1. cpu从硬盘中读取数据到内存中，将渲染数据（法线，纹理坐标，顶点颜色等）发送给gpu，即调用一次dc
2. 几何阶段
   1. 顶点着色器：坐标变换（从模型空间转到齐次裁剪空间）和逐顶点光照，
   2. 曲面细分着色器：细分图元
   3. 几何着色器：执行逐图元着色操作，或者用于产生更多的图元
   4. 裁剪：将摄像机视野外的顶点裁减掉，并提出某些三角图元的面片，例如：可以控制裁剪区域，或者可以控制裁剪正面还是背面，如果三角形一半在里面一半在外面，则会在边缘新建顶点，来舍弃屏幕外的
   5. 屏幕映射：将每个图片坐标转换到屏幕坐标系
3. 光栅化阶段:将三角形转为像素
   1. 三角形设置：计算每个图元覆盖了哪些像素，以及为这些像素计算他们的颜色
   2. 三角形遍历：检查每个像素是否被一个三角网格覆盖，如果被覆盖则会生成一个片元
   3. 片元着色器
   4. 逐片元操作
      1. 决定每个片元的可见性，例如深度测试、模板测试等
      2. 如果片元通过了所有测试，则进行混合

抗锯齿：

1. SSAA:用更大的分辨率来渲染场景，然后再采样至正常分辨率，一般是取平均值，但需要更多显存和着色计算（每个子采样点都需要进行光照计算，我的理解是比如屏幕分辨率是100x100，先渲染到200x200的buffer上，并且对每个像素都进行完整的片元着色，然后再以2x2为一个单位取颜色平均值，渲染到100x100的屏幕上）
2. MSAA：由于SSAA需要走n倍片元着色，很费
   1. 比如4Xmsaa,对于每一个像素有一个长度为4bit的coverage mask（覆盖标记），每一位代表对应采样点的覆盖情况
   2. MSAA也是和SSAA一样覆盖和遮挡信息都是在光栅化阶段到一个更大分辨率上进行
   3. 步骤
      1. 光栅化阶段到4x分辨率上
      2. 片元着色器
      3. 每个三角形对4个sample执行模板测试与深度测试，用mask记录，通过则复制颜色，没通过依然为背景色
      4. 通过一个对高层透明的Pass，将四个sample的颜色插值获得最终输出的像素颜色

渲染路径：

1. 延迟渲染：

gpu架构：

1. <https://www.imgtec.com/blog/a-look-at-the-powervr-graphics-architecture-tile-based-rendering/>
2. IMR：

