(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101419721 B
(45)授权公告日 2012.02.15

(21)申请号 200810201983.X

(22)申请日 2008.10.30

(73)专利权人 上海大学
地址 200444 上海市宝山区上大路 99 号

(72)发明人 余小清 万旺根 周俊玮 丁欢

(74)专利代理机构 上海上大专利事务所（普通合伙） 31205
代理人 何文欣

(51)Int.Cl.
G06T 15/20(2006.01)

(56)对比文件

封春生等．基于视域剔除和图像缓存技术的复杂场景快速绘制方法．《系统仿真学报》．增刊，2006，第 18 卷（第 1 期），94-98.

石祥滨等．一种适合室内 3D MMOG 场景的快速绘制方法．《小型微型计算机系统》．2008，第 29 卷（第 8 期），1548-1552.

王钧等．一种大范围复杂场景的快速绘制算法．《计算机工程与应用》．2003，（第 16 期），88-90.

高宇等．大规模外存场景的交互绘制．《计算机辅助设计与图形学学报》．2007，第 19 卷（第 6 期），792-797.

审查员 李芳

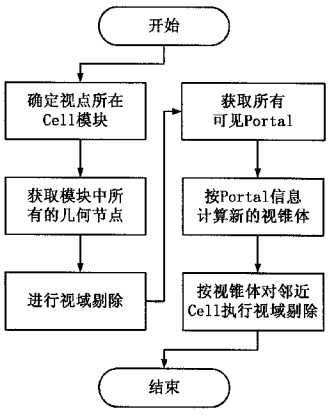
权利要求书 1 页 说明书 6 页 附图 3 页

(54)发明名称

一种基于视域剔除的复杂室内场景快速绘制方法

(57)摘要

本发明涉及一种基于视域剔除的复杂室内场景快速绘制方法。本方法首先把整个场景划分成 n 个区域，用 portal 入口把相邻的区域连接起来，再确定视点所在的位置，将该区域模块所包含的所有几何模块与视锥体进行相交检测，完成视域剔除工作，最后计算可视范围内的所有 Portal，计算新视锥体，并进行新一轮的视域剔除，如此递归下去，得到最终需要渲染的节点。实验结果表明，采用本发明的室内场景管理方法，在室内遮挡率高时，能够大大提高渲染效率。



CN 101419721 B

1. 一种基于视域剔除的复杂室内场景快速绘制方法，其特征在于首先将场景划分成若干区域，用 Portal 入口把相邻的区域连接起来，得到若干个新区域，再确定视点所在的位置，所述的每个新区域将区域包含的所有几何模块与视锥体进行相交检测，然后进行视域剔除工作，实现复杂室内场景的快速绘制；具体操作步骤如下：

a. 采用‘凸’多边形作为划分区域的最小单位；并且得到的每一个区域都是一个闭合的‘凸’多边形；

b. 用 portal 入口把相邻的区域连接起来，得到若干个新区域；

c. 确定视点所在的位置，将摄像机的空间坐标与所有区域依次进行相交判断，由此确定视点处于哪个区域中；

d. 将该区域包含的所有几何模块与视锥体进行相交检测，完成视域剔除工作；

e. 计算可视范围内的所有 Portal 入口，按 Portal 入口大小和视点方向重新计算新的视锥体，并在该 Portal 入口相连的区域内利用重新计算的新的视锥体进行新一轮的视域剔除；

f. 如此递归下去，直到所有的可视区域都完成了视域剔除，如此获得的所有几何节点形成的序列就是需要进行渲染的节点。

2. 根据权利要求 1 所述的基于视域剔除的复杂室内场景快速绘制方法，其特征在于：所述的步骤 a 中的采用‘凸’多边形作为划分区域中的最小单位中的‘凸’表示在区域内的任意两点间画一条线段，这条线段不会穿透该区域的任何一个多边形，而闭合表示从区域内发出一条光线，如果要让光线射到区域外，则此光线必须要穿透某个区域的多边形。

3. 根据权利要求 1 所述的基于视域剔除的复杂室内场景快速绘制方法，其特征在于：所述的步骤 b 中的 portal 入口表示为两个区域相连接部分的位置和大小。

4. 根据权利要求 1 所述的基于视域剔除的复杂室内场景快速绘制方法，其特征在于所述的步骤 e 中的按 Portal 入口大小和视点方向重新计算新的视锥体，这里会出现两种情况：

e-1..Portal 入口完全处于原视锥体范围内部；只要计算由 Portal 入口产生的视锥体为新的视锥体；

e-2.. 如果部分处于原视锥体内部；那么在内部以 Portal 入口产生的视锥体，即新视锥体为准，而超出原视锥体的部分还是按照原视锥体的边界进行计算所得。

一种基于视域剔除的复杂室内场景快速绘制方法

技术领域

[0001] 本发明涉及图形处理和虚拟现实，主要是一种场景图形引擎快速绘制方法，特别是一种基于室内场景图结构的图形引擎快速绘制方法。

背景技术

[0002] 由于计算机绘制的场景日益复杂，为了达到实时的效果，对计算机软硬件的要求只会越来越高，它要求采用先进的绘制算法来对复杂的场景进行规划和管理，对场景管理算法进行优化，加快绘制场景的速度，通常图形质量与计算量是成正比的，计算量越大，图形的质量也就越好，然而考虑到虚拟现实中实时性的要求，所以不得不在这两者中均衡考虑，在满足视觉效果的同时，尽量加快绘制的速度。

[0003] 目前用来加速场景绘制的主要技术有：层次细节算法，基于图像的绘制算法等快速绘制算法，层次细节算法的核心思想是在不影响画面视觉效果的前题条件下，根据视点距离，逐步简化场景表面细节来提高绘制的效率。但层次细节 LOD 转换时的图像跳跃比较大，是一种离散的，在离散 / 连续方面非常欠缺。基于图像绘制算法利用二维图像信息来表达和绘制虚拟场景，利用这种方法会有较好的真实感和实时性，因为它来自真实的照片，具有真实感强的图像，但是它的缺点是缺少交互性。

[0004] 对于一个虚拟现实系统来说，不仅有室外场景，还有相当一部分的室内场景，室内场景通常都有一个特性——场景内部模型相当复杂，并且从观测者的视点看去只能看到他所处环境内的物体，除非在这个环境中有其它 Portal 入口可以通往其它环境。由于室内场景的复杂，运用 LOD 肯定是不现实的，因为都是近距离的模型，而基于图像的绘制的算法也不能从根本上解决加速绘制的要求，一般针对这样的室内环境都是运用创建树结构，但树结构是相当低效的，因为通过 Portal 入口观测者只能看到相邻环境中非常小的一部分物体。换句话说，树结构不能很好的处理既有高复杂度又有高阻光度的场景。这个问题常常被称为“无效渲染”，会花费大量的时间去渲染最终被一堵墙挡住的相邻房间中的物体。

发明内容

[0005] 本发明的目的在于提供一种基于视域剔除的复杂室内场景快速绘制方法，能大为提高在绘制室内场景时的绘制速度，保证虚拟现实的实时性要求。

[0006] 为达到上述目的，本发明的构思是：克服现有图形引擎快速绘制技术，尤其是在绘制室内场景时的速度不够，提供了一种基于视域剔除的复杂室内场景快速绘制方法，它考虑了室内场景与室外场景的区别与特点，大大提高了在绘制室内场景时的绘制速度，保证了虚拟现实的实时性要求。

[0007] 根据上述发明构思，本发明采用的技术方案是：

[0008] 一种基于视域剔除的复杂室内场景快速绘制方法，其特征在于首先将场景划分成区域，用 Portal 入口把相邻的区域连接起来，再确定视点所在的位置，将前面连接起来的相邻区域包含的所有几何区域与视锥体进行相交检测，然后进行视域剔除工作，实现复杂

室内场景的快速绘制；具体操作步骤如下：

[0009] (1) 采用‘凸’多边形作为划分区域的最小单位；并且得到的每一个区域都是一个闭合的‘凸’多边形；

[0010] (2) 用 Portal 入口把相邻的区域连接起来；

[0011] (3) 确定视点所在的位置，将摄像机的空间坐标与所有区域依次进行相交判断，由此确定视点处于哪个区域中；

[0012] (4) 将该区域所包含的所有几何与视锥体进行相交检测，完成视域剔除工作；

[0013] (5) 计算可视范围内的所有 Portal 入口，按 Portal 入口大小和视点方向重新计算得到新的视锥体，并在该 Portal 入口相连的区域内利用新视锥体进行新一轮的视域剔除；如此递归下去，直到所有的可视区域都完成了视域剔除，如此获得的所有几何节点形成的序列就是需要进行渲染的节点。

[0014] 所述的步骤 (1) 中的采用‘凸’多边形作为划分区域中的最小单位中的‘凸’表示在区域内的任意两点间画一条线段，这条线段不会穿透该区域的任何一个多边形，而闭合表示从区域内发出一条光线，如果要让光线射到区域外，则此光线

[0015] 必须要穿透某个区域的多边形。

[0016] 所述的步骤 (2) 中的 Portal 入口表示为两个区域相连接部分的位置和大小。

[0017] 所述的步骤 (5) 中的按 Portal 入口大小和视点方向重新计算新的视锥体，这里会出现两种情况，Portal 入口完全 / 部分处于原视锥体范围内部，如果完全处于原视锥体内部，那么只要计算由 Portal 入口产生的视锥体为新的视锥体，如果部分处于原视锥体内部，那么在内部以 Portal 产生的视锥体为准，而超出原视锥体的部分还是按照原视锥体的边界进行计算所得。

[0018] 本发明有益的效果是：采用 Portal 入口技术对复杂的室内场景进行管理，基于视域剔除，使得在模型复杂的场景内无需渲染场景中的每个物体，这样大大提高了渲染了效率，很好的设计了 Portal 入口，区域的数据结构体，而避免了用树结构产生的“无效渲染”提高内存利用率。

附图说明

[0019] 图 1 本发明的方法流程图。

[0020] 图 2Portal 入口模型结构图。

[0021] 图 3 视锥体计算示意图 1。

[0022] 图 4 视锥体计算示意图 2。

[0023] 图 5 测试效果图。

具体实施方式

[0024] 本发明的一个优选实施例结合附图说明如下：本复杂室内场景快速绘制方法，共分五步：

[0025] 第一步：将整个场景划分成 n 个区域

[0026] 如图 1 所示，在 Portal 入口技术的概念中整个场景被划分 n 个区域 (cell)，每个区域都是一个闭合的凸多边形，这里“凸”表示在区域内的任意两点间画一条线段，这条线

段不会穿透该区域的任何一个多边形，而闭合表示从区域内发出一条光线，如果要让光线射到区域外，则此光线必须要穿透某个区域的多边形．因此可以形象地把一个区域看成是一个房间，区域的多边形构成了房间的墙壁、天花板和地板等（如果要描述一个“凸”的房间，需要将这个房间划分成若干个“凸”区域，然后用 Portal 入口将它们连接起来）。两个邻近的区域通过 Portal 入口连接。

[0027] 第二步：区域结构设计

[0028] 区域的划分是由美工前期确定的，划分有两个主要条件，一是区域内部不能有墙或者其他遮挡体形成较大范围的遮挡，二是该区域的形状必须是凸多面体，而所有的几何节点将以其所在位置和包围盒信息对号入座至相应的区域，两个区域将通过窗或门结构进行连通，每个房间就是一个区域，而连接房间与房间之间的门（窗）就是 Portal 入口。在视域剔除前我们就假定所有当前区域（视点处于的那个区域）中的几何节点都是可见的，而相邻的区域中只用那些处于门窗可视范围内的几何节点才可见，不相邻的区域中的所有几何节点均不可见。

[0029] 本发明中，设计的区域数据结构如下：

[0030] Class Cell

[0031] {

[0032] wstring mName；

[0033] TPortalList mPortals；

[0034] TNodeList mNodes；

[0035] TOBBList mBoxes；

[0036] bool mIsVisualize；

[0037] }

[0038] 其中 mName 是该区域的名字，也就是引擎中的标识号，mPortals 记录了属于这个区域的所有 Portal 入口结构，mNodes 记录了位于区域中的所有几何节点的信息，mBoxes 记录的是组成这个区域的所有 OBB 的序列，由于区域其实就是一个个房间，而房间不一定是方方正正的，所以有些情况下需要用多个 OBB 才能准确的描述一个区域，mBoxes 就是用来记录这些 OBB 的。最后一个布尔变量 mIsVisualize 是用来记录该区域是否可见的。

[0039] 第三步：Portal 入口结构设计

[0040] Portal 入口模型结构如图 2 所示，这里将 Portal 入口抽象成一个平面的长方形，所以在数据结构中只要记录其中心坐标，两个轴役及轴向长度即可，通过这些参数我们可以计算出 Portal 入口四个顶点的坐标值：

[0041]

$$A = P_c - k_0\vec{n}_0 - k_1\vec{n}_1 \qquad (1)$$

[0042]

$$B = P_c + k_0\vec{n}_0 - k_1\vec{n}_1 \qquad (2)$$

[0043]

$$C = P_c + k_0\vec{n}_0 + k_1\vec{n}_1 \qquad (3)$$

[0044]

$$D = P_c - k_0\vec{n}_0 + k_1\vec{n}_1 \qquad (4)$$

[0045] 式中 P_c 代表模型的中心坐标 mCenter，$\vec{n}$ 代表两个方向向量，k 代表方向向量上的长度。

[0046] Portal 入口是从属于区域的，也就是说每个 Portal 入口有且仅有一个区域作为其父节点。由于一般来说墙面是有厚度的，所以对于一个连接通道而言在墙的正反面需要两个 Portal，而这两个 Portal 入口模块就各自需要一个指向对方的指针来说明其连接情况，当然还要考虑到有些时候门窗关闭的情况，我们需要有一个变量来说明 Portal 入口的状态。主要数据结构如下所示：

[0047] Class Portal

[0048] {

[0049] TVector3 mCenter；

[0050] TVector3 mAxes[2]；

[0051] TVector2 mExtents；

[0052] TCell*mCell；

[0053] TPortal*mAdjacent；

[0054] bool mOpen；

[0055] }

[0056] 其中 mCenter 记录空间位置的坐标，mAxes 记录两个方向向量，利用一个二维向量 mExtents 记录两个方向向量上的长度值，mCell 记录所属区域的指针，mAdjacent 记录的是相连的 Portal 入口的指针，最后一个布尔向量记录该 Portal 入口的闭合情况。

[0057] 第四步：新视锥体生成算法

[0058] 新视锥体的生成分为两种情况：

[0059] 第一种情况是整个 Portal 入口完全处于原视锥体范围内部，在这种情况下，新的视锥体其实就是将原视锥体变窄后的模样如图 3 所示：图中 O 点代表的是视点（假设视点处于原点上），粗线条代表的是墙，墙上的缺口就是 Portal 入口，可以理解为门或者窗，ABCD 表示的是原视锥体，但是对 O 点来说墙后的物体是不可见的，所以连接 OF 和 OP 分别交 AD 于 E 点和 H 点，作 FG 平行于 BC，这样一个新的视锥体 FGHE 就生成了，FG 为新视锥体的近平面，HE 为新视锥体的远平面，三个方向轴不变。对于 O 点来说，处于墙另一侧的物体只有处在这个新的视锥体内才是可见的。新视锥体将通过如下步骤求得：

[0060] 1. 首先计算 Portal 入口的四个顶点到原视锥体近平面 BC 的最小距离，也就是求出各点指向 B 点的向量，然后求出这些向量与 BC 平面法向量的内积，取出其中的最小值，由图上可知最后求得的点是 F 点，最短距离为 IK 线段所代表的长度，具体公式如下（以 F 点为例）：

[0061]

$$D = (\overrightarrow{OB} - \overrightarrow{OF}) \bullet \vec{n} \qquad (5)$$

[0062] 式中 $\overrightarrow{OB}$ 和 $\overrightarrow{OF}$ 分别代表原点指向 B 点和 F 点的向量，$\vec{n}$ 代表平面法向量。

[0063] 2. 由于视点未改变所以新视锥体的三个方向向量以及中心点坐标与原视锥体完全相同，而新的近平面距离 (mNear) 就等于原近平面距离加上上一步求得的最小值 (IK)，

远平面 (mFar) 距离值保持不变和原值相同。

[0064] 3. 新视锥体的左值 (mLeft) 即图中 FI 线段的长度，可以将 OF 向量内积 mAxes0 的负向量求得。

[0065] 4. 新视锥体的右值 (mRight，图中 IG 线段长度的) 的求解相对复杂，首先需要求得 F 点和 G 点在 mAxes2 向量方向上的差值，然后求得 OG 在 mAxes0 向量方向上的投影，利用相互比例进行求解，具体公式如下 ：

[0066]

$$IG = (\overrightarrow{OG} \bullet \vec{n}_0) \times (IK + OK) / (IK + OK + \overrightarrow{GF} \bullet \vec{n}_2) \quad (6)$$

[0067] 式中$\vec{n}_0$和$\vec{n}_2$分别代表 mAxes0 和 mAxes2 这两条向量

[0068] 5. 由于图是俯视图所以竖直向量无法在图中表示，但利用同样的方法可以求得上值 (mTop) 和下值 (mBottom)。

[0069] 第二种情况是 Portal 入口部分处于原视锥体范围内部，在这种情况下，新的视锥体的形状相当于原视锥体的侧边半部中的一小部分，如图 4 是一个新视锥体计算的示意图 ：此图和第一种情况唯一的不同是墙面向左平移了一段距离，部分 Portal 入口位于视锥体外侧，像这样的情况我们首先连接 OP 交 AD 于 E，在 P 点做 BC 的平行线交 AB 于 G，四边形 AGPE 就是新的视锥体，且视点和三个方向向量和原视锥体保持一致。新视锥体通过如下方法可以计算获得 ：

[0070] 1. 首先计算处在视锥体内的 Portal 入口顶点到原视锥体近平面 BC 的距离，由图上可知需要求解距离的点是 P 点，最短距离为 IP 线段所代表的长度，具体公式如下 (以 P 点为例) ：

[0071]

$$D = (\overrightarrow{OB} - \overrightarrow{OP}) \bullet \vec{n} \quad (7)$$

[0072] 式中$\overrightarrow{OB}$和$\overrightarrow{OP}$分别代表原点指向 B 点和 P 点的向量，$\vec{n}$代表平面法向量。

[0073] 2. 这一步和第一种情况相同，新的近平面距离 (mNear) 就等于原近平面距离加上上一步求得的最小值 (IP)，远平面 (mFar) 距离值保持不变和原值相同。

[0074] 3. 新视锥体的右值 (mRight) 即图中 KI 线段长度的负值，可以将 OP 向量内积 mAxes0 向量求得 (求得直接为负值)。

[0075] 4. 由图上可知新视锥体与原视锥体共用一条左边，所以他们的左值是相同的，即新视锥体 mLeft 就等于原视锥体 mLeft。

[0076] 5. 由于图是俯视图所以竖直向量无法在图中表示，但利用同样的方法可以求得上值 (mTop) 和下值 (mBottom)。

[0077] 第五步 ：根据生成的新视锥体渲染顶点

[0078] 利用计算的视锥体进行视域剔除，直到所有的可视区域都完成了视域剔除，如此获得的所有几何节点形成的序列就是需要进行渲染的节点。这时渲染的顶点比处理前顶点的个数要少了很多很多，因为有很多顶点都由于 Portal 入口产生的视锥体进行了视域剔除而被排除在渲染之外，从而大大提高渲染效率。

[0079] 实验结果

[0080] 本实验在 vc.net 的编译环境下使用 Direct3D 渲染某室内场景，按照上述方法场

景管理。测试平台如下表所示：

[0081] 表 5.1 测试平台

[0082]

测试平台	
CPU	Inter® Core (TM) 2 2.4GHz
硬盘	WD320G、7200 转
内存	1GB×2 ， DDRII 667
显卡	Quadro FX 3450、 256M 显存

[0083] 基于上述硬件平台我们对于一个有 110 个节点、14 个区域、20 个 Portal 入口的场景进行测试。测试结果对比示于表 5.2。

[0084] 表 5.2 不同情况下渲染速度的比较

[0085]

测试的 不同遮挡程度	不应用 Portal 入口的帧数	应用 Portal 入口后 的帧数	提高的效 率
较少	61fps	81fps	32.79%
中等	39fps	79fps	102.56%
高	35fps	122fps	248.57%

[0086] 从测试数据中不难发现，遮挡率越高越能体现 Portal 入口算法的优越性，如表第三行所示，在高遮挡的情况下，渲染帧数竟然有 2.5 倍的提升，使渲染效率有了质的飞跃。测试效果图如图 5 所示。

[0087] 实验结果表明，本发明的一种基于视域剔除的复杂室内场景快速绘制方法，克服现有图形引擎绘制技术、尤其是在绘制室内场景时的速度不够的缺点，提供了一种基于视域剔除的复杂室内场景快速绘制方法，大大提高了在绘制室内场景时的绘制速度，保证了虚拟现实的实时性要求。

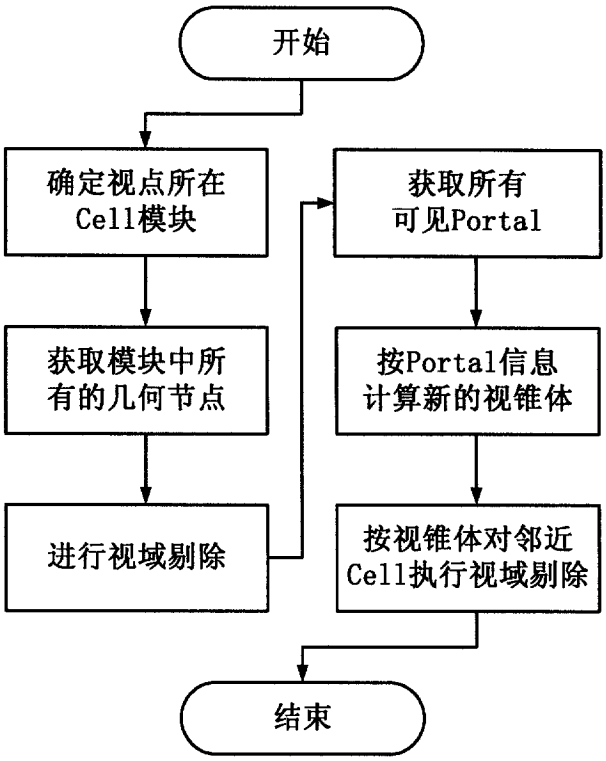

开始
确定视点所在Cell模块
获取模块中所有的几何节点
进行视域剔除
获取所有可见Portal
按Portal信息计算新的视锥体
按视锥体对邻近Cell执行视域剔除
结束

图 1

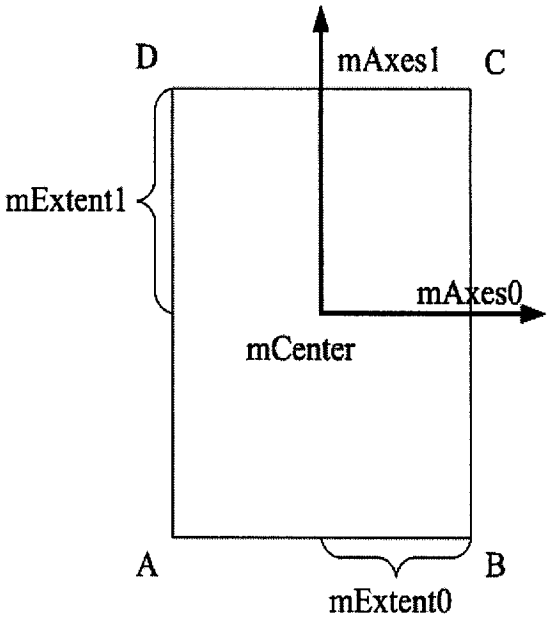

D
mAxes1
C
mExtent1
mAxes0
mCenter
A
B
mExtent0

图 2

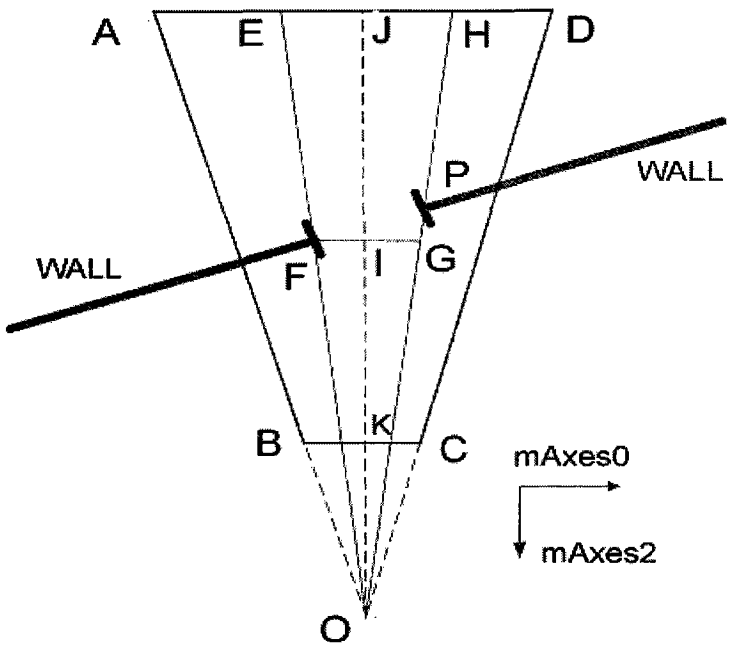

A
E
J
H
D
P
WALL
WALL
F
I
G
K
B
C
mAxes0
mAxes2
O

图 3

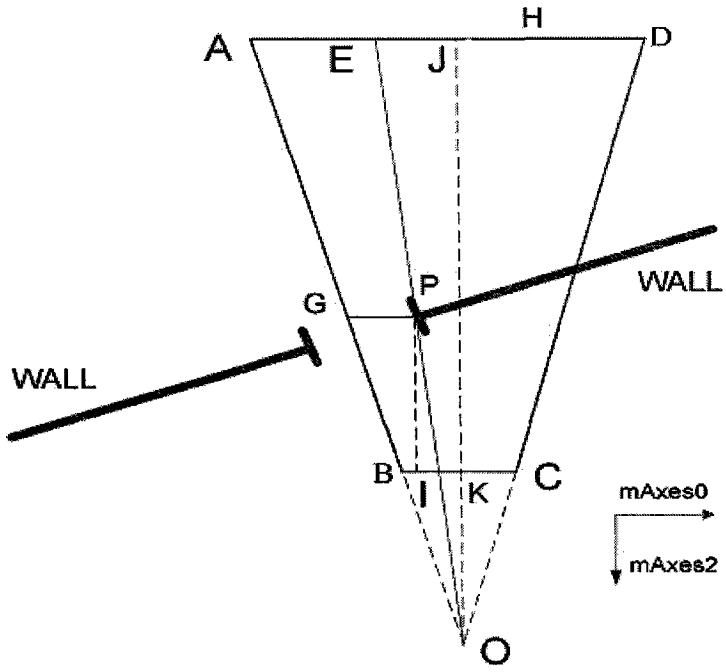

H
A
E
J
D
P
WALL
G
WALL
B
I
K
C
mAxes0
mAxes2
O

图 4

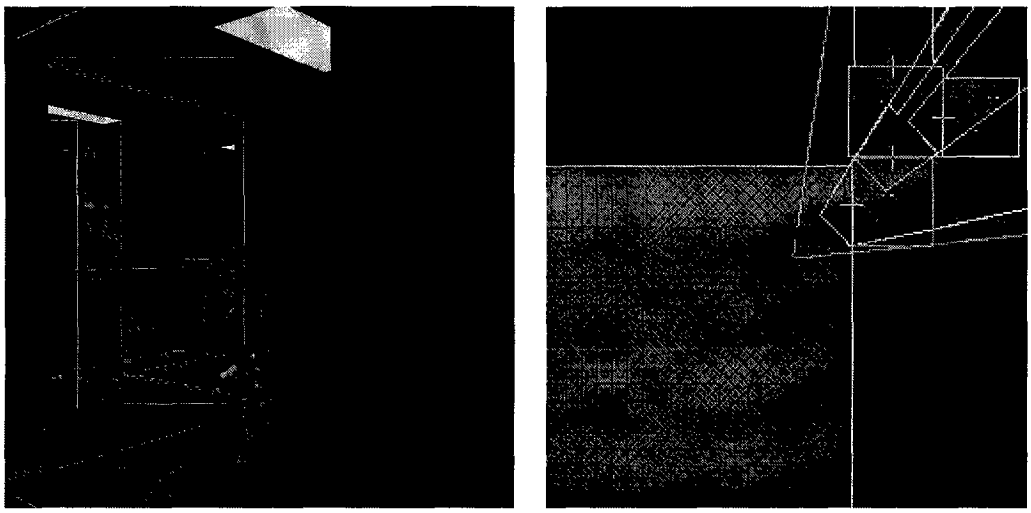


图 5