[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06T 15/00

[12] 发明专利说明书

[21] ZL 专利号 99814075.9

[45] 授权公告日 2005 年 5 月 25 日

[11] 授权公告号 CN 1203454C

[22] 申请日 1999.11.30 [21] 申请号 99814075.9
[30] 优先权
[32] 1998.12. 4 [33] US [31] 09/205,192
[86] 国际申请 PCT/US1999/026031 1999.11.30
[87] 国际公布 WO2000/034922 英 2000.6.15
[85] 进入国家阶段日期 2001.6.4
[71] 专利权人 法国电信公司
地址 法国穆利诺克斯
[72] 发明人 朱利安·西格内斯
审查员 袁丽颖

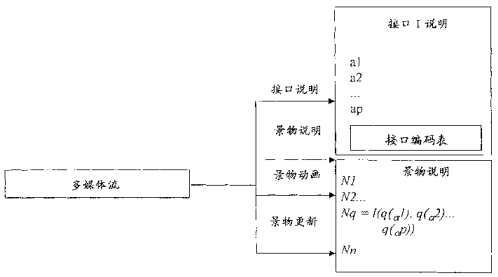
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 朱海波

权利要求书 1 页 说明书 14 页 附图 3 页

[54] 发明名称 利用动态原型控制多媒体流的方法

[57] 摘要

公开一种利用动态接口编码多媒体内容的方法和系统。通过对景物的内容提供一种可扩充的程序设计接口，和在每个帧中显式地规定视频流里的景物中的每个对象的各个特征相比，可以更加有效地对景物编码。



ISSN1008-4274

1. 一种用于控制媒体流的方法，其特征在于包括如下步骤:

创建一个媒体流，其中包括景物成分和接口成分，其中:

该景物成分包括关于（1）在景物中的数字对象和（2）这些对象如何被量化、更新或制作动画的信息;

该接口成分包括数字内容语言的方法的接口定义;

把多媒体流发送到一个接收器;

由该接收器对该多媒体流进行解码，并且通过改变在该接口成分中的至少一个参数而把在该接口成分中定义的所述方法应用于所述景物成分。

2. 根据权利要求 1 所述的方法，其中在该接口成分中定义的所述方法包括（a）用来量化值的量化方法、（b）对这些数字对象进行动画和（c）更新这些数字对象中的至少一种方法。

利用动态原型控制多媒体流的方法

本申请与下述二份于同一日提交的共同未决申请相关：（1）“数据数组预测编码的方法和系统”，代理审查号 2167－0106－2，序列号 09/205,191 和（2）“编码三维生成的景物中的旋转和法线的方法和系统”，代理审查号 2167－0105－2，序列号 09/205,190。二份申请的发明人都是 Julien Signes 和 Olivier Ondet，并把这二份申请收录为本文的参考资料。

技术领域

本发明涉及对计算机生成的图象序列编码的改进方法，并且更具体地涉及利用原型来对怎样对视频流的各部分量化、动画制作或者则予以更新进行编码。

背景技术

尽管以前通过在每个帧中不断地规定每个象素生成许多计算机生成的图象，现代计算机图象序列通常是通过合并多个多媒体对象（例如，视频和二维或三维图形图元）建立的－有时带有一个或多个声道。在多媒体领域中，利用语言构成图象和/或景物。例如，超文本标记语言（HTML）规定正文和图形的二维（2D）组合。类似地，虚拟现实标记语言（VRML）描述三维（3D）世界。

多媒体语言现在还包括动态改变合成图象的扩充。HTML 现在包括动态 HTML（DHTML）扩充，例如在 Danny Goodman 的“Dymamic HTML: The Definitive Reference”一书中说明，该书于 1998 年由 O'Reilly & Associates，Inc.出版，其内容收录作为参考。另外，HTML 的 XML 扩充提供一种元语言以说明如何修改或扩充自然语言。

类似地，VRML 包括对“原”定义的支持，这使得能使新图元成为“原型”。定义的一部分是一个代码块或多个代码块，它们由 VRML 环境执行以提供新的原型式的功能。

成立了移动图象专家小组（MPEG）以调查研究编码和译码图象流所需的技术。所产生的标准（现称为“MPEG－1”）充当另外二个 MPEG 标准（MPEG－2 和 MPEG－4）的基础。MPEG－4 是一个“进展中”的

标准。最终委员会草案是 ISO/IEC FCD 14496－1 MPEG－4 系统和－2 MPEG－4 Visual，这些最终委员会草案的内容收录为参考资料。这些草案包括各种对多媒体内容编码的方法。这些方法包括:（1）用于景物参数的量化方法，（2）用于编码和发送多媒体景物的动画流，以及（3）更新流以便随时间修改多媒体信息。MPEG－4 的最终草案包括支持:（1）用于景物的二进制格式下的量化（以下称为“BIFS”），（2）利用 BIFS 的动画（以下称为“BIFS－Anim”）和（3）利用 BIFS 的更新控制（以下称为“BIFS－Command”）。

在 BIFS 下使用量化包括用量化类别标记节点和字段。向每个节点的每个字段分配一个量化类别，当在量化处理期间对字段赋予一个量化参数结构以参数化时该量化类别施加到该字段上。

BIFS－Anim 定义一种用于多媒体数据的基于流式的和基于文件的动画的编码方法。BIFS－Anim 包括三种基本成分:（1）一个景物图，（2）一个动画屏蔽和（3）多个动画帧。景物包括（a）静止的未标记的对象和（b）要修改的带标记的对象。动画屏蔽设置要通过动画流修改的带标记对象的特征。

BIFS Command 是一种更新 BIFS 节点和景物图的协议。它通过发送用另一个景物代替整个景物、去掉一些带标记对象或者改变景物中带标记对象的特征的各种命令能把改变传送到景物。

发明内容

本发明的一个目的是为多媒体内容语言和/或多媒体置换语言提供动态编程接口和/或扩充。

本发明的另一个目的是对动画、修改和量化方法提供链接和对景物提供新的原型，从而可有效地表达、制作动画和修改景物。

本发明提供一种用于控制媒体流的方法，其特征在于包括如下步骤:创建一个媒体流，其中包括景物成分和接口成分，其中：该景物成分包括关于（1）在景物中的数字对象和（2）这些对象如何被量化、更新或制作动画的信息；该接口成分包括数字内容语言的方法的接口定义；把多媒体流发送到一个接收器；由该接收器对该多媒体流进行解码，并且通过改变在该接口成分中的至少一个参数而把在该接口成分中定义的所述方法应用于所述景物成分。

附图说明

通过参照下述详细说明，尤其在连带着各附图下，对本发明的更完整理解以及对它的各附属优点会变得相当清楚，附图是：

图 1 示意表示实现本发明的方法的计算机；

图 2 是接口说明和景物说明之间的相互关系的方块图；以及

图 3 是接口说明的一个伪码例子。

具体实施方式

现参照各附图，其中相似的参考数字在几张图中代表相同或对应的部分。图 1 是利用动态接口对多媒体内容编码的计算机系统的示意表示。计算机系统 100 实现本发明的方法，其中计算机机壳 102 容纳母板 104，母板 104 含有 CPU 106、存储器 108（例如，DRAM、ROM、EPROM、EEPROM、SRAM、SDRAM 和闪速 RAM）和其它选用的专用逻辑部件（例如，各个 ASIC）或可配置逻辑部件（例如，GAL 和可再编程 FPGA）。计算机 100 还包括多个输入部件（例如键盘 122 和鼠标）和控制监视器 120 的显示卡 110。另外，通过利用适当的部件总线（例如，SCSI 总线、增强型 IDE 总线或超 DMA 总线）连接，计算机系统 100 还包括软盘机 114，其它可移动介质部件（例如，光盘 119、带和可移动磁光介质（未示出））和硬盘 112 或其它固定式高密度介质机。也连接到相同的部件总线或其它部件总线，计算机 100 还可以包括读光盘 118、光盘读/写机单元（未示出）或自动光盘机（未示出）。尽管光盘 119 示成是在 CD 盒式机中，可把光盘 119 直接插入到不需要盒的 CD－ROM 机中。在一替代实施例中，按（1）机顶盒、（2）视频板和/或（3）接收机/回放机中之一或组合实现本发明。在另一替代实施例中，打印机（未示出）提供用于编码多媒体内容的接口组的打印列出。

该系统包括至少一种计算机可读介质。计算机可读介质的例子是：光盘 119、硬盘 112、软盘、磁带、磁光盘、各种 PROM（EPROM、EEPROM、闪速 EPROM）、DRAM、SRAM、SDRAM 等。本发明包括存储在任何一种计算机可读介质上的或组合上的软件，用于控制计算 100 的硬件和用于使计算机 100 能和人类用户交互。这种软件可包括但不限于：部件驱动程序、操作系统和用户应用，例如开发工具。这样的计算机可读介质还包括本发明的用于利用动态接口编码多媒体

内容的计算机程序产品。本发明的计算机代码部件可以是任何解释的或可执行的代码机构，包括但不限于：脚本、解释程序、动态链接库、Java 类和完整的可执行程序。

如图 2 中所示，多媒体流由本发明接收并且译码成景物成分和接口成分。景物成分包括（1）景物中的数字对象和（2）这些对象如何更新或制作动画的二方面信息。接口成分包括接口定义，这些接口定义译码后被内部存储，并且在一实施例中，把这些定义，至少部分地，存储在接口编码表（InterfaceCodingTable）中。具体地接口编码表和新接口相连并且存储给定接口定义中的编码元素以便向景物成分施加数字内容语言（例如，MPEG－4 BIFS 或 VRML）的方法（例如，景化、动画和更新方法）。

接口编码表包含以下信息：（1）更新方式、（2）动画方式（或 Anim 方式）、（3）量化类别、（4）量化参数和（5）动画类别。利用更新模式，可把任何参数说明为 "In" 类型。在显式地或者隐式地支持定义参数类型的语言（例如，VRML）中，参数无需单独地定义为 "In" 类型。具体地，VRML 允许把参数定义为 "In" 类型。具体地，VRML 允许把参数定义成 EventIn、ExposedField、EventOut 和 Field。按照定义，EventIn 和 ExposedField 都是 "In" 类型参数。在使用时，"In" 类型表明该接口可从多媒体流接收进一步的更新（例如，通过诸如 BIFS－Command 流的更新流）。通过用相应的动画类别标记参数，Anim 方式可在接口中的任何参数上操作。通过用相应的量化类别标记参数和发送任选的附加量化参数，Quant（量化）方式也可在接口中的任何参数上操作。

多媒体流可按如下使用接口编码表（ICT）信息：（1）控制量化，（2）更新对应接口的一部分，和（3）通过接口实现动画。具体地，利用 ICT 值 qp[i]以及来自景物的其它语境信息施加和接口参数 ai 相关的量化类别 qc[i]。这能使接口说明几乎最优地压缩。任何在 "In 方式" 下标记的给定接口参数可由将在一给定时间点修改该接口的值的更新流修改。利用连续的动画流持续更新值中的一个，可以使该接口

动画。在这种情况下使用动画类别 ac[i]。

接口编码表是接口说明的一部分并且具有作为例子在图 3 中示出的结构。下面说明各具体参数。

ICTmask: 布尔值的屏蔽以对每个参数设定哪些信息是可使用的。

useQuant: 布尔值，用于为每个参数设定是否发送量化参数。

useUpdate: 布尔值，用于为每个参数设定是否发送 "In 方式" 上的信息及更新类别信息。如前面所说明，在已经支持独立地定义参数是否可更新的语言中该参数是选用的。

useAnim: 布尔值，用于为每个参数设定是否发送 "dyn 方式" 上的信息和动画类别。

对接口中的每个参数 ai 进行其它处理。如后面参照原型接口定义（PROTOinterfaceDefinition）说明那样，在变量 "numFields" 中规定参数数量。从而，对每个参数（选用地）规定下述信息:

In 方式数据: 关于接口的该参数是否 "可更新" 的信息。该信息也可来自接口说明。

量化类别数据: 要使用的量化类别。当 "useQuant" 真时使用该数据。

量化参数数据: 用于这个类别的专用量化参数: 最小值和最大值以于压缩的信息——根据该类别的专用量化/反量化方法。当 "useQuant" 真时使用该数据。

动画参数数据: 为该具体字段选择的动画模式。当 "useAnim" 真时使用该数据。

在 MPEG-4 和 VRML 环境下，本发明允许通过 PROTO 和 EXTERNPROTO 定义新的接口。PROTO 和 EXTERNPROTO 之间的唯一差别是，在第一种情况下在相同的描述（PROTO）内提供实现该接口的代码，而在第二种情况下是在一个外部源中提供的。

在这二种情况下，使用带有原型说明的 ICT 能用所有必需的参数标记字段说明以便以后编码、动画和更新该节点。在 MPEG-4 中，

ICT 包括下述信息:（1）量化类别（见表 1),（2）动画类别（见表 2）和（3）量化参数（包括该字段的最小值和最大值)。

表 1 量化类别

类别	说明
0	无
1	三维位置
2	二维位置
3	绘图次序
4	SF 彩色
5	纹理坐标
6	角度
7	尺度
8	插值器关键字
9	法线
10	旋转
11	对象尺寸 3D（1）
12	对象尺寸 2D（2）
13	线性标量量化
14	坐标索引
15	保留

表 2 动画类别

类别	说明
0	无
1	三维位置
2	二维位置
3	保留
4	彩色
5	保留
6	角度
7	浮点
8	界限浮点
9	法线
10	旋转
11	二维尺寸
12	三维尺寸
13	整数
14	保留
15	保留

利用该信息以及 PROTO 定义，MPEG－4 终端能生成一个编码、更新和动画节点所需要的节点编码表（NodeCodingTable）。下面是一个用于 DirectionalLight 节点的节点编码表的典型例子。设置最小值、最大值以及量化（Q）类型和动画（A）类型。从字段说明中导出所有的 id。利用节点数据类型（SFWorldNode，SF3Dnode）以及它们的相应 id 确定该节点可在其中使用的语境。各个 PROTO 接收专用节点类别。

DirectionalLight	SFWorldNode SF3DNode				000100 0001		
字段名	DEF id	IN id	OUT id	DYN id	[m,M]	Q	A
环境亮度	000	000	000	00	[0,1]	4	2
彩色	001	001	001	01	[0,1]	4	2
方向	010	010	010	10		9	6
亮度	011	011	011	11	[0,1]	4	2
接通	100	100	100				

表 4 MPEG－4 类型

布尔	0000
浮点	0001
时间	0010
串	0011
Int32	0100
串	0101
Vec3f	0110
Vec2f	0111
彩色	1000
旋转	1001

二进制语法

以下是带有对应的接口编码表的 PROTO 的二进制语法。按照 MPEG－4 MSDL 语言说明二进制位流语法。

1.0 PROTOdeclaration

```
PROTOdeclaration() {
            PROTOinterfaceDefintion();
            PROTOcode();
            protoCodingTable();
}
```

2.0 PROTOinterfaceDefinition

```
class PROTOinterfaceDefinition {
      bit(idBits) id;          // The ID used to refer to the PROTO.
      Bit(5)      type;        // Context type, using last 5 bits, Tbl 1.
      bit(4)      fieldBits;   // Number of specifying number of fields.
      bit(fieldBits)    numFields;
                        // Number of fields and events in the proto.

      for (i=0; i<numFields; ++i) {
            bit(2) eventType; // eventIn, eventOut, field or
                              // exposedField

            bit(6) fieldType; // The field type, using table 1.
      }
}
```

3.0 PROTOcode

```
class PROTOcode {
      bit(1) isExtern          // Is this an extern proto?
      if (isExtern) {          // Read URL for location of PROTO defs.
            MFUrl locations;

      } else {
            do {
                  SFNode node(SFWorldNodeType,true);
                  bit(1) moreNodes;
            } while (moreNodes);
      }
}
```

4.0 SFNode

```
class SFNode(int nodeDataType,boolean ISED) {
      bit(1) isReused ;
      if (isReused) {
            bit(BIFSConfig.nodeIDbits) nodeID;
      }
      else {
            bit(GetNDTnbBits(nodeDataType)) localNodeType;
            nodeType = GetNodeType(nodeDataType,localNodeType);
            bit(1) isUpdateable;
            if (isUpdateable) {
                  bit(BIFSConfig.nodeIDbits) nodeID;
            }
            bit(1) MaskAccess;
            if (MaskAccess) {
                  if (ISED)
                  isedMaskNodeDescription node(MakeNode(nodeDataType,
                                                      nodeType));
                  else
                  MaskNodeDescription mnode(MakeNode(nodeDataType,
                                                nodeType));
            }
            else {
                  if (ISED)
                  isedListNodeDescription mnode(MakeNode(nodeDataType,
                                                      nodeType));
                  else
                  ListNodeDescription lnode(MakeNode(nodeDataType,
                                                nodeType));
            }
      }
}
```

5.0 isedMaskNodeDescription

```
class isedMaskNodeDescription(NodeData node) {
      for (i=0; i<node.numDEFfields; i++) {
            bit(1) Mask;
            if (Mask) {
            if (node.nodeType == PROTO) {
                  bit(1) ISedField;
                  if (IsedField)
                        bit(node.numDEFfields;) protoField;
                  else
                        Field value(node.field[node.def2all[i]]);
            }
            else {
                  Field value(node.field[node.def2all[i]]);
            }
      }
```

6.0 isedListNodeDescription

```
class isedListNodeDescription (NodeData node) {
      bit(1) endFlag;
      while (!EndFlag){
            int(node.nDEFbits) fieldRef;
            if (node.nodeType == PROTO) {
                  bit(1) ISedField;
                  if (IsedField)
                        bit(node.numDEFfields;) protoField;
                  else
                        Field value(node.field[node.def2all[i]]);
            }
            else {
                  Field value(node.field[node.def2all[i]]);
            bit(1) endFlag;
      }
}
```

7.0 protoCodingTable

```
InterfaceCodingTable() {
            InterfaceCodingMask;
            InterfaceCodingParameters;
}
```

8.0 protoCodingMask

```
InterfaceCodingMask() {
            boolean useQuant;
            boolean useUpdate;
            boolean useAnim;
}
```

9.0 protoCodingParameters

```
InterfaceCodingParameters() {
      for (i =0; i < proto.NbdefFields; i++) {
            if (useQuant) {
                  ui nt(4) quantCategory;
                  if (quantCategory == 13)
                        uint(5) nbBits;
                  bit(1) hasMinMax;
                  if (hasMinMax) {
                        SFField(fieldType) minFieldValue;
                        SFField(fieldType) maxFieldValue;
                  }
            }           // REM: Here we do not need an "isUpdatable"
                        // since there is it automatically comes
                        // from the "event and field type"
            if (useAnim) {
                  boolean isDyn;
                  if (isDyn) {
                        uint(4) animCategory;
                  }
      }
}
```

本发明的一个示例使用利用 MPEG－4 应用中的各种 PROTO 类型来实现一个协同专业应用，该应用能使多个工程师在不同的远程位置同时参与建模和设计（例如，船的模型）并且仍能实时地观看改变。工程师们进行设计试验并且希望所有场地同时得到实验结果，并且同时利用最有效的压缩传输来适应低比特率网络连接。另外，该系统应足够通用以便容纳各方面的设计。通过为该应用采用 MPEG－4 框架，得到一个用于整个通信和应用设计的通用并有效的框架，以代替为各种情况设计专用的并且可能是子最优的应用。

对于该例子，假定一个三维的船模型包括 20,000 个顶点构成的 10,000 个多边形。在该实验中，工程师们希望分析并控制因某些风造成的船的变形。通常会通过数个参数，例如风速、风向和波浪的高度控制该变形。接着把该船定义为 MPEG－4 增强型 PROTO（原型），并且一旦接收新的值，该 PROTO 内的 Script 计算新的顶点参数。以这种方式，工程师们可进行直接仿真，并且利用 BIFS－Anim 协议只发送一些优化编码的参数而不是重新发送 20,000 个顶点。另外，每次要检查新结果时，工程师们只需要利用 BIFS－Command 协议发送一组新参数。

根据该例子，可定义下面的 PROTO 类型：

```
PROTO WindBoat [
      exposedField      SFVec3f          windSpeed
      exposedField      SFVec3f          windOrientation
      exposedField      SFFloat          waveHeight
] {
      Script { # Changing vertices according to parameters
      }
      IndexedFaceSet {  # The 3D Model of the boat
      }
}
```

ICT 参数

该 ICT 中可关联下述编码参数。

WindBoat	PROTO				10001		
字段名	DEF id	IN id	OUT id	DYN id	[m,M]	Q	A
WindSpeed	00	00	00	00	[-I,+I]	11	1
WindOrientation	01	01	01	01	[0,1]	9	9
WaveHeight	10	10	10	10	[0,100]	138	11

除了增加功能性外，本发明产生更高效编码的位流。通过对使用 PROTO 接口发送信息和不使用 PROTO 接口发送信息进行比较，可得比特率的比较。可以为 PROTO 实例（把船放到景物中）、为 PROTO 更新（或远程地改变值）或者为动画制作，测量增益。对每种要发送的数据的以字节为单位的尺寸进行比较。请注意若使用不带 ICT 的普通 PROTO，则不能得到更新功能和动画功能。

实验	原始数据 尺寸(字节)	VRML PROTO 数据尺寸(字节)	MPEG－4 BIFS PROTO 尺寸(字节)
示例(场景中 n 条船)	n*400,000	400,000+35*n	30,000+n*5
更新	240,000	不能得到	5
动画(每个平均 amin 帧)	240,000	不能得到	3

作为第二个例子，本发明可用于对跳动的球进行动画。替代在每帧中发送更新该球的每个多边形的命令流，建立一个控制动画的接口。该接口定义一个控制该球离地面的高度的参数。为了对该球动画，只需要对每个帧改变高度参数－该接口的脚本实现各顶点的实际更新。类似地，若该球接口包括用于彩色的第二参数，则会通过脚本而不是通过分别对各多边形的彩色进行动画改变每个多边形的彩色。

尽管上面通过扩展语言所提供的方法的接口说明本发明，本发明还指向向接口提供接口。正如对象可从超类继承行为那样，接口可从以前例示的接口继承功能。

如一般的业内人士所清楚，在不会背离本发明的精神下可以和本文中所明确的描述不同地实现本发明。从而，本说明书不是用作限制性的，而本发明的限制仅由权利要求书的范围定义。

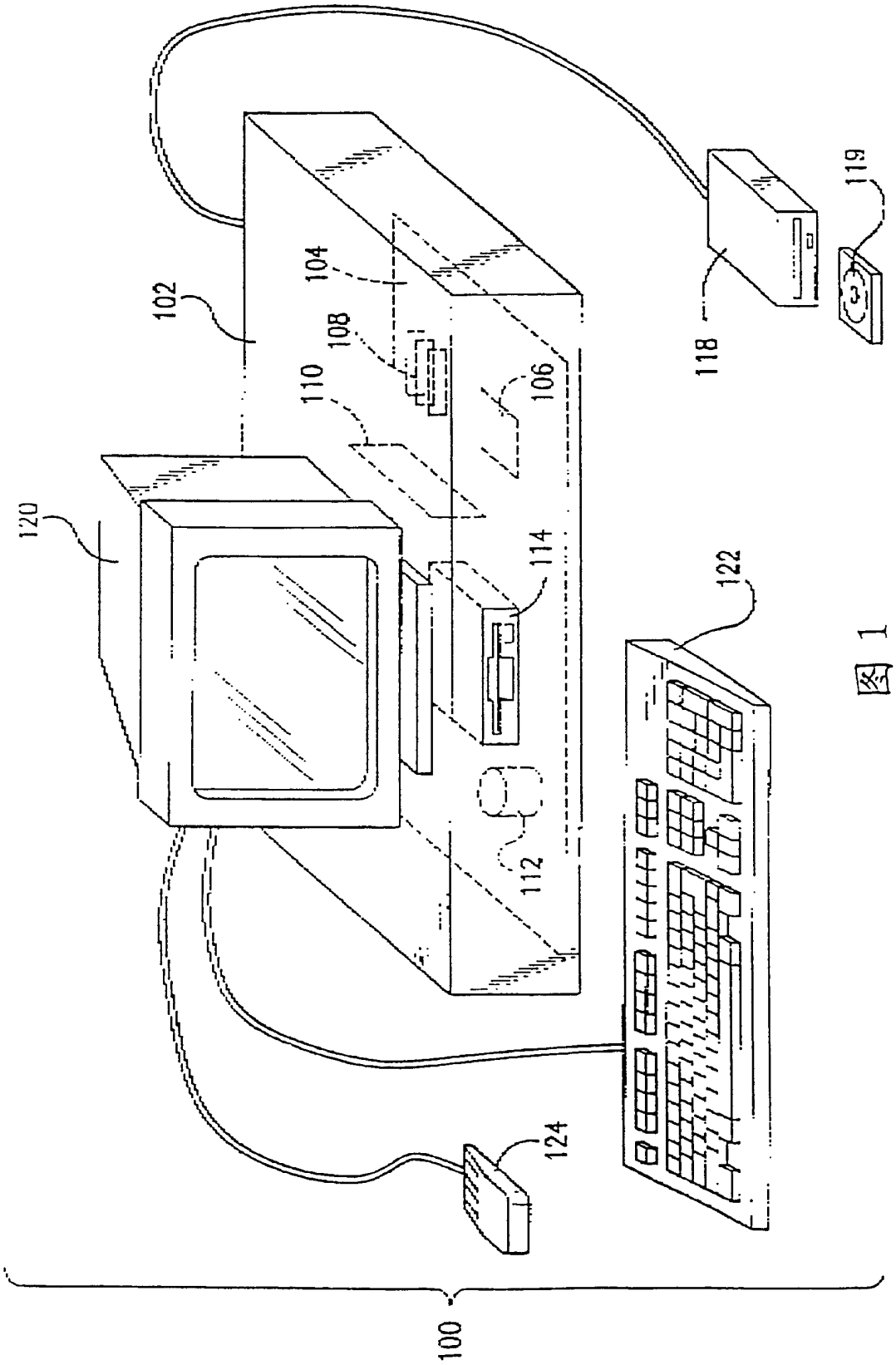


图 1

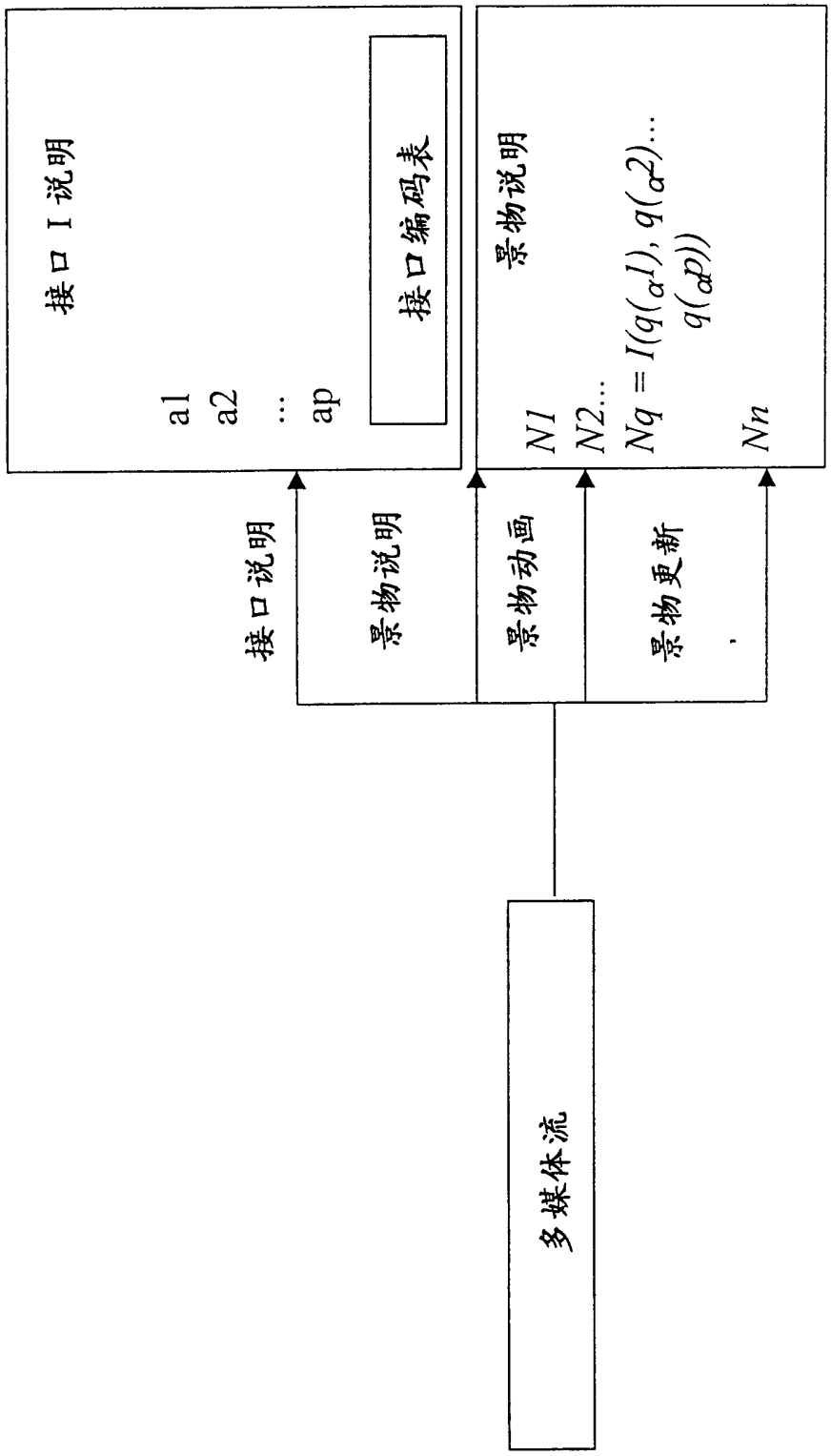

接口 I 说明
a1
a2
...
ap
接口编码表
景物说明
N1
N2...
Nq = I(q(α1), q(α2)...
q(αp))
Nn
接口说明
景物说明
景物动画
景物更新
多媒体流

图 2

```
InterfaceDeclaration() {
          //Specific declaration information…
          InterfaceCodingTable();
}

InterfaceCodingTable() {
          InterfaceCodingMask;
          InterfaceCodingParameters;
}

InterfaceCodingMask() {
          boolean useQuant;
boolean useUpdate;
          boolean useAnim;
}

InterfaceCodingParameters() {
          for (I =0; I < interface.NbParameters; I++) {
               if (useQuant) {
                    int quantCategory;
                    QuantParameter(quantCategory);
               }
               if (useUpdate) {
                    boolean isUpdatable;
               }
               if (useAnim) {
                    boolean isDyn;
                    if (isDyn) {
                         int animCategory;
                    }
          }

}
```

图 3