基于CUDA和OSG技术的城市震害模拟

清华大学防灾减灾工程研究所 韩博

2012-04-11

内容提要

1.课题简介

2.弹塑性时程分析程序的并行化

3.地震响应展示程序的实现

4.程序实例

5.总结

内容提要

1.课题简介

2.弹塑性时程分析程序的并行化

3.地震响应展示程序的实现

4.程序实例

5.总结

1.课题简介

本课题以CUDA和OSG作为技术手段，分别完成了城市一般建筑震害模拟的计算模块和显示模块，并将成果应用于清华大学的区域震害模拟上，取得了良好的效果。从而能够高效、廉价、快速地进行大范围城市中大量一般建筑的震害模拟。

1.课题简介

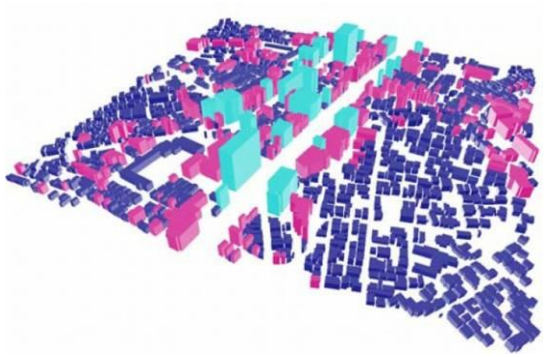
- 中国是世界上地震灾害最为严重的国家之一



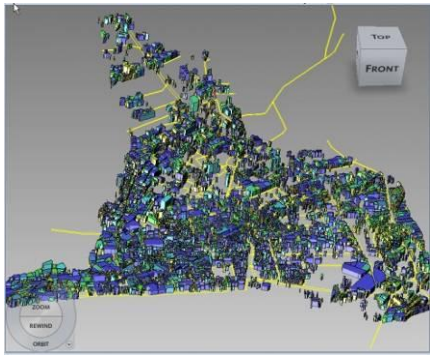
- 对地震灾害进行科学预测是我国目前的一个重要任务

1.课题简介

- 目前，采用高性能计算平台，利用精细化模型，对城市区域震害进行模拟成为了关注的焦点



(a) Urban regional seismic damage simulation for Tokyo by Utokyo.

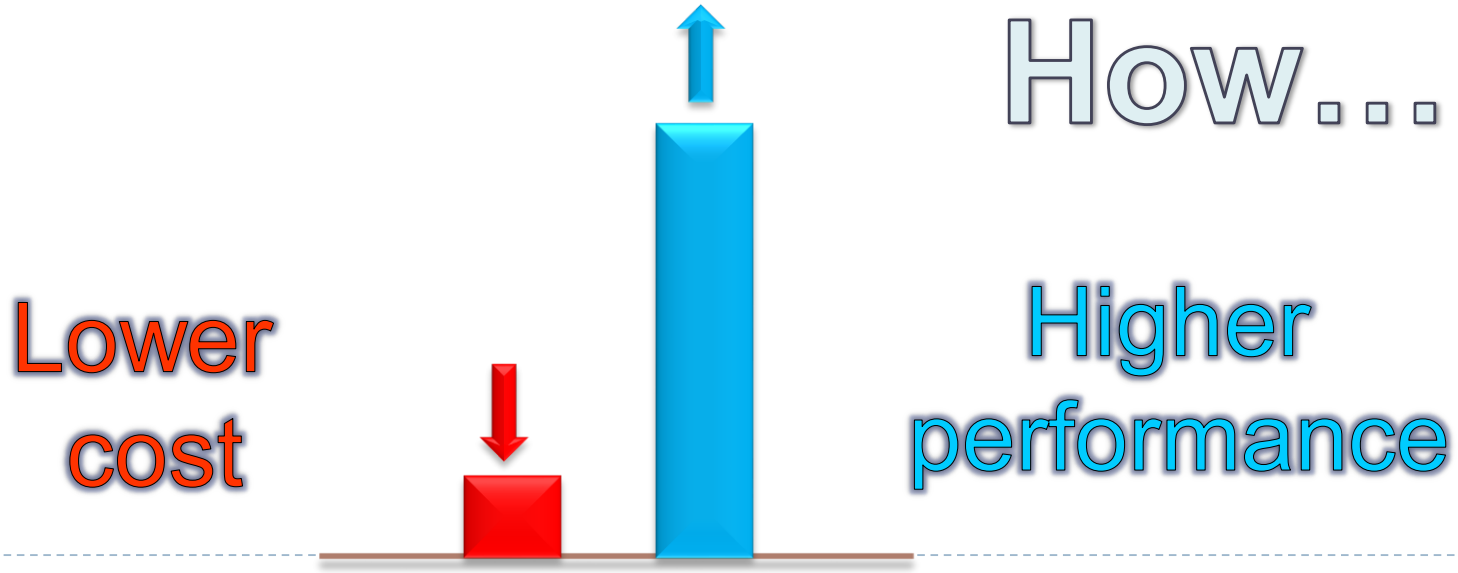


(b) Urban regional seismic damage simulation for Shantou by THU.

But…

1.课题简介

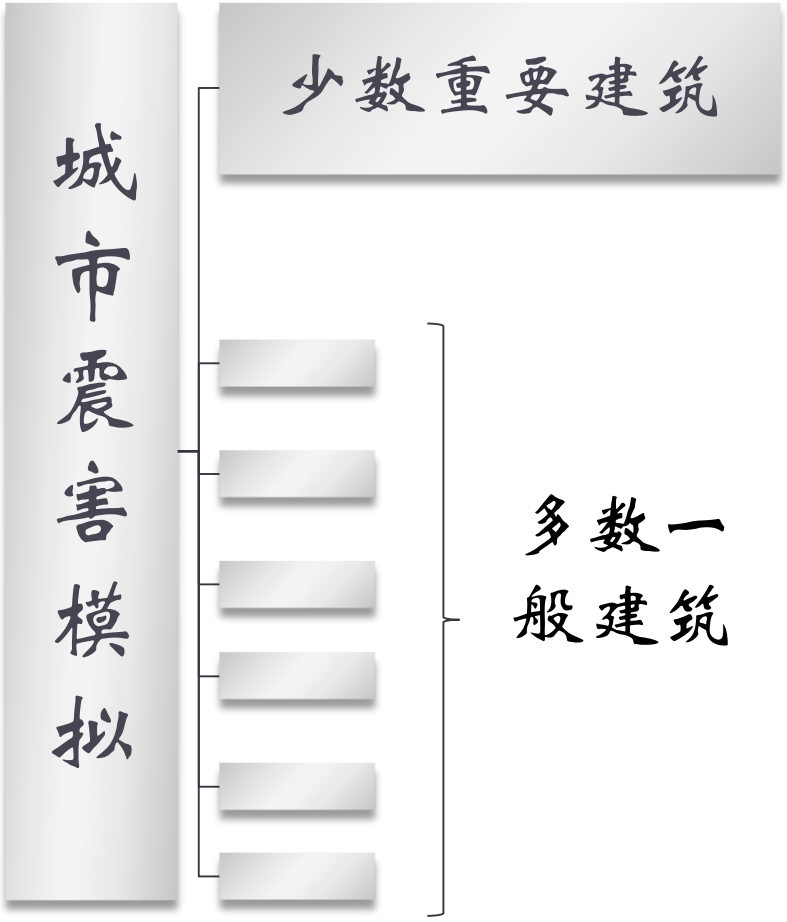
- 目前的研究大多都基于传统的CPU平台
- 超级计算机
  - 昂贵
  - 笨重
  - 维护费用高



1.课题简介

- GPU(Graphic Processing Unit)，即图形处理单元（狭义上说就是显卡）。
  - 特点
    - 单核计算能力相对较弱
    - 核心数很多
    - 价格相对较低
    - 适合大量并行运算和浮点计算
- 2007年6月，NVIDIA推出了CUDA，将GPU推向通用计算领域

1.课题简介



需要复杂模型，适合通用有限元软件，采用CPU计算。

模型简单，采用层模型计算就能满足精度要求

1.课题简介

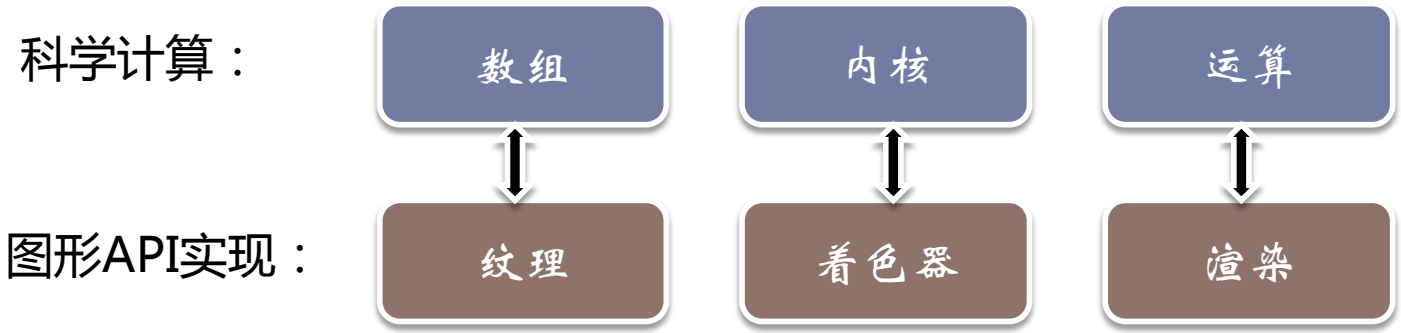
- 采用GPU计算的优点

采用GPU计算一般建筑		GPU计算特点
模型简单，但模型自由度少	⟺	单核计算能力相对较弱
建筑之间无需数据交换	⟺	线程之间数据交换越少越好
建筑数量巨大	⟺	适合并行计算

低价格　　合适　　高性能

1.课题简介：CUDA

- 传统GPGPU计算
  - 图像显示：每个点一个向量，每秒上亿次运算，相关性低，适合并行计算
  - 传统上，GPU的应用被限制于图形渲染上的计算，通用科学计算则采用图形API进行计算。
    - 缺点：(1)图形API操作困难 (2)线程间无法通信



1.课题简介：CUDA

- CUDA
  - 2007年6月，NVIDIA推出了CUDA (Compute Unified Device Architecture，统一计算设备架构) 与以往的GPU相比，在架构上有了显著的改进.
    - 1.采用了统一处理架构，即将过去用于纹理计算的顶点渲染器和像素渲染器进行了整合。
    - 2.引入了片内共享储存器，使得线程间可以进行通讯

1.课题简介：OSG

- OSG
  - OpenSceneGraph（简称OSG）使用OpenGL技术开发，是一套基于C++平台的应用程序接口（API）。
  - 快速、便捷地创建高性能、跨平台的交互式图形程序。

1.课题简介：OSG

- 优点
  - 便捷的三维世界和场景管理系统
  - 跨平台接口
  - 可以优化渲染性能
  - 支持多边形细分

1.课题简介

- 研究内容
  - 1. 基于CUDA技术的城市区域地震响应计算模块
  - 2. 基于OSG技术的城市区域地震响应显示模块
- 预期目标
  - 1. 编写完成计算模块和显示模块
  - 2. 将计算模块和显示模块应用于实例

内容提要

1.课题简介

2.弹塑性时程分析程序的并行化

3.地震响应展示程序的实现

4.程序实例

5.总结

内容提要

1.课题简介

2.弹塑性时程分析程序的并行化

3.地震响应展示程序的实现

4.程序实例

5.总结

2.弹塑性时程分析程序的并行化

- 程序介绍

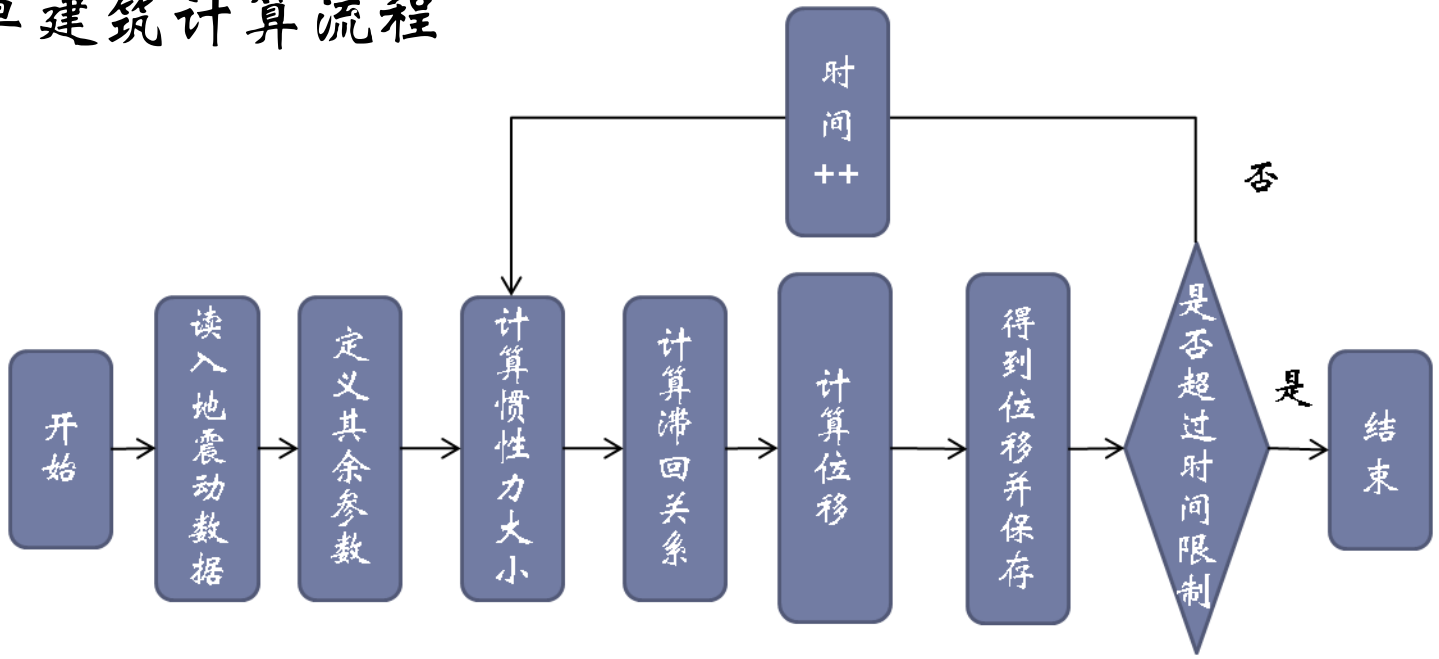
  层剪切模型

  陆新征—曲哲塑性铰模型

  中心差分法

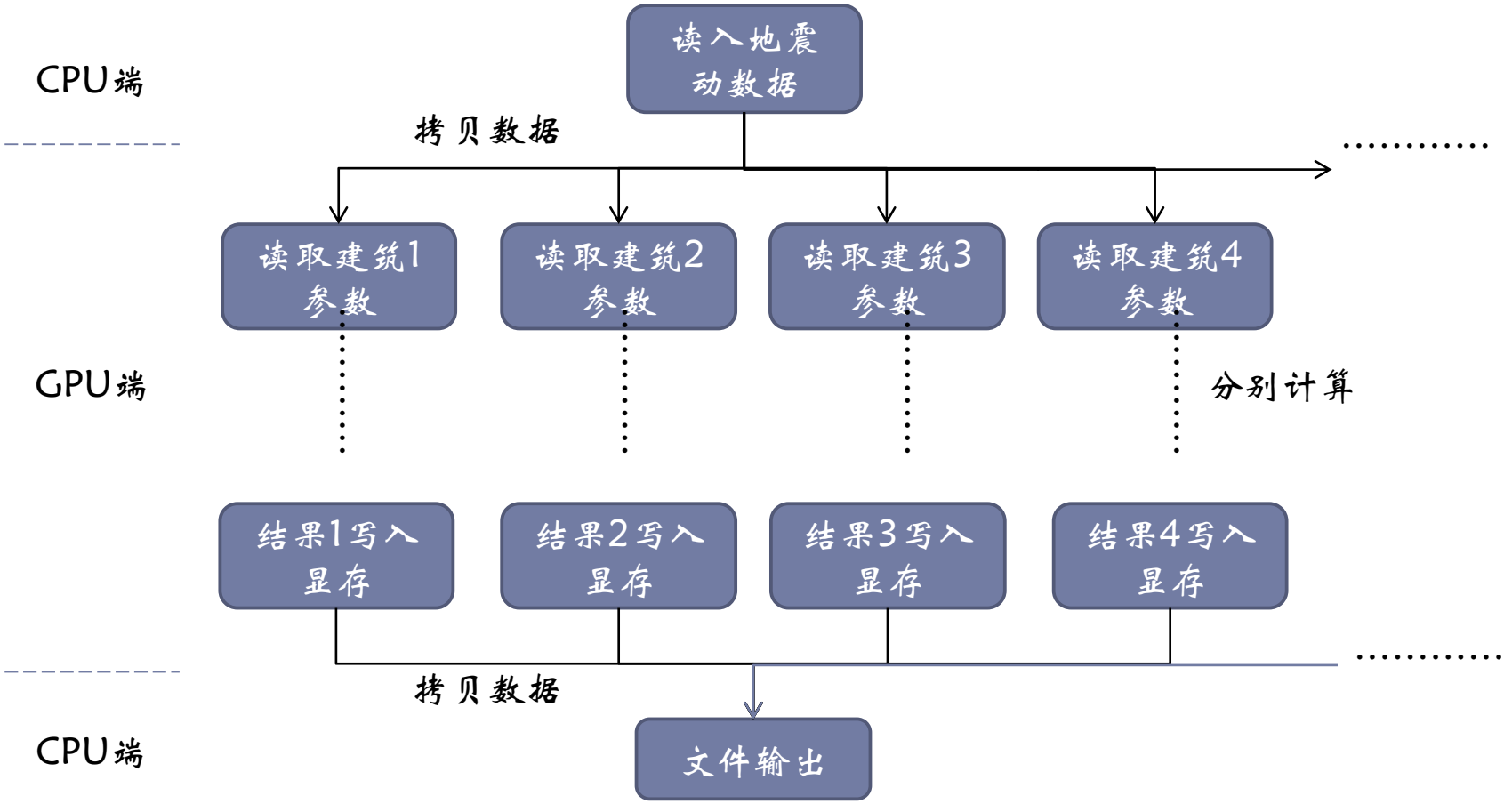
计算方法简单，满足精度要求，且能够考虑倒塌。

- 单建筑计算流程



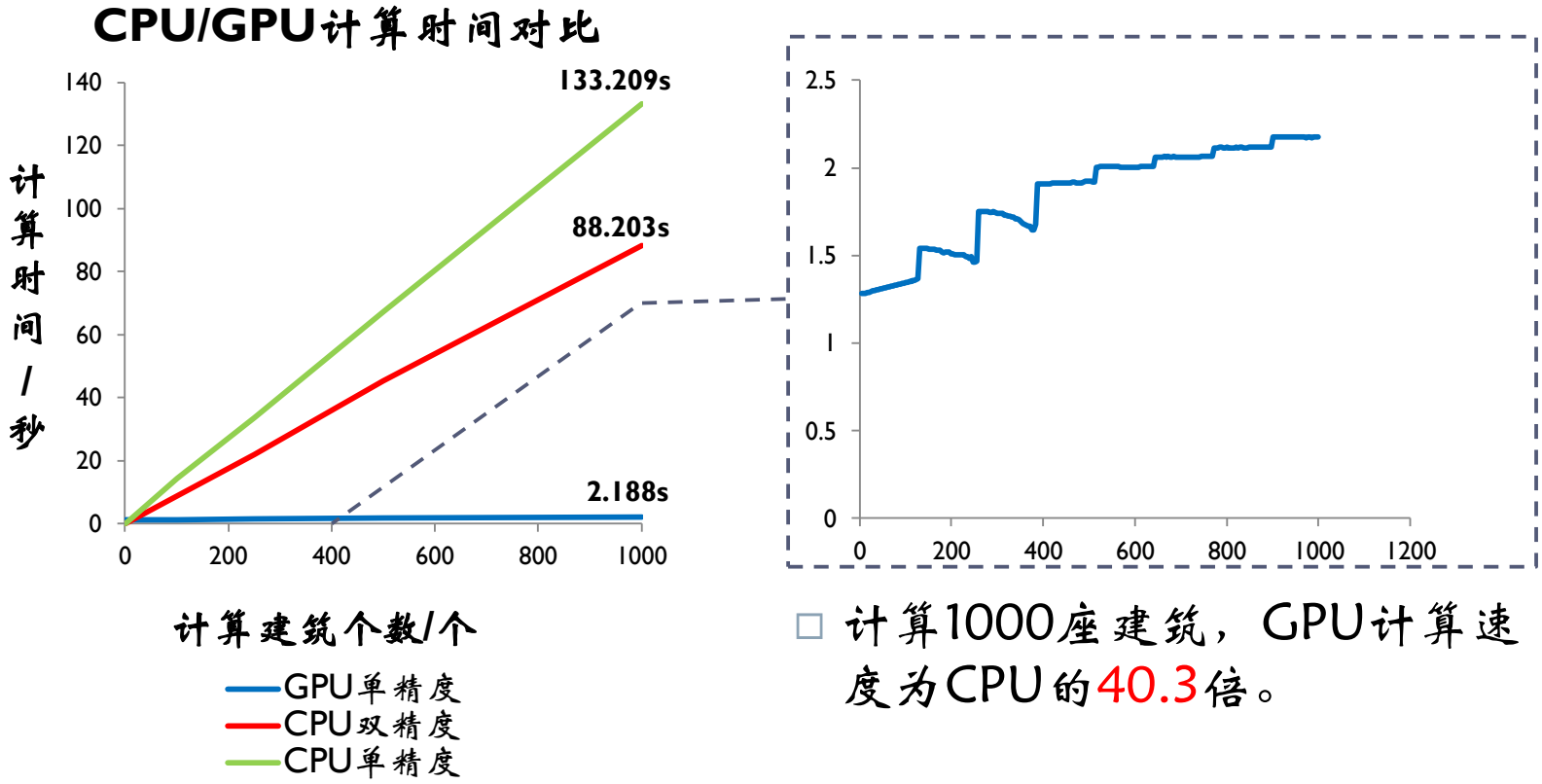
2.弹塑性时程分析程序的并行化

- 程序流程如图



2.弹塑性时程分析程序的并行化

- 程序性能测试



- 计算1000座建筑，GPU计算速度为CPU的40.3倍。

3.程序性能测试

- CPU/GPU计算结果对比（楼层位移）
  - 40.000s（最后一个数据）

位移 (10^{-2})	1层	2层	3层	4层	5层	6层
GPU	0.52284	0.70995	0.24754	0.19642	0.19633	0.19628
CPU	0.52272	0.70970	0.24729	0.19616	0.19607	0.19602
误差	0.02%	0.04%	0.10%	0.13%	0.13%	0.13%

  - 累积误差较小，可接受

内容提要

1.课题简介

2.弹塑性时程分析程序的并行化

3.地震响应展示程序的实现

4.程序实例

5.总结

内容提要

1.课题简介

2.弹塑性时程分析程序的并行化

3.地震响应展示程序的实现

4.程序实例

5.总结

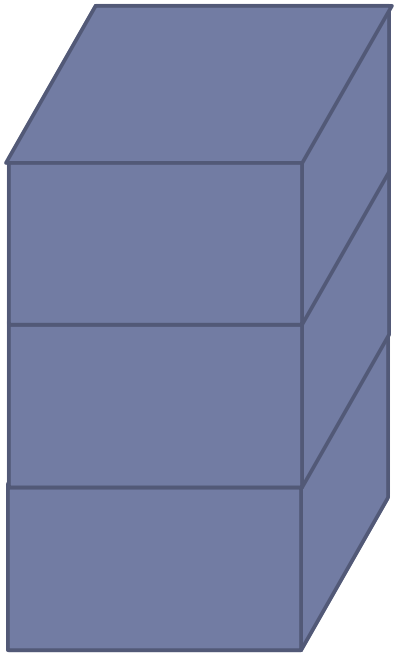
3.地震响应展示程序的实现

- 展示程序实现

  1. 静态展示→三维建模

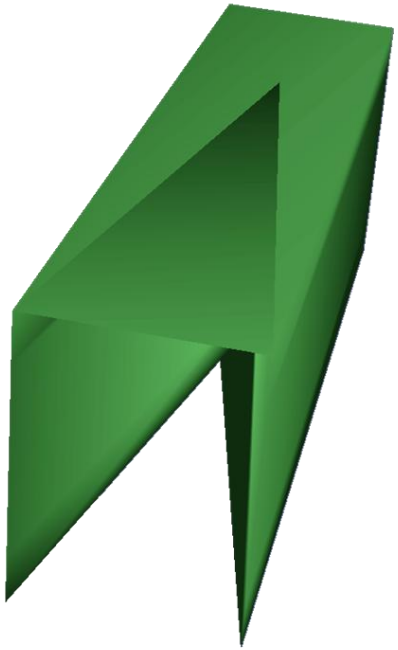
  2. 动态展示→动态更新响应

- 最直接的方法
  - 转角点坐标、层高、层数
  - 所有顶点编入同一Geometry
  - 定义顶点更新序列，进行更新



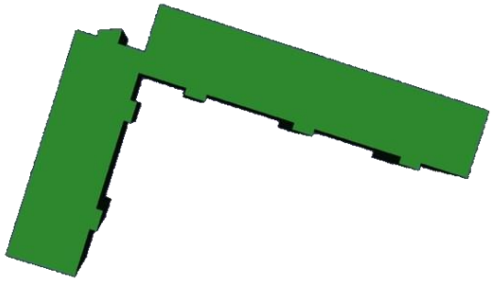
3.地震响应展示程序的实现

- 问题：
  - 顶面，底面存在凹多边形
  - OSG并不支持直接绘出



3.地震响应展示程序的实现

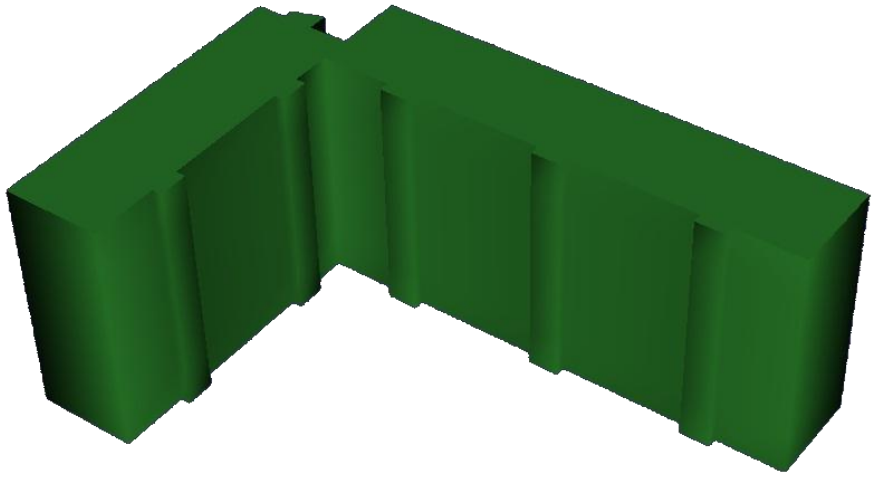
- 解决方案：分格化
  - 1.自编程序分格化
    - 顶点数量增加，且增加数量不定
    - 不利于编写顶点更新序列
  - 2.采用OSG的osgUtil::Tessellator类
    - 应用于Geometry
    - 可处理凹多边形和自交多边形



```
ref_ptr<Tessellator>tscx=new Tessellator();
tscx->setTessellationType(Tessellator::TESS_TYPE_GEOMETRY);
tscx->setBoundaryOnly(false);
tscx->setWindingType(Tessellator::TESS_WINDING_ODD);
tscx->retessellatePolygons(*(geom.get()));
```

3.地震响应展示程序的实现

- 静态展示
  - 1. 侧面：同一个Geometry，所有建筑的侧面
  - 2. 顶面+底面：单独建立Geometry，并应用Tessellator



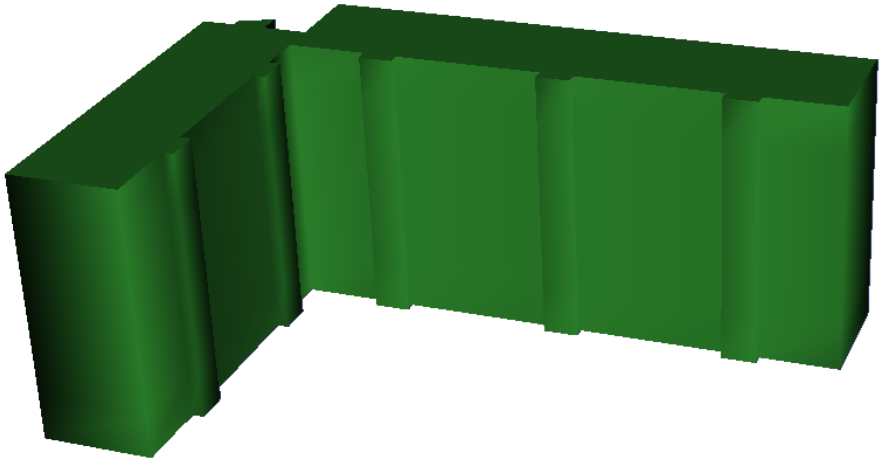
3.地震响应展示程序的实现

- 动态展示
  - 顶面、底面和侧面不共享顶点
  - 无法采用统一的更新序列
- 采用osg::Drawable::UpdateCallback
  - 分别定义不同的更新类，重载其中update函数

```
virtual void update( osg::NodeVisitor* nv, osg::Drawable* drawable )
{
        osg::Geometry* geom = dynamic_cast<osg::Geometry*>( drawable );
        if ( !geom ) return;
        osg::Vec3Array* vertices = dynamic_cast<osg::Vec3Array*>( geom-
>getVertexArray() );
        if (vertices)
                for (int i=0;i<vertices->getNumElements();i++)
                        (*vertices)[i].set(x[i][period],y[i][period],z[i][period]);
        vertices->dirty();
}
```

3.地震响应展示程序的实现

- 动态展示



3.地震响应展示程序的实现

- 相机操作
  - 采用osg::Camera::DrawCallback类
  - 重载operator ()函数
  - 添加截图等操作
    - 产生bmp序列

```
itoa(frame,name,10);
strcat(name,".bmp");
viewer->getCamera()->setFinalDrawCallback(new WindowCaptureCallback(GL_BACK, name));
viewer->renderingTraversals();
frame++;
if (frame>40) frame=0;
```

内容提要

1.课题简介

2.弹塑性时程分析程序的并行化

3.地震响应展示程序的实现

4.程序实例

5.总结

内容提要

1.课题简介

2.弹塑性时程分析程序的并行化

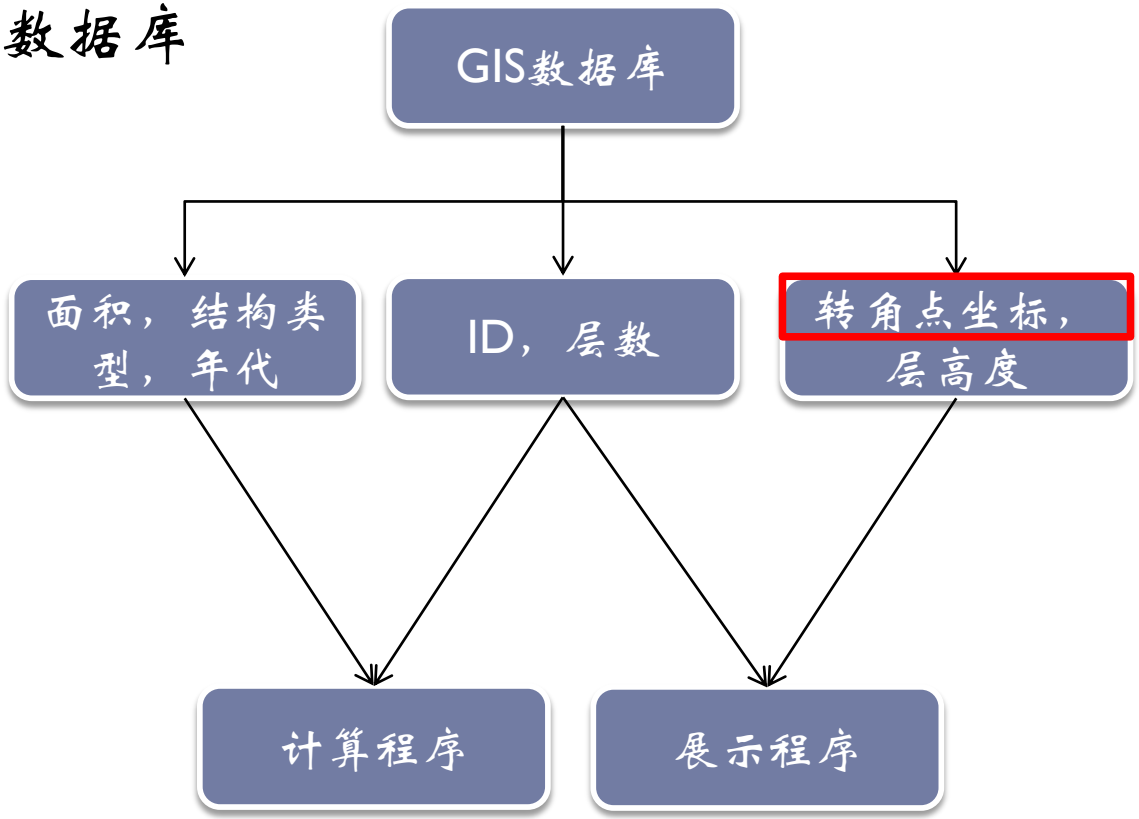
3.地震响应展示程序的实现

4.程序实例

5.总结

4.程序实例

- 数据来源
  - GIS 数据库

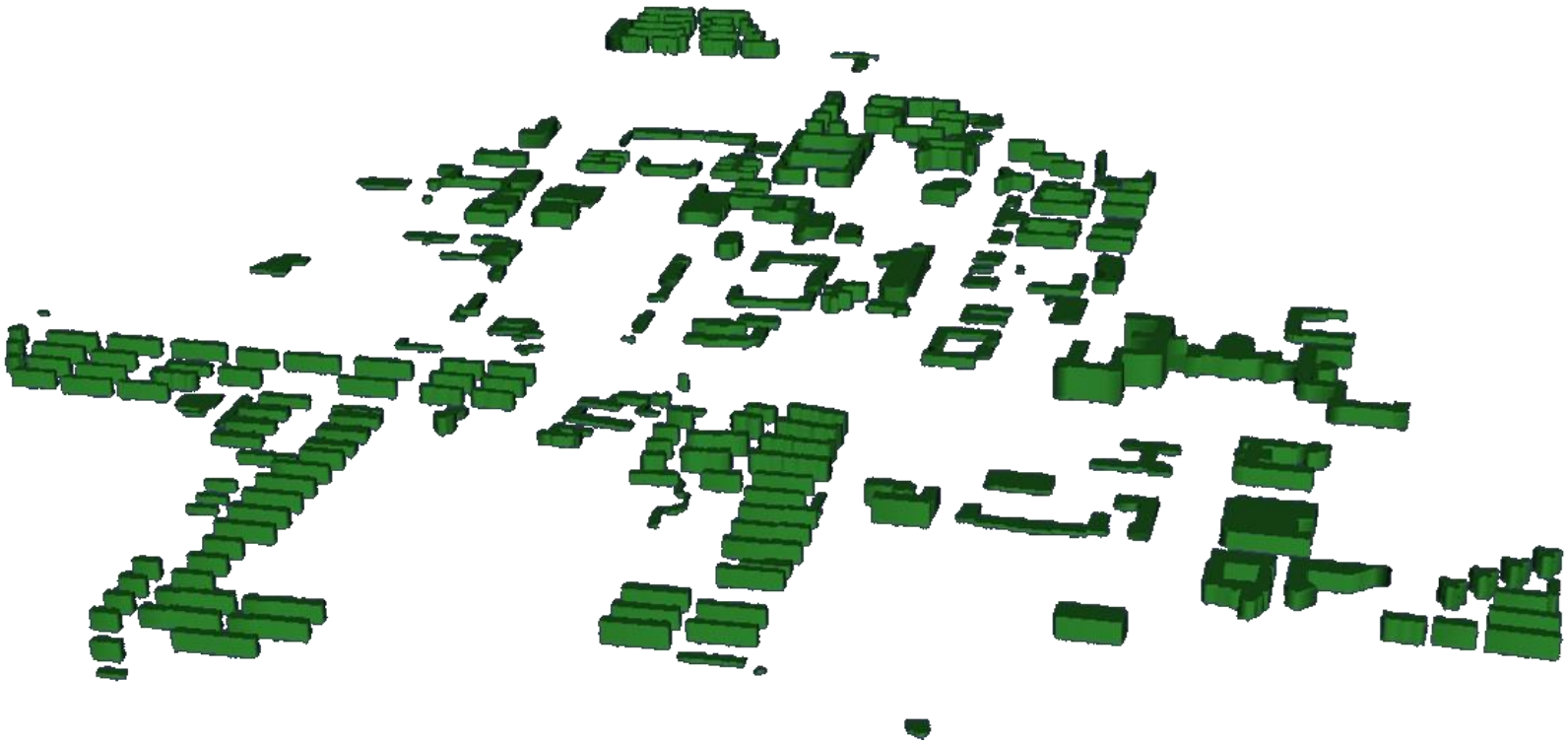


采用ArcGIS自带描点插件获得

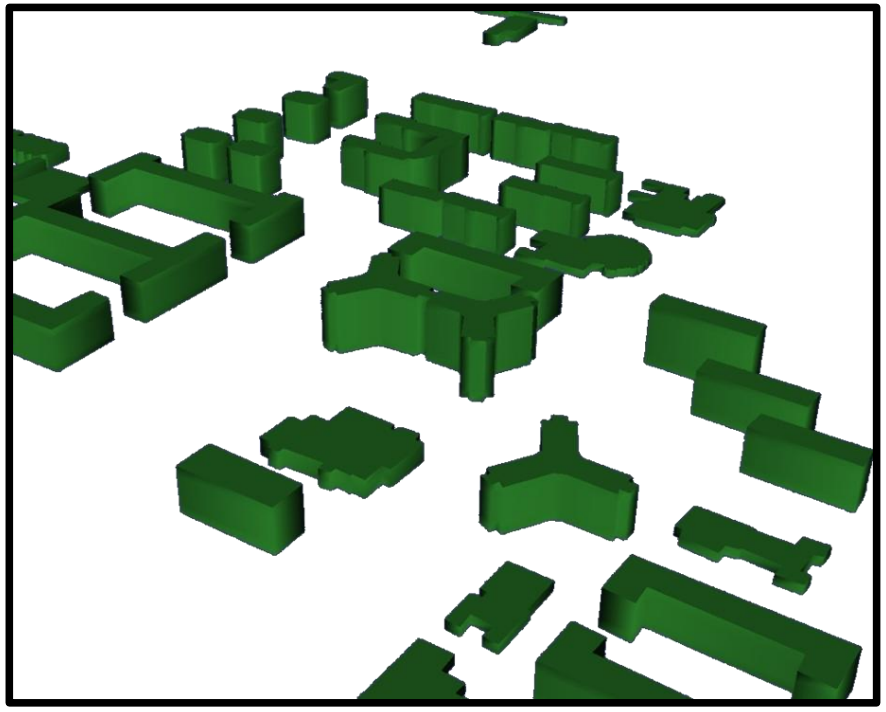
4.程序实例

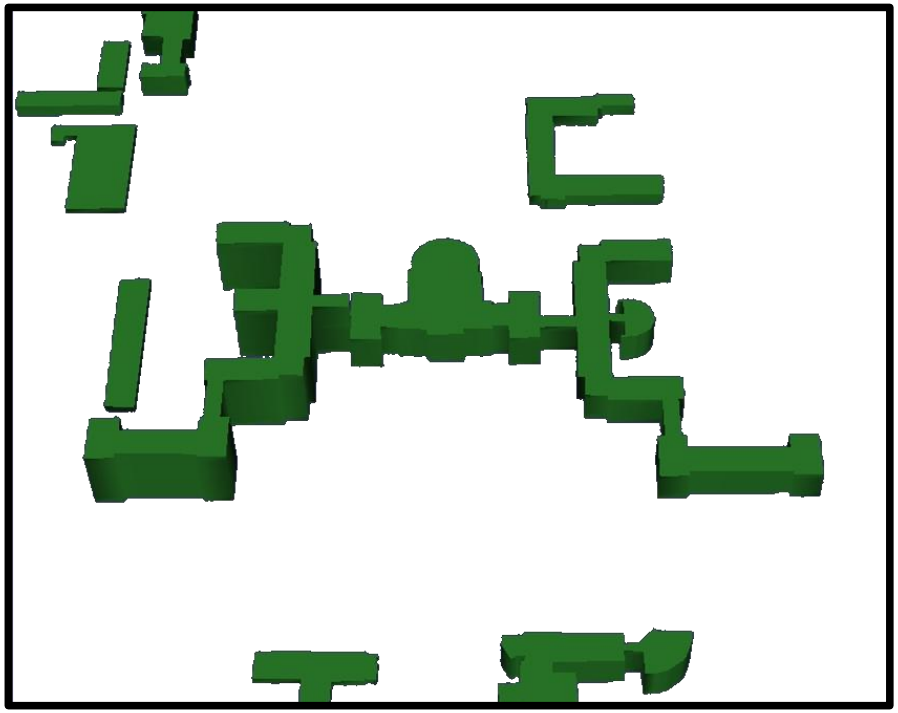
- 展示成果：清华
  - (1) 静态模型
    - 36967个顶点
    - 21356个四边形
    - 769个三角形

4.程序实例



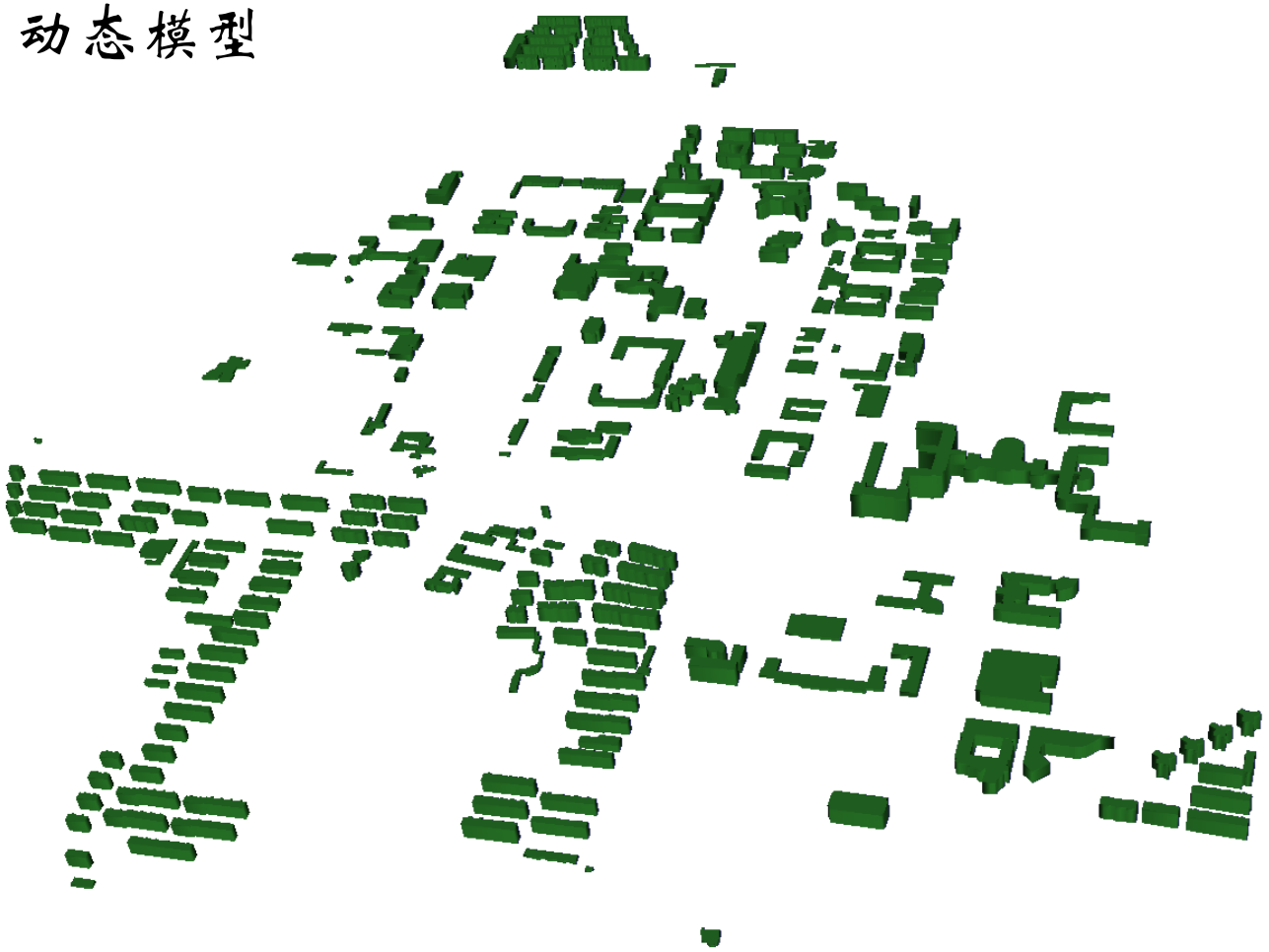
4.程序实例



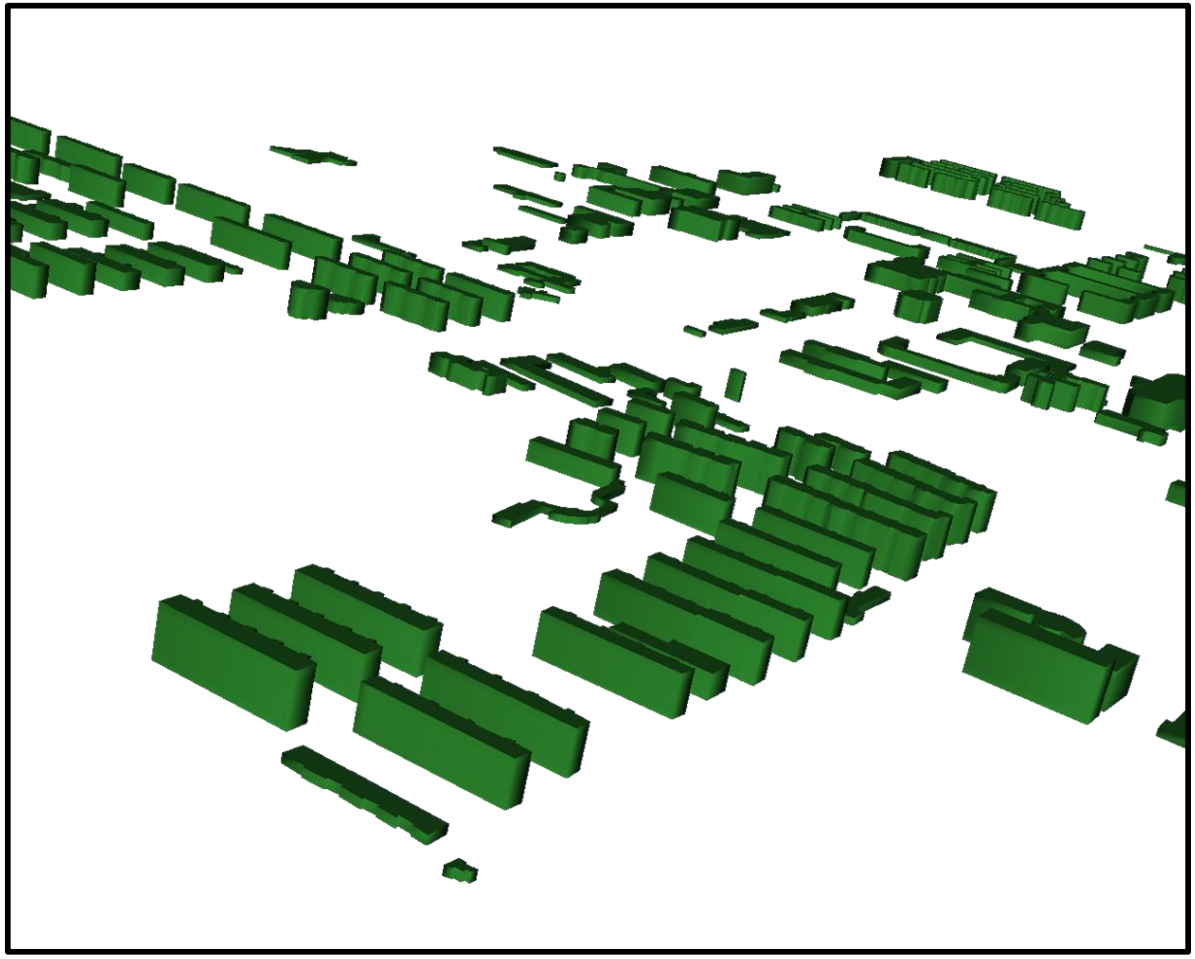


4.程序实例

- (2) 动态模型

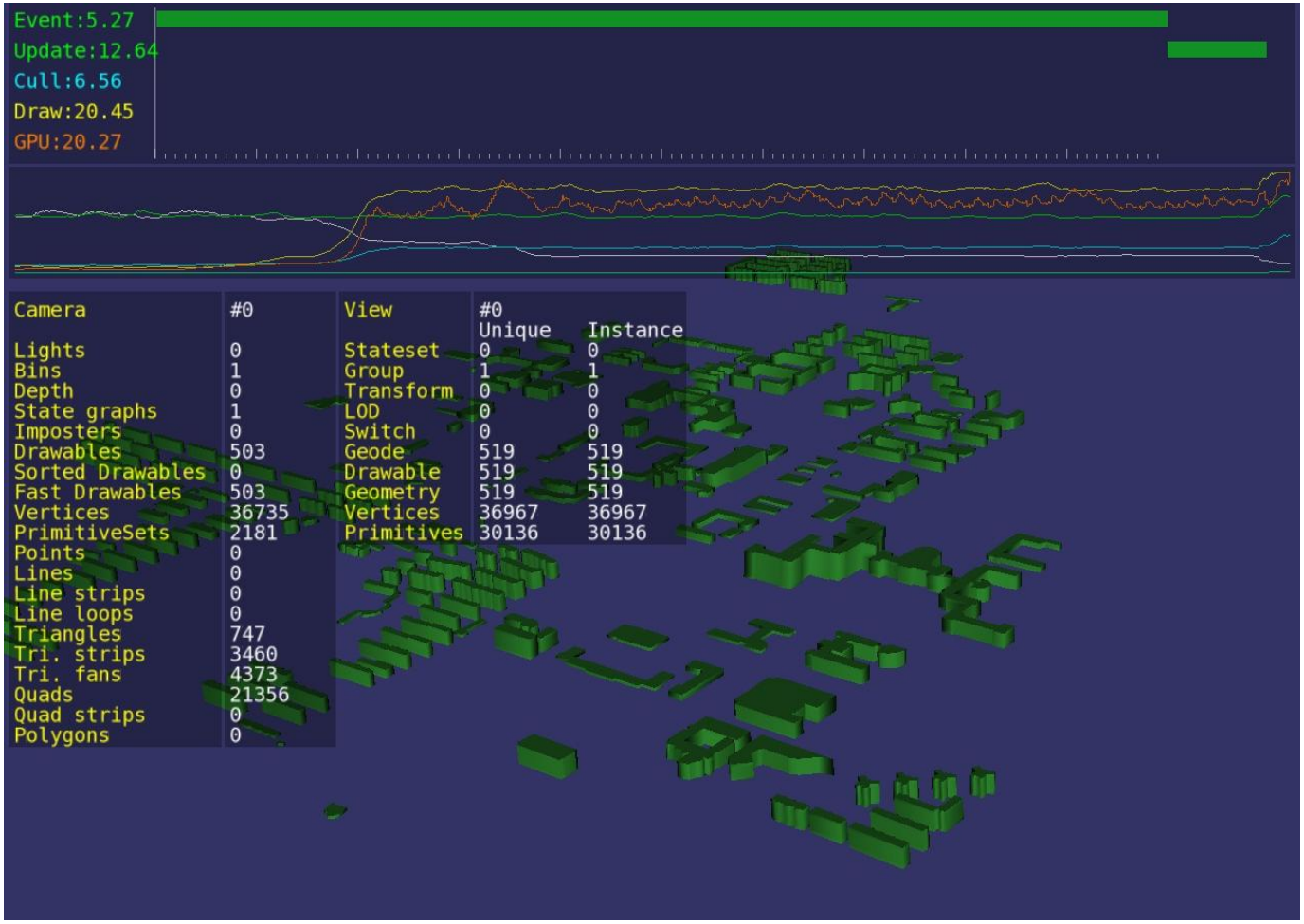


4.程序实例



4.程序实例

- 动态渲染帧速可达30fps



内容提要

1.课题简介

2.弹塑性时程分析程序的并行化

3.地震响应展示程序的实现

4.程序实例

5.总结

内容提要

1.课题简介

2.弹塑性时程分析程序的并行化

3.地震响应展示程序的实现

4.程序实例

5.总结

5.总结

- 结论：
  - 本课题基于CUDA和OSG，成功完成了城市一般建筑的地震响应的计算模块和展示模块，并进行了测试
    - 使用CUDA计算效率可达CPU的40倍
    - 利用OSG能够较为高效地进行震害模拟
  - 将程序应用于清华的地震响应，得到了较为良好的效果，证明了GPU进行结构地震响应的可行性

5.总结

- 展望
  - 计算模块
    - 进一步完善GPU并行化程序，完善结构参数数据库
  - 显示模块
    - 引入物理引擎，增加倒塌判定和倒塌特效
    - 引入贴图和纹理，使效果更加真实
    - 改进节点更新相应，提升效率

Thank you!

欢迎老师和同学提问