基于混合多细节层次技术的实时绘制算法*

周 昆, 潘志庚, 石教英

(浙江大学 CAD&CG 国家重点实验室,浙江 杭州 310027)

E-mail: kzhou@cad.zju.edu.cn

http://cad.zju.edu.cn/home/kzhou

摘要: 多细节层次是实时图形生成的一项重要技术.提出一种把视点无关和视点相关两类多细节层次技术结合起来的网格模型实时绘制算法.该算法首先根据应用领域的不同要求或用户给出的误差范围,对模型进行与视点无关的预处理简化,然后把简化后得到的模型用在与视点相关的实时简化算法中.实验表明,这种网格模型简化和绘制算法能在损失很小的屏幕像素误差的前提下大大提高绘制速度.

关键词: 网格简化;实时绘制;细节层次;虚拟现实

中图法分类号: TP391 **文献标识码**: A

一般说来,实时绘制技术包括可见性删除、多细节层次(level of detail,简称为 LoD)、分布式并行计算和基于图像绘制等.可见性删除技术是把不在视野范围内的、不可见的模型部分删除,并不进行绘制[1].LoD 技术根据视点的不同选择不同精细程度的模型进行绘制[2].这些不同精细程度的模型或者是预先生成,或者是实时生成.分布式并行计算技术把绘制计算任务分成若干个子任务到不同的处理器上运行,最后生成结果图像.这需要分布式硬件环境和分布式绘制算法的支持[3].基于图像绘制技术用已经预先生成好的图像代替整个场景或场景的某些部分,并从这些图像中合成出新的图像以适应视点的变化.基于图像绘制技术的最大优点是,绘制时间只与图像的分辨率有关,而与场景的复杂度无关[4].

LoD 技术可以分成与视点无关和与视点相关两大类.(1) 与视点无关的 LoD 技术.主要是根据不同的误差标准在尽可能保持模型外形的前提下,自动生成不同精细程度的简化网格模型,在绘制过程中,根据视点的位置选择相应的网格模型进行绘制[2,5~13].(2) 与视点相关的 LoD 技术.根据视点动态地生成简化模型[14~16].目前,国内对 LoD 技术也开展了一些研究工作[17~22].

与视点无关的 LoD 技术要预先生成并存储多个简化模型,一方面需要很多存储空间,另一方面,当随着视点变化在不同精细程度的模型间切换时会产生视觉效果跳变.与视点相关的 LoD 技术则根据视点动态地生成简化模型,可以保证模型连续平滑地过渡,不会产生视觉效果的跳变.但现有的技术都是预先为整个场景生成一棵简化操作顶点树,这棵树要记录大量与简化操作相关的正向和逆向的信息,当场景很大时,这棵树就会耗费庞大的内存.

本文提出了一种把与视点无关的 LoD 技术和与视点相关的 LoD 技术结合起来的实时绘制技术,即先根据应用领域的要求或用户指定的误差范围,用与视点无关的网格简化算法对模型进行处

* **收稿日期**: 1999-04-12; **修改日期**: 1999-10-25

基金项目: 国家自然科学基金资助项目(69823003;60083009)

作者简介: 周昆(1977—),男,湖南岳阳人,博士生,主要研究领域为虚拟现实,计算机视觉;潘志庚(1965—),男,江苏淮安人,研究员,博士生导师,主要研究领域为虚拟现实,多媒体计算,分布式图形处理;石教英(1937—),男,浙江宁波人,教授,博士生导师,主要研究领域为虚拟现实,科学计算可视化,多媒体技术.

理,再把得到的简化模型提交给与视点相关的网格简化技术来处理,进行实时绘制.另外,本文中的与视点相关的网格简化算法具有自己的特点,该算法基于使用浮动包围盒的顶点聚类算法,综合考虑视点、视线和分辨率因素,并与视见体裁剪和背面删除结合起来,可在控制屏幕像素误差的前提下实时生成简化网格.

本文第1节简单介绍与视点无关的网格简化预处理.第2节详细描述与视点相关的实时网格简化算法.第3节介绍整个实时绘制算法的实现和一些试验结果.最后是结论.

1 与视点无关的网格简化预处理

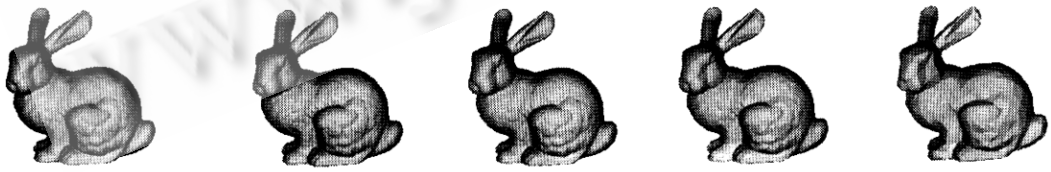
在进行与视点无关的网格简化预处理时,并不是要预先生成一系列简化网格模型,而是根据应用领域的不同要求或用户指定的误差范围生成一个简化模型,这个简化模型将代替原始模型用于后继的实时绘制过程.之所以能这样做,主要是基于下面的一些事实:一方面,大多数网格模型是通过成熟的造型软件或三维测量设备自动生成的,为了保证对模型造型的精确性,对模型的采样是有很大的冗余度的(例如,机械零件CAD系统生成的零件模型有很多共面的顶点、边和三角形;科学计算可视化中的等值面抽取算法Marching Cube产生的大量三角面片也有冗余);另一方面,不同的应用领域需要不同精细程度的模型,很多应用领域对绘制实时性的要求都高于对绘制精度的要求.我们可以根据应用领域的不同要求或用户指定的误差范围,预生成满足要求的简化模型.

在文章开始部分中列出了多种与视点无关的网格简化算法,只要是能够控制简化操作带来的全局误差的算法都可以用在我们的预处理过程中.本文采用我们提出的基于三角形折叠的简化算法.由于三角形网格模型是最常见的多边形网格模型,而且任意多边形很容易转化成三角形,本文只讨论三角形网格,算法的详细过程可以参考文献[20].

图1给出了我们的算法对兔子(bunny)模型作简化处理的结果.为了衡量简化模型和原始模型之间的误差[11],我们用下面的公式计算原始网格 M_o 和简化网格 M_s 之间的误差 E_s.

$$E_s=\frac{1}{|X_0|+|X_s|}\left(\sum_{v\in X_o}d(v,M_s)+\sum_{v\in X_s}d(v,M_o)\right),$$

其中 X_o 和 X_s 分别为原始网格 M_o 和简化网格 M_s 的顶点集合,$|X|$ 表示顶点集合 X 中顶点的数目,$d(v,M)=\min_{p\in M}\|v-p\|$ 为点 v 到网格 M 的最短距离.表1给出了对应于图1简化模型的误差值.根据我们的试验,当相对误差值(即表1中最后一列)小于0.02%时,肉眼很难分辨简化模型与原始模型之间的差别.



(a) Original model (69 473 triangles) (b) 17 279 triangles (c) 8 584 triangles (d) 3 416 triangles (e) 1 362 triangles

(a) 原模型(69 473个三角形) (b) 17 279个三角形 (c) 8 584个三角形 (d) 3 416个三角形 (e) 1 361个三角形

Fig. 1 View-Independent simplification of bunny model

图1 兔子模型的与视点无关的简化

Table 1 Approximation errors according to the bunny models

表 1 兔子模型的简化误差

	The size of bounding box of the original model①	Approximation error②	Approximation error/length of diagonal of the bounding box of the original model③(%)
(a)	2.00000×1.98330×1.54913	0.000 00	0.000 00
(b)	2.00009×1.98292×1.55044	0.000 41	0.012 64
(c)	2.00149×1.98247×1.54983	0.000 66	0.020 58
(d)	2.00089×1.98035×1.55222	0.001 33	0.041 33
(e)	2.00839×1.97537×1.56133	0.002 65	0.082 49

①原模型的包围盒大小,②误差值,③误差值/原始模型的包围盒对角线.

2 与视点相关的实时网格简化算法

考虑到实时性,我们采用基于顶点聚类的网格简化算法[7,10].文献[10]针对由原始顶点聚类简化算法生成的简化网格质量较差的不足,提出了浮动单元顶点聚类简化算法,该方法把权值较大(即较重要的)顶点作为顶点聚类的中心.虽然该算法在计算顶点聚类包围球半径时也考虑了视点到模型的距离,但很大程度上只是作为自动进行简化运算的输入参数,而不是侧重于实时简化之运算,算法是作为一种静态网格简化算法而提出来的.而我们则把它用在动态网格简化中.与 Low 的算法相比,本文的算法使用一种简单的方法计算顶点的权值,着重考虑控制模型绘制在屏幕窗口内的误差(以像素计算),根据视点、视见体和屏幕分辨率实时生成简化模型.同时,为了加快绘制速度,本文的算法还与视见体裁剪、背面删除结合起来.整个算法包括顶点相对曲率值预计算和实时简化运算两部分.

2.1 顶点相对曲率值预计算

这一步要计算第 1 节生成的简化网格的各顶点相对曲率值,并根据相对曲率值从大到小对所有顶点排序,这一步也是作为预处理来完成的.排序得到的序列将用在第 2.2 节中的实时简化运算中.

顶点相对曲率值的计算可以用很多种方法.对于非边界顶点 V_i(即共享该顶点的所有三角形构成一个环),设共享该顶点 V_i 的三角形集合为 Ts_i,则该顶点的相对曲率值 C_i 可以用下面的方法计算:

$$\bar{n}_i=\frac{\sum_{T\in Ts_i}\Delta T\cdot\bar{n}_T}{\sum_{T\in Ts_i}\Delta T},\quad C_i=\frac{\sum^{T\in Ts_i}\angle(\bar{n}_i,\bar{n}_T)}{|Ts_i|}.$$

其中 ΔT 表示三角形 T 的面积,$\bar{n}_T$ 表示三角形 T 的法向量,$\angle(\bar{n}_i,\bar{n}_T)$表示向量 $\bar{n}_i$ 和 $\bar{n}_T$ 之间的夹角,$|Ts_i|$表示三角形集合 Ts_i 中三角形的数目.

顶点相对曲率值能代表顶点的几何位置的重要性:相对曲率值越大,顶点的位置就越重要.

2.2 实时简化运算

实时简化运算根据视点、视见体和屏幕分辨率实时生成简化模型.为了加快绘制速度,这里的简化算法与视见体裁剪、背面删除结合起来.虽然很多图形软件包(如 OpenGL)包含了视见体裁剪和背面删除,但在执行两种操作之前,还要执行图形绘制流水线前面的一些操作.本文提出的算法能很好地与它们结合起来,一方面节省了绘制时间,另一方面还有助于保持模型绘制在屏幕上的轮廓(silhouette),关于这一点,我们在后面会进一步加以说明.下面,先给出算法框架,然后详细讨论算法细节.

(1) 算法框架

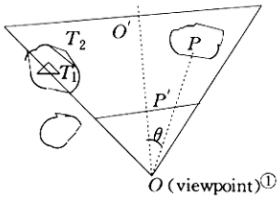
设 N 为网格模型的顶点数目,$int\ queue[N]$为从第 2.1 节得到的排序队列,$int\ Map[N]$为顶点映射数组,它是算法中最关键的数据结构.本文提出的实时网格简化算法可以用下面的伪代码来描述:

```
begin
//步骤 1.初始化,每个顶点映射到自身.
for (i:=1;i<=N;i++)
  Map[i]:=i;
//步骤 2.视见体裁剪和背面顶点删除
for (网格中的每个顶点 Vi)
  if (Vi 不在当前视见体中或者 Vi 是背面顶点)
      Map[Vi]:=-1;
//步骤 3.顶点聚类.
for (i:-1;i<=N;i++)
{
  Vi:=queue[i];
  if (Map[Vi]=Vi)
  {
      根据当前视点、视见体和屏幕窗口分辨率计算包围球半径 Ri;
      for (每个落在以 Vi 为球心,Ri 为半径的球内的顶点 Vk)
          if (Map[Vk]=Vk)
              Map[Vk]:=Vi;
  }
}
//步骤 4.生成简化网格模型.
for (网格的每个三角形 Ti)    //设 Vi1,Vi2和 Vi3分别是 Ti 的 3 个顶点
{
  if (Map[Vi1]≠-1 并且 Map[Vi2]≠-1 并且 Map[Vi3]≠-1)
  {
      if (Map[Vi1]=-1)
          V[1]:=Vi1;
      else
          V[1]:=Map[Vi1];
      if (Map[Vi2]=-1)
          V[2]:=Vi2;
      else
          V[2]:=Map[Vi2];
      if (Map[Vi3]=-1)
          V[3]:=Vi3;
      else
          V[3]:=Map[Vi3];
      把顶点 V[1],V[2]和 V[3]构成的三角形输出到简化网格;
  }
}
end
```

(2) 视见体裁剪和背面顶点删除

图 2 以二维的方式显示了视点、视见体以及模型之间的关系.视见体对应锥体的 6 个面,在进

行视见体裁剪时,先求出6个面的方程(用Plane[6][4]表示)和视见体内任意一点的齐次坐标(例如中心$Q[4]$),则判断空间任意一点$P[4]$是否在视见体内的方法为:对所有$1\leqslant i\leqslant 6$,若都有$(\text{Plane}[i]\cdot Q)*(\text{Plane}[i]\cdot P)>0$,则$P$在视见体内;否则$P$不在视见体内.



①视点.

Fig. 2 Viewing frustum, line of sight and models
图2 视点、视见体和模型

背面顶点的判断方法是:设从该顶点到视点的向量为$d[3]$,该顶点的法向量为$n[3]$,如果$(d\cdot n)<0$,则为背面顶点.

需要特别指出的是,网格模型中某些三角形会有一个或两个顶点在视见体裁剪或背面删除操作中被删除,还有两个或一个顶点没有被删除,本文的算法不删除这些顶点而保留整个三角形,而这些三角形一般都在模型投影到屏幕窗口内的轮廓对应处,这就保持了模型绘制在屏幕上的轮廓.

(3) 顶点聚类

顶点聚类的关键在于包围球半径的计算.下面由图2分析如何根据屏幕窗口的像素误差计算包围球的半径.

如果屏幕绘制窗口的分辨率为$Width*Height$,视见体近平面(即视区)的大小为$W*H$,那么绘制窗口内的一个像素对应于近平面上区域的面积为$(W*H)/(Width*Height)$.假设模型上的一点P投影到近平面上的P'(如图2所示),绘制误差控制在K个像素内,则P点包围球的半径R可以用下式来计算:

$$R=\sqrt{\frac{K*W*H}{Width*Height}}*\frac{\|OP\|}{\|OP'\|}.$$

考虑视线的因素,远离视线中心的景物会比较模糊,设OP和视线中心线OO'的夹角为θ,我们假设包围球半径随θ角的增加以2次幂函数(也可以选择其他函数)的形式增加,则包围球半径R的计算可修正如下:

$$R=\sqrt{\frac{K*W*H}{Width*Height}}*\frac{\|OP\|}{\|OP'\|}*\left(\frac{90^\circ}{90^\circ-\theta}\right)^2.$$

(4) 生成简化网格模型

当顶点聚类得到顶点映射表后,只要判断原网格中的每个三角形在顶点被映射后是否还构成三角形并输出这些三角形,就可以生成简化网格模型.图3给出了在不同像素误差下对兔子模型生成的简化网格.

(a) The original model (69 473 triangles) (a) 原始模型(69 473个三角形) (b) 1 pixel error (b) 1个像素误差 (c) 2 pixels error (c) 2个像素误差 (d) 4 pixels error (d) 4个像素误差 (e) 8 pixels error (e) 8个像素误差

Fig. 3 Simplify bunny model with different rendering pixel errors
图3 不同像素误差下兔子模型的简化

3 算法实现和实验结果

3.1 算法实现

本文提出的实时绘制算法已经在我们的实时绘制系统中实现,实现环境为内存 64M 的 Pentium Pro 233M 微机,采用 Visual C++ 5.0,基本图形绘制软件包是 OpenGL. 用户可以在该系统中通过改变视点、视线方向、视见体以及屏幕窗口绘制误差来观察模型,系统还提供了记录观察路径和按给定路径漫游的功能.

算法包含的与视点无关预处理和与视点相关实时计算两部分都集成在一个程序里. 公用的数据结构有顶点表、三角形表、法向量. 另外,预处理部分还需要误差矩阵表[20],在作实时简化时,还需要顶点映射表和对网格模型的八叉树剖分.

3.2 实验结果

为了测试算法的有效性,我们分别对直升飞机模型和兔子模型作了两段漫游路径. 图 4 给出了直升飞机模型在漫游中的 3 帧画面*. 其中图 4(a)、图 4(b)和图 4(c)是直接用原始模型并且不用 LoD 技术绘制得到的画面,图 4(d)、图 4(e)和图 4(f)是对原始模型不作简化预处理而只用实时网格简化算法绘制得到的画面,图 4(g)、图 4(h)和图 4(i)是用本文的实时绘制算法(包括预处理简化和实时简化)得到的画面,图 4(j)、图 4(k)和图 4(l)则是图 4(a)、图 4(b)、图 4(c)和图 4(g)、图 4(h)、图 4(i)之间的差别. 从这些画面可以看出,本文算法能够很好地控制绘制的屏幕像素误差.

表 2 给出了对绘制帧速率的统计数据,可以看出,如果不对模型进行预处理简化而直接用实时简化算法,那么绘制的速度提高是有限的. 加上预处理简化后,带来的屏幕像素误差很小,绘制速度却大大提高了.

Table 2 Frame rates of the rendering processes

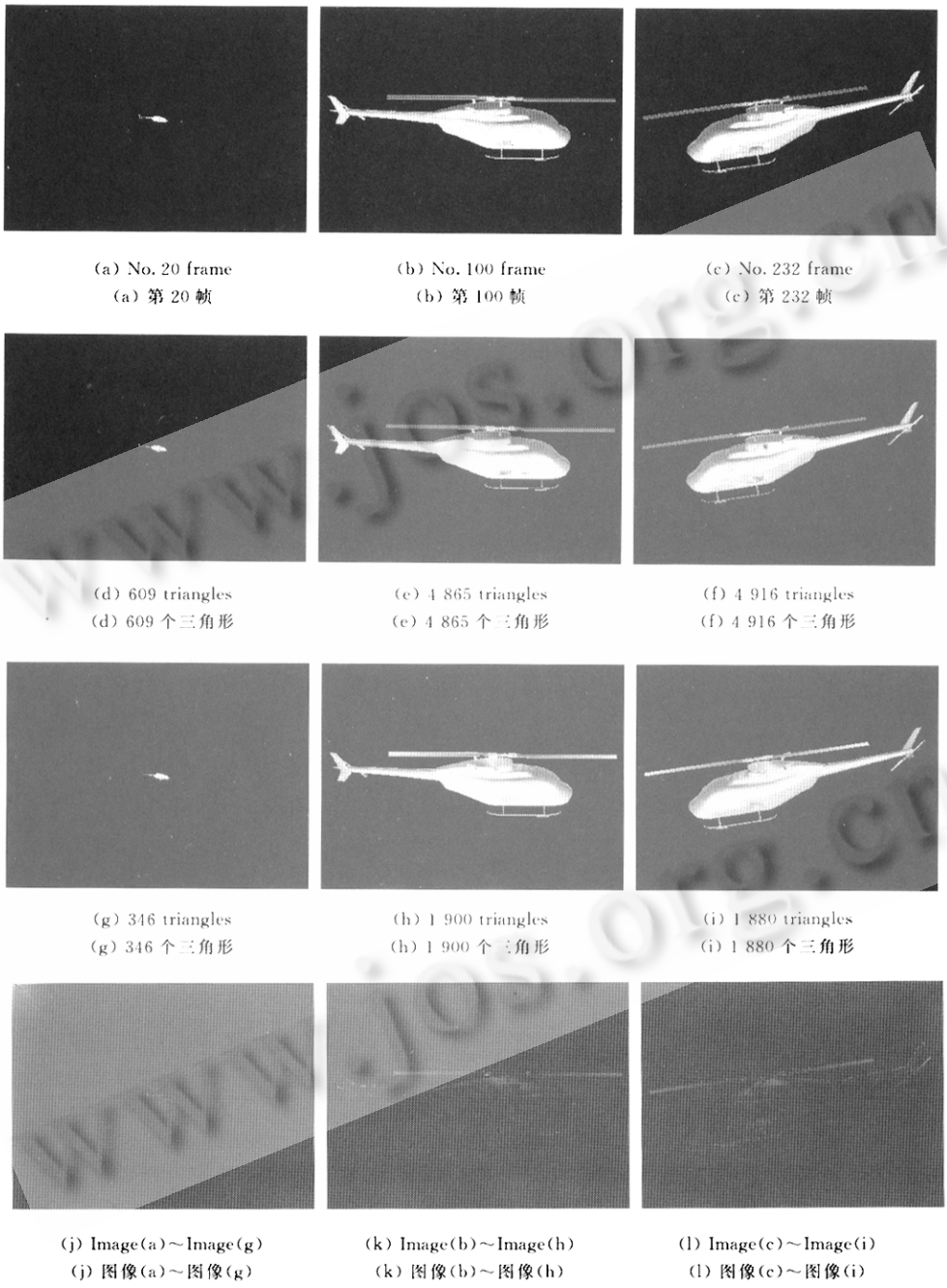
表 2 绘制帧速率数据

	Helicopter①	Bunny②
The resolution of the screen window③	400×300	400×300
The rendering pixel error④	1 pixel⑤	1 pixel
The number of frames⑥	600	300
The number of triangles in the original model⑦	14 168	69 473
The number of triangles in the pre-simplified model⑧	3 514	17 279
Approximation error between the pre-simplified model and the original model⑨(%)	0.008 96	0.012 64
Frame rate without using LoD method⑩	5.76	1.30
Frame rate with using dynamic LoD method (without pre-simplification)⑪	8.90	3.23
Frame rate the the algorithm⑫	22.39	8.81

①直升飞机,②兔子,③屏幕窗口分辨率,④屏幕窗口像素误差,⑤像素,⑥漫游帧数,⑦原始模型的三角形数目,⑧预处理得到的简化模型的三角形数目,⑨原始模型与预处理得到的简化模型之间的相对误差值,⑩不用 LoD 技术绘制的帧速率,⑪不对原始模型进行预处理简化直接用实时简化算法绘制的帧速率,⑫用本文算法绘制的帧速率.

另外,还需要指出的是,这里给出的两个例子都是对单个模型进行交互式观察得到的,并不是对虚拟环境的漫游. 虚拟环境中往往包括很多模型,只要这些模型各自独立成为网格模型,本文的算法还是可以适用的,并且随着模型的增多,视见体裁剪的效果会更明显,算法的效率会更高.

* 更多的信息在 http://cad.zju.edu.cn/home/kzhou/HybridLoD.html 可检索到.



(a) No. 20 frame
(a) 第 20 帧

(b) No. 100 frame
(b) 第 100 帧

(c) No. 232 frame
(c) 第 232 帧

(d) 609 triangles
(d) 609 个三角形

(e) 4 865 triangles
(e) 4 865 个三角形

(f) 4 916 triangles
(f) 4 916 个三角形

(g) 346 triangles
(g) 346 个三角形

(h) 1 900 triangles
(h) 1 900 个三角形

(i) 1 880 triangles
(i) 1 880 个三角形

(j) Image(a)~Image(g)
(j) 图像(a)~图像(g)

(k) Image(b)~Image(h)
(k) 图像(b)~图像(h)

(l) Image(c)~Image(i)
(l) 图像(c)~图像(i)

Fig. 4 Rendering of helicopter
图 4 直升飞机模型的绘制

4 结 论

本文提出了一种把与视点无关的 LoD 技术和与视点相关的 LoD 技术结合起来的实时绘制算

法.该算法综合了两种 LoD 技术的优点,在保持一定视觉效果的前提下大大加快了绘制速度.另外,本文中的与视点相关的网格简化算法与已有的方法相比有自己的特点,该算法基于浮动包围球的顶点聚类算法,综合考虑视点、视线和分辨率因素,并与视见体裁剪和背面删除结合起来,可在控制屏幕像素误差的前提下实时生成简化网格.本文算法可望应用在虚拟现实系统、交互式可视化系统和 CAD 中.

进一步的研究内容包括:(1) 对网格模型的颜色和纹理属性进行处理;(2) 研究如何保持绘制的恒帧速率;(3) 把 LoD 技术和其他实时绘制技术(如基于图像绘制)结合起来.

致谢: 感谢审稿专家对本文提出的修改意见.

References:

[1] Airey, J. Towards image realism with interactive update rates in complex virtual building environments. Computer Graphics (Symposium on Interactive 3D Graphics), 1990,24(2):41~51.

[2] DeHaemer, Jr. M., Zyda, M. J. Simplification of objects rendered by polygonal approximations. Computers & Graphics, 1991,15(2):175~184.

[3] Pan, Zhi-geng, Zhang, Ming-min, Shi, Jiao-ying. Time-Critical computing in virtual environment. In: Zhou, Ji, ed. Proceedings of the SPIE. Bellingham: SPIE Press, 1996. 712~717.

[4] Chen, S. E., Williams, L. View interpolation for image synthesis. Computer Graphics (SIGGRAPH'93), 1993,27:279~288.

[5] Schroeder, W. J., Zarge, J. A., Lorensen, W. E. Decimation of triangle meshes. Computer Graphics, 1992,26(2):65~70.

[6] Turk, G. Re-Tiling polygonal surface. Computer Graphics, 1992,26(2):55~64.

[7] Rossignac, J., Borrel, P. Multi-Resolution 3D approximation for rendering complex scenes. In: Falcidieno, B., Kunii, T., eds. Geometric Modeling in Computer Graphics. Heidelberg: Springer-Verlag, 1993. 455~465.

[8] Hoppe, H. Progressive meshes. Computer Graphics (SIGGRAPH'96), 1996,30:99~108.

[9] Cohen, J., Varshney, A., Manocha, D., *et al*. Simplification envelopes. Computer Graphics (SIGGRAPH'96), 1996, 30:119~128.

[10] Low, Kok-Lim, Tan, Tiow-Seng. Model simplification using vertex-clustering. In: Dan, A. V., ed. Proceedings of the 1997 Symposium on Interactive 3D Graphics. RI: ACM Press, 1997. 75~81.

[11] Garland, M., Heckbert, P. S. Surface simplification using quadric error metrics. Computer Graphics (SIGGRAPH'97), 1997,31:209~216.

[12] Soucy, Marc, Guy, Godin, Marc, Rioux. A texture-mapping approach for the compression of colored 3D triangulations. The Visual Computer, 1996,12(10):503~514.

[13] Cohen, J., Olano, M., Manocha, D. Appearance-preserving simplification. Computer Graphics (SIGGRAPH'98), 1998, 2(3):189~198.

[14] Hoppe, H. View-Dependent refinement of progressive meshes. Computer Graphics (SIGGRAPH'97), 1997,31(3):189~198.

[15] Xia, J., El-Sana, J., Varshney, A. Adaptive real-time level-of-detail-based rendering for polygonal models. IEEE Transactions on Visualization and Computer Graphics, 1997,3(2):171~183.

[16] Luebke, D., Erikson, C. View-Dependent simplification of arbitrary polygonal environments. Computer Graphics (SIGGRAPH'97), 1997,31(3):209~216.

[17] Pan, Zhi-geng, Ma, Xiao-hu, Shi, Jiao-ying. An automatic generation algorithm of LoD models in virtual environment. Journal of Software, 1996,7(9):526~531 (in Chinese).

[18] Zhou, Xiao-yun, Liu, Shen-quan. A polyhedral model simplification algorithm based on feature angle criteria. Chinese Journal of Computers, 1996,19(9):217~223 (in Chinese).

[19] Liu, Xue-hui, Wu, En-hua. An efficient rendering method for terrain surfaces. In: Lin, Zhong-kai ed. Proceedings of the CAD/Graphics'97. Beijing: International Academic Publisher, 1997. 89～94.

[20] Zhou, Kun, Pan, Zhi-geng, Shi, Jiao-ying. A new mesh simplification algorithm based on triangle collapse. Chinese Journal of Computers, 1998,21(6):506～514 (in Chinese).

[21] Li, Jie, Tang, Ze-sheng, Guo, Hong-hui. Multi-Resolution modeling based on fractional dimension. Chinese Journal of Computers, 1998,21(9):780～786 (in Chinese).

[22] Zhou, Kun, Pan, Zhi-geng, Shi, Jiao-ying. A new mesh simplification algorithm based on vertex clustering. Journal of Automation, 1999,25(1):1～8 (in Chinese).

附中文参考文献:

[17] 潘志庚,马小虎,石教英.虚拟环境中多细节层次模型自动生成算法.软件学报,1996,7(9):526～531.

[18] 周晓云,刘慎权.基于特征角准则的多面体模型简化方法.计算机学报,1996,19(9):217～223.

[20] 周昆,潘志庚,石教英.基于三角形折叠的网格简化算法.计算机学报,1998,21(6):506～514.

[21] 李捷,唐泽圣,郭红晖.基于分形维数的层次分辨率模型.计算机学报,1998,21(9):780～786.

[22] 周昆,潘志庚,石教英.一种新的基于顶点聚类的网格简化算法.自动化学报,1999,25(1):1～8.

A Real-Time Rendering Algorithm Based on Hybrid Multiple Level-of-Detail Methods

ZHOU Kun, PAN Zhi-geng, SHI Jiao-ying

(*State Key Laboratory of CAD & CG, Zhejiang University, Hangzhou* 310027, *China*)

E-mail: kzhou@cad.zju.edu.cn

http://cad.zju.edu.cn/home/kzou

Received April 12, 1999; accepted October 25, 1999

Abstract: Level of Detail (LoD) is a key technique for real-time rendering. A new real-time rendering algorithm incorporating view-independent algorithms and view-dependent algorithms is presented in this paper. Firstly, a polygonal mesh model is symplified view-independently according to the different requirements of applications or the user-specified error. Then the simplified model is used in a view-dependent real-time rendering algorithm. Examples illustrate that the proposed mesh simplification algorithm and rendering algorithm can accelerate the rendering speed greatly, while the rendering error (evaluated by pixel) is very small.

Key words: mesh simplification; real-time rendering; level of detail; virtual reality