第37卷 第16期
Vol.37 No.16

计 算 机 工 程
Computer Engineering

2011年8月
August 2011

·开发研究与设计技术· 文章编号：1000—3428(2011)16—0276—03 文献标识码：A 中图分类号：TP391

大尺度空间场景可视化中浮点精度问题研究

汪荣峰，廖学军

(装备指挥技术学院试验指挥系，北京 101416)

摘 要：在具有极大数值范围的空间场景中实现可视化时，为避免图形处理器以32 bit单精度浮点表示所产生的“Wobbling”、“Pieces”等现象，提出分组分次绘制的解决方法。针对上述现象产生的原因，将场景中的对象分为大实体和小实体，根据空间场景的特征和对象之间的关系将场景中的对象分组，设计组内对象的绘制方法。实验结果表明，基于以上方法实现的空间场景可视化图像正确稳定。

关键词：空间场景；可视化；浮点精度；图形处理器

Research on Floating Point Precision Problem for Large-scale Space Scene Visualization

WANG Rong-feng, LIAO Xue-jun

(Department of Testing and Command, Academy of Equipment Command and Technology, Beijing 101416, China)

【Abstract】To avoid “Wobbling” and “Pieces” phenomena generated by 32 bit representation of Graphics Processing Unit(GPU) when space scene visualization with high numerical range is implemented, a draw solution with grouping and gradation is put forward. The objects in scene are classified as a big entity and a small entity based on reasons of phenomena above. Objects are grouped according to relations between objects and characteristic of space scene. The draw method for objects in group is designed. Experimental results show that the realization of space scene visualization image based on the method introduced above is correct and stable.

【Key words】space scene; visualization; floating point precision; Graphics Processing Unit(GPU)

DOI: 10.3969/j.issn.1000-3428.2011.16.093

1 概述

不同于传统的陆海空战场[1]，空间场景中的对象具有较大的尺度和数值变化范围。太阳、地球、月球等天体的尺寸和运动轨道的半径等都需要用极大的值来表示，而卫星、空间站等人造实体的组成部分常需要以厘米甚至毫米表示。本文分析空间场景可视化中由于这些因素而产生的问题，并结合空间场景的特征，给出解决方法。

2 场景可视化中存在的问题

2.1 现代GPU的浮点精度问题

可视化技术与应用的发展离不开硬件的支持，空间场景的可视化也与计算机图形处理器(Graphics Processing Unit, GPU)密切相关。虽然目前GPU具有很高的主频、足够的带宽、很强的可编程能力，能支持更多的显存、64 bit的纹理，但是在可预见的一段时间内，其所支持的浮点精度仍将是32 bit[2]。对于大部分的图形绘制任务，如小范围战场可视化等，单精度浮点已经足够，但是在空间场景可视化中，将会出现明显问题。

在现代图形绘制流水线中，图1是输入图元的顶点坐标变换到输出窗口的屏幕坐标的典型过程。典型情况下，将根据视点等参数的设置生成模型视点变换矩阵和投影变换矩阵，这些矩阵将在图形绘制流水线中被应用到每个输入顶点。即使顶点在CPU中是以64 bit浮点数表示，但在GPU中，顶点的表示和变换矩阵都以32 bit浮点数表示。

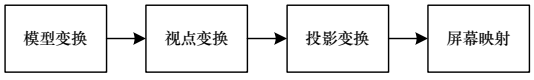


图1 模型顶点坐标变换屏幕坐标的过程

当场景中同时存在尺度差异较大的实体时，如地球和人造卫星，当人造卫星距离视点较远时，可以采用例如圆圈的某种简洁图形加文字表示。随着视点改变，如果文字是以位图形式实现，则会出现微小的抖动，虽然可以利用矢量字体来消除抖动，但矢量字体对生成比较小的字的效果并不好。总体而言，这种文字的抖动不会超过一个像素，很不明显，属于视觉可以接受的范围；当人造卫星距视点较近时，场景中需要同时绘制天体和人造卫星的三维模型，由于天体尺寸远大于卫星尺寸，而GPU所支持的IEEE单精度浮点标准只有23 bit用于表示尾数，因此小的数值将被大的数值“淹没”，超出精度范围导致的问题是当视点移动时产生明显的“Wobbling”现象，在屏幕上的抖动可达几个像素之多，人的视觉将无法容忍。

假设整个空间场景采用的坐标系为地心坐标系，当不加任何处理绘制人造卫星和地球时，地球半径约为6 371 001 m，而此时构成地球实体的三角形顶点的坐标也是这个尺度，假设人造卫星在某时刻距离地面800 km，其中，某个顶点坐标为(6 378 001.01, 0, 0)(实体坐标点并非此值，而是经流水线坐标变换之后的理论值)，此时，需要的浮点精度至少要9 bit的有效数字，超出GPU表达能力，这就是“Wobbling”现象产生的原因。

文献[3]在实现全球地形的实时可视化中对此问题进行分析，32 bit浮点精度处理全球数据所能提供的最高精度约

基金项目：国家部委基金资助项目

作者简介：汪荣峰(1973－)，男，副教授、硕士，主研方向：计算机图形学，空间信息处理系统；廖学军，教授、博士、博士生导师

收稿日期：2011-02-25 **E-mail：**wangrongfeng@sina.com

为 0.5 m。显然，在空间场景可视化中单精度浮点数所能支持的精度要远低于全球地形可视化中所要求的精度。

2.2　Z 缓存有限位数和非均匀分布造成的现象

在目前的图形绘制流水线中，经观察变换、视点变换和投影变换将图元坐标变换到单位立方体中，而光栅化时的消隐依赖于投影变换产生的 Z 值。但表示 Z 值的浮点数最多也只能为 32 bit，而且投影变换所产生的 Z 值在观察坐标系的近平面和远平面之间并非均匀分布。当绘制大尺度范围的空间场景时，近平面和远平面之间的比值可能非常大。如果一个天体所处的位置满足如下条件：(1)天体在屏幕上具有一定投影面积，应该显示一定细节；(2)表示天体的点的坐标投影得到的 Z 值非常接近，其差距超出 Z 值表示的精度范围。此时，图形绘制流水线将不能正确地计算天体组成图元的位置关系，消隐不正确，视点不动时天体表现的如同碎片(Pieces)或显示混乱(Confusion)，碎片现象如图 2 所示，而当视点移动时该天体将产生忽隐忽现的闪烁(Flickering)现象。

图 2　Z 值的单精度表示和非均匀分布造成的碎片现象

3　分组分次绘制的解决技术

工程上常用的解决类似问题的技术有 2 种：(1)不同的模型在各自的局部坐标系下进行绘制。这种办法确实可以避免上述现象，但会导致整个场景消隐不正确，理论上按模型到视点的距离绘制可以保证消隐正确，但各实体有时会交错遮挡，如天体会遮挡一部分轨道，当视点以比较倾斜的角度观察地表时，地球距离视点更近，将导致地表实体与地形之间消隐错误。(2)将所有模型转换到以视点为原点的坐标系。当视点距离地球表面较近，且同时显示地形和实体时，这种方法可以有效避免“Wobbling”现象，但当卫星距离视点很近、场景中其他天体也需要显示时，“Wobbling”现象非常明显，同时因 Z 缓冲精度不足而导致的各种现象也很明显。

由以上分析可知，已有方法可解决一定尺度范围或实体间不存在交错遮挡的场景的可视化，但对于解决大尺度空间场景可视化还是无能为力，即使根据视点位置综合运用各种方法，仍然无法彻底解决问题。正如上述方法(1)所述，多次绘制是解决问题的有效技术，如 BSP 树[4]等一些空间数据结构，可将场景中几何数据组织起来，通过该数据结构确保按距离视点的远近顺序绘制图元，但需要针对图元进行多次绘制，这样做显然代价太大。

因此，本文采用的对策是将场景中的实体根据其相关关系进行分组，不同组之间不会出现交错遮挡，对排序后的各组分别绘制，多次绘制的结果构成整个可视化空间场景。每次绘制时都将 Z 缓冲区清除，以充分利用 Z 值精度。

由于每次绘制是相对独立的，在局部坐标系进行的绘制，分组内的坐标值范围比较接近，因此可以去除影响精度数值部分(分组内还需进一步处理，见第 5 节)，从而保证图形流水线中的变换矩阵和顶点坐标的精度。

在相对独立的坐标系下，同样可更有效地控制视锥近、远平面的取值。由于 Z 值的单精度浮点表达造成“Flickering”现象，当在局部坐标系下绘制实体时，不是设置固定的近、远平面值，而是根据实体的包围盒在视线方向的投影距离来设置近、远平面，这样的范围远远小于整个场景的近、远平面范围，因此可以充分利用 Z 缓存，达到正确消隐的目的。

4　场景中对象的分组

空间场景较之地面场景，显示内容相对单一[5]，因此可基于空间场景特性与对象关系设计一种比一般空间数据结构更简洁的分组算法，保证正确的分组间消隐关系。

4.1　对象的定义

空间场景需要显示的对象主要有天体、航天器、轨道、传感器和地面目标，这几类对象都由公共对象派生，图 3 表示其类图。传感器只考虑与航天器之间存在依赖关系，而未支持地面传感器；虽然天体也存在运行轨道，但只考虑航天器轨道；地面对象包括车辆、飞机等，其与天体间依赖关系的定义也是为后续的局部坐标系绘制。

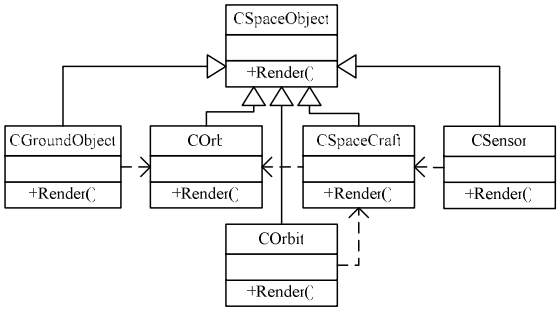


图 3　场景中对象类图

4.2　对象的分组策略

结合空间场景的特点以及对象之间的关系，对象的分次绘制主要应用以下策略：

(1)对象分类。将所有需要绘制的对象分为大实体和小实体 2 类，小实体指航天器、地面目标等模型精度达米级以下的实体；大实体指天体、轨道和传感器(主要是绘制传感器探测范围，并非传感器自身)等模型尺度很大的实体。根据第 2 节分析可知，小实体主要会产生“Wobbling”现象，大实体则会产生“Flickering”现象。

(2)利用实体间的依赖关系进行对象分组。分次绘制的难点在于有些对象存在交错遮挡关系，对于这类对象的解决，必须将对象分到同一组，可利用对象之间的依赖关系进行分组，这样比基于距离进行分组更为合理和有效。

(3)分组内透明对象的处理。传感器主要是以半透明形式表现传感器能力，为保证透明表面后的实体不会由于先计算透明物体的 Z 值而被遮挡，一般是先绘制完不透明实体之后再绘制透明物体。故传感器的探测范围也在分组内所有非透明实体绘制完之后再绘制。

4.3　对象的分组算法

定义如下数据结构管理同次绘制的所有对象，其中，SpaceObjectVector 为对象数组。

```
SpaceObjectGroup={
minDistance, maxDistance:double;
SpaceObjects:SpaceObjectVector;}
```

定义 GroupData 为 SpaceObjectGroup 结构的数组，用来管理所有的分组，将场景中所有对象进行分组的算法如下：

(1)对于每个天体，建立一个新的分组并加入到 GroupData，分组中的距离设为天体到视点的最近和最远距离。

(2)对于其他的每个大实体，根据依赖关系加入对应天体分组，不改变分组的距离值。

(3)计算所有小实体包围盒(用球表示)在屏幕的投影，当投影面积大于阈值时，需要绘制实体模型，称为“模型可见实体”，如果这类实体没有地面目标，则转(4)，否则转(5)。

(4)对于每个小实体，判断其是否可见，判断方法是计算视点到该实体中心的射线是否与实体所依赖的天体有交，如果可见，则建立一个新的分组并加入到 GroupData，否则不必处理，转(6)。

(5)对于每个小实体，如果可见，且所依赖的天体与模型可见的地面目标所依赖的天体相同，则将其分组到该天体的 CSpaceObjectGroup 中，否则建立新分组并加入 GroupData，转(6)。

(6)GroupData 中的数据按距离排序。

5 分组内对象的绘制

根据图形绘制流水线的工作原理可知，分组绘制时需要保持视点到实体的距离、视线、视角、窗口大小等参数与整个场景的一致，而近、远平面距离只会影响裁剪和消隐，对窗口图像的大小等没有影响，同时根据前面的分析可知，如果仅改变近、远平面距离，只能有效消除第 2 节中的第 2 个问题，而对于第 1 个问题则无能为力。故分组内绘制主要是调整视点、近、远平面距离 3 个参数决定。

由于“Wobbling”现象由小实体产生，因此根据分组内是否存在小实体确定的绘制方法略有区别：(1)如果分组内没有小实体，取分组内天体中心作为新的坐标系的原点；(2)如果分组内有小实体，取距离视点最近的小实体中心为新的坐标系的原点。

绘制算法如下：

(1)在本分组绘制时，清除深度缓存，不清除颜色缓存。

(2)设地心坐标系视点位置为$(x_{eye}, y_{eye}, z_{eye})$，新坐标系原点在地心坐标系下的坐标为$(x_{new}, y_{new}, z_{new})$，则新的视点为$(x_{eye}+x_{new}, y_{eye}+y_{new}, z_{eye}+z_{new})$，新的观察中心为$(x_{new}, y_{new}, z_{new})$，保持视点到实体的距离和视线不变，视角等参数保持不变。

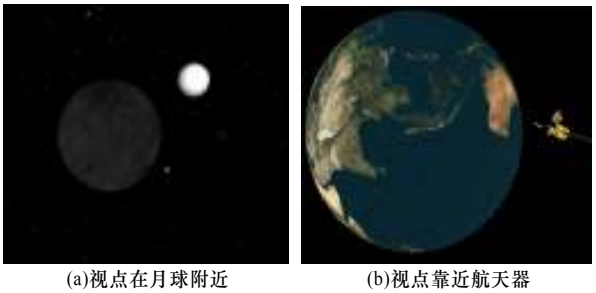
(3)计算所有实体包围盒沿视线方向的投影距离，并根据其投影距离设置近、远平面值。

(4)绘制分组内所有不透明的实体，如果分组内有小实体，则每个顶点的坐标减去矢量$(x_{new}, y_{new}, z_{new})$，否则按原坐标绘制并进行相应的平移变换，对象的旋转变换正常进行。

(5)绘制透明实体，顶点坐标处理等与(4)相同。

在以上过程中，第(3)步设置的近远平面范围远小于整体绘制的范围，解决 2.2 节的问题；当存在距离视点较近的小实体时，第(4)步、第(5)步压缩坐标值的范围，解决 2.1 节的问题。

基于上述技术实现空间场景的显示，图 4(a)同时绘制太阳、地球和月球，视点在月球附近，图 4(b)同时绘制地球和航天器，视点距离航天器很近。



(a)视点在月球附近 (b)视点靠近航天器

图 4 空间场景可视化效果

6 结束语

本文以具有极大数值变化范围的空间场景为基础，分析其在单精度浮点的 GPU 中实现可视化时所产生的“Wobbling”、“Pieces”等现象，并针对这些现象提出分组分次绘制的解决方法，实验结果证实该方法的有效性。下一步将结合 Vertex Shader 方法并在场景中加入更多空间对象，为更好地实现可视化进行更深入的研究。

参考文献

[1] 曾 鹏, 陈长征, 李苏军. 基于数字地球的虚拟海战场环境仿真[J]. 计算机工程, 2009, 35(8): 269-270.

[2] 吴恩华. 图形处理器用于通用计算的技术、现状及其挑战[J]. 软件学报, 2004, 15(10): 1493-1504.

[3] Lindstrom P, Koller D, Ribarsky W, et al. An Integrated Global GIS and Visual Simulation System[EB/OL]. (2007-11-07) http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.6067&rep=rep1&type=pdf.

[4] Berg M D, Kreveld M, Overmars M, et al. 计算几何——算法与应用[M]. 邓俊辉, 译. 2 版. 北京: 清华大学出版社, 2005.

[5] 郑 娟, 姜振东. 空间战场可视化系统的设计与实现[J]. 计算机仿真, 2005, 22(1): 36-39.

编辑 刘 冰

(上接第 275 页)

参考文献

[1] 封松林, 叶甜春. 物联网/传感网发展之路初探[J]. 中国科学院院刊, 2010, 25(1): 50-54.

[2] 任丰原, 黄海宁, 林 闯. 无线传感器网络[J]. 软件学报, 2003, 14(7): 1282-1291.

[3] 周 笑, 李 明, 卜佳俊, 等. 移动远程医疗监护系统的设计与实现[J]. 计算机工程, 2010, 36(10): 251-253.

[4] 毛 玲, 张国敏, 孙即祥. 一种心电信号 QRS 复杂形态自动分析算法[J]. 信号处理, 2009, 25(11): 1680-1685.

[5] Kohler B U, Henning C, Orglmeister R. The Principles of Software QRS Detection[J]. IEEE Engineering in Medicine and Biology, 2002, 21(l): 42-57.

[6] Mallat S, Zhang Sifeng. Characterization of Signal from Multiscale Edge[J]. IEEE Transaction on Pattern Analysis and Machine Intelligence, 1992, 14(7): 681-693.

编辑 刘 冰