[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06T 17/10
G06T 17/00 A63H 33/00

[12] 发明专利说明书

[21] ZL 专利号 00804119.9

[45] 授权公告日 2004 年 9 月 22 日

[11] 授权公告号 CN 1168048C

[22] 申请日 2000.1.21 [21] 申请号 00804119.9
[30] 优先权
[32] 1999. 1.22 [33] DK [31] PA199900095
[86] 国际申请 PCT/DK2000/000027 2000.1.21
[87] 国际公布 WO2000/043959 英 2000.7.27
[85] 进入国家阶段日期 2001.8.22
[71] 专利权人 英特莱格公司
地址 瑞士巴尔
[72] 发明人 迈克尔·托姆森 保尔·克罗格
奥尔加·蒂姆森科
审查员 韩 燕

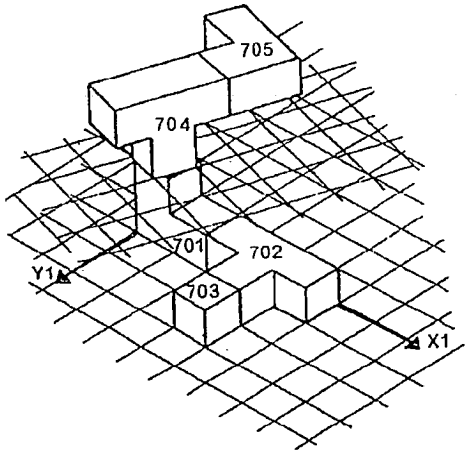
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 李 强

权利要求书 7 页 说明书 32 页 附图 12 页

[54] 发明名称 生成/解释几何对象的计算机可读模型的方法和计算机系统

[57] 摘要

一种几何对象模型，包括：第一数据结构中的位，这些位得到编码，以标识组件集合的几何形状之表示集合中的第一组组件，并利用第一坐标系中的整数坐标来表示组件的位置；第二数据结构中的位，这些位得到编码，以表示第一坐标系相对于第二坐标系的空间变换。 从而可以创建几何对象模型的紧凑表示。 初步看来，该模型似乎很受约束，实际上，可以非常灵活、方便地访问该模型。



ISSN1008-4274

1. 一种生成几何对象的计算机可读模型的方法，该模型是在能够访问数据库的计算机上生成的，该数据库包含组件的一个集合的几何形状表示，该方法包含以下步骤：

对一个第一数据结构内的位进行编码，以标识集合内的第一组组件，并利用一个第一坐标系中的整数坐标来表示组件的约束位置；

对一个第二数据结构内的位进行编码，以表示一个第二坐标系相对于第一坐标系的空间变换；

将第一和第二数据结构作为一个几何对象的计算机可读模型。

2. 根据权利要求 1 的方法，还包括以下步骤：

对一个第三数据结构内的位进行编码，以标识所述集合内的一个第二组的组件，并利用第三坐标系中的整数坐标，表示组件的位置；以及将这些位包含到计算机可读模型中；

对所述第二数据结构内的位进行编码，以表示第三坐标系相对于第二坐标系的约束空间变换。

3. 根据权利要求 1 或 2 的方法，还包括以下步骤：

对所述第二数据结构内的位进行编码，以表示轴的位置，并表示第一坐标系相对于第二坐标系围绕该坐标轴的旋转。

4. 根据权利要求 1 或 2 的方法，其中对所述第三数据结构内的位进行编码的步骤还包括以下步骤：对所述第二数据结构内的所有位进行编码，以利用第二坐标系中的整数坐标表示轴的位置。

5. 根据权利要求 1 或 2 的方法，其中对第三数据结构内的所有位进行编码的步骤还包括以下步骤：对所述第三数据结构内的所有位进行编码，以便通过标识沿一组预定的互相垂直的坐标轴中一个坐标轴的方向，表示该坐标轴的方向。

6. 根据权利要求 2 的方法，还包括以下步骤：对所述第一和第三数据结构内的所有位进行编码，以分别表示第一和第三坐标系中组件的方向。

7. 根据权利要求 1 或 2 的方法，还包括以下步骤：对所有位进行编码，以便通过标识沿一组预定的互相垂直的坐标轴中一个坐标轴的方向，表示该组件的方向。

8. 根据权利要求 1 或 2 的方法，还包括以下步骤：对一个第四数据结构内的所有位进行编码，以表示第一和第二坐标系相对于第四坐标系的空间变换；以及将第四数据结构包含到几何对象的计算机可读模型中。

9. 根据权利要求 1 或 2 的方法，其中各整数坐标能够标识三维体积，反之亦然，并且其中三维体积的大小适合于利用整数坐标处理两个组件的位置表示，从而两个组件结合以形成一个粘在一起的几何对象。

10. 根据权利要求 1 或 2 的方法，其中组件包含类似于众所周知的玩具建筑组件的几何形状的表示。

11. 根据权利要求 1 或 2 的方法，其中组件包括其至少一部分形状类似组件可能存在的互连的连接装置或形象的几何形状的表示。

12. 根据权利要求 1 或 2 的方法，其中组件集合的几何形状表示包括坐标系中 real 类型的数值给出的坐标。

13. 根据权利要求 1 或 2 的方法，还包括以下步骤：对所述第三数据结构内的所有位进行编码，以表示相对于第二坐标系之原点的整数坐标；原点定义与第一坐标系中的组件进行互连的坐标；响应第二坐标系中的互连的新坐标，定义第三坐标系的新原点；作为新原点的结果，计算第三坐标系内组件的新位置；对所述第三数据结构内的所有位进行编码，以表示计算结果。

14. 根据权利要求 3 的方法，还包括以下步骤：对所述第一数据结构内的所有位进行编码，以表示轴的位置，从而定义第一坐标系中的第二坐标系中的组件的互连坐标；定义第一坐标系内的互连的新坐标，以表示第一坐标系内的组件相对于第二坐标系的新位置；以及对所述第一数据结构内的所有位进行编码，以表示坐标轴的新位置，新位置为互连的新坐标。

15. 根据权利要求 1 或 2 的方法，还包括以下步骤：提供表示一个用户与一个用户界面之间的交互的信号，该交互包括修改和/或处理计算机可读模型的几何表示；以及，将该信号转换为计算机可读模型或改进的计算机可读模型。

16. 根据权利要求 1 或 2 的方法，还包括以下步骤：经由连接到计算机的通信网络，传输经过编码的位或计算机可读模型。

17. 根据权利要求 16 的方法，还包括以下步骤：经由连接到计算机的通信网络，在计算机上接收经过编码的位或计算机可读模型。

18. 根据权利要求 1 或 2 的方法，还包括以下步骤：

连接到本地数据库和远程数据库以便进行数据传送；

对于各组件，首先查询本地数据库，以检查是否可以从本地获得几何形状的表示；

第二，如果能够从本地获得几何形状的表示，则从本地数据库检索几何形状，或者作为选择，如果不能从本地获得几何形状的表示，则从远程数据库检索几何形状，如果远程数据库中存在该几何形状的话。

19. 根据权利要求 2 的方法，还包括以下步骤：生成表示几何形状的可显示数据，作为根据第一、第二和第三数据结构描述的空间相互关系。

20. 一种计算机系统，该系统被编程以生成几何对象的一种计算机可读模型，该模型是在能够访问数据库的计算机上生成的，该数据库包含组件的一个集合的几何形状表示，该计算机可读模型的生成包含以下步骤：

对一个第一数据结构内的位进行编码，以标识集合内的第一组组件，并利用一个第一坐标系中的整数坐标来表示组件的约束位置；

对一个第二数据结构内的位进行编码，以表示一个第二坐标系相对于第一坐标系的空间变换；

将第一和第二数据结构作为一个几何对象的计算机可读模型。

21. 一种解释一个几何对象的一个计算机可读模型的方法，该模

型是在具有用户界面的计算机上得到解释的，该计算机能够访问一个数据库，该数据库包含一个组件集合的几何形状的表示，该方法包括以下步骤：

对一个第一数据结构内的位进行解码，以标识集合内的第一组组件，并利用一个第一坐标系中的整数坐标来标识组件的位置；

对一个第二数据结构内的位进行解码，以标识第一坐标系相对于第一坐标系的空间变换；

获取所标识的几何形状的表示，并生成表示几何形状的可显示数据，而它们的空间相互关系根据第一和第二数据结构而得到描述。

22. 根据权利要求 21 的方法还包括以下步骤：

对一个第三数据结构内的所有位进行解码，以标识所述集合内的一个第二组的组件，并利用第三坐标系中的整数坐标，标识组件的位置，以及将这些位包含到计算机可读模型中；

对所述第二数据结构内的位进行解码，以标识第三坐标系相对于第二坐标系的约束空间变换。

23. 根据权利要求 21 或 22 的方法还包括以下步骤：

对所述第二数据结构内的所有位进行解码，以标识轴的位置，并标识第三坐标系相对于第二坐标系围绕该坐标轴的旋转。

24. 根据权利要求 21 或 22 的方法，还包括以下步骤：对所述第二数据结构内的所有位进行解码，以利用第二坐标系中的整数坐标标识轴的位置。

25. 根据权利要求 21 或 22 的方法，还包括以下步骤：对所述第三数据结构内的所有位进行解码，以便通过标识沿一组预定的互相垂直的坐标轴中一个坐标轴的方向，标识该坐标轴的方向。

26. 根据权利要求 22 的方法，还包括以下步骤：对所述第一和第三数据结构内的所有位进行解码，以分别标识第一和第三坐标系中组件的方向。

27. 根据权利要求 26 的方法，还包括以下步骤：对位进行解码，以便通过读取沿一组预定的互相垂直的坐标轴中一个坐标轴的方向，

标识该组件的方向。

28. 根据权利要求 21 或 22 的方法，还包括以下步骤：对一个第四数据结构内的所有位进行解码，以标识第一和第二坐标系相对于第四坐标系的空间变换；以及修改可显示数据，以根据空间变换进行显示。

29. 根据权利要求 21 或 22 的方法，其中各整数坐标能够标识三维体积，反之亦然，并且其中三维体积的大小适合于利用整数坐标处理两个组件的可显示位置，从而两个组件结合以形成一个粘在一起的几何对象。

30. 根据权利要求 21 或 22 的方法，其中组件包含类似于众所周知的玩具建筑组件的几何形状的表示。

31. 根据权利要求 21 或 22 的方法，其中组件包括其至少一部分形状类似组件可能存在的互连的连接装置或形象的几何形状的表示。

32. 根据权利要求 21 或 22 的方法，其中组件集合的几何形状表示包括坐标系中 real 类型的数值给出的坐标。

33. 根据权利要求 22 的方法，还包括以下步骤：

对所述第三数据结构内的所有位进行解码，以表示相对于第二坐标系之原点的整数坐标；原点定义与第一坐标系中的组件进行互连的坐标；

响应第二坐标系中的互连的新坐标，定义第三坐标系的新原点；

作为新原点的结果，计算第二坐标系内组件的新位置；

对第三数据结构内的所有位进行编码，以表示计算结果。

34. 根据权利要求 22 的方法，还包括以下步骤：

对第一数据结构内的所有位进行解码，以表示轴的位置，从而定义第一坐标系中的互连的坐标；

定义第一坐标系内的互连的新坐标，以表示第一坐标系内的组件相对于第二坐标系的新位置；

对第一数据结构内的所有位进行编码，以表示坐标轴的新位置，新位置为互连的新坐标。

35. 根据权利要求 25 的方法，还包括以下步骤：

提供表示一个用户与一个用户界面之间的交互的信号，该交互包括修改和/或处理计算机可读模型的几何表示；

将该信号转换为计算机可读模型或改进的计算机可读模型。

36. 根据权利要求 21 或 22 的方法，还包括以下步骤：通过计算与该数据结构相对应的几何形状在平面或曲面上出现时的投影，将计算机可读模型转换为可显示数据。

37. 根据权利要求 21 或 22 的方法，还包括以下步骤：从表示计算机可读模型的二进制文件类型中读取计算机可读模型。

38. 根据权利要求 21 或 22 的方法，还包括以下步骤：经由连接到计算机的通信网络，传输所有位或计算机可读模型。

39. 根据权利要求 38 的方法，还包括以下步骤：经由连接到计算机的通信网络，在计算机上接收所有位或计算机可读模型。

40. 根据权利要求 21 或 22 的方法，还包括以下步骤：

断开数据库到本地数据库和远程数据库的连接；

对于各组件，首先查询本地数据库，以检查是否可以从本地获得几何形状的表示；

第二，如果能够从本地获得几何形状的表示，则从本地数据库检索几何形状，或者作为选择，如果不能从本地获得几何形状的表示，则在远程数据库中存在该几何形状的情况下从远程数据库检索几何形状。

41. 一种计算机系统，该系统被编程以解释一个几何对象的一个计算机可读模型，该模型是在具有用户界面的计算机上得到解释的，该计算机能够访问一个数据库，该数据库包含一个组件集合的几何形状的表示，该计算机可读模型的解释包括以下步骤：

对一个第一数据结构内的位进行解码，以标识集合内的第一组组件，并利用一个第一坐标系中的整数坐标来标识组件的位置；

对一个第二数据结构内的位进行解码，以标识第一坐标系相对于第一坐标系的空间变换；

获取所标识的几何形状的表示，并生成表示几何形状的可显示数据，而它们的空间相互关系根据第一和第二数据结构而得到描述。

生成/解释几何对象的计算机可读模型的方法和计算机系统

发明领域

本发明涉及依靠预定几何组件的虚拟现实的计算机辅助建模领域。

发明背景

虚拟现实的计算机辅助建模的任务是在计算机系统中创建几何对象的模型，解释模型，处理模型，以及处理几何对象的模型。

从第一个观点看，虚拟现实的建模是一个感兴趣的话题，因为可以在现实世界中实现想法之前观察想法。与在现实世界中开发和改善几何对象的过程的相比，如果虚拟现实模型的修改非常容易，则在该过程中能够节省大量时间。在现实世界中，给对象涂色的简单任务可能需要几小时，而计算机可以在几毫秒或几秒内应用一种新的颜色，以观察模型。

从第二个观点看，虚拟现实的建模是非常有趣的，因为可以创建现实世界中存在的对象的模型，并且借助计算机在某种意义上观察并处理该模型。因此，可以为不同目的，如为高级文档管理的目的，存储现实世界中对象的模型。

尽管存在计算机辅助虚拟现实建模的大量应用，但是一种特殊应用是使用虚拟现实建模进行娱乐或教育。因为在现实世界中，在发明电子计算机很久以前，就已经熟知各种类型的建模概念。特别是，使用积木或半积木概念的概念在过去、在现在都是非常普遍的。通常，上述概念提供一组预制组件，其中可以根据预制组件模块，以某种预定方式，互连预制组件。预制组件类似适合特定建模任务的众所周知的对象。因此，在例如构建房屋模型时，该组件可类似墙砖、瓦、门和窗户。以此种方式选择组件的目的在于，与每次构建新模型时均需定义房屋的所有细节的情况相比，能够显著降低构建房屋模型所包含

的工作。然而，需要为构建模型的简洁性，折衷选择构建房屋或其他对象时全部自由。

在计算机辅助虚拟现实建模技术中，上述具有预定组件的方法是众所周知的。同时，具有可互连的积木组件的概念也是众所周知的。但是，当在计算机中表示此类模型时，现有技术不能完全实现具有预定组件和积木系统的概念的便利。

只要计算机还要用于计算机辅助设计和建模，需要大量计算工作的任务就是模型的可视化，包括计算模型的出现方式。扩充虚拟现实模型之复杂性和精巧程度的原因之一在于，追随最近可用的计算机技术和计算能力。

当传送或交换表示模型的数据时，虚拟现实模型的复杂性和精巧程度的缺点表现的非常清楚。现在，将以存储部件或计算机通信网络形式的某些类型的外部部件，连接到计算机，从而扩展了存储、加载和/或传送、接收模型的信号通路。通常，与单一计算机内的信号通路相比，此类信号通路具有较低的带宽。因此，需要一种表示模型的有效模式。

现有技术

一种表示几何对象模型的模式是众所周知的“虚拟现实建模语言”，也称为“VRML”。通常在因特网上使用此模式。VRML 基于利用一组参数定义的某些类型的多边形。通过指定参数，如在三维空间中定义多边形的各角的位置，在上述多边形上建立所有模型。依靠公共变换，可以定义一组多边形的位置、方向和比例。

依靠上述建模模式，可以根据需要模拟所有可能的建模任务，这是因为该模式并不对用户进行任何限制。事实上，这也正是该模式的缺点之一，因为需要利用大量表示支持每个多边形。在计算机上，此表示包含模型中各参数的大量 Real 类型。另外，由于大部分建模范例的本性，需要大量多边形 — 从而需要大量 Real 类型的参数。例如，考虑一个曲面，在非常粗糙的模型中，可以依靠单一多边形模拟此类曲面，然而，如果需要一个精细模型，则该曲面必须由许多多边形组

成，至少包括许多多边形以至该曲面以足够的光滑度出现。

当处理 VRML 模型时，必须利用定义各种可能处理的元描述，扩展几何描述。因此，如果 VRML 模型最初不提供此类元描述，则不能处理该模型。这是完全自由建模模式的结果。

因此，现有技术存在以下问题，即利用计算机辅助建模获得的模型需要大量存储能力。

在现有技术中，存在各种类型的压缩模式，然而，此类压缩模式需要处理大量数据以便有效地压缩数据。此外，在执行压缩和解压缩时，此类压缩模式需要比较大的时隙。通常，当开始包含例如通过计算机网络传送模型的交互建模任务时，这是很关键的。因此，需要一种能够快速生成、解释、并且不需要附加压缩模式的几何模型表示。

本发明的概述

本发明的第一目的在于提供几何对象模型的一种紧凑表示。

本发明的第二目的在于提供几何对象模型的一种紧凑表示，该表示允许模拟包含可移动和/或可转动部分的复杂对象。

本发明的第三目的在于提供一种能够快速生成和解释的模型结构和该模型结构的表示。

本发明的第四目的在于提供一种易于表示由积木玩具构建块或组件组成的对象的模型结构和该模型结构的表示。

本发明的第五目的在于提供一种模型结构表示，当必须可视化该模型、必须生成或处理该模型时，在较低带宽需求和较少处理时间的意义上，易于通过计算机网络分发该模型结构的表示。

本发明的第六目的在于提供一种模型结构和该模型结构的表示，其中当必须处理该模型时，该模型结构的表示并不需要几何对象的附加元描述。

根据本发明，在权利要求书定义的本发明的范围内，实现在上述各段中提及的目的。具体而言，当几何对象的模型包含：第一数据结构中的所有位，对所有位进行编码，以标识组件集合的几何形状之表示集合中的第一组组件，并依靠第一坐标系中的整数坐标，表示组件

的位置；第二数据结构中的所有位，对所有位进行编码，以表示第二坐标系相对于第一坐标系的空间变换。

从而获得几何对象的非常紧凑的表示，同时该模型便于模拟包含可移动部分的复杂集合结构。事实上，该模型区分固定部分，并将组件的位置约束在此固定部分内，以便获得紧凑表示。然而，当需要表示此类固定部分或段的空间变换时，可以应用相互关系的非约束表示或较少约束的表示。从而也能够提供相对灵活的模型表示。

由于组件位置的整数表示，所以能够在计算机上一再处理该模型，而不会在各自的坐标系内引入舍入误差。因此，其某段内包含大量组件的模型不会由于舍入误差的结构而变形。

由于以下事实，即该模型仅包含对组件之几何表示的引用这一事实，所以利用所选的组件集解释该模型，而无需修改模型。因此，可以选择特定的图形格式，以便需要显示该模型时，在特定计算机上方便地解释，例如在提供足够快的绘制时间方面。

由于各自坐标系内组件位置的整数约束表示的结构，所以隐式给出组件的所有可能位置，即旋转、自由位移等在坐标系内是不可能的。

由于通过约束组件获得模型的紧凑表示，所以不需要附加的压缩，因此，通过写入/读取一连串的用于表示模型的语句，能够顺序生成并顺序解释模型结构。

附图说明

以下将参照附图连同最佳实施方式详细解释本发明，其中附图为：

图 1 表示根据现有技术的虚拟现实建模语言的几何对象模型的层次结构；

图 2a 表示根据本发明建模的第一对象主题；

图 2b 表示根据本发明建模的第二对象主题；

图 3 表示根据本发明的几何对象模型的层次结构；

图 4 表示两组组件；

图 5 表示构建第三几何对象模型过程中的第一步；

图 6 表示构建第三几何对象模型过程中的第二步；

图 7 表示构建第三几何对象模型过程中的第三步;

图 8 表示根据本发明建模的第四对象;

图 9 表示用于根据本发明设计、存储、处理并共享几何模型的客户计算机系统;

图 10 表示用于根据本发明处理几何构造的服务器计算机系统;

图 11 表示需要处理段和模型时并行构造段和模型的方法的流程图;

图 12 表示访问并共享可由多台客户机访问的公用坐标的方法的流程图; 以及

图 13 表示传送模型信息的方法的流程图。

最佳实施例的详细描述

图 1 表示根据现有技术的虚拟现实建模语言的几何对象模型的层次结构。对熟练技术人员而言, 诸如 VRML 之类的虚拟现实建模语言是众所周知的。VRML 的基本组件是利用其三维直角坐标系中的坐标定义的二维非凹多边形。如果将三维直角坐标系的轴分别表示为 x 轴、y 轴和 z 轴, 则可以利用矩形的四个角的坐标, 定义可以为矩形的二维凸多边形。即四组 (x, y, z) 坐标。由于面向多边形的 VRML 结构, 所以设计虚构或现有的现实世界对象的虚拟现实模型的设计者, 必须将该对象划分为多边形。因此, 具有弯曲的光滑曲面的对象之精确、精细的 VRML 模型, 必须由大量多边形构成, 以便使得该模型以具有精细的光滑曲面的形式出现。然而, 可以由较少数目的多边形构成该对象之粗糙的 VRML 模型, 因此, 降低模型的复杂度。为了使设计者能够模拟任何对象, 必须能够在三维坐标系 (在下文中, 将此表示为 “三维空间” 或仅表示为 “空间”) 中, 自由放置二维多边形。在计算机中, 利用许多所谓的 Real 类型, 即浮点数, 表示多边形的坐标。

为了处理更加简单的模型, 以倒置的树结构层次 101 排列多边形。将该树的每个分支 102 连接到某个节点, 以定义层次。该层次给定层上的各节点为该层次较低层上的其他节点或多边形的公共引用。层次在作为该层次最底层的多边形的向下方向上终止。

利用坐标 x、y 和 z 定义多边形 103 和 104，其中 x、y 和 z 通过接口 105 和 106 引用同一节点 107。具有多边形（未示出）的其他节点 108 也可以引用节点 107。通过节点 107 的接口部分 109，可以在利用公共变换直接或间接连接到节点 107 的层次中的较低层上，平移、旋转和缩放多边形和节点。因此，可以对一组多边形进行分组，并通过包含平移、旋转和缩放的单一变换处理该组。

在多边形和节点的定义中使用的所有参数，即坐标、平移、旋转和缩放，为 Real 类型，因此在计算机中用浮点数表示。选择此表示以便能够处理所有建模任务。然而，此表示需要大量存储能力，并且当需要存储和传送模型时 — 包括实时处理模型时，此表示实际上是一个障碍。

图 2a 表示根据本发明建模的包含第一和第二段的简单对象。该对象的模型具有两段 201 和 202，各段分别具有组件 203、204、205 和 206、207。轴 x_1、y_1 和 z_1 跨越坐标系 209，并且属于段 201，轴 x_2、y_2 和 z_2 跨越坐标系 211，并且属于段 202。

请注意，仅根据坐标系 209 的模块 208，放置第一段的组件 203、204 和 205。同样，仅根据坐标系 211 的模块 210，放置第二段的组件 206 和 207。

下面说明将该对象划分为两段的目的。

图 2b 表示图 2a 所示的简单对象，其中第二段相对于第一段旋转。在此图中，显然存在两个分别由轴 x_1、y_1、z_1 和轴 x_2、y_2、z_2 定义的不同的、不一致的坐标系。可以仅根据坐标网格 209 的模块 208，放置段 201 内的组件。相应地，可以仅根据坐标网格 211 的模块 210，放置段 202 内的组件。从而相对于其位置，约束各段内的组件。以二维坐标网格的形式表示坐标网格，但是，通常坐标网格为三维坐标网格，以便在三维空间中排列组件。另外，相对于各段的三个坐标轴（分别为 x_1、y_1、z_1 和 x_2、y_2、z_2），约束各段内所有组件的方向，以 90°的步长旋转。

根据段 201 的坐标网格 209 放置段 202，但是，允许段 202 相对

于轴 z_1 旋转。现在，扩展了上面的简单模型，并且可以模拟包含一连串可旋转段的高级对象。请注意，尽管将以上旋转特征引入到简单模型中这一事实，也仅仅使用了表示旋转角度的一个数值来扩展其表示，这是由于将段 202 的位置约束到坐标网格 209 以及将组件 204 和 205 约束到坐标网格（即坐标网格 211）的缘故。

通过将更多组件逐渐添加到该段中，并互连具有组件的新段和现有段 201 和 201，可以扩展该模型。事实上，尽管以位置和方向约束形式的大量约束，仍然可以按上述方式模拟更复杂的几何对象。上述大量约束是模拟复杂几何对象之模型的紧凑表示的关键。

具体而言，当需要模拟的对象包含能够互相移动或旋转的比较少的固定部分时，并且当必须利用许多组件来模拟固定部分时，根据本发明的模型比现有技术的虚拟现实模型更优越。例如，此类对象的示例可以为具有两片桥页的桥梁。此对象的模型可以仅包含两个可移动的固定部分，即桥页，但是各桥页和支撑桥页的平台可以包含固定结构，如必须依靠许多组件模拟的栅栏、人行道和桥塔。

以下说明实际生成根据本发明的表示的方法，以及将其他特征添加到根据本发明之模型中的方法。

图 3 表示根据本发明的几何对象模型的层次结构。

层次的目的不仅在于提供服从公共变换的几何原型的逻辑分组，而且在于添加在该层次的各层中使用的参数的不同类型的约束，从而能够在考虑约束并在建立该模型的紧凑表示的意义上，选择参数的方便表示。

请注意，该层次涉及组件之间的空间关系 — 而不是该组件的几何形状的实际表示。可以以任何方便格式，如 VRML 或 3DMF，选择组件形状的实际表示。

通过包含四个基础层次的层次 316，组织根据本发明之某一实施方式的模型，其中四个基础层次为：World Level(通用坐标层)、Model Level（模型层）、Segment Level（段层）和 Element Level（组件层）。

在 World Level 上，定义表示虚拟世界的 World Block（通用坐

标块）315。World Block 的主要目的是定义具有原点的三维空间，其中原点为该层次的较低层的公共参考点。

在 Model Level 上，存在许多表示虚拟世界中的模型的 Model Block（模型块）311、314。Model Block 包含字段 312 和 313，以及以 Segment Block（段块）309 和 307 形式的块。在最佳实施方式中，Model Block 中的字段为“Translate（平移）”、“Rotate（旋转）”和“Scale（缩放）”。这些字段定义虚拟世界中模型的位置、方向和大小。允许自由定义模型的位置、方向和大小，即该模型不受任何有关字段约束的限制。在计算机中，依靠 Real 类型的数值，即浮点数，表示字段 Translate、Rotate 和 Scale。

在 Segment Level 上，存在许多表示模型中所有段的 Segment Block（段块）307、309。Segment Block 包含字段 308、310，以及 Element Block（组件块）形式的块 301、302 和/或从属 Segment Block 305。Segment Block 中的字段 310 为 Offset（偏移）、Rotate（旋转）和 Axis（轴）。这些字段定义该段内轴的位置，环绕该轴的旋转角度，以及该轴的方向。从而该轴的位置定义用于将 Segment Block 305 连接到从属 Segment Block 305 的连接。

将该段内轴的位置限制在该段的约束允许的位置。可以依靠三维坐标系设置此类约束，其中可以以模块步长定位所有组件。在本发明的基础实施方式中，不限制环绕该轴的角旋转。将轴的方向限制为以 90°步长围绕坐标系的三个直角轴旋转，例如参见图 2b。

字段 Offset 可以具有三个参数，其中在计算机中利用 Integer 类型的数值表示。作为选择，也可以使用具有三个 Integer 类型参数的字段 Translate。

另外，字段 Rotate 可以具有三个参数，该参数包含三个 Real 类型数值的数组，可以利用三个 Short Integer 类型数值的数组表示 Axis，或者作为选择，通过利用一个比特表示该坐标系中三个坐标轴中的一个坐标轴来表示字段 Axis。

在 Element Level 上，存在许多表示各段内所有组件的 Element

Block（组件块）301、302。Element Block 不包含任何从属块，但是仅包含字段 303、304。因此，组件块沿向下方向终止层次。Element Block 中的字段为 Offset、Orientation（方向）和 Shape Reference（形状引用）。有关位置和方向组件是模型的主要约束部分。字段 Offset 是约束为离散值的三维偏移。如果该约束是通过三维坐标系设置的，其中以模块步长定位所有组件，则可以利用 Integer 类型数值的数组来表示字段 Offset。另外，以在 Segment Block 中约束字段 Axis 的相同方式，约束字段 Orientation，即约束为以 90°步长围绕坐标系的三个直角轴旋转。因此，可以利用三个 Short Integer 类型数值的数组表示 Axis，或者作为选择，通过表示该坐标系中三个坐标轴中的一个坐标轴来表示字段 Axis。

通过字段 Shape Reference，组件包含几何描述的引用，可以解释几何描述并用于可视化目的。形状引用最好为引用数值，其中可以在包含各组件之几何描述的数据库的查询中，使用该引用数值。通常，模型将基于一组不同类型的组件。数据库包含根据所有最佳几何描述的各组件的几何描述。最佳几何描述可以为 VRML、3DMF 等。另外，在该数据库中，可以存储各组件大量的不同类型的几何描述。因此，可以在某些意义上，如在提供快速获取模型的可视化和获得足够高分辨率的可视化的折衷的意义上，选择合适的几何描述。应该强调的是，仅仅将组件的位置限制为整数坐标 — 而并不限制用于表示组件形状的坐标或其他类型的参数。

Element Block 301、302 也可以包含其他字段，这些字段提供该组件构成材料的引用，装饰或纹理的引用。可以将此类字段表示为“Material Reference（材料引用）”和“Decoration Version（装饰方案）”。至于几何描述，可以在数据库中以合适的格式存储材料和装饰或纹理的描述。

因此，与 VRML 模型结构相比，根据本发明约束模型结构的影响是基于以下事实，在大部分的建模任务中，位于层次最底层上的结构的使用最频繁。因此，通过精确约束最底层上的结构，并使上述结构

的表示恰好足够，则有可能获得非常紧凑的模型表示。当从整体上考虑层次时，Model Block 允许自由变换，Segment Block 允许部分约束的变换，而 Element Block 仅允许完全约束的变换。

在定义了组件和段的可能几何解释，并定义了在其中组织组件、段、模型和通用坐标的层次后，现在的论题是定义用于表示实际模型的对应代码结构的最佳实施方式。

请注意，本发明的基本实施方式可以包含单一段，其组件具有自由通用坐标系（如任意坐标系）中的约束坐标系中的坐标定义的约束位置。

代码结构

代码结构包含两类原型：块和字段。利用块名和围绕其内容的大括号{ }定义块。一个块可以围住许多字段和其他从属块。利用字段名和字段名之后的许多参数定义字段。

块和字段

在下文中，利用其块名后跟括号内的上下文写入块。上下文是可以在其中存在的块；最高层称为文件（File）。术语文件用作例如计算机存储器之类的计算机可读介质、CD-ROM、数字视盘（DVD）、软盘、硬盘、易失存储器（如随机存储器）等上存储的电子文件的引用。

各块具有一个简短描述。为了降低此定义的复杂性，仅写入从属块的块名。

World(File)

就结构而言，重要原型为层次中的块，即 World Block、Model Block、Segment Block 和 Element Block。World Block 引用包含通用坐标描述的文件“File”。

利用以下语法定义通用坐标

World

```
{
    Options{...}
    Model{...}
}
```

在根据本发明的虚拟世界中，语句“Model{...}”中可以存在许多模型。然而，可以在通用坐标中设置按照或不按照本发明描述的其他特征。此类特征可以为背景、音乐、预定事件等。

否则，World Block 用于设置通用坐标中模型（和其他特征）的公共引用。此引用最好采取坐标系的形式。

Model(World)

Model Block 可以包含许多模型或许多 Segment Block。Model Block 中的字段为：Rotate、Translate 和 Scale。

```
Model
{
    Rotate [Real; Real; Real]
    Translate [Real; Real; Real]
    Scale [Real; Real; Real]
    Segment {...}
}
```

请注意，所有的旋转和方向参数均基于众所周知的右手坐标系，即拇指沿某个坐标轴的正值方向前进，而沿其他手指的方向计算正的角度值。

Rotate 字段包含三个参数，这些参数确定模型在 World 中的朝向。最好以众所周知的欧拉角的方式指定这些参数，即首先根据指定围绕直角坐标系中第一坐标轴之角度的三个参数中的第一参数，旋转该模型，第二，根据指定围绕其方向由于该模型环绕第一坐标轴的第一次旋转而改变的第二坐标轴之角度的第二参数，旋转该模型。该过程继续，直至按照需要根据 Rotate 字段中的第三参数旋转了该模型。然而，

可以应用指定对象之方向和/或旋转的其他过程。

在 Model 之上述定义中的 Rotate 字段的语法中，参数的类型为 Real，然而也可以使用其他类型，如 Integer，指定的枚举类型（如包含所有可能值[0, 10, 20, ..., 350]度的枚举类型），或其他类型。

字段 Translate 也包含三个 Real 类型的参数。每个参数指定该模型沿坐标系中三个坐标轴中某一坐标轴的平移。因此，Translate 允许未限制到积木坐标系之模块的平移。

Segment(Model, Segment)

Segment Block 包含由 Element Block 构成的主体部分和可选的从属 Segment Block。Segment 中的字段为 Offset、Orientation、Axis 和 Rotate。

```
Segment
{
    Offset [Integer; Integer; Integer];
    Orientation [2 bit; 2 bit; 2 bit];
    Axis [3 bit];
    Rotate [Real];
    Element {...}
    Segment {...}
}
```

Offset 字段描述当前段相对于另一端（即该层次中当前段上层的段）之原点的原点位置。如果不存在上层段，则省略 Offset 字段。Offset 字段包含三个描述位置的 Integer 类型的参数。通过将此位置约束为整数值，可以显著降低表示全部模型所需要的数据量。

Orientation 字段（在图 3 中省略）描述该段的方向，即该段的朝向是否倒置，是否向左转 90°等。Orientation 字段包含三个 2 比特参数，用于以 90°步长描述围绕该坐标系中三个坐标轴之某一坐标轴或多个坐标轴的方向。

Axis 字段描述可围绕某坐标轴旋转该段的坐标轴的方向。仅使用

一个 3 比特参数来指定坐标系中三个坐标轴中的一个坐标轴，这是由于相对于当前段之原点描述方向的缘故。

Rotate 字段描述当前段围绕 Axis 字段中先前定义的坐标轴的旋转。Rotate 字段包含一个 Real 类型的参数。此参数可用于指定当前段的静态角度，或者修改此参数以模拟处理，如以指定的角速度旋转当前段。

因此，通过利用上述定义的约束，可以显著降低表示全部模型所需要的数据量。能够获得上述益处而并不会使得根据本发明创建模型的用户感到不满，这是由于在积木构造系统或积木虚拟现实中期望约束的缘故。然而，如果在部分建模任务中约束是限制因素，则可以应用例如在 Model Block 中定义的 Translate 字段以克服或解除约束。例如，可以使用利用 Real 类型的非整数数值的位移。

语句“Element {...}”和“Segment {...}”表示，正在考虑的段可以包含根据本发明之模型层次的从属组件和段。

Element(Segment)

Element Block 是根据本发明的几何原型。它通过字段 ShapeReference 引用几何主体。然而，也可以定义其他引用字段：MaterialReference 和 DecorationVersion。其他字段为：Orientation 和 Offset。

```
Element
{
    Offset [Integer; Integer; Integer];
    Orientation [2 bit; 2 bit; 2 bit];
    ShapeReference [Integer];
    MaterialReference [Integer];
    DecorationReference [Integer];
}
```

Offset 字段用于定义段内的组件根据积木坐标网格之模块所处的位置。因此可以使用 Integer 类型的参数。作为选择，可以在字段

ShapeReference、MaterialReference 和 DecorationReference 中使用包含字母数字字符串类型的参数。

利用 Orientation 字段和 3 个 2 比特参数定义组件的方向。因此，将组件的方向约束到该段的积木坐标网格。

为了以有效方式处理特定模型，Element Block 包含组件的几何描述的引用，而不是包含可能包含大量几何数据的完整几何描述。因此，通过利用另一个 ShapeReference 参数互换 Element Block 中的 ShapeReference 参数，就能改变模型的外观。

现在转到根据本发明的特定模型示例。

图 4 表示两组组件。第一组包含组件 401、402 和 403。第二组包含组件 404、405 和 406。每个组件均具有一个与其引用关联的几何描述，以至当请求具有指定引用的组件时，可以检索几何描述。通过利用计算机辅助设计软件设计该组件，可以提供几何描述，从而生成指定格式的数据，其中能够在方便介质上存储并从该介质上检索数据。指定格式可以为例如 VRML 和 3DMF 等。因此，可以检索该数据，以便根据本发明从视觉上解释虚拟模型。

将每个组件设计具有引用位置和方向，以至当在模型中使用该组件时，可以相对于引用位置和方向，描述组件在模型中的位置和方向。作为说明目的，为了定义上述引用位置和方向，在本公开中仅将组件放置在具有坐标轴 X、Y 和 Z 的直角坐标系中。

尽管各组件的外观不同，各组件仍然具有与各段之积木坐标网格内的模块相对应的各自的引用模块。利用虚线/围绕框表示上述的引用模块。

现在转到特定组件。组件 401 为占据 3 个坐标网格模块的 L 型组件。对此组件分配类型标识“T401”。组件 402 为占据一个坐标网格单位的立方体组件。对此组件分配类型标识“T402”。 组件 403 为占据 4 个坐标网格单位的 T 型组件。对此组件分配类型标识“T403”。

相应地，组件 404 为占据 3 个坐标网格模块的 L 型管组件。对此组件分配类型标识“T404”。组件 405 为占据一个坐标网格单位的直

管组件。对此组件分配类型标识“T405”。 组件 406 为占据 4 个坐标网格单位的 T 型管组件。对此组件分配类型标识“T406”。

因此，可以依靠类型标识，即引用，借助根据本发明的虚拟建模语言，存储各组件 401-406 的几何描述，并检索特定组件的几何描述。

最好将各个组内的组件设计为，能够与其各组内的其他组件互连。这可以直观表示为，将组件设计为包含以下表面形式的可识别界面，即该表面与两个相邻坐标网格模块之间的平面一致。考虑管状组件 404、405 和 406，可识别的界面可以为管子末端的形式。例如，美国专利 4,214,403、4,176,493、5,795,210、4,124,949、5,645,463 说明了其他组件。

图 5 表示构建几何对象模型过程中的第一步。将类型 T401 的组件 501，放置在具有坐标轴 X1、Y1、Z1 定义的直角坐标系的段中。坐标轴仅用于说明目的，当需要可视化该模型时，可以显示也可以不显示坐标轴。

在虚拟现实模型的最佳实施方式中，定义最小的可以接受的模型必须包含至少一个段，因此，将组件 501 放置在一个段中。请注意，组件 501 具有相对于坐标轴 X1、Y1、Z1 的方向，该方向与图 4 所示的该组件类型之定义中的引用方向不同。最好以该组件的 90°步长的倍数的形式，描述以上方向。通过沿平行于坐标轴 X1 的坐标轴，将该组件旋转 270°，确定组件 501 的方向，其中根据右手坐标系测量角度（即，拇指沿 x 坐标轴的正向，而沿其他手指的环行方向取正值 — 将在下文中使用此定义）。

用于表示具有单一组件 501 的段的最佳代码为：

```
CODE BEGIN:
    Segment
     {
       Element
        {
```

```
            ShapeReference (T401);
            Orientation (270, 0, 0);
        }
    }
CODE END
```

根据代码语法的定义，定义具有组件的段。另外，依靠 ShapeReference 字段，定义组件的类型为 T401。依靠 Orientation 字段以及参数(270, 0, 0)说明该组件的方向，其中参数(270, 0, 0)表示相对于 X1 坐标轴将该组件旋转 270°。请注意，以度为单位说明此例中的参数，然而，由于 90°约束的缘故，可以将此参数表示为每参数仅有两位。

因此，通过根据上述代码中的给定语句，加载该组件的几何描述，计算该组件的位置和方向，并计算在平面或球面上显示时所需的投影以便显示装置显示，可视化解释程序可以解释以上代码。

与用于显示该组件的分辨率相比，选择的坐标网格的大小比较粗糙。通常，使坐标网格的大小适合于能够仅处理互连，或只是允许在某些意义上调整一组组件中的组件。

尽管依靠整数坐标将组件放置在坐标系中，组件的几何形状的定义或表示仍然可以包含非整数坐标。其重要方面在于，依靠整数表示组件的位置。

图 6 表示构建几何对象模型过程中的第二步。在此步骤中，添加更多组件，并且根据本发明的虚拟模型的效果将更加清楚。现在，几何对象包括 3 个组件 601、602 和 603，其类型分别为 T401、T403 和 T402。利用以下代码给出各组件的位置和方向。

```
CODE BEGIN:
    Segment
     {
```

```
        Element
        {
            ShapeReference (T401);
            Orientation (270, 0, 0);
        }
        Element
        {
            ShapeReference (T403);
            Orientation (0, 0, 270);
            Offset (3, 0, 0);
        }
        Element
        {
            ShapeReference (T402);
            Offset (3, 2, 0);
        }
    }
CODE END
```

已经介绍了定义组件 601 的代码的第一部分。正如将出现的那样，相对于此组件（601）定义该段中的其他组件（602 和 603）。

代码的下一部分定义类型 T403 的组件 602。正如看到的那样，此组件的方向与该组件类型定义中的方向不同；因此，必须添加 Orientation 字段，该字段规定沿 Z1 坐标轴将该组件旋转 270°。另外，通过沿 X1 坐标轴偏移 3 个模块值，放置组件 602。

代码的最后部分定义类型 T402 的组件 603 的位置。通过沿 X1 坐标轴偏移 3 个模块值，然后沿 Y1 坐标轴偏移 2 个模块值，放置此组件。

由于可以仅根据坐标网格模块来放置段内的组件，所以非常有效

地表示用于定义组件之位置和方向的参数，如利用整数值。这样，如果需要的话，可以向该段添加其他组件。

如果处理，则容易确定由于积木坐标网格引起的约束的缘故，不能相对于对方处理段内的组件。

图 7 表示根据本发明构建几何对象模型过程中的第三步。在此步骤中，还可以将更多组件添加到该模型中，并且根据本发明的虚拟模型的效果将更加清楚。另外，在此步骤中，无需非要将组件添加到段内指定的位置和方向约束上。这需要模型的扩展描述，然而，由于仅在需要时（例如，在描述旋转时）才使用此扩展描述，所以仍能保持模型的紧凑性和方便性。

现在，该几何对象包含 5 个组件 701、702、703、704 和 705，其类型分别为 T401、T403、T402、T403 和 T401。利用以下代码给出各组件的位置和方向。

```
CODE BEGIN:
    Segment
     {
      Element
      {
         ShapeReference (T401);
         Orientation (270, 0, 0);
      }
      Element
      {
         ShapeReference (T403);
         Orientation (0, 0, 270);
         Offset (3, 0, 0);
      }
      Element
```

```
        {
            ShapeReference (T402);
            Offset (3, 2, 0);
        }

        Segment
         {
            Offset (0, 0, 2);
            Orientation (90, 0, 0);
            Rotate (225);
            Axis (0, 0, 1);
            Element
            {
                ShapeReference (T403);
                Offset (0, -1, 0);
            }
            Element
                {
                ShapeReference (T401);
                Offset (-3, -1, 0);
                Orientation (270, 0, 0);
            }
        }
        }
CODE END
```

代码的第一部分定义在图 6 中介绍的组件 701、702 和 703。代码的下一部分定义另一段，其中相对于与 Z 坐标轴（即垂直坐标轴）平行的坐标轴将该段旋转 45°，并且该段具有组件 704 和 705。

可以以不同方式生成定义组件 704 和 705 的代码部分，然而，以下包含步骤 1-4 的循序渐进过程是最容易理解的。

步骤 1：在此代码生成步骤中，确定（如响应用户命令）是否应将一个新段添加到此模型中。鉴于此，将组件 704 作为第一组件添加到新段中，其中新段具有与坐标网格一致的积木坐标网格（未示出），或类型 T403 之组件（参见图 4）的定义中定义的坐标系。因此，无需指定 Offset 或 Orientation 字段。接着，将组件 705 添加到新段中。然而，将该组件沿 X1 坐标轴偏移整数值-3，并相对于类型 T401 之组件的定义中的组件方向，沿 X 坐标轴将其方向旋转 270°。利用以下代码中的 Offset 和 Orientation 字段进行定义：

```
CODE BEGIN:
        Segment
         {
           Element
           {
              ShapeReference (T403);
           }
           Element
              {
              ShapeReference (T401);
              Offset (-3, 0, 0);
              Orientation (270, 0, 0);
           }
        }
CODE END
```

步骤 2：在此代码生成步骤中，确定新段的原点，以确定将新段连接到旧段的位置，即 T 型组件 704 的底部。因此，重新计算组件 704

和 705 的位置，以与该段的新坐标原点一致。

```
CODE BEGIN:
        Segment
         {
           Element
           {
              ShapeReference (T403);
              Offset (0, -1, 0);     //添加
           }
           Element
              {
              ShapeReference (T401);
              Offset (-3, -1, 0);        //修改
              Orientation (270, 0, 0);
           }
        }
CODE END
```

请注意，以上处理包括，为组件 704 添加 Offset 字段，并修改组件 705 的 Offset 字段，以反映新坐标原点。

例如，如图 2b 所示，如果在新段中的不同位置，将新段连接到旧段，则必须重新计算。但是，由于 Offset 参数为 Integer 类型，所以与使用 Real 类型的参数相比，可以非常快速地执行重新计算。由于采用整数表示，所以尽管在实际模型中将执行许多加、减运算，并不会引入舍入误差。

步骤 3：在此代码生成步骤中，确定新段将与旧段互连的位置。通过指定相对于旧段之坐标原点平移新段之坐标原点的方式，说明上述问题。其实现方式是，将下述 Offset 字段插入到步骤 2 给出的第二

Segment 块中。

Offset (0, 0, 2);

因此，沿 Z1 坐标轴，将新段的坐标原点平移 2 个整数值。

步骤 4：在定义了新段的位置以及新段内的组件后，可以指定新段相对于旧段的方向。其实现方式是，将下述 Rotate 字段插入到步骤 2 给出的代码中。

Rotate (90, 225, 0);

因此，首先围绕 X 坐标轴将新段旋转 90°，接着，在已经旋转 90° 之后，围绕 Y 坐标轴将新段旋转 225°。

如上所述，如果处理，则容易确定由于积木坐标网格引起的约束的缘故，不能相对于对方处理段内的组件。但是，也容易确定由于段的隐式定义的缘故，允许相对于对方旋转该段。因此，无需定义附加的元描述（如，移动或旋转）。如果需要相对于对方平移该段，而不受各自积木坐标网格的约束，则可以添加定义平移操作的简单 Translate 字段。

图 8 表示根据本发明建模的第四简单对象。可以利用图 7 说明的同一模型说明本对象，只需利用图 4 定义的第二组组件中各组件的类型，互换利用 ShapeReference 字段参数定义的各组件的类型。该模型包含组件 801、802、803、804 和 805，这些组件分别对应于图 7 中的组件 701、702、703、704 和 705。

因此，可以在该模型中使用另一组组件而不会损失有效描述。在最佳实施方式中，该组组件适合于一类建模范例，从而该组件表示众所周知的现实世界中的对象，其中可以找到适合给定建模任务的现实世界中的对象。

另外，根据上述公开的最佳实施方式，并根据本发明的目的，对熟练技术人员而言，显然可以通过修改各语句的参数，依靠几何描述隐式给出元描述，即几何对象的可能处理的描述。例如，考虑两段的互连，可以利用 Axis 字段和 Rotate 字段给出此类互连。通过修改旋转角度，可改变几何描述，同时处理（如旋转）发生。在此类情况中，

在几何描述中定义旋转坐标轴，这样，将元描述嵌入到几何描述中。当遇到段内的组件时，由于几何描述中的所有参数均不考虑旋转的元描述，所以显然不能旋转这些组件。然而，在实际实施方式中，如果需要的话，可以向用户提供控制，以应用新的元描述。事实上，因为该模型结构中仅存在有限的描述集合，并将其参数约束到有限的值集合，所以很容易应用新的元描述。

在根据本发明定义了模型的结构后，将公开模型的修改和处理，以及用于执行上述任务的计算机系统。

图 9 表示用于根据本发明设计、存储、处理并共享几何构造的客户计算机系统。可以将一般由 901 指示的客户计算机系统用作独立系统，或客户机/服务器系统中的客户机。客户机包含存储器 902，其中以诸如硬盘和随机存取存储器（RAM）之类的易失和非易失存储器的方式，部分实现存储器。存储器包含过程或程序代码解释程序 907，代码生成程序 908，UI 事件处理程序 909，和可由中央处理器 903 执行的建模应用程序 910。另外，存储器也包含模型数据 911。

代码解释程序 907 适合于根据本发明读取并解释定义模型的代码。在最佳实施方式中，代码解释程序适合于根据本发明读取模型，然后将此模型转换为众所周知的图形格式，以便在计算机显示器上显示。当熟练技术人员了解先前定义的用于表示对象之模型的代码结构时，通过应用在图形计算领域内熟知的图形原则，可以实现上述转换。以下给出代码解释程序 907 实现的部分函数的列表:

函数 1: 当读取 World Block 时，代码解释程序（CI）将设置后继语句将引用的坐标系。

函数 2: 当读取 Model Block 时，CI 准备将包含 Segment 和 Element 的模型，放在函数 1 定义的坐标系内。这是根据 Rotate 和 Translate 字段实现的。

函数 3: 当读取 Segment Block 时，CI 根据函数 2 中指定的位置和方向并根据规定该段的方向的 Orientation 字段，准备放置包含 Element 的段。此外，读取 Offset、Axis 和 Rotate 字段，以便确定从

属段的位置。

函数 4: 当读取 Element Block 时，CI 通过访问以模型数据 911 的方式存储的数据库，读取 ShapeReference 字段中组件类型引用指定的组件的几何描述。

提供顺序和/或交互调用函数 1、2、3 和 4 所产生的结果，作为投影到平面或球面上的数据，或者作为须经投影的数据。

UI 事件处理程序 909 适合于将用户与用户界面的交互，转换为代码生成程序 908 可识别的正确的用户命令。可能的、可识别的命令集包括：从组件库获取组件，将一个组件连接到另一个组件，断开组件的连接，废除组件，处理段，如开始旋转等。随同各命令一起，关联如上定义的各字段的一组不同参数。

代码生成程序 908 适合于根据本发明按上述方式，响应用户命令，修改描述实际模型的代码。以下给出代码生成程序 908 实现的部分函数的列表（假设已经指定一段）：

函数 5: 当从组件库中选择组件时，利用 ShapeReference 字段指定其类型。

函数 6: 当将该组件连接到段内的某个组件时，标识该段，确定该段内所连接组件的位置和方向，并利用插入到模型表示中的 Offset 和 Orientation 字段表示其位置和方向。

函数 7: 当断开组件连接时，删除表示该组件的位置和方向的代码。

作为并发或后继任务，可执行代码解释程序，以表示代码生成程序的结果。

建模应用程序 910 适合于控制存储器、文件、用户界面等。

用户 905 能够利用用户界面 906 与客户计算机系统 901 进行交互。

为了加载模型、几何描述或其他数据，客户计算机系统包含一个输入/输出部件（I/O）904。可以将输入/输出部件用作不同类型存储介质和不同类型的计算机网络（如因特网）的接口。另外，输入/输出部件（I/O）904 可用于与其他用户以交互方式交换模型。

利用数据总线 912，实现存储器 902、中央处理器（CPU）903、用户界面（UI）906 和输入/输出部件 904 之间的数据交换。

图 10 表示用于根据本发明处理几何构造的服务器计算机系统。一般由 1001 指示的服务器服务多个客户计算机系统。服务器包含存储器 1002，其中以诸如硬盘和随机存取存储器（RAM）之类的易失和非易失存储器的方式，部分实现存储器。存储器包含过程或程序代码解释程序 1005，代码生成程序 1006，和可由中央处理器 1003 执行的会话处理程序 1007。另外，存储器也包含例如几何描述、模型描述等形式的模型数据 1008。

为了加载模型、几何描述、或其他数据，抑或为了向客户计算机系统分发数据，服务器计算机系统包含一个输入/输出部件（I/O）1004。可以将输入/输出部件用作不同类型存储介质和不同类型的计算机网络（如因特网）的接口。

利用数据总线 1009，实现存储器 1002、中央处理器（CPU）1003、和输入/输出部件 1004 之间的数据交换。

现在转到参照图 9 和图 10 之描述的函数/计算机程序的详细说明。

图 11 表示需要处理段和模型时并行构造段和模型的方法的流程图。此流程图也描述为根据本发明处理模型而在客户机和/或服务器计算机上所执行的操作过程。

在步骤 1101 中，接收利用用户控制输入的用户命令，其中用户控制为用户界面 1102 的一部分。用户控制可具有不同类型，例如包括与用户显示 1121 进行交互的计算机键盘、游戏杆或鼠标，从而向程序提供信号，以表示用户与用户界面的交互。例如，通过利用计算机鼠标，使用熟知的拖放操作，用户可以选择在用户界面 1121 上显示的组件，然后将其移动到所需位置。因此，生成表示以上组件移动的用户命令。此类用户界面操作被认为是事件驱动的程序设计。

在流程图的块 1113 中，接收用户命令，用户命令用于根据本发明创建、更新或处理模型数据 1112。通过输入/输出设备 1117，将作为块 1113 中的处理结果的模型数据 1112，传送到通信网络。因此，可

与其他用户交换模型，共享模型，从服务器加载模型，在服务器上保存模型等。通常，依靠转换程序 1118，合并模型数据 1112 与表示所存储组件之外形的数据。该转换程序使用先前定义的 ShapeReference，以从数据库 1119 中获得此外形的几何表示。另外，在转换程序 1118 中，组合该几何表示以及先前定义的描述特定模型之形状的 Block 和字段，然后转换为可由 3D 虚拟现实引擎 1120 解释的格式。3D 虚拟现实引擎 1120 提供表示该模型的信息，从而可以在用户显示 1121 上可视化该模型。

现在转到该流程图的块 1113 的详细说明。

当接收到用户命令时，确定该用户命令是否涉及组件操作、段操作或模型操作。如果将组件添加到该模型中，从该模型中删除组件，或者将组件从模型内的第一位置移动到第二位置，则在步骤 1116 中检测到组件操作。如果移动、删除或处理该模型内的段，或作为选择，如果将一个新段添加到该模型中，则在步骤 1115 中检测到段操作。如果处理模型，如不受坐标网格的约束自由移动模型，或者如果在不受坐标网格之约束的情况下自由放置段，则在步骤 1114 中检测到模型操作。

根据检测到的操作，创建、更新或处理模型数据。

如果用户从例如包含图 4 所示组件的组件库中选择一个组件，则进入步骤 1110“获取新组件”，并将所选组件标识为例如类型 T401。随后，在步骤 1111 中，将所选组件连接到通用坐标中已经存在的另一组件，或作为选择，单独放置此组件，从而作为一段进行处理。当连接所选组件时，将表示此组件、其位置和方向的代码添加到已有代码中，或生成新代码。

如果通用坐标中存在另一组件，用户可以决定选择此组件，然后移动此组件或从其当前位置断开此组件的连接。如果检测到此类事件，则删除表示该组件的现有代码部分，然后临时存储。

可以废除所选组件，或将所选组件连接/移动到其他位置。如果检测到废除组件的事件，则该组件将从用户界面上消失，并在步骤 1109

中删除临时代码。作为选择，如果检测到将该组件连接到另一位置的事件，则生成表示该组件、其新位置和方向的代码，然后插入到现有代码中。

为了加快生成或修改现有模型过程的速度，可以作为单一对象处理包含其他段和组件的完整段。

如果通用坐标中存在一个段，则用户可以决定选择此段，然后依靠作为移动或断开连接对象的所有段以及该段内所有组件的公用操作，移动该段或从其当前位置断开该段的连接。如果检测到此类事件，则删除表示以上整段的现有代码部分，然后临时存储。

如果检测到废除整段的事件，则以上整段将从用户界面上消失，并在步骤 1106 中删除临时代码。作为选择，如果检测到将该组件连接到另一位置的事件，则生成表示该段、其新位置和方向的代码，然后插入到现有代码中。

此外，可以处理段并且可以移动模型。如果通用坐标中存在段，则可以处理它们，如不改变需要处理的该段内的组件或其他段，但是可以将该段旋转指定的角度，或者规定以指定的角速度旋转。通过检测来自用户界面的指定事件，并在步骤 1104 中修改需要处理的段的代码，例如通过修改 Segment Block 的 Rotate 字段中的参数，可以实现上述处理。

如果通用坐标中存在模型，可以类似于处理段的方式移动模型，其中在步骤 1103 中修改 Model Block 的 Rotate 和 Translate 字段中的参数。

当发生预定事件时，该过程可以在步骤 1112 中终止或停止，包括存储创建的段和模型。

图 12 表示访问并共享可由多台客户机访问的公用坐标的方法的流程图。在本发明的最佳实施方式中，服务器保持通用坐标的模型。可以将该模型分发到允许其处理或改变通用坐标中的段和模型，或可以向通用坐标添加新模型和段的多台客户机。可以将各台客户机进行的修改，传送到共享同一通用坐标的其他客户机。因此，各台客户机将

会体验到，如果其他客户机改变通用坐标的模型，则其通用坐标的本地副本将改变。如果必须支持大量用户并且必须避免过多的更新时间，则必须快速有效地更新通用坐标的客户机的本地副本。本发明能够实现上述方式的更新，并且这也正是提供可由多台客户机访问的公用通用坐标的关键，其中由于不同客户机与通用坐标交互的缘故，通用坐标不断改变。

在流程图 Load World（加载通用坐标）的第一步骤 1201 中，执行客户机和服务器之间的登录会话，包括识别客户机名称，正确的数据传输协议等。在下一步骤 1202 中，将通用坐标的服务器描述传送到客户机。在上述数据传送之前、之中或之后，客户机和服务器交换信息，以确定该客户机是否了解该模型中的所有构件，如模型、段、组件等。在步骤 1203 中执行上述验证。如果该通用坐标中有任何未知模型，则在步骤 1204 中传送相关模型的正确描述。如果没有未知模型，则认为该客户机具有通用坐标的副本，因此，允许处理模型和段，或者在通用坐标中进行更改。

在下文中，说明根据本发明向客户机提供与模型的层次结构相对应的其他信息的方法。假设模型、段、组件以及模型的其他组件具有唯一标识号。因此，服务器可以请求客户机特定模型是否出现，如果特定模型出现，则无需传送其他数据，否则，服务器可以请求客户机特定模型的特定段是否出现。这样，只需将未知的模型组件传送到客户机。

图 13 表示用于获取未知模型、段和组件的方法的流程图。然而，由于通过继续调查模型层次的各层，该流程图普遍适用于获取模型、段和组件，所以仅说明获取未知模型的方法。

在流程图 Get Model（获取模型）的第一步骤 1301 中，进入获取模型的方法。在步骤 1302 中，以任意形式验证该模型是否在客户机可用，如果可用的话，则在步骤 1305 中，将该模型本地传送到运行时数据库。运行时数据库负责保存以下构件，即客户机希望与其交互的通用坐标的一部分。作为选择，如果不能从本地获得该模型，但可远程

（即从服务器）获得，则将该模型从服务器传送到位于客户机的运行时数据库。在步骤 1303 进行此判定，并且在步骤 1305 传送数据。如果在步骤 1303 中判定也不能从远程获得该模型，则必须在步骤 1304 中处理错误。

如果需要从服务器传送到客户机的模型包含该客户机不了解的段，则像未知模型一样，互换这些段。在步骤 1306 中，验证交换模型中是否有未知段。如果有未知段，则在步骤 1308 中交换它们，否则在步骤 1307 中，允许继续修改通用坐标模型。

可以以硬件、软件或其组合的方式，实现图 9、10、11、12 和 13 所示的最佳实施方式。

可以利用各种数据存储器或数据传输介质分发该软件。该介质包括软盘、CD-ROM、Mini Disc、光盘或网络。例如，网络可以为因特网。经由网络，通过下载计算机程序，也可以分发本发明的软件实施方式。

另外，也可以利用任意一种上述数据存储器或数据传输介质，分发或交换组件、部分模型或整个模型的几何描述。

请注意，尽管利用易于读取的字母和数字规定为各字段定义的参数的表示，但根据本发明的模型表示决不限于此类表示。事实上，通常使用参数的压缩表示或编码表示。可以利用数据压缩领域中的熟知方法，实现此类表示。

此外，请注意，可以将代码结构中的语句转换为记号表示中的位。因此，可以紧凑数据结构或数据包的方式，在计算机之间传送记号表示。

在以下引用的参考文献中，有适合于利用几何描述（如 VRML、3DMF 等）进行描述并用作根据本发明的模型中的组件的公开组件。

转让给 Interlego AG 的美国专利 5,645,463，公开了一种玩具建筑组件，包括一个基板和一个枢轴安装在该基板上的一个圆孔中的圆盘。基板和圆盘包含按积木坐标网格排列的连接装置。在最佳实施方式中，根据本发明模拟建筑组件，其方法是，将基板作为第一段或第一段的

一部分，将圆盘作为第二段或第二段的一部分。根据本发明的上述说明，利用 Axis 字段和 Rotate 字段，可以模拟圆盘在基板中的旋转。在基板中心垂直于基板平面的方向上，模拟旋转轴。

转让给 Interlego AG 的美国专利 4,214,403，公开了一种玩具铰链组件，包括一对盒状建筑积木，其中利用两块积木中第一块积木一角的管状衬套以及两块积木中第二块积木中互补枢轴轴销，以铰链方式，互连两块积木。在最佳实施方式中，根据本发明模拟建筑组件，其方法是，将第一块积木作为第一段，将第二块积木作为第二段。利用 Axis 字段和 Rotate 字段，可以模拟第一块积木相对于第二块积木的旋转。在枢轴轴销中央沿枢轴轴销的方向上模拟旋转轴。在 4,214,403 的说明中，以刚性互连方式，将两块积木连接到其他积木。可以将其他积木分别模拟为第一段或第二段内的组件。

转让给 Interlego AG 的美国专利 5,645,463，公开了另一种玩具建筑组件，包括许多类型的连接装置，如用于支撑轴的圆孔，连接双头螺旋和组合件，以及用于将组件卡入到互连装置的固定装置。根据本发明可以模拟以上玩具建筑组件，包括适合于与其互连的组件的模型。利用上述连接装置，可以将其他组件连接到所公开的组件。当根据本发明模拟此类组件时，可以将该组件指定为，支持在与连接装置的位置一致的位置与其他段进行互连的段。通过添加如上所述的 Asix 字段和 Rotate，可以实现上述处理。

转让给 Magic Mold 公司的美国专利 5,061,219，公开了一种建筑物玩具建筑组件，包括许多类型的连接组件，如用于支撑轴的圆孔，具有一个或多个固定座孔的像轮轴一样的连接器，以及具有像撑杆一样外型的建筑组件，其中将撑杆的尾部作成可插入到固定座孔中的形状。可以以相对于第一坐标轴有 4 个可能方向、相对于垂直于第一坐标轴的第二坐标轴有 4 个可能方向、相对于垂直于第一和第二坐标轴的第三坐标轴有 8 个可能方向的组件的方式，模拟该建筑物玩具建筑组件。可以使用像轮轴一样的连接器的中央圆孔作为描述该玩具组件之位置的参考点。

另外，包括美国专利 5,423,707、5,137,486、5,199,919、5,421,762 和 5,238,438 中公开的建筑物玩具建筑组件。

显然可以使用本发明来模拟除上述所述建筑系统之外的其他积木建筑系统。

尽管参照直角坐标系说明了本发明，但也可以使用其他坐标系来说明本发明使用的定义。

本发明并不限于所示代码，选择此代码的目的是作为最佳实施方式。因此，本发明并不限于特定块和字段语句（如 Segment Block，Translate 字段等）。在本公开中，以易于读取、易于理解其表示的方式，使用其参数表示（如偏移值 4，90°的旋转角等）。然而，对熟练技术人员而言，将此表示转换为足够方便在计算机可读介质上的数据结构中进行存储的表示，是众所周知的。

采用的数据结构可以为块和字段语句的直接表示，包括相关参数以及标识块的范围的括号。因此，可以以直接 ASCII 表示的方式，存储该数据结构。作为选择，可以以编码列表项目结构的方式选取数据结构，其中每个项目表示利用二进制编码标识的一个块和一个字段，并且例如以二进制编码格式存储与所有字段关联的参数。然而，也可以选择用于根据本发明存储必需信息的其他合适数据结构。由于括号定义块的范围，所以通过采用其数据结构为块和字段定义以及该示例中所用代码的直接表示，可以将块和字段的层次自动嵌入到该数据结构中。然而，可以利用其他方式指定以上层次，例如，对于各组件，标识该组件属于哪个上层段、模型和通用坐标，同样，对于段，标识该段属于哪个模型和通用坐标。作为选择，可以通过标识上层块来指定层次。

在本说明书中，应将“几何对象”理解为需要根据本发明建模的现实世界或虚拟世界中的对象。

在本说明书中，应非常广泛地解释术语“处理”，包括移动和/或旋转模型的几何解释部分的操作。

在该说明中，术语“元描述”是处理的实际描述。

不要将普通术语“建模（模拟）”或“根据本发明的模型”与术语“**Model Block**”相混淆。

本发明的基本思想是，模型是严格约束的 — 初步看来 — 似乎限制了建模概念的能力。然而，事实上，利用上述思想能够选择非常紧凑的模型表示，这又会导致非常灵活的模型结构，因为其表示非常紧凑并且易于存取，因此，允许根据本发明进行快速、有效的修改、交互或处理。另外，可以经由传输网络，非常快地传输相当复杂的对象的模型。

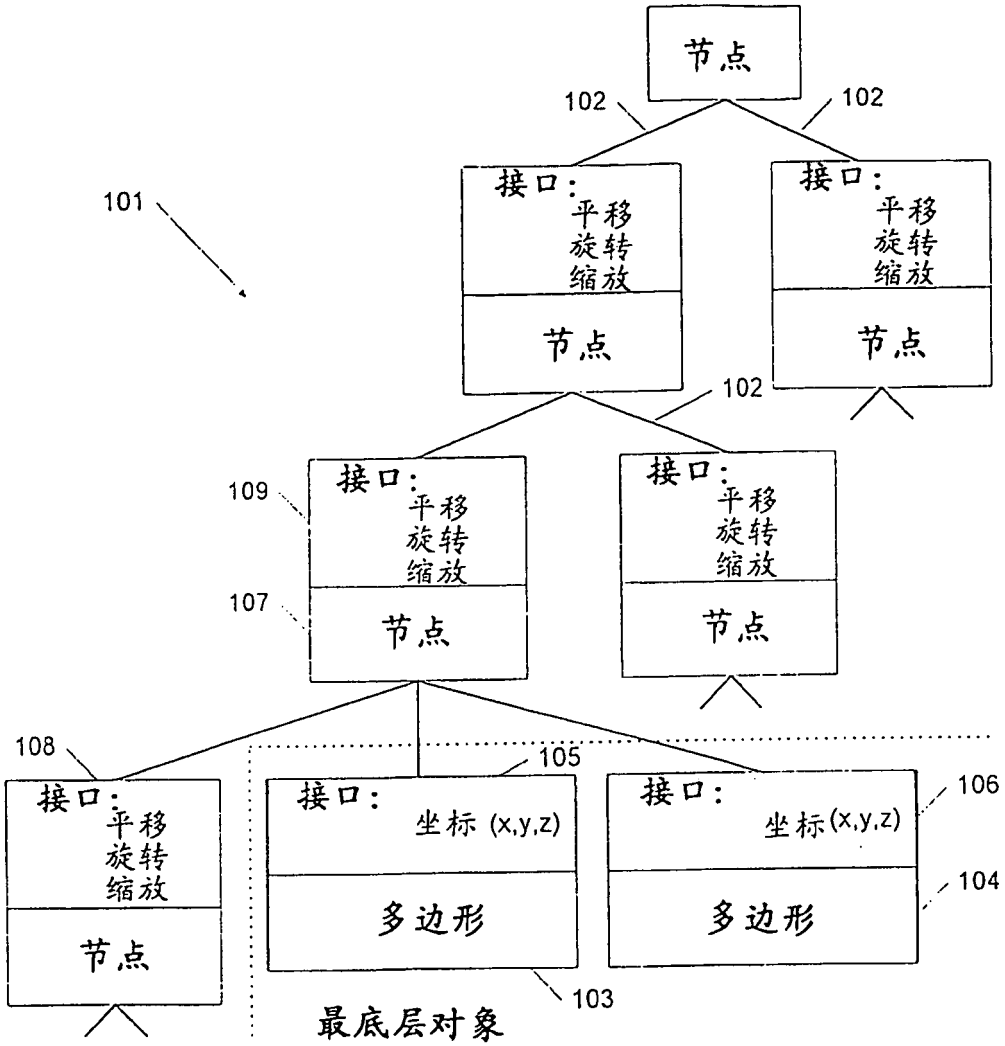

节点
102
102
101
接口：
平移
旋转
缩放
节点
接口：
平移
旋转
缩放
节点
102
109
接口：
平移
旋转
缩放
107
节点
接口：
平移
旋转
缩放
节点
108
105
106
接口：
平移
旋转
缩放
节点
接口：
坐标 (x,y,z)
多边形
接口：
坐标 (x,y,z)
104
多边形
103
最底层对象

图 1

现有技术

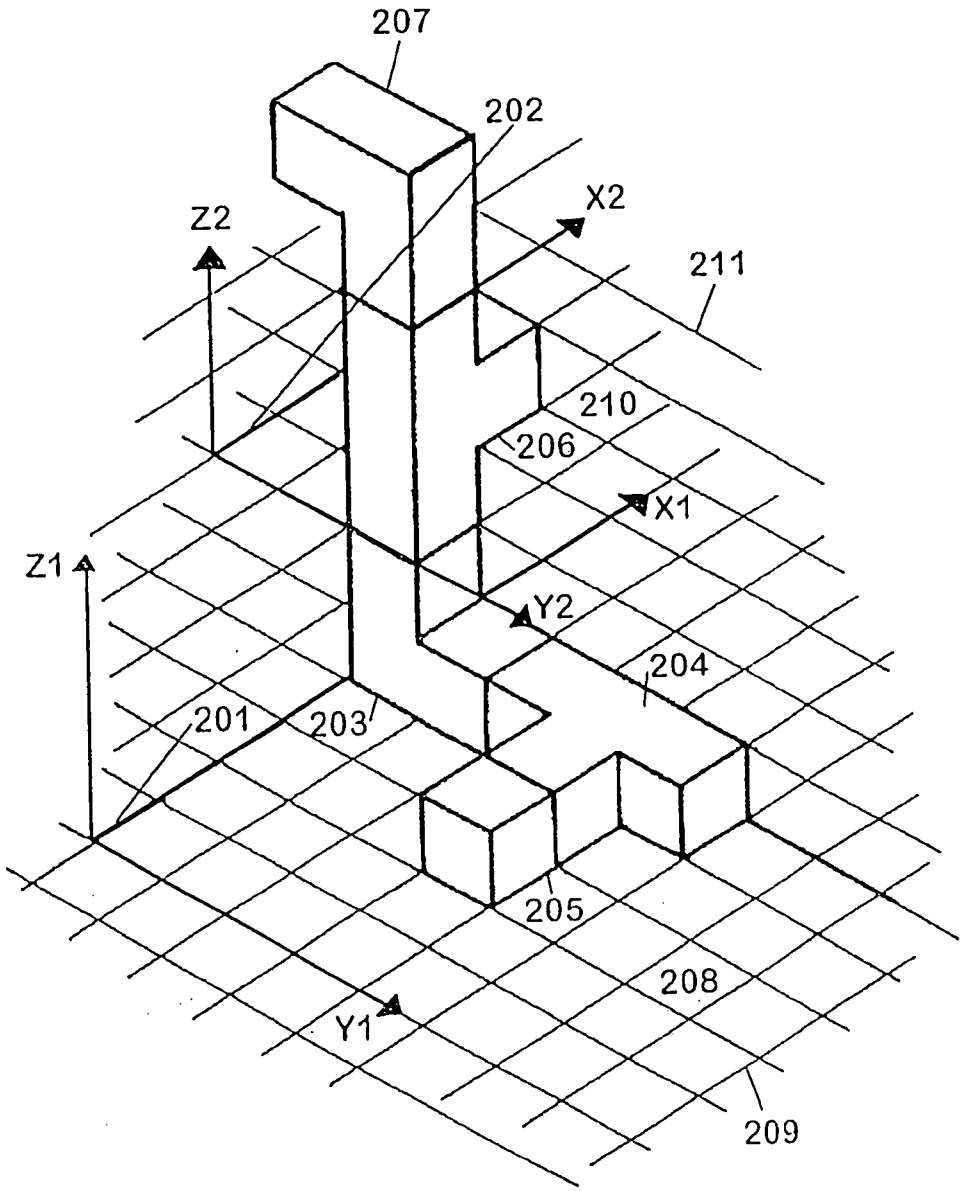


图 2a

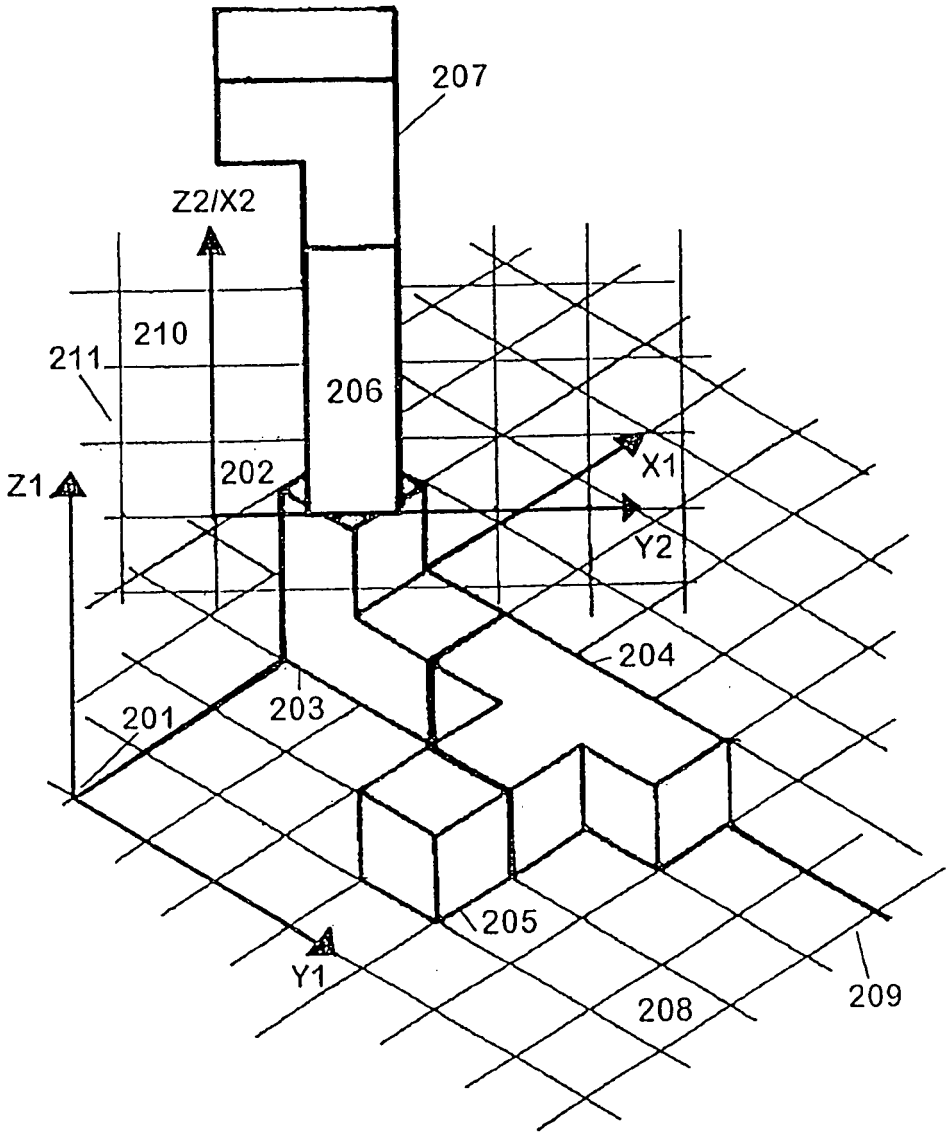


图 2b

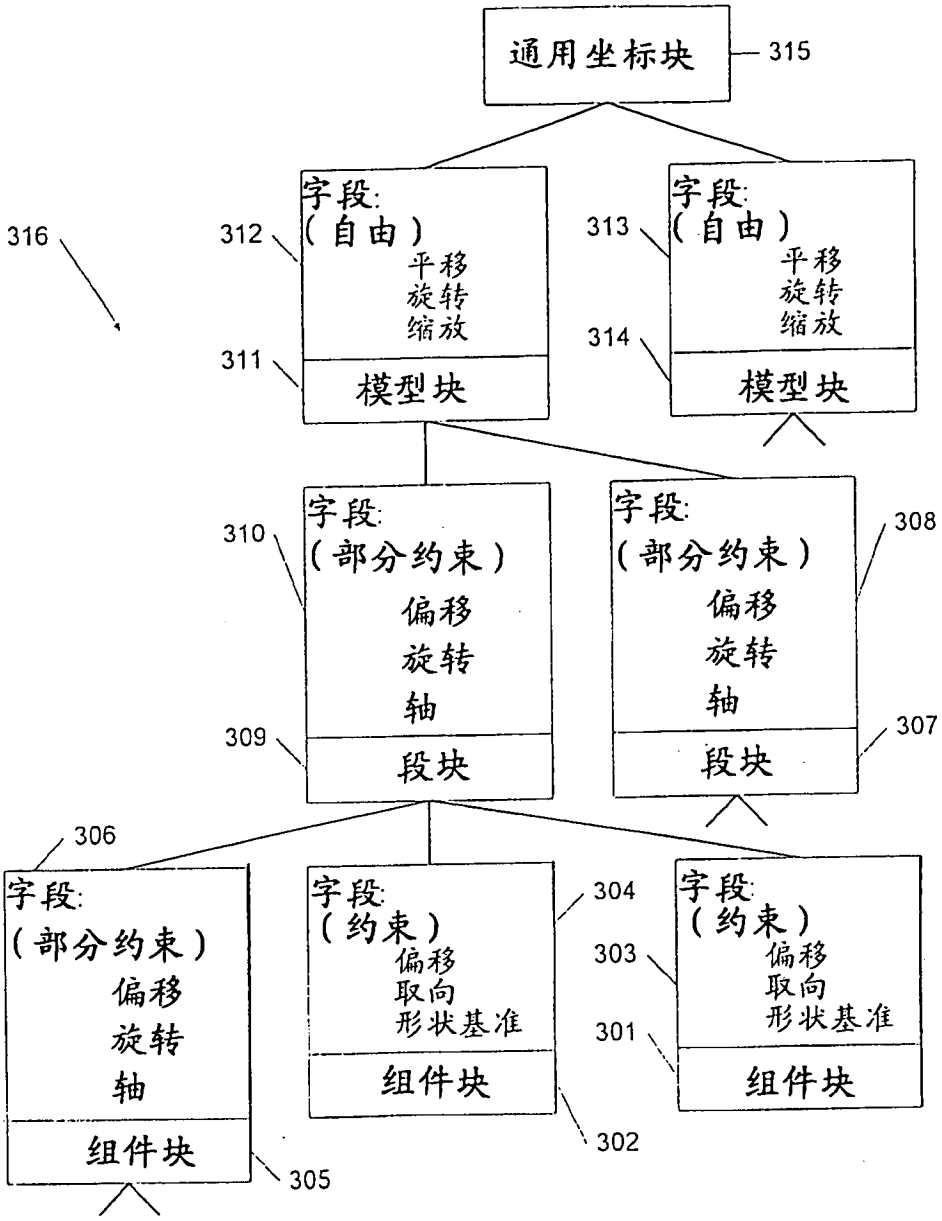

通用坐标块
315
316
312
字段:
(自由)
平移
旋转
缩放
311
模型块
313
字段:
(自由)
平移
旋转
缩放
314
模型块
310
字段:
(部分约束)
偏移
旋转
轴
309
段块
308
字段:
(部分约束)
偏移
旋转
轴
307
段块
306
字段:
(部分约束)
偏移
旋转
轴
组件块
305
304
字段:
(约束)
偏移
取向
形状基准
组件块
302
303
字段:
(约束)
偏移
取向
形状基准
301
组件块

图 3

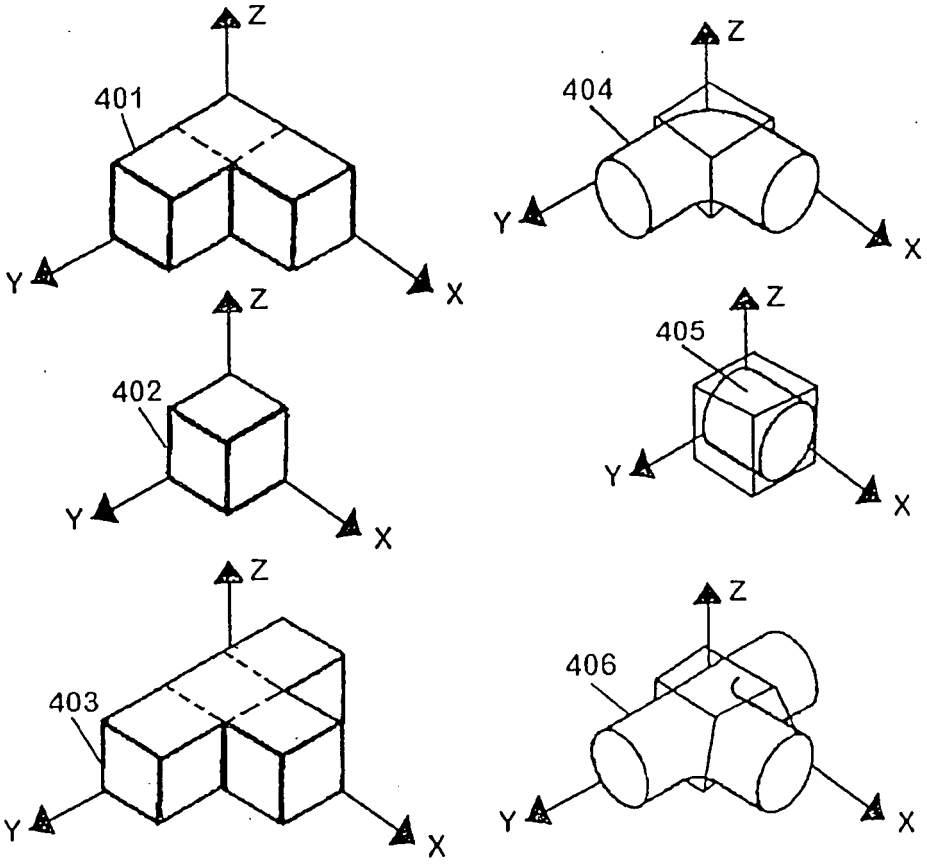


图 4

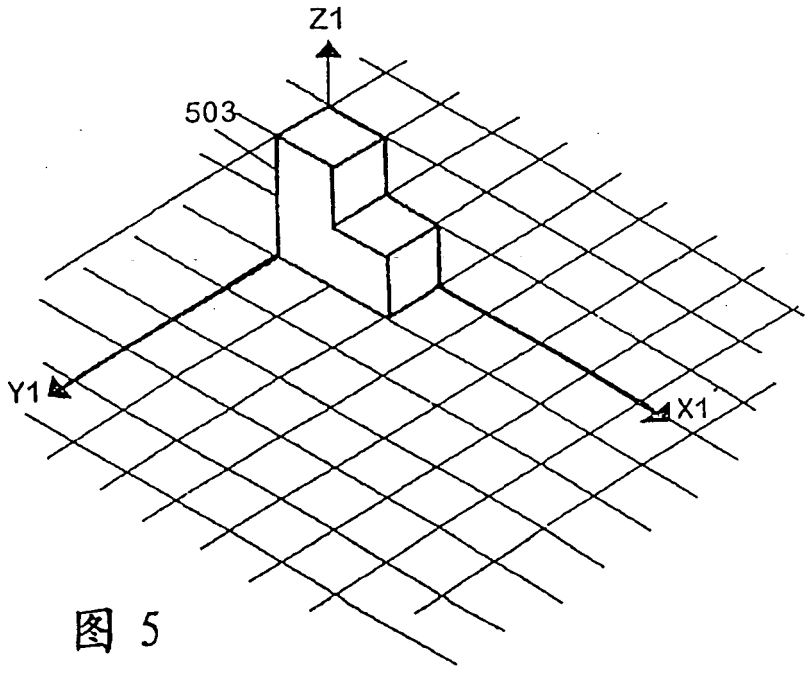


图 5

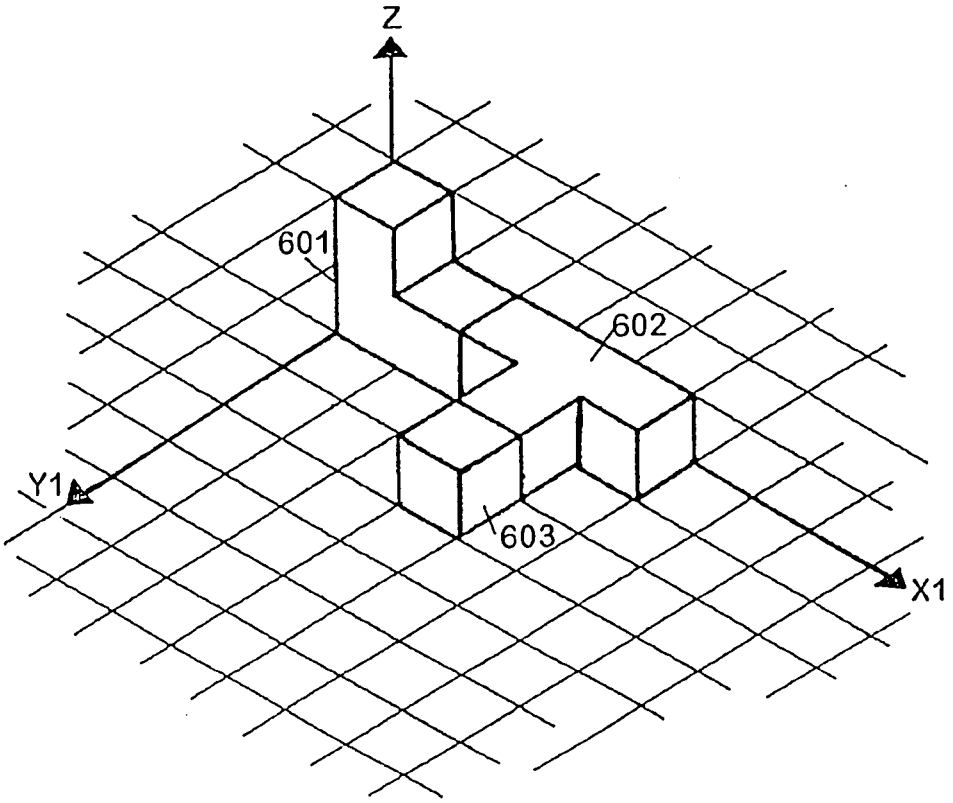


图 6

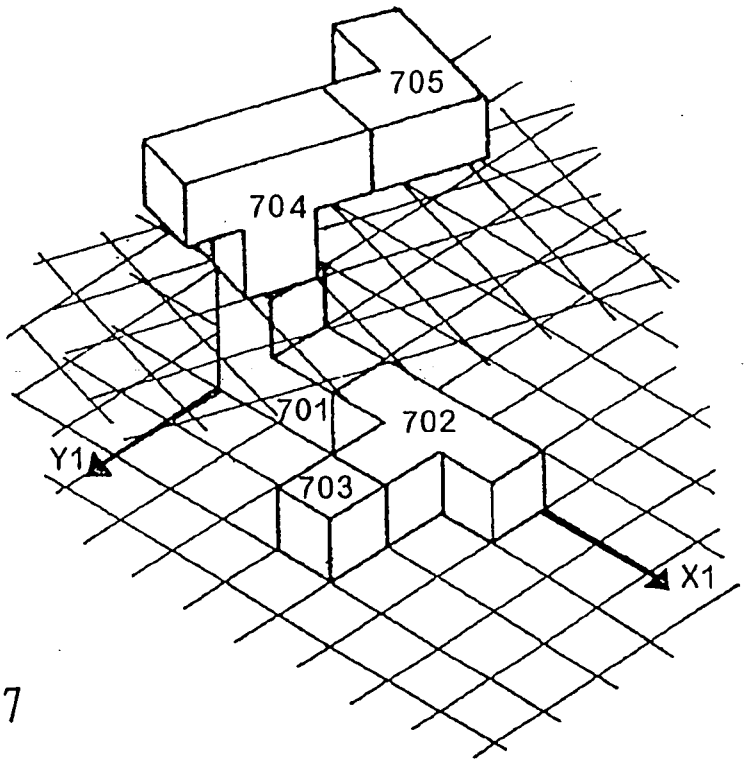


图 7

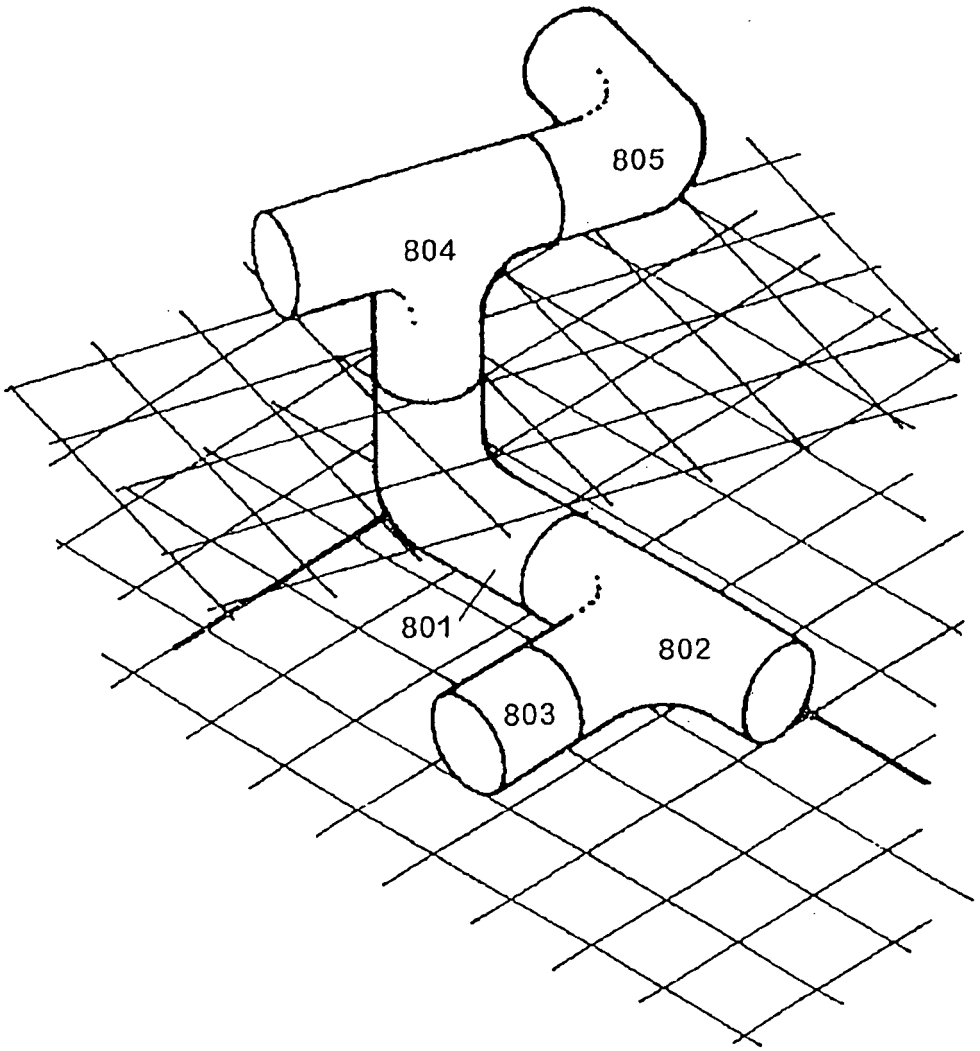


图 8

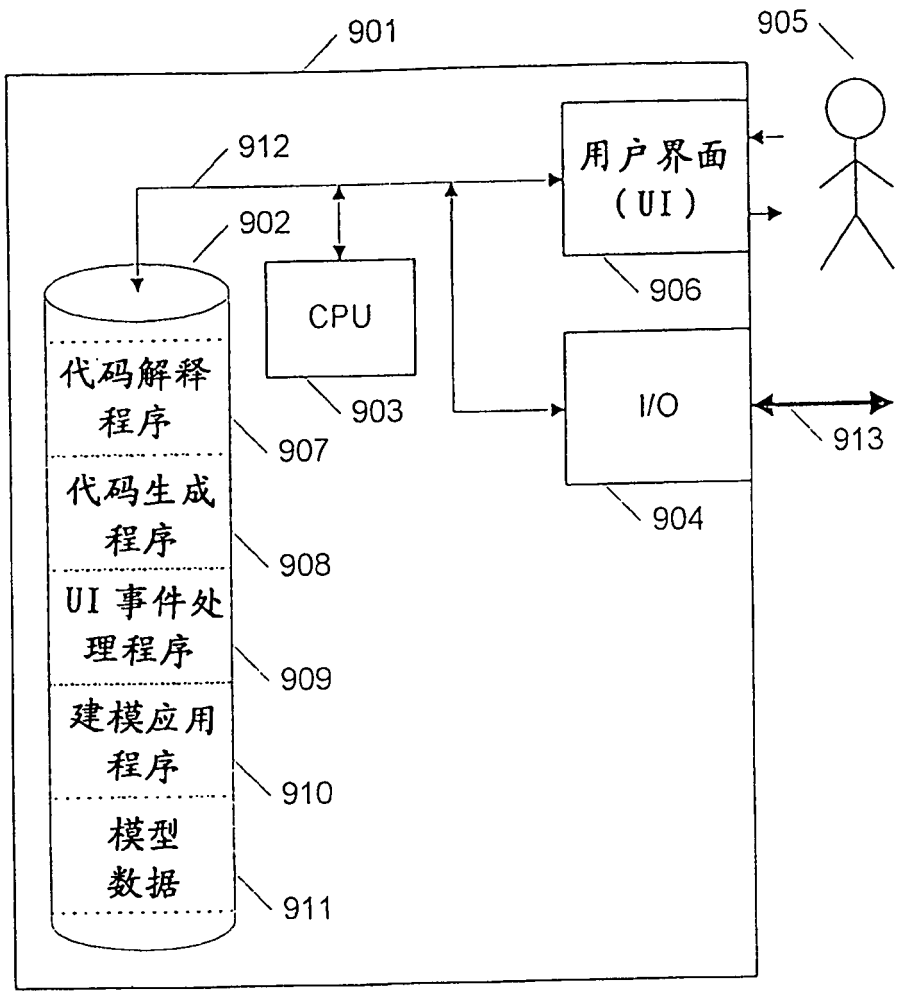

901
905
912
902
用户界面
(UI)
906
CPU
903
I/O
913
904
代码解释
程序
907
代码生成
程序
908
UI 事件处
理程序
909
建模应用
程序
910
模型
数据
911

图 9

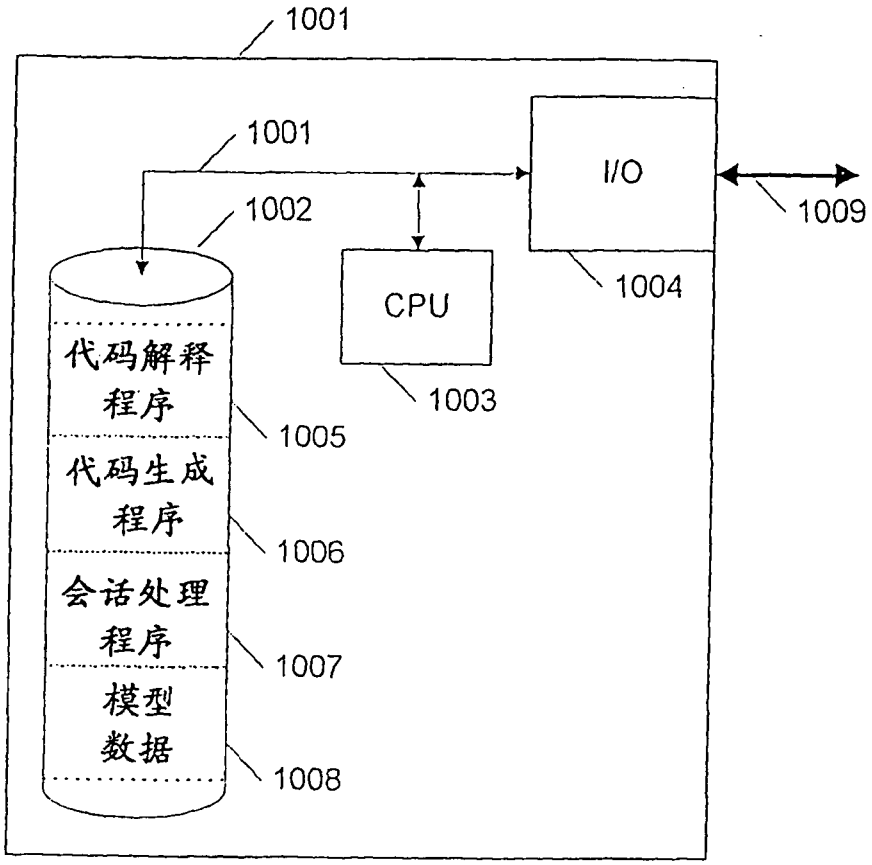

1001
1001
1002
I/O
1009
1004
CPU
1003
代码解释程序
1005
代码生成程序
1006
会话处理程序
1007
模型数据
1008

图 10

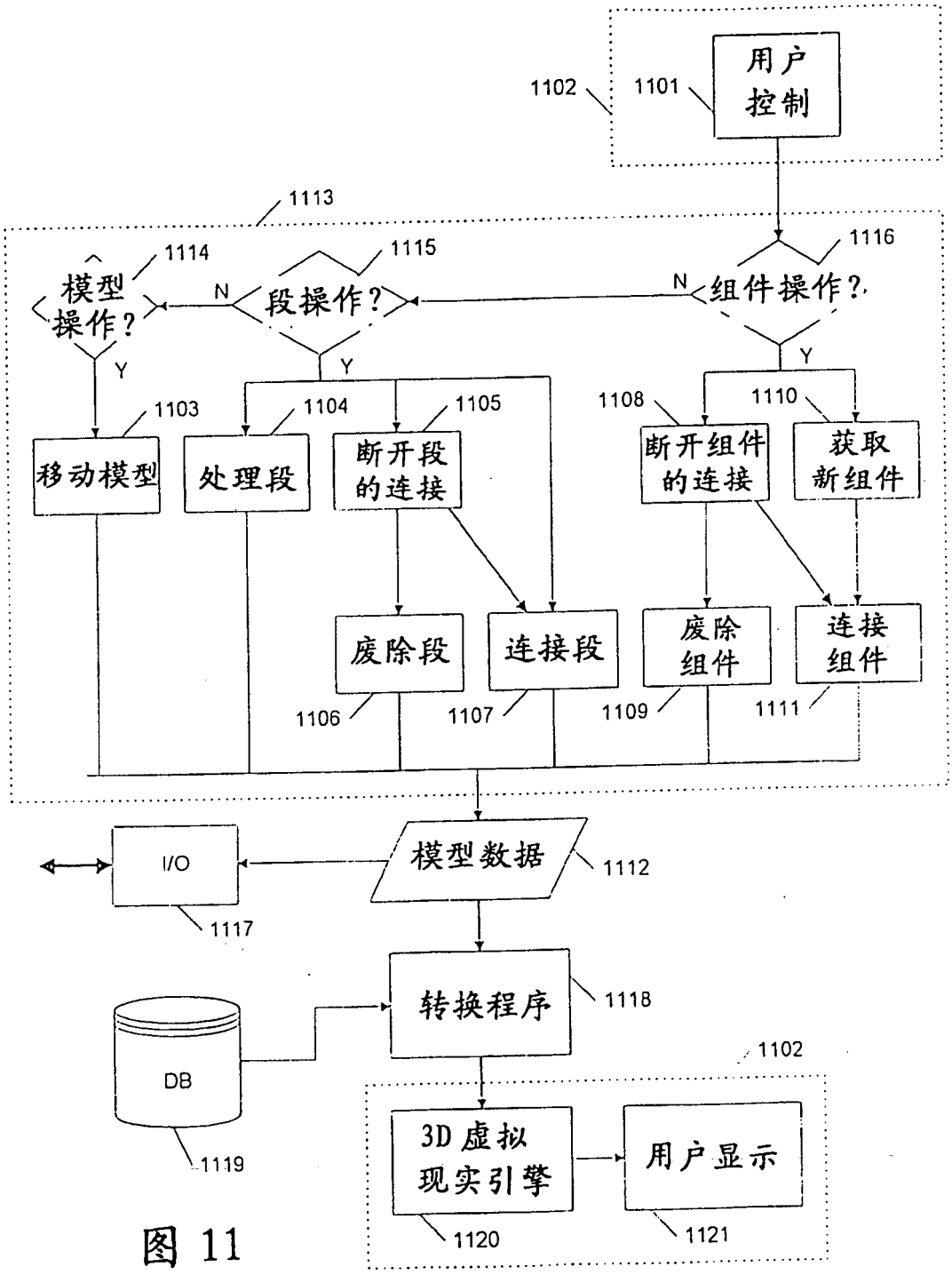

用户控制
1101
1102
1113
1114
模型操作?
1115
段操作?
1116
组件操作?
N
N
Y
Y
Y
1103
1104
1105
1108
1110
移动模型
处理段
断开段的连接
断开组件的连接
获取新组件
废除段
连接段
废除组件
连接组件
1106
1107
1109
1111
I/O
模型数据
1112
1117
转换程序
1118
DB
1119
1102
3D虚拟现实引擎
用户显示
1120
1121

图 11

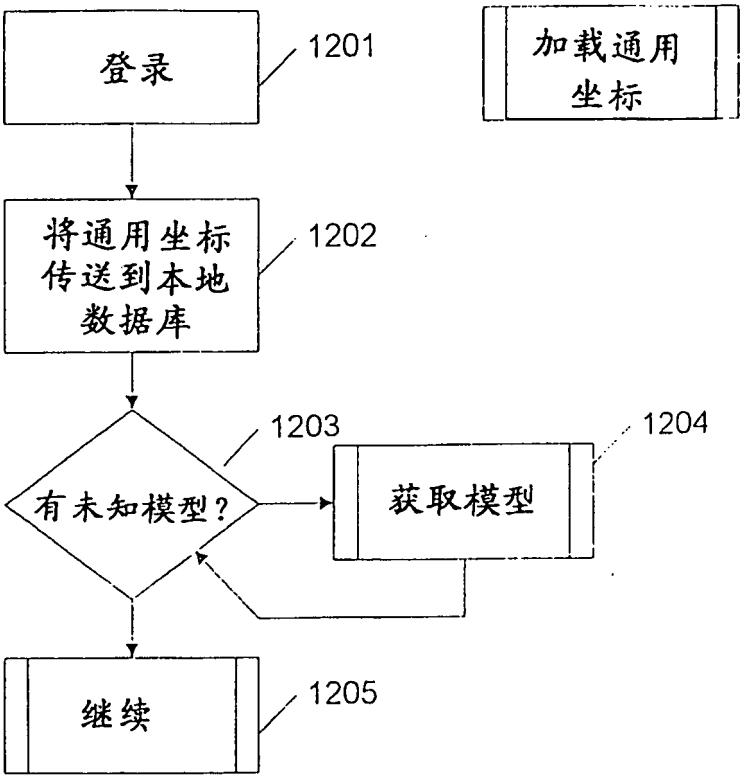

登录
1201
加载通用坐标
将通用坐标传送到本地数据库
1202
有未知模型？
1203
获取模型
1204
继续
1205

图 12

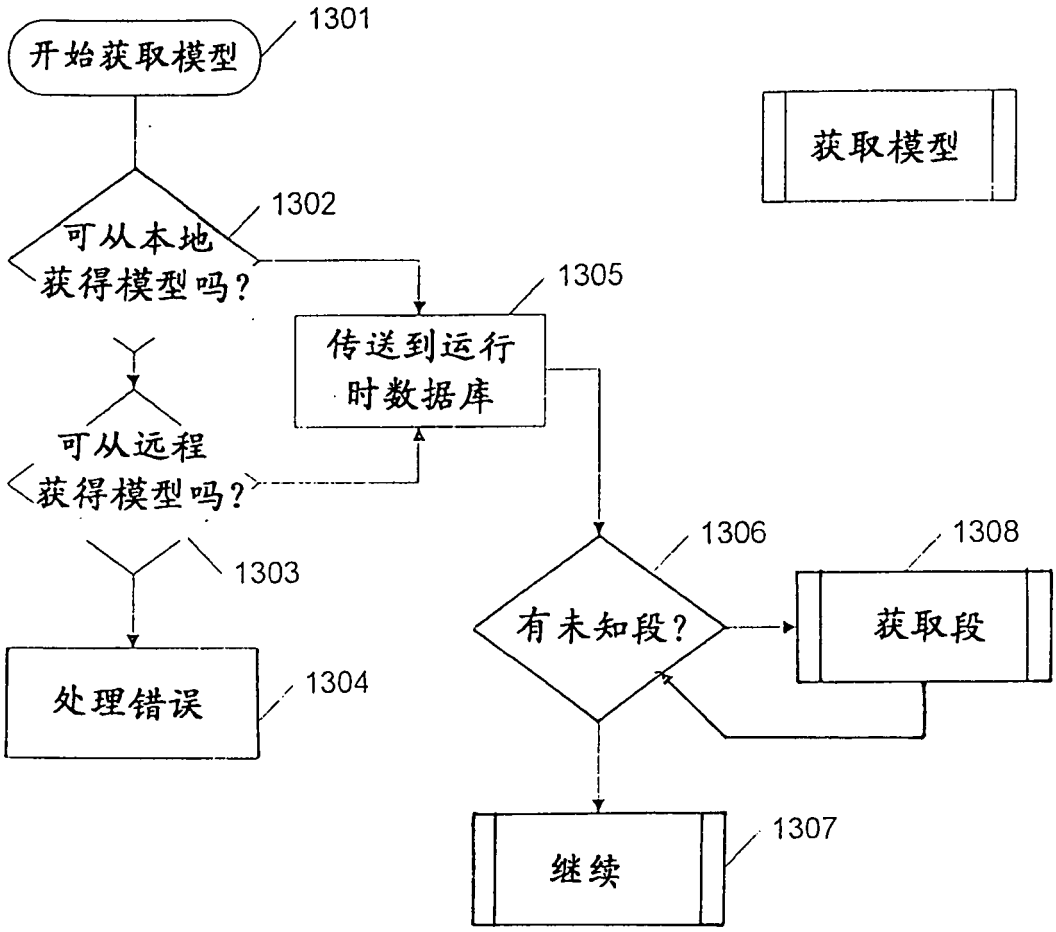

开始获取模型
1301
获取模型
可从本地
获得模型吗?
1302
1305
传送到运行
时数据库
可从远程
获得模型吗?
1306
1308
1303
有未知段?
获取段
处理错误
1304
继续
1307

图 13