基于动态 Impostor 技术的树木快速绘制方法

孙学波， 张小苏
(辽宁科技大学 计算机科学与工程学院，辽宁 鞍山 114051)

摘 要：树木快速绘制技术研究一直以来都是计算机图形绘制的一个热点领域，因为树木复杂的细节结构使其在绘制上存在着相当大的困难。提出了一种基于动态 impostor 技术的树木绘制方法，不仅可以快速地绘制树木，而且具有较高的真实感效果。该方法首先基于树木模型进行平面投影去创建 impostor 平面，然后将纹理映射到已经创建好的 impostor 平面上，最后通过绘制 impostor 纹理平面形成树木图像。当参考视点移动时，可以根据误差来决定是否动态更新 impostor 平面的内容。

关键词：树木绘制；动态更新；impostor 技术；真实感效果；纹理平面
中图法分类号：TP391.41 **文献标识码**：A **文章编号**：1000-7024 (2008) 12-3126-04

Method on fast tree rendering based dynamic impostor technology

SUN Xue-bo, ZHANG Xiao-su
(School of Computer Science and Engineering, University of Science and Technology Liaoning, Anshan 114051, China)

Abstract: Research on fast tree rendering technology is a focus field of computer graphic rendering all the time, because complicate detail structure of tree makes it very difficult to rendering. A method on tree rendering based on dynamic impostor technology is presented, which not only can fast rendering tree, but also has highly realistic effect. This method firstly projects to the plane to create the plane of impostor based tree model, secondly maps the texture on the plane of impostor which has created, thirdly forms tree image through rendering impostor texture plane. When reference viewpoint moves, it can decide whether dynamically update the content of impostor plane according error metric.

Key words: tree rendering; dynamically update; impostor technology; realistic effect; texture plane

0 引 言

用计算机绘制逼真的场景是计算机图形绘制领域的一个非常重要且富有挑战性的课题，这种挑战性源于场景的多样性和绘制场景的复杂性。树木是自然场景的重要构成因素，其种类繁多、形态各异、复杂的结构使其无论在造型、存储还是在绘制上都存在相当的困难，因此各种加速树木绘制性能的技术日益丰富起来。树木快速绘制方法包括基于几何简化的绘制技术、基于图像的绘制技术、点绘制技术、体纹理技术。

基于几何简化的绘制技术是在不影响画面感知效果的条件下，通过逐次简化场景的表面细节来减少需要绘制的场景面片数目，从而提高绘制算法的效率。如果模型离视点较远，且在屏幕空间的投影区域覆盖较少的像素，则用粗糙的模型来进行表示；如果模型离视点较近，则采用精细的模型进行绘制。

Marshall 提出了一种用不规则四面体拟合植物的方法[1]，Lluch 提出了过程多精度模型的方法[2]，Remolar 提出了一种基于树木简化算法的多精度模型[3]。但是如果为了追求速度从而加大简化面片的数量则会造成严重失真的现象。

基于图像的绘制技术是从一些预先生成好的真实感图像出发，通过一定的插值、混合、变形等操作生成一定范围内不同视点处的真实感图像。Max 提出了并行投影 z-buffer 图像[4]以及多层 z-buffer 的方法[5]；Meyer 提出了一种基于公告板的方法[6]，Lluch 提出了一种基于层次图像的方法[7]。这种方法不足之处在于只适用于静态场景，而且视点被限制在一定范围内。

体纹理技术是从 Kajiya 和 Kay 为了绘制毛皮表面提出纹元 (texel) 模型发展而来的[8]。Neyret 提出了用体纹理绘制植物的方法[9]。后来这个模型被发展成体纹理，使得它可以更方便地被用来构造植物等复杂场景。这种方法虽然可以消除几何面造成的混淆，能够构造和准确绘制高度复杂的自然场景。缺点则是由于绘制算法实现比较复杂，运算时间复杂度比较大。

Levoy 和 Whitted 提出了点绘制技术[10]。点绘制技术是从三维离散点采样直接重建出三维空间并投影至二维屏幕进行计算。Weber 提出了基于点绘制的建模和绘制树木算法[11]，Deussen 提出了利用点线等基本元素来绘制复杂植物的方法[12]。作为一种新的技术，在很多地方还远未完善。比如当视点离

收稿日期：2007-12-14 E-mail：pupoly@163.com
作者简介：孙学波 (1964－)，男，辽宁鞍山人，硕士，副教授，研究方向为虚拟现实、人工智能理论；张小苏 (1981－)，男，硕士研究生，研究方向为计算机图形学、真实感图形绘制。

模型非常近或非常远时，怎样克服视觉走样；如何确定采样密度又确保预处理时间不会成倍增加。

本文简要地讨论了各种加速树木绘制性能的技术，分析了这些方法的优缺点。在此基础上，提出了一种基于动态impostor技术的树木绘制方法，不仅可以快速地绘制树木，而且具有较高的真实感效果。这种方法通过纹理映射创建绘制所需要的impostor平面，并且当参考视点进行移动时，可以根据误差来决定是否动态更新impostor平面的内容。这种方法的绘制花费同树木的原始几何面片数没有直接关系，只与在屏幕上形成的impostor纹理平面分辨率有关。

1 Impostor技术简介

使用二维impostor技术在计算机动画和游戏中变得越来越普遍，它的目标在于：通过使用缓冲存储部分场景内容为impostor纹理图像从而减少三维场景的几何复杂性，并且在帧连续的时候可以重复使用前面帧的光栅化图像，这样就可以加速绘制场景从而减少绘制场景的时间。

1.1 Impostor技术定义

Maciel首次提出了基于impostor技术的绘制方法[13]。这种方法在预计算阶段把某些指定对象经投影变换转换成带有透明信息的纹理图像，映射到四边形平面后，再放置到场景中该对象原来所在的位置上，并将这类对象称为图像替身。

1.2 Impostor技术基础

Impostor可以提前创建，称之为静态impostor；提前产生的静态impostor可以改进帧速，但是随着场景数量的增加这种方法变得不太可行因为所需纹理所占用的内存空间太大，这就使得动态产生的方法变得极为重要。在参考视点移动过程中动态创建，称之为动态impostor。在绘制过程中，impostor平面替代原有的场景，这种方法不但可以提供相同的外观特性，同时还可以加速绘制。绘制一个impostor平面的时间基本仅仅依赖于屏幕中显示的impostor平面的分辨率大小，而与原有场景部分的几何复杂度无关，这使得这种方法可以快速显示任意的场景。为了创建一个impostor平面，需要指定一个单一的视点，称之为参考视点，这样就可以产生场景各自的几何外表信息从而生成impostor平面。

1.3 Impostor技术优缺点

Impostor技术的优点在于：

(1) 快速绘制场景，绘制impostor平面的时间远小于绘制大堆多边形的时间；

(2) 绘制静态的远处场景非常有效；

(3) 绘制不同层次细节条件下的同类场景非常有效；

(4) 与简化模型相比，impostor平面不以牺牲细节为代价，能够克服低精度的层次细节(LOD)限制；

(5) 帧连续的时候可以重复使用前面帧所产生的相同光栅化图像；

(6) 图像纹理精度仅仅取决于最后屏幕上所形成impostor平面的分辨率大小，因此用来存储纹理所占用内存空间消耗不大。

Impostor技术的缺点：在参考视点移动过程中，需要根据误差来确定是否动态更新impostor平面的内容，详细内容后文。

2 创建impostor平面以及绘制过程

2.1 Impostor技术的适用范围

当参考视点距离场景足够远，不需要绘制场景的全部几何细节；并且在视点所观察到的三维场景和用纹理绘制的场景之间的细节差距并不是很大，这时候就可以创建impostor纹理平面替代场景几何细节。

2.2 创建impostor平面

Impostor平面可以通过三维场景的包围盒得到，这个包围盒可以限定场景的范围。当参考视点确定以后，包围盒将三维场景投影到对应的平面空间上，这个平面空间装入了三维包围盒的所有投影顶点，在平面空间上就形成了impostor平面，如图1所示。

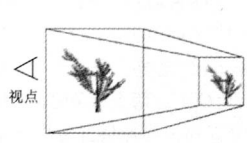


图1 三维场景创建impostor平面

创建impostor平面需要计算存储以下几个信息：

(1) 确定创建impostor平面的顶点和位置。

(2) 参考视点确定以后，顶点可以通过计算三维场景的包围盒在投影平面的信息而得到；并且记录下每一个顶点的深度值，相互比较每一个顶点的深度值从而决定所有顶点的最小深度值。平面空间的顶点和最小的深度值作为impostor平面的位置。

(3) 计算绘制平面纹理所需的投影矩阵信息。

(4) 投影矩阵信息是用来映射绘制平面的纹理。它可以通过计算对应的impostor平面而得到，也就是计算上面叙述的投影平面。

因此，当一个参考视点确定以后，将三维树木划分为多个部分，将每一个部分都通过三维包围盒投影到对应的平面空间上，形成多个impostor平面，如图2所示。这个时候，每一个三维树木的投影顶点都包括在各个impostor平面中，每个impostor平面的顶点、深度、位置、投影矩阵信息都可以通过上面的方法计算出来。接下来就是绘制这些impostor平面。

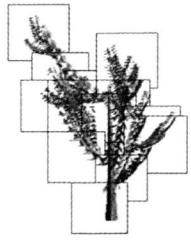


图2 树木创建成多个impostor平面

2.3 绘制过程

Impostor平面绘制包括两个阶段。第1个阶段是视觉依赖的纹理映射操作，称之为描绘impostor平面。根据上面投影impostor平面时候所确定的参考视点与场景之间的角度，初始

化每一个投影impostor平面的纹理：将投影时候所确定的纹理图像映射到每一个相对应的impostor平面上，如图3所示。这些impostor纹理平面用于第2个阶段的绘制。

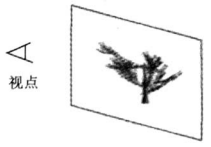


图3 impostor平面纹理映射

在这个过程中需要注意像素的深度问题，这是因为一个四边形只代表一个单一平面的深度，但是三维场景中可以包括多个深度，因此常常会出现遮挡错误的问题。因此，在描绘impostor平面的阶段，impostor纹理平面包括了一个alpha混合技术，当alpha等于1.0时，impostor平面是不透明的；当alpha等于0.0时，impostor平面是完全透明的。被部分透明的impostor平面(alpha值处于这两个极值之间)所替代的屏幕像素无论在颜色方面还是缓冲存储值方面都不会受到影响。因此，一个像素P创建为impostor平面像素I，然后绘制为帧缓冲像素F，这样就可以产生作为最后结果的像素R，它的公式为：$R=P+F\times(1-P\alpha)$，它的推导方式如下：

创建impostor平面的结果为

$$I=P+0\times(1-P\alpha)$$

$$I\alpha=P\alpha+0\times(1-P\alpha)$$

绘制为帧缓冲产生的结果为

$$R=I+F\times(1-I\alpha)=P+F\times(1-P\alpha)$$

第2个阶段称之为绘制impostor平面。当第一个阶段描绘impostor平面完成之后，剩下就是根据impostor平面的深度、位置信息绘制这些impostor平面，也就是依据深度、位置信息确定impostor纹理平面的先后顺序问题。最后得绘制结果如图4所示。深度信息存储在纹理图像的alpha通道，保持正确的深度信息可以使得这个阶段的绘制能够连续并且使得整个绘制过程的深度接近准确，这样最后绘制的结果真实感效果更佳。

图4 绘制impostor平面

3 Error Metric以及动态更新

3.1 静态Error Metric

计算impostor平面纹理的分辨率可以通过使用下面这个公式表示

$$resolution_{texture}=resolution_{screen}\frac{object\ size}{object\ dist}$$

式中：$resolution_{texture}$——impostor平面纹理的分辨率，$resolution_{screen}$——屏幕的分辨率，*object size*——场景的大小，*object dist*——场景离视点的距离。

为了具体确定impostor平面的有效性，可以通过与屏幕中像素的最大角度βscr相比较，这个角度使得参考视点能够确定是否要更新impostor纹理平面的内容。只要用户观察角度小于这个固定值βscr，impostor平面则被认为有效。

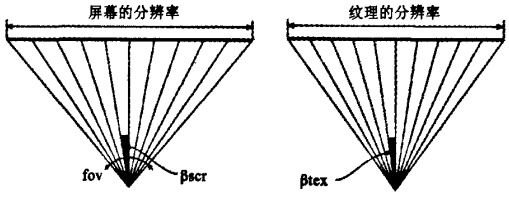
屏幕中像素的最大角度(如图5(a)所示)可以这样计算

$$\beta_{scr}=\frac{fov}{resolution_{screen}}$$

Impostor平面纹元(纹理中的像素)的角度(如图5(b)所示)也可以同样计算

$$\beta_{tex}=\frac{fov}{resolution_{texture}}$$

这里fov指的是图5(a)中所表明的视点范围。



(a) 屏幕像素的最大角度　(b) impostor平面纹元的角度

图5 屏幕中像素的最大角度与impostor平面纹元的角度

当βscr大于βtex的时候，impostor纹理平面不需要更新；反之，则需要更新将impostor纹理平面更新替代为原来的几何平面细节。

3.2 动态更新

由于静态impostor纹理平面的分辨率是固定的，因此最后绘制出来的树木各种属性也是固定不变的。当参考视点开始运动时，树木图像仍然保持不变，这就使得绘制的树木图像与参考视点之间不存在任何视差，这一现象说明绘制过程出现了误差。因此，应该通过计算得出impostor平面的误差估计值用来确定impostor平面是否有效，是否应该动态更新。

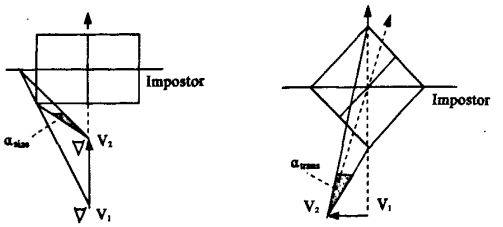
当参考视点移近impostor平面到一定程度时，会产生视觉人造物现象，这一类的error metric称之为zoom error，如图6(a)所示。当参考视点从V1移动到V2时，在投影面上就产生了一个角度差α size，这个角度差可以计算出来。一旦这个角度差大于动态更新所需的最大角度βscr，这个时候面向参考视点的impostor纹理平面无效，需要发生动态更新。

同样，当参考视点平移运动时，如果视角跟初始impostor平面绘制的视角不相一致，则会产生几何图像的扭曲或失真，这一类称之为translation error，如图6(b)所示。当参考视点从V1移动到V2时，在投影面上就产生了一个角度差α trans，这个角度差也可以计算出来。一旦这个角度差大于动态更新所需的最大角度βscr，这个时候面向参考视点纹理impostor平面无效，需要发生动态更新。

3.3 动态更新算法

动态更新impostor平面的算法不需要太多的计算，同时算法的复杂度也很小。

使用impostor绘制场景0{



(a) zoom error 和角度差α size　　(b) translation error 和角度差α trans

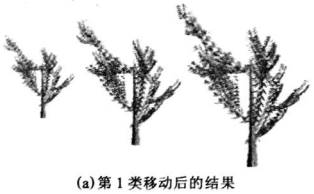
图 6　zoom error、translation error 和角度差

```
初始化βscr;
计算 βtex;
while (βtex<βscr){
计算 α size 和α trans;
   if(αsize>βscr 或者α trans>βscr)
     动态更新 impostor();
   绘制 impostor();
}
if (βtex>βscr)
  绘制几何平面细节();
}
```

3.4　实验结果

本文的树木绘制结果是基于 OpenGL 环境下的得到的，所使用计算机的性能为：CPU 2.4 G，主存 256 MB，显存 64 MB。树木被分成了 14 个 impostor 平面，每个 impostor 平面的分辨率是 256×256，每个像素都有 R、G、B、A 和深度值。树木的原始面片为 31 593，原始的纹理数据所占用的内存空间为 45 MB，使用本文的方法后只需要占用内存 7.5 MB。

在绘制实验中，对于静态的树木，实验结果如图 4 所示。本文的方法能够在快速绘制过程中减少多边形的绘制数量，提高绘制的速度。当参考视点开始运动后，树木图像就会根据误差不断进行动态更新，这就使得在绘制过程中不同参考视点下的同一颗树木出现了不同的绘制结果，如图 7 所示。图 7(a)表示第 1 类移动后的结果，也就是参考视点从远处不断移近后树木细节增多的过程，明显可以看出后一幅比前一幅质量更好；图 7(b)表示第 2 类移动后的结果，也就是参考视点不断平移后所看到的树木细节不断发生变化的过程，明显可以看出视角发生变化后树木细节跟着变化。



(a) 第 1 类移动后的结果

(b) 第 2 类移动后的结果

图 7　移动后的结果

4　结束语

基于 impostor 技术的树木绘制方法，在图像质量、绘制性能上提供了很好的平衡。基于树木模型进行平面投影，将纹理映射到平面上创建 impostor 纹理平面，通过绘制形成树木图像。当参考视点进行移动时，可以根据误差来决定是否动态更新 impostor 纹理平面的内容。

本文是讨论针对树木的快速绘制方法，还有多个地方可以进一步研究和提高，例如：添加动态光照效果和阴影，进一步加强树木的真实感效果；当距离近到一定程度的时候，需要绘制树木的完全几何细节的时候，如何避免过渡过程中出现失真的效果。

参考文献：

[1] Marshall D,Fussell D.Multiresolution rendering of complex botanical scenes[C].Graphics Interface'97,1997:97-104.

[2] Lluch J,Camahort E,Viv'o R.Procedural multiresolution for plant and tree rendering [C]. 2nd International Conference on Computer Graphics,Virtual Reality Visualisation and Interaction of ACM,2003:31-38.

[3] Remolar I,Chover M,Ribelles J,et al.View-dependent multiresolution model for foliage[J].Journal of WSCG,2003,11(2):370-378.

[4] Max N,Ohsaki K.Rendering trees from precomputed Z-buffer views[C].Eurographics Workshop on Rendering,1995:45-54.

[5] Max N.Hierarchical rendering of trees from precomputed multilayer Z-buffers [C]. Eurographics Rendering Workshop, 1996: 65-174.

[6] Meyer A,Neyret F,Poulin P.Interactive rendering of trees with shading and shadows[C].Eurographics Workshop on Rendering, 2001:183-196.

[7] Lluch J, Camahort E, Vivó R. An image-based multiresolution model for interactive foliage rendering [J]. Journal of WSCG, 2004,12(1/3):507-520.

[8] Kajiya J,Kay T.Rendering fur with three dimensional textures[J]. Computer SIGGRAPH,1989,23(3):27-280.

[9] Neyret F.Synthesizing verdant landscapes using volumetric textures[C].Rendering Techniques,1996:215-224.

[10] Levoy M,Whitted T.The use of points as a display primitive[R]. University of North Carolina,Department of Computer Science, 1985.

[11] Weber Penn. Creation and rendering of realistic trees [C]. Los Angeles: Computer Graphics Annual Conference Series, 1995: 119-128.

[12] Deussen O,Colditz C,Stamminger M.Interactive visualization of complex plant ecosystems [C].IEEE Visualization Conference, 2002.

[13] Maciel P,Shirley P.Visual navigation of large environments using textured clusters[C].Interactive 3D Graphics,1995:95-102.