BIM模型轻量化以及高并发实现技术

**1 BIM三维建模**

BIM一般指建筑信息模型（建筑学、工程学及土木工程的新工具）。

建筑信息模型（Building Information Modeling）是建筑学、工程学及土木工程的新工具。建筑信息模型或建筑资讯模型一词由Autodesk所创的。它是来形容那些以三维图形为主、物件导向、建筑学有关的电脑辅助设计。当初这个概念是由Jerry Laiserin把Autodesk、奔特力系统软件公司、Graphisoft所提供的技术向公众推广。

BIM（Building Information Modeling）技术是Autodesk公司在2002年率先提出，已经在全球范围内得到业界的广泛认可，它可以帮助实现建筑信息的集成，从建筑的设计、施工、运行直至建筑全寿命周期的终结，各种信息始终整合于一个三维模型信息数据库中，设计团队、施工单位、设施运营部门和业主等各方人员可以基于BIM进行协同工作，有效提高工作效率、节省资源、降低成本、以实现可持续发展。

BIM的核心是通过建立虚拟的建筑工程三维模型，利用数字化技术，为这个模型提供完整的、与实际情况一致的建筑工程信息库。该信息库不仅包含描述建筑物构件的几何信息、专业属性及状态信息，还包含了非构件对象（如空间、运动行为）的状态信息。借助这个包含建筑工程信息的三维模型，大大提高了建筑工程的信息集成化程度，从而为建筑工程项目的相关利益方提供了一个工程信息交换和共享的平台。

BIM有如下特征：它不仅可以在设计中应用，还可应用于建设工程项目的全寿命周期中；用BIM进行设计属于数字化设计；BIM的数据库是动态变化的，在应用过程中不断在更新、丰富和充实；为项目参与各方提供了协同工作的平台。我国BIM标准正在研究制定中，研究小组已取得阶段性成果。

**2 BIM轻量化分析**

传统三维模型建模的工作流程是：

CAD+照片=>3DMax建模=>Unity开发=>三维软件。

结合BIM建模模型则演变为：

CAD资料=>BIM建模=>3dMax处理渲染=>Unity开发=>三维软件。

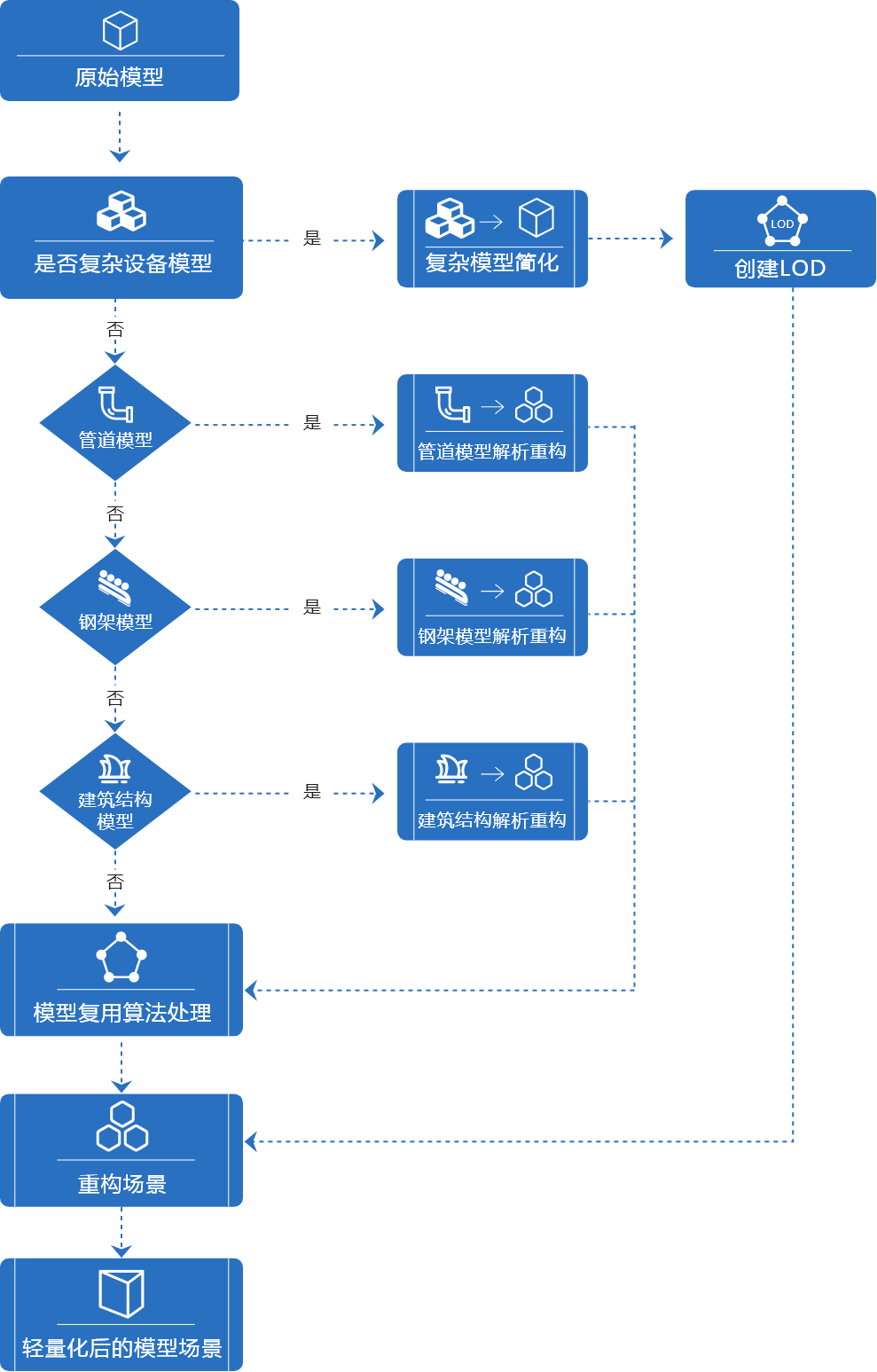
使用BIM建模软件建模的过程标准、高效、精确，能快速基于CAD资料构建完整的精准场景，能够极大的提高精准建模的工作效率。但同时BIM建模软件中的模型的基本单位的模型对于作为三维软件的模型来说有点面多、数量多、贴图简单、细节多的特点，需要进行“轻量化”处理。除此之外，栏杆系统、管道系统创建出来的模型也需要专门进行解析和处理。

为了保证运行效率和显示效果，下面会做一些模型处理和采用一些技术，但总的思路都是“按需加载”，不显示当前不需要显示的模型。

## 2.1 重构场景

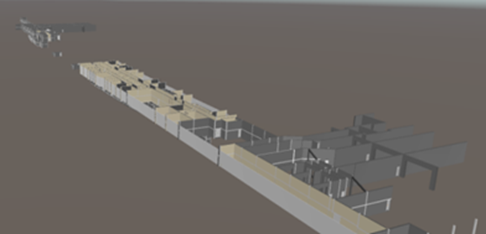
重构场景是通过对BIM建模软件生成的模型进行分析，根据模型的名称进行分类，建立相应的类别信息数据。将整个场景分主要为框架模型和部件模型，部件模型又细分为管道模型、钢架模型、建筑结构模型、复杂结构模型（如栏杆）和复杂设备模型。

总的模型处理流程，将原始模型根据名称类别分配到不同的处理子流程，处理完成后在重建场景。



2.1.1框架模型

框架模型指的是有孔洞的墙壁、不规则的地板或者天花板 、非标准的模型，这些模型的特点是结构简单、点面少，相互关联，无法作为部件复用。处理方式是设置为静态模型，作为场景的整体框架和背景。可结合光照阴影，做一些效果上的渲染优化如图1所示。将模型设置为静态模型后，可以事先做复杂的渲染，将渲染结构保存到光照贴图中，减轻该部分模型在实时渲染时的CPU占用。



**图1 框架模型**

2.1.2 部件模型

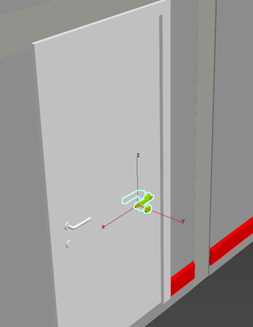
部件模型指的是门、窗、桌椅、各种箱、柜、连接件、仪表、灯具、以及其他设备，即各种三维模型族。处理方式如下：

（1）解析并获取模型的转换信息(Transform)，包括坐标、角度、缩放，以及模型类型、名称等信息，保存到一个json文件中。

（2）将每个类型BIM模型导出一个FBX模型作为预设。

（3）在Unity中根据json的信息使用预设重建场景。

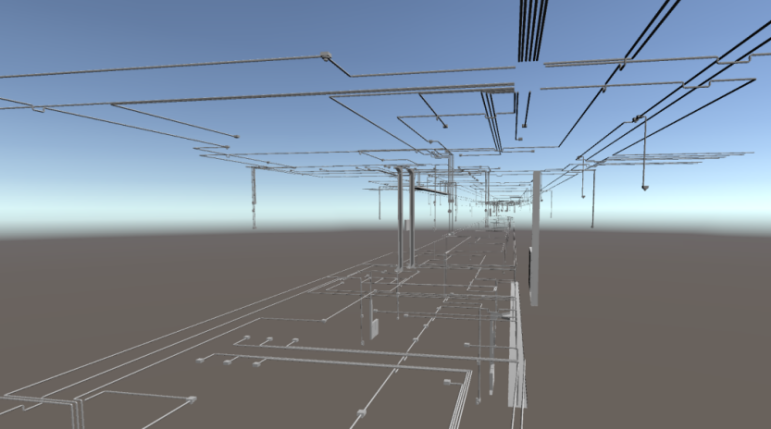
经过前面的步骤并加载框架模型，已经能够重建整个场景。通过将场景分解成一个个部件，可以实现精确的模型控制，按类别加载显示不同种类的模型，同时不显示不需要的模型。



**图2 部件模型**

2.1.3 管道模型

管道模型指的是BIM建模软件中的各种“线管”，如“钢性非金属线管”，管道的特点是虽然单个（或者说单断）管道的点面数量不是很多，50-100个点，但是数量巨大（4000以上），在消防与照明场景中结合相应的“弯头”模型占据了80%-90%的点面数量。

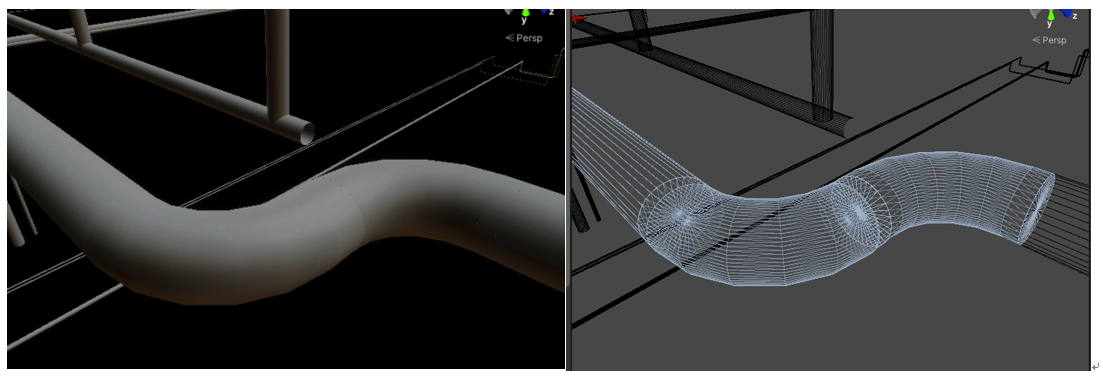


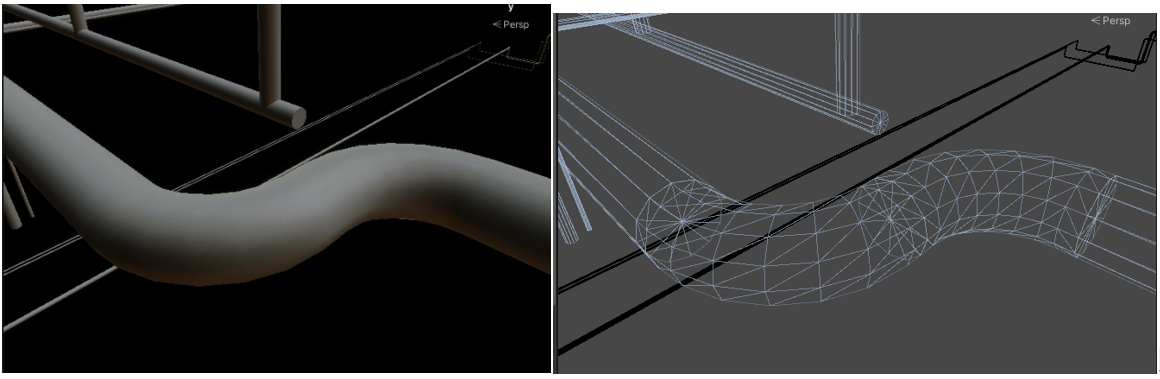
**图3 管道模型**

处理方式：将管道模型中的管线(PipeLine,PipeFlange)，弯头(PipeElbow、PipeBend)、三通(PipeTee)、变径(PipeReducer)、焊缝(PipeWeld)等进行模型解析重建，将其他相关设备或者管道连接件则作为部件模型进行复用处理。

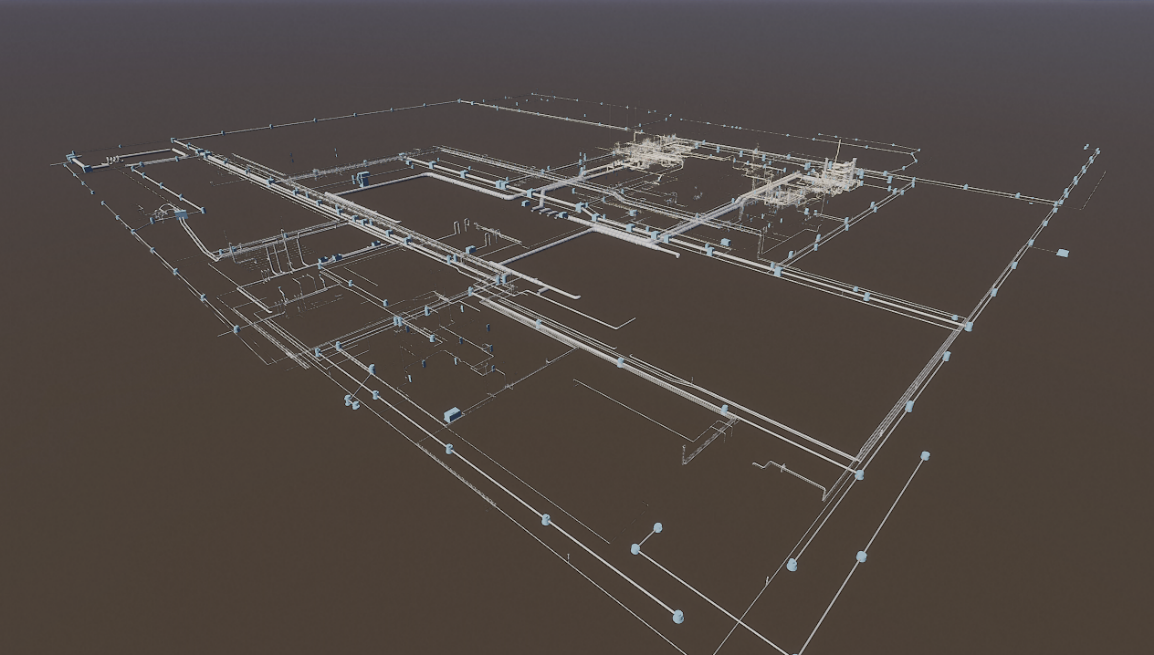
根据不同的管道模型类型进行模型解析获取关键参数（如PipeLine的起点、终点、半径；PipeElbow的两端点、弯头角度、弯头半径；PipeBend的经过点坐标、半径），并创新创建简化的管道模型，以便达到减少点面的作用。

如下图所示，原始模型是36面的，基本上和简化成12面的模型效果上是没区别的，这是从源头上将模型进行简化。

原始模型

简化后的模型：

完善后的话，程序中可以只存储管道参数信息，在程序运行过程中动态创建管道。这样整个程序中的管道部分的模型占用的存储空间能够节省很多，渲染效率也会提高一些。

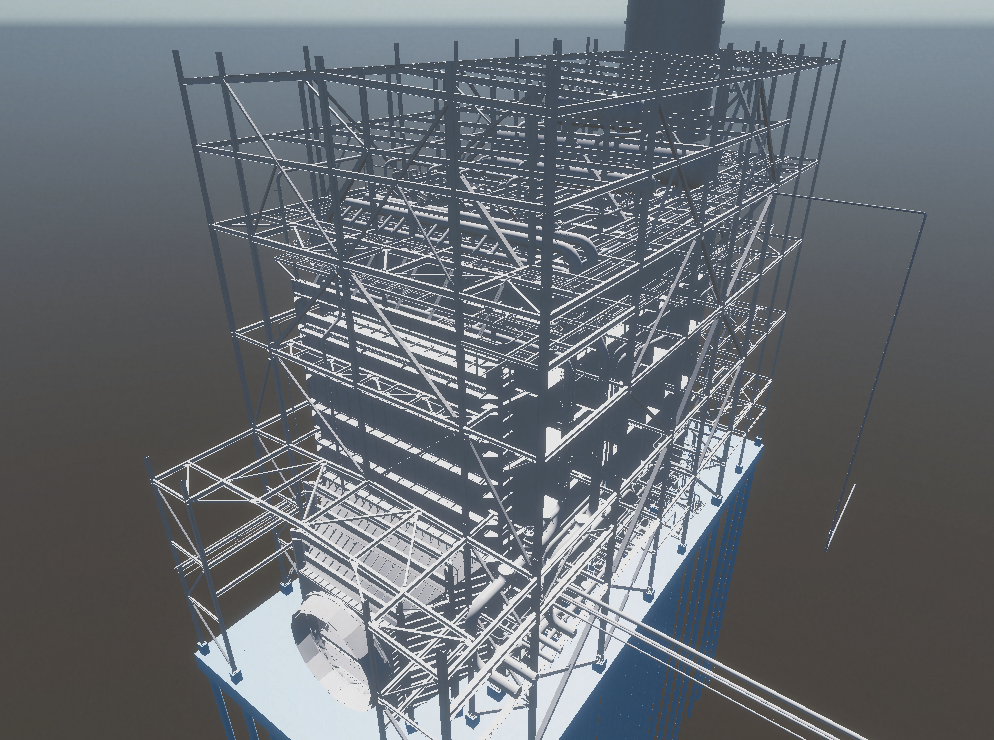
测试管道场景：

管道测试场景的处理结果是最终的顶点数量为原始模型的9.3%左右，不考虑连接件的复用率的话仅仅是管道本身顶点数量简化为原始模型的3.5%。测试数据如下：

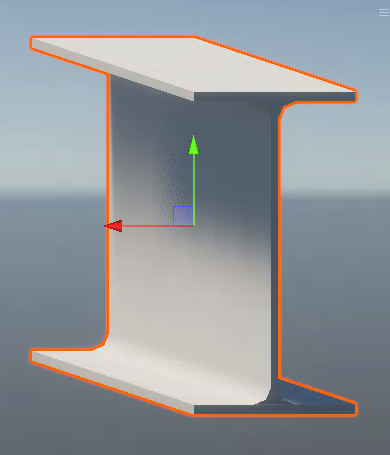
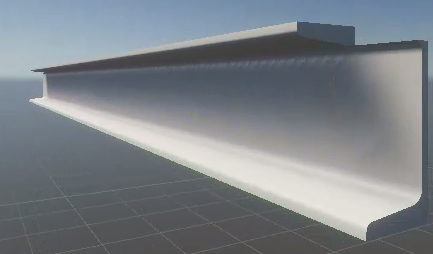
|  |  |  |  |
| --- | --- | --- | --- |
| 测试管道 | 原始顶点数量(万) | 处理后顶点数量(万)和比例 | **最终的(模型复用后)的顶点数量(万)** |
| HH(有焊缝无连接件) | 176 | 37(21.2%) | **2.7(1.5%)** |
| JG(有焊缝无连接件) | 236 | 48(20.2%) | **1.6(0.7%)** |
| SG(有焊缝无连接件) | 543 | 116(21.4%) | **1.5(1.6%)** |
| 全部(有焊缝无连接件) | 1856 | 433(23.3%) | **66(3.6%)** |
| 全部(有焊缝有连接件及相关设备) | 3051 | 1448(47.5%) | **284(9.3%)** |

2.1.4钢架模型

钢架模型指的是建筑结构中的钢架结构，

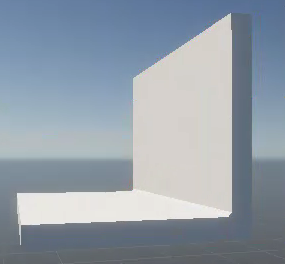
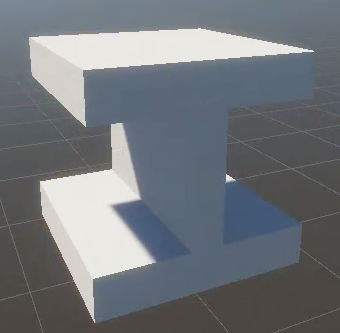
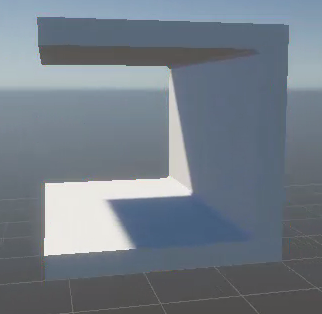


主要有长方体模型、H型钢架、L型钢架、C型钢架，以及其他其他钢架部件组成。

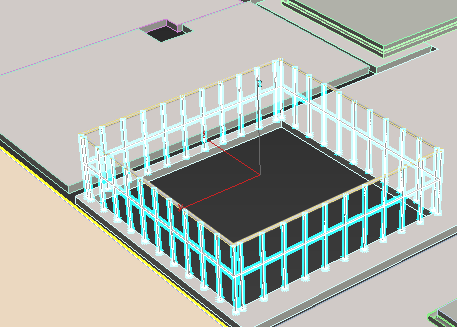
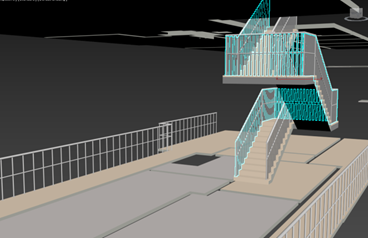
处理方式：将可基本的几种钢架模型进行解析成关键参数（类型、起点、终点、尺寸）并重建成简化模型，将其他的模型作为部件模型进行复用处理。

简化后的基本钢架模型为

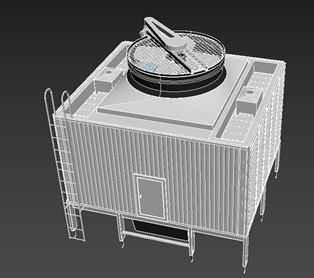
2.1.5 栏杆模型

栏杆模型指的是栏杆系统创建的模型，主要是楼梯两边的扶手栏杆，也包括楼板上的栏杆图4所示，都是由一根根管子组成，而且更加复杂。处理方式：也是和上面的管道类似，解析并重建。不过由于栏杆更加复杂，这些管子是合并在一个模型内的，需要多一个分解的过程。即分解->重建->合并的过程。



**图4 栏杆模型**

2.1.6 复杂设备：指的是一个模型就具有十几万甚至更多点，一般有复杂内部结构，如空调系统的冷却塔。处理方式：手工创建简化后的模型，将细部结构用贴图代替，删除没不用展示的结构（内部结构、外部细节）。最终在场景中显示简化后的模型，但摄像头拉近时显示精细模型。



**图5 复杂模型**

## 2.2 模型复用技术

**模型复用的原理：**

假如一个模型文件(FBX、Obj等)中有2个相同的模型，这2个模型的坐标、角度、比例可能不同也可能相同，但是经过坐标、角度、比例的变换后是可以完全重叠在以前的。将该模型文件加载到三维引擎中并创建模型场景，这时运行时内存中会有2份模型数据。如果将一个模型在三维引擎中拷贝一个副本（Instance），并变换坐标角度等重叠到另一个模型上，删除另一个模型。这时内存中则只有一份模型数据，显示的2个模型都是一个模型的副本。

在三维引擎中搭建模型场景一般是使用一些基本的模型单位，通过创建不同的副本来构建完整的场景的，这样的内存占用小，运行效率高。而BIM模型转换过来的模型会存在多个相同模型的问题，通过模型复用算法，能够减少内存显存的占用情况，提高运行效率。

默认情况下的多个相同模型的Instance渲染时在GPU中还是多次渲染的，需要使用GPUInstance技术才能达到一次渲染多个Instance的效果。

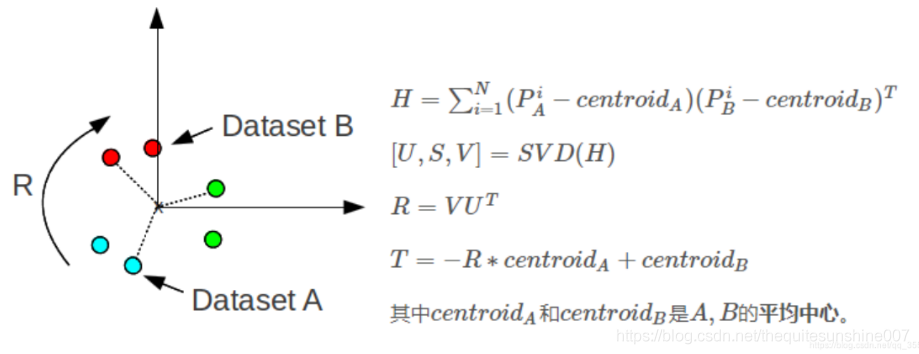
Gpu Instance是一种用来提高渲染大量物体效率的技术。这是最新渲染api提供的一种技术，如果绘制1000个物体，它将一个模型的vbo提交给一次给显卡，至于1000个物体不同的位置，状态，颜色等等将他们整合成一个per instance attribute的buffer给gpu，在显卡上区别绘制，它大大减少提交次数，它在不同平台的实现有差异，这种技术对于绘制大量的相同模型的物体由于有硬件实现，所以效率最高，最为灵活，避免合批的内存浪费，并且原则上可以做gpu skinning来实现骨骼动画的instancing。

**模型匹配算法：**

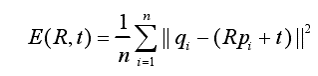
模型的三维信息包括坐标、旋转角度、缩放比例，这些数据实际上都是一个三维变换矩阵。两个模型之间的坐标、角度变换就是两个三维矩阵之间的变换。该问题的最简单形式就是根据对应的三维点估计2个刚体变换的旋转平移矩阵（RT矩阵）。

RT矩阵原理：只要算出变换矩阵，就可以算出A坐标系的一个点P在坐标系B里的对应点坐标，即T为3x3的转换矩阵， t 为3x1的位移变换向量，这里点坐标均为3x1的列向量（非齐次形式，齐次形式下为4x1列向量，多出的一个元素值补1而已）。

理论上只要给定至少3对点，就能计算出 T 和 t 。自然的，点对越多，计算出来的转换就越精确。详细的原理请参考《Estimating 3-D Rigid Body Transformations: A Comparison of Four Major Algorithms》，它使用SVD方法计算 T 和 t 。



模型本质上是由点组成的，两个模型的变换也就是两个点云之间的变换。两个模型之间的对应点一般不是能之间获取到的，不同的对应点得到的RT矩阵是不同的，这是就需用用到ICP算法。

ICP算法的基本原理是：分别在带匹配的目标点云P和源点云Q中，按照一定的约束条件，找到最邻近点（pi，qi），然后计算出最优匹配参数R和t，使得误差函数最小。误差函数为E（R，t）为： 

其中n为最邻近点对的个数，pi为目标点云 P 中的一点，qi 为源点云 Q 中与pi对应的最近点，R 为旋转矩阵，t为平移向量。

ICP算法步骤：

（1）在目标点云P中取点集pi∈P；

（2）找出源点云Q中的对应点集qi∈Q，使得||qi-pi||=min；

（3）计算旋转矩阵R和平移矩阵t，使得误差函数最小；

（4）对pi使用上一步求得的旋转矩阵R和平移矩阵t进行旋转和平移变换，的到新的

对应点集pi’={pi’=Rpi+t,pi∈P}；

（5）计算pi’与对应点集qi的平均距离； https://img-blog.csdn.net/20180511160030251

（6）如果d小于某一给定的阈值或者大于预设的最大迭代次数，则停止迭代计算。

否则返回第2步，直到满足收敛条件为止。

ICP算法关键点：

（1）原始点集的采集：均匀采样、随机采样和法矢采样

（2）确定对应点集：点到点、点到投影、点到面

（3）计算变化矩阵：四元数法、SVD奇异值分解法

## 2.3 轻量化处理与动态显示

在解析重构场景的基础上，可以对每个部件模型进行简化处理，将点面多的模型，简化创建简单的模型，并在程序中动态切换不同精细程度的模型。

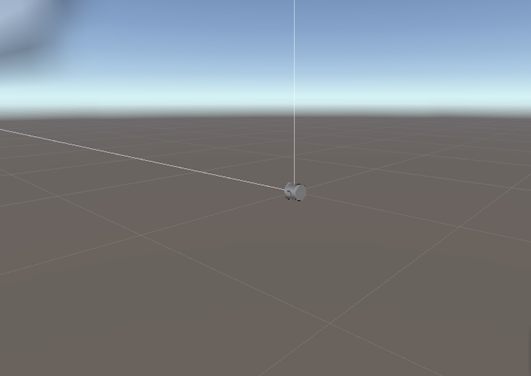
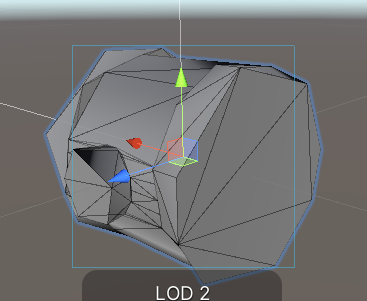
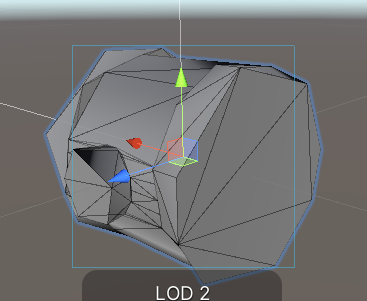
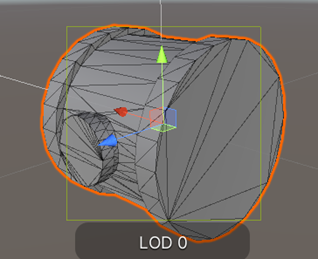
模型动态显示的技术有两种LOD和遮挡剔除。LOD，即Levels of Detail的简称，意为多细节层次。LOD技术指根据物体模型的节点在显示环境中所处的位置和重要度，决定物体渲染的资源分配，降低非重要物体的面数和细节度，从而获得高效率的渲染运算。使用LOD技术简单来说就算距离模型越远，切换到越简单的模型，最终到一定距离时直接不显示该模型。

遮挡剔除 (Occlusion Culling) 功能可在对象因被其他物体遮挡，当前在相机中无法看到时，禁用对象渲染。该功能不会在三维计算机图形中自动开启，因为在大部分情况下，离相机最远的对象最先渲染，离相机近的对象覆盖先前的物体，该步骤称之为“重复渲染 (overdraw)”。使用遮挡剔除简单来说就是被墙壁挡住的模型不用渲染。

另外还有个视锥体剔除 (Frustum Culling) 只禁用相机视野外的对象渲染，不禁用视野中被遮挡的任何物体的渲染。不过这个是三维引擎默认都具备的功能，不需要另外设置开发。

创建简化模型的过程，则是创建相应“高模”的不同等级的“低模”。对于一般模型，可以用程序自动创建低模，而对于复杂模型，需要手工创建。因为自动创建出来的低模可能导致一些问题，如破面、法线、贴图出错。LOD需要结合简化模型，遮挡剔除则需要模型分解一个个独立的部件。自动创建简化模型的操作可以在3dmax中做，也可以在unity中做，最理想是手工创建每个模型的简化模型。

这里是一个“异径三通-沟槽式\_标准”的模型（图6所示），LOD0是原始模型具有504个点，LOD1有338个点(66%简化)，LOD2有184个点(33%简化)。从LOD0到 LOD2模型轻量化效果明显。没有LOD3，因为这是已经看不到了。



**图6 “异径三通-沟槽式\_标准”的模型**

可以看到LOD2时模型以及很粗糙了，但是实际上这时的距离是这样的，完全没必要使用精细的模型。LOD等级、数量、距离都是可以调整的。

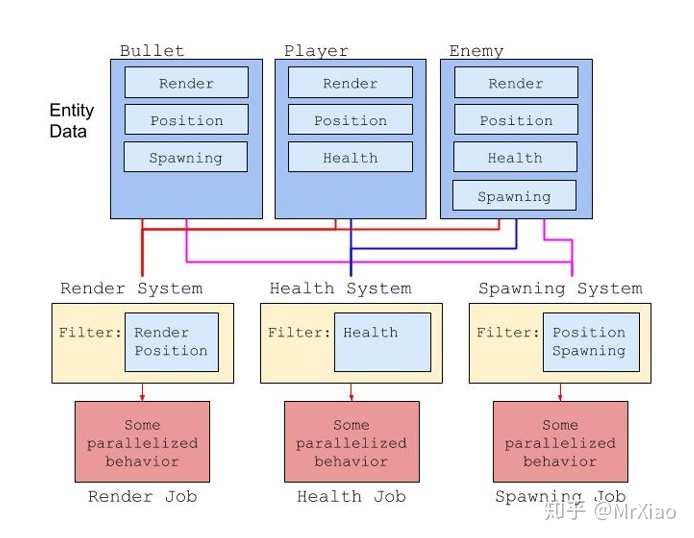
通过简化和动态加载，能始终确保场景中的点面数保持在一定范围以下，根据具体电脑的配置大概200W-500W个点，保证帧率达到40-60，程序能流畅运行。参考每秒的帧数(fps)或者说帧率表示图形处理器处理场时每秒钟能够更新的次数。高的帧率可以得到更流畅、更逼真的动画。一般来说30fps就是可以接受的，但是将性能提升至60fps则可以明显提升交互感和逼真感。对于Unity来说60fps以上是不存在的，没必要的，所以限制起来的。

需要注意的是，使用LOD技术实际上是增加了内存中的点面的，因为除了原始模型外，不同等级的简化模型也会被创建并加载到内存中，内存占用总体增加了一倍左右。但现在的电脑硬件条件下，内存是成本最低的。

## 2.4 ECS技术

ECS（Entity-Component-System）是一种新的架构模式（当然是针对Unity本身来说），这是一个取代GameObject / Component 的模式。其模式遵循组合优于继承原则，游戏内的每一个基本单元都是一个实体，每个实体又由一个或多个组件构成，每个组件仅仅包含代表其特性的数据（即在组件中没有任何方法）。系统便是来处理拥有一个或多个相同组件的实体集合的工具，其只拥有行为（即在系统中没有任何数据）。

ECS旨在比GameObject / MonoBehaviour更容易处理大量物体。ECS由于是面向数据的设计，所以很容易并行高速处理，以及与Unity开发的C#JobSystem/Burst Compiler一起工作,从而获取更高的项目运行性能。



**Entity-Component-System的优点**

1. 高性能
2. 易编写出高度优化的代码
3. 易编写出可重用的代码
4. 数据在内存中是紧密排列的
5. 和Job System、Burst一起可以获得更高的性能

## 2.5超大场景加载

前面的处理都是基于一个独立场景的，比如一个地铁站的BIM模型，或者是一栋大楼。对于“超大场景”的加载的场景，比如GIS结合BIM，做数字城市、城市大脑之类的项目，在GIS上显示建筑的BIM模型。处理方式还是按需加载，在远处时只显示建筑外壳，在拉近后再逐步显示细节，最后进入建筑场景中再加载BIM模型。进入建筑后则和上面的应用场景一样了。

另外就算没有结合GIS，对于大场景，也是按照分块按需加载的。其在当前区域外的模型都是没有加载的，随着视角的移动才按需加载相应的区域以及细节。

## 2.6 美化处理

在前面的场景解析、轻量化处理与动态加载的同时，可以对模型进行材质设置，贴图设置，渲染灯光等，以便达到更加好的显示效果。另外可以加上一些动画效果，如告警闪烁、管道流向、烟雾等。

BIM建模软件默认的材质效果导入3dmax和unity中时能用是能够的，但是还不够美观，需要根据场景具体设置不同类别模型族的材质效果。具体来说墙壁、地板、天花板，这种整体框架的模型，需要创建几种通用美观的材质库，通过软件自动设置并渲染。

**3 总结**

（1）轻量化处理其实是将BIM和三维平台直接创建一个桥梁，使BIM模型能够通过一个标准化、自动化的工具的处理，快速的导入三维平台中，并保证运行效率和展示效果；

（2）轻量化处理过程还需要结合不同的BIM场景文件优化完善，并细化各个部分的处理模块，同时创建必要的模型库、材质库；

（3）通过轻量化处理模型减少了运行过程中的内存、显存的资源占用，通过高效渲染技术（动态加载显示、LOD、遮挡剔除、GPUInstance、ECS等）减少了运行中CPU、GPU使用率的，最终大大提高了最终发布程序的并发量。

（4）设计团队、施工单位、设施运营部门和业主等各方人员可以基于BIM进行协同工作，有效提高工作效率、节省资源、降低成本。