一、Instancing

1.基本工具

ThreeJS中提供了两种缓冲区BufferAttribute和InstancingBufferAttribute，BufferAttribute缓冲区用于存放每个网格顶点的信息，InstancingBufferAttribute缓冲区用于存放每个对象的信息。

Instancing方式渲染多个相同对象时，GPU只从内存中取一次数据，减少了GPU的访存次数，所以速度较快，但是传统的GUP创建的对象基本上都是完全相同的，且一般不用于处理带动画的模型。

2.动画处理

gl\_Position = projectionMatrix \* modelViewMatrix\* matrix2 \* matrix1 \* vec4( position, 1.0 );

（1）动画与骨骼的关系

Time=time1+k\*(time2-time1);

Bones[i].position=k\*tracks[time1]+(1-k)\*tracks[time2];

（2）骨骼与点的关系

Matix1=weight[b1]\*bone[k1]+weight[b1]\*bone[k2]+weight[b1]\*bone[k3]+weight[b1]\*bone[k4];

3.贴图多样性

Texture=texture2D(text[type],UV);

4.颜色多样性

gl\_FragColor = vec4 (Texture.r+color[0],Texture.g+color[1],Texture.b+color[2],1.0);

5.将模型分区处理

if(vPosition.y<0.15)type\_part=0.0;//位于下半身  
else if(vPosition.y<0.59) type\_part=1.0;//位于上半身  
else type\_part=2.0;//位于头部

6.shader程序的优化

顶点着色器中的代码每一帧的每个点都会调用一次，因为执行次数非常多，所以这里代码的细微变化会对系统性能产生极大的影响。

shader程序中最好不要出现分支结构，但是有些分支结构对系统性能的影响不大，

二、PM

1.基本工具

PM是建立在QEM模型简化算法的基础之上的，是QEM模型简化算法的逆过程，QEM算法的核心是找出当前最优先的删除点,QEM算法的评价标准是曲度，弯曲程度大的地方后简化，弯曲程度小的地方先简化。

2.PM处理的流程

a将模型格式进行规范化

b使用含有插件的meshlab进行模型简化

c将PM所需信息分段

d在场景中还原模型

3.PM与LOD的结合

PM在场景中还原模型时，可以将中间计算结果存储下来用于LOD

4.PM与视点依赖

PM可以选定某个区域先简化或者最后简化

----视点依赖功能还未实现

精细的模型的处理

PM+LOD

简单相同部件的处理

实例化渲染

大规模人群的处理

细粒度化

提高资源的复用度

3D模型

模型网格

切分

纹理贴图

使用不同纹理贴图进行搭配组合

不同像素精度的多级处理

骨骼动画

预处理

设计了专门用于骨骼动画的数据存储格式

人群

资源复用->3D模型分级复用

第一级，所有对象都共用的数据 网格（UV）、骨骼

第二级，选择性使用的数据 纹理贴图、动画

人物多样性->参数化差异

坐标变换差异（变换矩阵从移动、放缩、旋转三个维度丰富多样性）

第一级，人物坐标变换

形态差异

通过放缩实现人物整体上的形态差异

位置差异

人物的移动和旋转

第一级，网格点坐标变换（通过骨骼来实现）

骨骼导致的形态差异

动画差异

动画播放速度

播放的动画类型

材质差异

纹理贴图的搭配组合

色调调整

色调对换

色调加强