工业软件研发中处理超大模型(3)

原创 邓子平 多物理场仿真技术

工业软件研发中处理超大模型

工业软件研发中处理超大模型(2)

上文(点击链接查看)介绍了超大模型的一些概念和处理原则方法

这里的大模型处理包括：

1. 模型文件的加载，传输，文件保存；
2. 模型数据在UI上的编辑操作
3. 模型在网络上的传输
4. 模型在渲染视图上的显示
5. 模型在视图上的交互，比如创建模型，拾取，捕捉等操作
6. 模型的几何数据编辑
7. 模型的网格生成，显示，编辑
8. 模型的合理分解

回顾一下之前介绍的大模型处理方法和原则：

1. 识别非线性操作，并尽量避免
2. 数据压缩
3. 分治，避免合并
4. 动态加载，分步操作
5. LOD（Level of details）

6. 非阻塞多线程

7. 避免高频操作，批次合并

8. 大数据唯一，避免拷贝

9. 合理使用树结构

10. 利用业务逻辑取代一般性操作

11. 解耦性能瓶颈模块

12. 开发相应调试工具

本文再用一个流程做一下总结

1. 文件导入

原则上我们不会导入一个大的实体模型做编辑，而是对局部模型做编辑后，然后做集成。比如BIM里有族Family的概念，机械里装配体，EDA设计里IC，PCB集成，CFD里模型分解建立Block等等。从业务上需要尽量避免编辑大模型。外部模型导入后，需要重新建立索引，优化结构，然后存储为新的格式。通常导入模型的开销都能忍受，但实际上通过保存视图数据到文件，分开保存几何模型和业务模型，优化索引排序，使用多线程读入等方法，可以将任意超大模型的读入时间压缩在分钟级别以内。

2. 内存模型数据

文件导入后，在内存中要保证数据的唯一性，避免数据产生拷贝，还要检查硬件内存大小是否能完全完成数据导入后各种操作。windows有虚拟内存，当内存不够会使用硬盘虚拟内存，性能会大幅下降。内存模型数据需要保证数据的完整性，连续性和唯一性。对于大模型，尽量减少各种不必要的数据操作，比如备份，事务，锁，历史记录等等。

3. UI更新

对于UI而言，关注的是模型里对象的数目，而非单个模型的复杂度，因为UI主要是将数据加载到对应的界面上。当模型数目过多时，UI会产生各种性能问题。最常用的控件Table，List，Tree当元素达到一定数量时会有性能问题。在操作中会夹杂业务逻辑，从而造成UI操作各种延迟卡顿。UI中最常用的解决办法就是缓存，延迟和局部加载，各种操作后避免刷新大数据，此前也多有描述。

4. 渲染视图显示

大模型三维视图显示对于渲染是一个大的挑战。目前市面上主流三维显示引擎，在不做任何加速的情况下，当三角面片达到千万级别时，都会产生性能问题。视图显示性能这块其实个大话题，有非常多的加速方法，有硬件的，有软件的，有算法的，有业务的。长远看，渲染引擎显示这块不会成为瓶颈，都有对应的技术手段解决。毕竟十亿面片在UE5里已经能在台式机上流畅显示了。

5. 几何计算

几何计算是大模型里常见的性能瓶颈。单单对模型本身进行一个拓扑和几何有效性检查就是一个漫长的过程。更不用说整体干涉检查，模型操作里的拾取，捕捉等需要大量实时计算的操作。这块一方面需要使用各种加速比如树结构，另外也要和业务紧密关联，减少一般性的无效计算。像外科手

术一样根据具体需求和操作，做精准的定位和有效计算。像树这种结构不仅在几何中可以使用，在网格生成，图形渲染，求解器求解都能派上用场，是一种通用数据结构。

6. 一般计算

数据量大时，要避免一般性的迭代循环查找，遍历等操作，尤其是双重乃至三重循环。在设计上根据实际情况选择合适的数据结构，比如标准模板库中的set,map,list等。涉及到性能瓶颈的问题，有条件要自己设计实现。这一块也是比较考验数据结构设计以及基础算法功底的部分。

7. 广泛使用多线程处理问题

Native C++和QT，OpenMP等对多线程的支持越来越好，尽量使用多线程方法解决问题。必要时可使用多进程，以及封装使用外部工具。

8. 使用自己开发的调试工具定位解决性能问题。

9. 在软件设计架构层面，建立自己的事务，事件以及消息处理机制，让程序更加灵活，便于大模型的调试和处理。

关于硬件也需要注意，通常前后处理需要大的内存和好的显卡，而计算求解要求大内存和好的CPU，分布式还要求网络传输要好。超大模型的求解器仿真计算对软硬件的资源调度，负载均衡也有要求。求解器里对超大模型的处理，是一个非常大的话题，也是工业仿真软件核心中的核心，后面单独介绍。

阅读：null

在看：null