OpenSceneGraph 是一个开源的三维引擎，被广泛的应用在可视化仿真、游戏、虚拟现实、科学计算、三维重建、地理信息、太空探索、石油矿产等领域。OSG采用标准C++和 OpenGL编写而成，可运行在所有的Windows平台、OSX、GNU/Linux、IRIX、Solaris、HP-Ux、AIX、Android 和FreeBSD 操作系统。OSG在各个行业均有着丰富的扩展，能够与使用OpenGL书写的引擎无缝的结合，使用国际上最先进的图形渲染技术，让每个用户都能站在巨人的 肩上。

**面向用户**

OSG是一个开源的三维实时场景图开发引擎，被广泛应用在可视化（飞行、船舶、车辆、工艺等仿真）、增强现实以及医药、教育、游戏等领域。

**平台支持**

OSG可以支持几乎所有的操作系统平台，它使用OpenGL ES使得可以支持手持台、平板以及其它嵌入式设备，使用OpenGL使得其可以在所有的家用电脑以及中型大型机和集群上进行工作。

**许可**

LGPL，在国内很少有人完全明白各种开源许可是怎么回事，但是大家都在使用开源工程。在中国使用一个引擎就是对该引擎发展的最大贡献，用户多就意味着繁荣和对该引擎越来越多的完善（非原文，译者加）。

**技术**

OSG采用C++书写，使用了标准模版库（STL）。OSG使用场景树的方式来管理三维场景，使用逻辑组来构建场景树，以便进行高效的渲染和遍历等。

OSG使用运行时对各种显卡扩展的实时检测，使得OSG支持从OpenGL1.0到OpenGL4.2以及OpenGL ES 1.1 到2.0的所有设备，所以不管设备新旧，操作系统如何，OSG均能及时识别出它支持什么版本的OGL或OES，然后完好的在其上运行。

OSG采用模块化的设计，降低了OSG内部模块的耦合性，使得用户更加容易理解。并且OSG提供了丰富的示例，通过阅读这些示例可以很好的学习这些 模块（学习例子对于学习OSG是非常重要的---FreeSouth注）。模块化的设计使得用户不仅可以只学习和使用自己需要的模块，也可以根据需要定制 自己的模块。

OSG的关键特点可以使用如下关键词进行总结：高性能、可扩展性、接口化等，具体如下：

* **高性能**

支持基于视锥体的裁切、基于遮档的裁切以及其它的小特性裁切，支持LOD、OpenGL 状态排序、VAO、VBO以及着色语言、显示列表等所有的图形学里经常提到的提高效率的招数。它使得OSG成为一个效率高，表现力好的引擎。OSG同样支 持客户化的LOD，客户可以自己定制基于分页的四叉树场景结构用来实现复杂场景，具体可以看一下VTP和Delta3D以及osgEarth。

* **Productivity**

OSG的核心支持所有的OpenGL扩展，哪怕是刚发布的最新扩展，对其进行封装，优化使得用户不用关注OpenGL那些底层的代码和扩展等，就可以快速的搭建基于最新特性的三维应用程序。

除 对底层代码的封装外，OSG还有着与其它系统类似Performer以及OpenInventor等各种现代高级系统的结合，这些结合的案例可以使得用户 快速的将OSG与自己的系统相结合提供帮助。OSG和现有与三维相关的，尤其是基于OpenGL的系统有着丰富的结合案例，可以看一下业内知名人士 array的osgRecipes、osgXI以及osgCookbook从中获取三维系统与OSG相结合的方案灵感。

* **数据加载**

OSG支持市面上几乎所有的数据格式，无论是图片还是三维模型，以及字体等都能很好的读取。

除了支持单一的格式外，OSG还有VPB、osgEarth以及其它不常用的扩展来支持对海量数据的处理和读取。

* **工具类**

OSG提供一些工具类用来完成一些相互独立的功能，列举如下：

* + osgParticle-粒子系统。（OSG的粒子系统从OSG的1.2版本以来，鲜有改变，八年了，它没发展，推荐使用Spark粒子系统，其与OSG的结合array的osgXI还是osgRecipes中有示例--FreeSouth注）。
  + osgText-文字处理与显示。
  + osgFX-特殊效果。
  + osgShadow-阴影。
  + osgManipulator-对模型的局部操作器。
  + osgSim-一些可视化效果。
  + osgTerrain-地形渲染。（针对地形，推荐使用VPB或osgEarth--FreeSouth注）。
  + osgAnimation-动画。
  + osgVolume-体渲染。
* **接口化**

OSG做到不依赖任何与操作系统有关的中间件，只使用标准C++和OpenGL，早期在IRIX上开发，随后扩展到Linux、Windows、Mac、AIX以及Andriod和其它中国人不关心也用得少的操作系统。

OSG的接口化保证了其高度独立，这也使得其除了跨各种平台以外，还可以支持各种UI，比如MFC、QT、SDL、GLUT、WxWidget、Cocoa等。OSG的示例中有这些UI与OSG相结合的例子。（国内使用最多的是QT和MFC--FreeSouth注）。

* **可伸缩性**

OSG可以运行在多核的CPU和GPU上，这缘于OSG对OpenGL显示列表和纹理单元以及拣选、绘制遍历等过程实施了保护措施，使这些阶段可以单独为一个线程也可以在一个线程中串行执行。可以通过osgViewer以及所有的例子来配置当前OSG应用程序的线程模型。

* **多语言支持**

Lua、Python、甚至JAVA都有与OSG的结合。

动态分页数据库：

数据的分页技术在根多领域都十分常见。例如网络上的海量数据库搜索，往往会采取分页的方式，

只搜索数据库的一部分内容-当用户浏览到后面的页面之后，再继续搜索对应的内容。在三维场景的浏览中同样会面临速十问题，如果用户需要制览的数据量很大，比如地形模拟、虚

拟小区和城市，甚至是虚拟地球的工程中，都不可避免地要用到场景数据库的分页技术，否则将对计算机系统产生极大的负担。纵然，采用视锥体裁减等方法可以保证每帧中只有一部分数据被传送到染染管道，而LOD等场景结构方案可以通过牺牲一部分渲染质量来换取效率的提升，但是这些都不能解决在内存中可能要存储海量数据”这一问题——数百GB甚至TB级别的数据（例如中国境内的山形地貌，或者整个北京市的高楼矮房）是不可能全部载入内存中的，就算未来的计算机能够将它们一次性读入，也已经损耗了太多的系统性能。此时数据的分页，也就是数据的动态调度机制就显得尤为重要：在显示当前视域中的场景元素（可见元素）的同时，预判断下一步可能载入的数据（预可见元素）．以及那些短时间内不可能被看到的对象（不可见元素），从而作出正确的数据加载和卸载处理．确保内存中始终维持有限的数据额度，并且不会固此造成场景浏览时重要信息的丢失或者过于迟缓

LOD

LOD技术即Levels of Detail的简称，意为多细节层次。LOD技术指根据物体模型的节点在显示环境中所处的位置和重要度，决定物体渲染的资源分配，降低非重要物体的面数和细节度，从而获得高效率的渲染运算。

LOD技术在不影响画面视觉效果的条件下，通过逐次简化景物的表面细节来减少场景的几何复杂性，从而提高绘制算法的效率。该技术通常对每一原始多面体模 型建立几个不同逼近精度的几何模型。与原模型相比，每个模型均保留了一定层次的细节。在绘制时，根据不同的表准选择适当的层次模型来表示物体。LOD技术 具有广泛的应用领域。目前在实时图像通信、交互式可视化、虚拟现实、地形表示、飞行模拟、碰撞检测、限时图形绘制等领域都得到了应用，已经成为一项要害技 术。很多造型软件和VR开发系统都开始支持LOD模型表示。