论文技术研究 01

父子： 父子继承 独立和协同

继承和传递： 交互逻辑

LOD Levels of Detail 层级细节

方案

1. 离散 LOD 模型：根据需要使用不同的分辨率副本
2. 连续 LOD 副本：算法实时生成不同分辨率 mod，据视角到 mod 距离调整分辨率
3. 多分辨率：对单个 mod，局部分辨率根据视角远近显示不同层次细节 - 对单个 mod 不同 part 与视角有关且连续变化

优劣：

1. 闪烁 2. 需优化顶点计算 3. 适用于大场景

不同层级 LOD 模型构建思路

1. 顶点→网格 优化
2. 研究 - 构建 LOD：
   - 2.1 3d mod 几何顶点往平滑 curve 无价值
   - 2.2 网格优化算法，渐近网格概念，生成算法
   - 2.3 基于多分辨率，小波变化的连续细节层次模型算法

动态加载技术

① 预分块 - 低细节层次：背景、扩展视野

\ 高层次： 高精度、漫游沉浸感

\ 思路1：①bxb 宫格划分

②视点移动，计算高模面积，据 S 覆盖加载对应高模

思路2：①大地形分块加载（如果起伏不大）

②切为小地形 ③据相机获小块位置加载附近小块

④新旧内容对比，增新删旧

遮挡剔除算法

core：识别不可见物体，从列表中移除

视锥体裁剪：

1. 视锥体相交部分才会被渲染
2. 优势：易实现，但遮挡物体、复杂场景效率低

空间分割：（Octree、BSP）

1. scene → 空间单元
2. 递归遍历空间分割树，确定可见物体
3. 优势：适用复杂场景，减少计算资源

obj 简介

顶点　v x y z　　float

从顶点列表开头

顶点向量 Normals

vn　x y z

面

f　v1/vt1/vn1　v2/vt2/vn2　v3/vt3/vn3

↓纹理

纹理材质：

mtllib　filename .mtl

usemtl　material_name

纹理坐标（二维）- 0~1

vt　u　v

mtl 介绍

Kd 漫反射光照：基础色 - RGB

Ns 反光度：　粗糙度　光滑　粗　光　(0~1000)

Ka 环境光反射　三个float　0~1　RGB

Ka 0 0.5 0 灰色（反射颜色）

map-Kd 模型表面漫反射纹理映射文件的路径

/

pny.

论文技术探究 02

three.js

1. scene：本体，所有对象的父亲

2. camera：视角，视野范围

PerspectiveCamera　OrthographicCamera

近大远小（人眼）

3. renderer：使用WebGL实现3D图形渲染

scene+3D object → para（画布大小）→ 画布（web）

相机

4. mesh：网格（基本object）

= 几何体+材质

5. object3D / Group：object3D是所有可渲染对象的基类

Group是特殊的object3D

2者皆可承载其它对象，形成层机关系，

城市对象：

实体对象（单个模型）

聚合对象（小的组合）

白模（不重材质、纹理）只重形状

- 区块白模
- 实体白模

---

linux　/home　　win　F:\

- string
- regex
- JSON

} 常转义F:\\

三角剖分

需求：modern engine 支持三点以上面，
但为了兼容性，需要手动划分三角面，避免 three.js
出现错误面

2种剖分算法：

1. 简单剖分

loop
找1个+2个相邻点 ⇒ Δ
if Δ在内部
删这个点
直至点集余下 ≤3

1.1 优化一 凹多边形过滤
three.js 对凸多边形划分不会出问题
经多次剖分，余下凸多边形，不再划分，少1个三角面

1.2 优化二 点密集程度分析
针对那些形似凸多边形，但有凸起的（凹多边形），将这些凸起的轮廓控制点，先处理 ⇒ real 凸多边形

方法 KDE（核密度估计）
一点，高斯核函数叠加
↳ 从密度最高的点，开始剖分

Delaunay

2. Delaunay 三角剖分（适合处理嵌套轮廓）

主要特点：
将点集组织成无重叠的三角形网络

特性
1. 凸性：三角形网络凸的，任意Δ外接圆内不包含其它点.
2. 最小角度：避免瘦长三角形
3. 唯一性：从任一区域开始，结果一致
4. 区域性：增、减、改某一点，只影响临近三角形

论文技战术探究 07

Delaunay 三角剖分流程

输入点集 P

1. 构建超级△（2个）包含 P 在内部 一矩形包围盒
2. Super △ 加入 △ set 中
3. 逐点插入，对于 P 中每一点 p
   - 3.1 找到 △set △的外接圆
   - 3.2 p 在外接圆 >1 时
   - 3.3 删除 对应的公共边（空腔）← 此处 △ 和 P 所处△
   - 3.4 点 p 与空腔轮廓点连接
4. 删除、all super △ 相关 △
5. 输出

3种方法对比

① 优化后简单部分 三角形数 少于其它 2种（显著）

② runtime 三者接近

结果 ① 简单 → 优化后简单部分

② 内轮廓 → Delaunay 三角部分

obj 构建

① 平面轮廓点集 + 高度

② 顶点（底、顶）面生成 → v x y z → .obj

③ normal：垂直顶点所属面（叉乘）→ vn x y z → .obj

④ 顶/底 面片组合：

三角剖分结果作为 face ⇒ .obj

f v索引/n索引

⑤ 侧面面片：顶 2个点 + 底 2个点、

内轮廓 内外高度不一致，加入额外面即可

mtl文件构建　ka (Ambient Color)　(0~1, 0~1, 0~1)

↳ 无直接光照时颜色

kd (Diffuse Color)

↳ 直接光照下的漫反射颜色

ka取 (1,1,1)　ks取 (1,1,1) 反光度权衡

交互式实体对象建模　层级/实体 - 增删

一为了避免权的断层，一拷贝对象做为hat

① 删层次

执行上级与下级包含关系的计算

缺少所属对象则添加一层 hat

② 增层级

一开始非空，会有 hat 拷贝 避免断裂

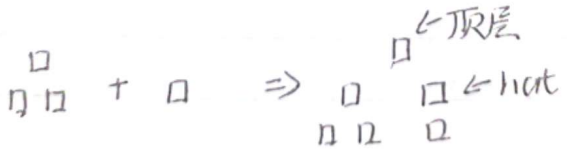
a 增对象

首先找父亲，无则加hat

b 删对象：

删后使用hat替代消失的对象

片区合并　以层级多者为准，补hat，合并树



论文研究 04

响应式动态渲染策略

前提：Sb mod 已是白模，再无细化 → 粗化可能

则使用层次树 配合 相机distance 阈值

↓ ↓

实时 compute

↓

渲染不同层级 mod

算法 ① 从 root 开始 > 阈值 则加载

< 阈值 遍历子节点

② 对比 增删 mod，减少闪烁（不是全部重来）

阈值与体积挂钩 → 外接球

阈值 = 外接球半径 × (2.5 / 3) 倍

↓

需计算 顶点坐标均值（中心）

↓

以最远距顶点为半径

延迟加载

前景 { 变化幅度大，场景多次渲染（load-mol 队列）

不断交换 ⇒ 卡顿

以加场景单位 /s，底层模型变化远大于中高层

↓

解决 使用画面比例 v 变化为依据

① v > 阈值，停止响应式渲染，直至 slow

② 执行缩放 → 层级变（场景单位数变）→ 重算标准速度