄像机的图形在屏幕上显示的时候也变小，这使得 我们的效果更加逼真的转换成2D图像。 D3D同样也给我们一个简便的投影变换矩阵：D3DXMaterxPerspectiveFovLH。

              8、视口变换 视口变换就是将投影窗口变换为屏幕上一个矩形区域的可靠变换，这个矩形区域 就是所谓的视口。 D3D中有一个结构体帮助我们设定视口的相关属性D3DVIEWPORT9结构；并且D3D还 给我们提供了另外一个重要的函数g\_pDevice->SetViewPort(D3DVIEWPORT结构的一个 变量地址)。 对于视口变换的矩阵的相关重要知识点下节进行重要分析。

              9、光珊化处理 经过视口变换这些个图像就到屏幕上去了。光栅化所处理的是计算需要显示的每个 三角形中每个点的像素值。它总是应该通过硬件来进行处理，因为它太过于繁琐了如果 微软自己编码处理的话。 在计算完了像素值之后呢，它就将这个成形的2D图像凸现出来啦！

        说了这么多，我感觉最后需要总结一下子，你觉得呢？ 本章重点阐述了D3D的固定渲染流水线。涉及9小点，那你如何记忆呢？ 还是之前说的D3D初始化那个记忆方法： 分为3个流程：

设计 --- 生产 --- 显示

设计：本地 ---> 世界 -->视图空间(摄像机空间) --> 拣选 -->光照 ---> 裁剪(超出平截台体的部分)

生产：按照以上设计的规格进入生产流程 --> 投影(将设计中的3D物体转换成现实的2D图像，并转到投影上去) --->视口变换(将2D图像映射到屏幕的某个区域内，给他提供可供显示的空间)

显示： 光栅化处理

批注： 光栅化做了什么什么事情: 光栅化就是把矢量的图形像素化,比如说3个顶点组成的一个三角形把它转化为屏幕上的像素的过程。 简单的说，就是把要显示的东西，显示在显示器上过程(给图像加颜色实际上)。