[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
A63F 13/00
G06T 15/70

[12] 发明专利说明书

[21] ZL 专利号 97190661.0

[45] 授权公告日 2005 年 4 月 13 日

[11] 授权公告号 CN 1196508C

[22] 申请日 1997.6.4 [21] 申请号 97190661.0
[30] 优先权
[32] 1996. 6. 5 [33] JP [31] 165224/1996
[32] 1996. 6.27 [33] JP [31] 186666/1996
[86] 国际申请 PCT/JP1997/001896 1997.6.4
[87] 国际公布 WO1997/046295 日 1997.12.11
[85] 进入国家阶段日期 1998.2.5
[71] 专利权人 世嘉股份有限公司
地址 日本东京
[72] 发明人 加来徹也 峰裕一朗 小野孝志
芳贺宪夫 大崎诚 山口贵之
关根纪裕 油井亮弥 西川沙织
杉本哲也 吉田茂 中谷学
内田真澄

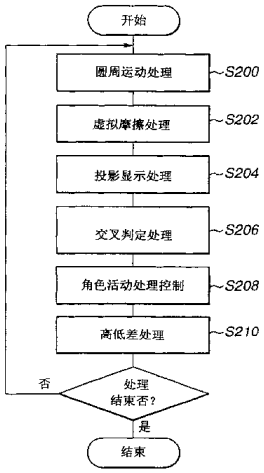
审查员 陈善学
[74] 专利代理机构 中原信达知识产权代理有限责任公司
代理人 余 朦

权利要求书 2 页 说明书 42 页 附图 60 页

[54] 发明名称 游戏用图象处理装置

[57] 摘要

在本游戏用处理装置中已在虚拟空间内设定了规定数目的模型(角色)，并进行控制使得一虚拟视点为基础的虚拟空间显示在显示器上。 本装置具有图像处理装置，用于把虚拟的向心力加到模型上，使得更为实时性地显示模型的活动。 本装置具有残像显示处理装置，用于把活动路径显示为残像，而且，该处理装置具有用于存储活动数据的装置。 此外，还有一个显示控制装置，用于显示已存储的带有实际活动数据的数据。



ISSN1008-4274

1. 一种用于打斗型游戏的图像处理设备，其特征在于：

该设备设置成：i) 在虚拟空间内设置由游戏者控制的第一模型（30）和第二模型（32），所述的第一模型和所述的第二模型相互打斗；ii) 控制这些模型使它们在上述虚拟空间内在规定的方向上移动，并且使显示装置（5）显示以虚拟视点为基础的该虚拟空间的图像；

所述的设备包括图像处理装置（10），用于进行图像处理（S200），从而执行预定的移动操作，而且，通过游戏者控制第一模型，在进行对峙的所述第一模型和所述第二模型之间设定虚拟的引力，并且使所述第一模型自动地围绕所述的第二模型运动，所述的第二模型是虚拟引力的中心。

2. 根据权利要求 1 所述的图像处理设备，其中图像处理装置（10）进行虚拟摩擦处理（S202），以使施加到模型上的摩擦力的量根据模型是移动的或模型是静止的情况而改变。

3. 根据权利要求 1 或 2 所述的图像处理设备，其中图像处理装置（10）进行角色的投影显示处理（S204），以便与放置上述模型的舞台的表面形状相吻合地显示上述模型的投影象。

4. 根据权利要求 3 所述的图像处理设备，其中图像处理装置（10）进行交叉判定处理（S206），以便判定从虚拟视点产生的视场相对于要被显示的模型的重叠情况，以及相对于不设为显示对象的另一模型的重叠情况，如果判定结果为肯定的，则另一模型不被显示。

5. 根据权利要求 4 所述的图像处理设备，其中图像处理装置（10）进行角色的活动处理控制（S208），以便对上述模型的预定的移动轨迹的终点，设定与该终点不同的目标点，并对移动轨迹进行内插，使得上述终点与该目标点一致。

6. 根据权利要求 5 所述的图像处理设备，其中图像处理装置（10）进行角色的高低差处理（S210），以便求把上述模型设置于上述虚拟空间内的高低差，并根据该高低差内插施加在该模型上的动作。

7. 一种包括如权利要求 1 所述的图像处理设备的游戏设备。

8. 根据权利要求 7 所述的游戏设备，进一步包含输入装置（2b），用于控制上述模型的移动，所述输入装置包括方向键（26b），用于指定相应的模型垂直或水平进行的动作的方向，所述图像处理装置可操作用于在响应于上述方向键的操作来执行图像处理，以便使上述模型产生自动的圆周运动。

游戏用图像处理装置

本发明涉及在虚拟设定的游戏空间（以下称为“虚拟空间”）中的图像（以下称为虚拟图像）处理，特别是涉及可以更为现实地表现设定在虚拟空间中的虚拟模型（例如，称之为角色）在画面上的动作的游戏用图像处理。该图像处理的手法特别适合于三维视频游戏机。

由于计算机图形学（CG）技术的发展，已变得可以立体性地实时性地表现虚拟设定的虚拟空间（也叫做‘虚拟世界’）。这就是视频游戏领域，即近年开发的可进行高速运算的中央处理装置（CPU）、视频显示处理器（VDP），以及高速且经济地利用该计算机图形学技术。

在这样的游戏机中，由于要根据使用者（也可以叫做游戏者或比赛者）的行为使游戏的内容不断地变化，所以必须使被显示物体在虚拟空间中以高速自由地活动。为此，通常用由三角形或四边形组成的一组多边形构成被显示体的模型（例如，角色），通过同时地改变这些多边形的空间位置来显示这种模型的运动。

此外，在角色的胳膊和腿等被显示物体的特定部分或者面部要同时动作的情况下，可以把多个多边形构成的多边形数据群作为一个单位，并对每一多边形数据群指定一个空间位置，从而使这些特定的部分或者面部同时动作。

近些年来，在用多边形构成角色，使之在监视器上表现出从规定的视点捕捉到的虚拟空间内的角色的动作的图像的所谓三维游戏机市场引起人们注目，特别是模拟多个勇士的武打的游戏机很受人喜爱（例如，SEGA ENTERPRISES 公司生产的‘残忍的勇士’（batcher fighter）（商标））。

在该武打模拟游戏中，游戏者快速地操作已附加到控制器上的操作杆或键盘或者按键使显示在画面上的勇士根据游戏操作杆等的操作而决定的指令进行动作。勇士的动作被称之为‘活动’戏，为实现该活动所必须的数据用“活动捕获”技术取得。然后，根据需要对这些数据进行

处理，作为最后的活动数据可以在3D游戏机中利用。

在这样的游戏机中，为了提高其商品价值，希望用更真实地表现角色的动作。就是说，例如，通过增加更多的变量使之极力接近实际的打斗的动作。但是，由于可以预想的角色的动作极其多种多样，所以在实现上述的目的方面还有许多尚未改善的地方。虽然也考虑过或者预先作成全部所需要的动作并存到存储器中，以及获得用于取得这样的活动的特征表示，但这样作的话，就存在着数据量将变得过大，不能进行实时处理等的问题。

因此，本发明的主要目的是提供一种可以极力压低图像处理所需要的运算量和数据量，且可以更为实时地和真实地表现角色在画面上的动作的游戏用图像处理的方法。

从该主要目的的观点来看，现有的视频游戏机所具有的问题的具体的第1方面如下所示：

A1、在3D视频游戏中，由于是利用投影变换来表现来自虚拟视点的二维画面上的图像，所以难于使勇士在画面的纵深方向上移动，即在游戏者的目光方向上移动，因此，要想使一个勇士在另一勇士的周围转圈就毫无办法。所以，把改善这样的环绕角色的动作，作为本发明的一个具体目的。

A2、现有的3D视频游戏机中，由于图像的显示是根据虚拟视点进行的，所以如果在虚拟空间中的一个能遮挡勇士的位置上设置诸如一堵墙这样的结构，就存在将勇士遮挡起来的问题。所以，如上所述，进行改善角色被构造物遮挡起来这种事态的显示，是本发明的另一具体的目的。

A3、在现有的三维视频游戏机中，角色的活动采用了例如连续使用仿样（spline）函数的方法，或者在连续帧中影响预定的再现图形的方法。但是，在现有的视频游戏机中，这种活动被固定下来，所以不能对活动进行修正以与对方角色的动作相匹配。这样，使得可以进行这种修正，是本发明的另一个具体的目的。

此外，作为从上述主要目的派生出来的事项，为了提高商品价值，

希望除上述第 1 方面之外，再加上下述一点：对角色的动作进行提高戏剧效果的画面显示。鉴于这一希望的现有的视频游戏机的问题的第 2 方面可以具体地说明如下。

B1、作为提高 CG 画面的戏剧效果的技术，人们知道“活动模糊”（motion blur）。若采用该活动模糊，则一个象素中可以产生许多光线（ray），采用涂上取其平均的颜色的办法，就可以制作有焦点之外的(out of focus)或有动作的画面。

另一方面，近些年来，在视频游戏等的 CG 领域中，为了进一步提高戏剧效果，角色的动作，可以显示人类视觉生理上常见的残像的效果，例如，在勇士正在挥舞的大刀的轨迹中加上残像。因此，本领域专业人员要计算多边形并构成与勇士的活动相匹配的残像，并使得该残像在勇士的附近进行显示。

但是，由于可以想像的角色的动作极其多种多样，要想编辑与所有的情况匹配的许多形态的残像用的多边形并使之存储到存储器中不仅将限制有限的计算机图形学装置的能力，与角色的活动相匹配地计算残像的多边形对计算机图形学装置的负荷也太大。所以，在形成 CG 图像方面，使得即便是考虑到提高戏剧效果也不会产生大的计算负荷（说得更详细点的话，使得减小这样的负荷），且可以使残像图像与实像图像同时进行显示，是本发明的另一个具体的目的。

B2、在利用了现有的图像处理装置的游戏装置中，可以在画面上显示飞溅的砂子和飞溅的水花等的飞行体（例如，SEGA ENTERPRISES 公司生产的“Rally”（商标）。但是这样的飞溅的砂子或飞溅的水花，因为仅仅是使纹理映象到多边形上，所以不能把模型（车等）的动作正确地反映到飞行体上。所以，使模型的动作更正确地反映到飞行体上，是本发明的再一个具体的目的。

B3、在现有的图像处理装置中，在模拟在虚拟空间内下落的模型的活动方面的品质是不充分的。所以，本发明的再一个具体的目的是使在虚拟的空间中的自由下落的模拟做成为更高的品质。

B4、在现有的图像处理装置中，先判定虚拟空间内的区域和运动模型之间的接近程度，如果判定是肯定的，则限制模型的活动使得模型不

超越所述区域。通常，作为这样的区域被设定为四角形或圆形等的定型的类型。在这种情况下，由于区域是普通的形状，所以把角色从所划分的区域中赶出去的运算也是容易的。但是，在该区域是不规则形状的情况下，在现有的 CG 装置中，就不容易处理这种问题。所以，本发明的再一个具体的目的是即便是不规则的区域，也可以准确且容易地执行区域和模型之间的冲突判定肯定后的模型的动作（活动）的计算处理。

B5、以前，在适用于游戏装置的图像处理的区域中，在表现一系列周期性的画面（例如，反复进行的波浪的绘画和跳绳等）的情况下，先手工编辑这样的画面的列并作为纹理的列，然后，采用顺次附地把它映象到多边形上去的办法，来表现反复进行同一动作的一连串的图像。

作成这样的纹理的列，由于需要很多手工作业，故人们试图利用应用软件。作为这样的应用软件，人们熟悉例如下列的商品名称。这些商品是： Alias/Wavefront(Alias/Wavefront 公司（110 Richomond， East Toronto ,Ontario Canada M5c lpl）生产)。在利用该软件的情况下，只要把预定的参数赋予软件，就可以得到纹理列

但是，在利用现有的这种的软件的情况下，通常是在纹理列的最初和最后，即纹理的图形会变成为不连续。

因此，在现有的图像处理的区域中，在反复进行的列的制作中还有进行改良的余地。所以本发明的再一个具体的目的是使得可以用图像处理做成这种的纹理列。

B6、在现有的图像处理装置中，为了提高回放图像的戏剧效果，常进行慢放，以将角色的动作减慢。但是，这种慢放的演出，其效果受限制。所以，本发明的再一个具体的目的是提供一种游戏用图像处理手法，该手法可以使在营造出更为有效的戏剧感方面的回放速度进行变化。

应用本发明，则可以从各种各样的视点提供可以极力压低图像处理所需的运算量或数据量，且可以更为实时地和真实地表现角色在画面上的动作的游戏用图像处理手法。在本发明中，为了实现从其主要目的派生出来的种种的具体的目的，采用了如下所述的种种的构成。

构成本发明的一个方面的一连串的构成如下所述。就是说，本发明

是一种下述游戏用图像处理装置：把规定数目的模型设定于虚拟空间内，并控制这些模型以使其在虚拟空间中规定的方向上移动，使以该视点为基础的该虚拟空间的图像显示于显示装置上，包括将虚拟的向心力施加到模型上的装置。此外，还提供具有该游戏用图像处理装置游戏装置。另外，还提供已记录下含有执行该图像处理装置的一连串的步骤的信息的记录媒体。

另外，根据本发明，一种游戏用图像处理装置包括一种图像处理器，其中当模型正在运动时或静止时，施加于其上的摩擦力的量会变化。

此外，根据本发明的游戏用图像处理装置中还具备一种图像处理器，用于显示与设置上述模型的舞台的表面形状相吻合的模型的投影象。

还有，根据本发明的游戏图像处理装置还具备一种图像处理器，用于执行虚拟视点的视场相对于作为显示对象的上述模型和尚未设定为显示对象的另一模型的重叠进行判定，在肯定该判定的情况下，不显示另一模型。

根据本发明的游戏用图像处理装置，还具备一种图像处理器，用于相对于上述模型的移动轨迹的预定终点，设定与该终点不同的目标点，并对移动轨迹进行内插以使上述终点与该目标点一致。

根据本发明的游戏用图像处理装置，还具备一种图像处理器，用于寻求上述模型设置于上述虚拟空间内的高低差，并根据该高低差对于该模型修正所给定的动作。

采用这样的构成，就可以以更为丰富的现实感显示角色等已设置在虚拟空间内的模型的动作。具体地说，可以容易地生成一个模型在另一个模型的周围转圆圈的图像。此外，还可以把已设置有模型的舞台的状态反映到模型的动作上去。还有，可以把舞台的凹凸正确而简单地反映到模型的投影图像上。再者，即便是在把模型遮挡起来的位置上，在墙壁等的构造物设定在虚拟空间内的情况下，也可以进行显示而不会把模型挡住。此外，还可以使模型的活动与对手一侧的模型的动作合在一起进行修正，所以，可以真实地表现模型的丰富多彩的动作。由于已经考虑到了模型的高低差，所以可以更富有现实感地生成优良的图像。

此外，构成本发明的一连串的构成如下所示。作为其中之一的具体构成，本发明提供了一种在虚拟空间内设定规定数目的模型并进行控制，使得该模型在上述虚拟空间内在规定的方向上移动，同时，使以该视点为基础的该虚拟空间的图像显示于显示装置上的游戏用图像处理装置，其特征是：包括一种用于把模型的移动轨迹作为残像进行表现的残像处理器，其具备存储装置，用于原封不动地存储上述模型的当前活动以前的活动数据；显示控制装置，用于和上述模型的当前活动数据一起显示该存储数据。该显示控制装置还可以具备把戏剧效果加到上述存储装置的数据上去进行显示的演出装置。该演出装置可以进行半透明的图像处理。

作为另外的一种具体的构成，本发明提供了一种在虚拟空间内设定规定数目的模型并进行控制，使得该模型在上述虚拟空间内在规定的方向上移动，同时，使以该视点为基础的该虚拟空间的图像显示于显示装置上的游戏用图像处理装置，其特征是：具备下述装置：计算装置，用于计算上述模型的移动特性；飞行体控制装置，用于用上述运算装置计算出来的计算结果控制飞行体的一连串的运动。此外，作为另外的具体的构成，本发明的特征是：在模拟设定在虚拟空间内的模型的自由落下运动的游戏用图像处理装置中，具备模拟上述模型的全体运动的的第 1 装置和把转圆圈运动赋予上述模型的第 2 装置。

此外，作为另外的具体的构成，本发明提供一种在虚拟空间内设定规定数目的模型并进行控制，使得该模型在上述虚拟空间内在规定的方向上移动，同时，使以该视点为基础的该虚拟空间的图像显示于显示装置上的游戏用图像处理装置，其特征是：具备下述装置：区域划分装置，用于把不规则形状的区域设定于上述虚拟空间内；矢量设定装置，该装置是一把规定的矢量设定于上述区域与上述模型之间，如果模型要试图移出该的区域，将该矢量设定在使模型不越过所划分的区域的方向上；模型移动控制装置，用于根据该矢量控制上述模型的移动。

另外，作为另外的具体的构成，在本发明的装置中，所形成的纹理在给定规定的参数后会周期性地变化，其特征是：包括产生第一纹理列的第一装置和产生第二纹理列的第二装置，当将参数给予各个装置时，所述纹理列是这样形成的，即使第一纹理列的第一或最后的纹理与第二纹理列的第一或最后的纹理变成为连续的形态，第一纹理列和第二纹理列中的至少一列的透明度逐渐增加或逐渐减少，其目的是从两纹理列的

最初纹理开始顺序进行重叠。

本发明是一种形成周期性地变化的纹理的方法，其特征是：包括先在规定的方向上变化的第一纹理列的第一步骤，和同样地在规定的方向上变化的第二纹理列的步骤，并且其中第一纹理列的最后或第一纹理与第二纹理列的第一或最后纹理以连续方式形成，第一纹理列或者第二纹理列中的至少一方的透明度逐渐增加或逐渐减少，从而两纹理列的最初纹理彼此间开始顺序进行重叠。

此外，作为另一具体的构成，本发明提供一种在虚拟空间内设定规定数目的模型，并进行控制使得该模型在上述虚拟空间内在规定的方向上移动，同时，使以该视点为基础的该虚拟空间的图像显示于显示装置上的游戏用图像处理装置，其特征是：包括第一装置，用于判定在两个模型之间规定的条件是否已成立；第二装置，用于将已建立的条件反映到各个模型的运动中去；第三装置，用于在该条件已经成立的时候使一个模型的回放速度对于另一模型发生变化。

倘采用这样的构成，则在可以更为实时地表现在虚拟空间中的模型的动作的同时，还可以生成演出感丰富的游戏图像。说得更详细点的话，倘采用其中之一的具体的构成，则可以提供一种游戏用图像处理装置：即便是在形成游戏图像方面，有提高演出效果的考虑，也不会产生大的计算负荷。另外，倘采用另外的具体的构成，则可以提供能够使模型的动作正确地反映到飞行体上的游戏用图像处理装置。倘采用另外一具体的构成，则可以提供可以使在虚拟空间内自由落下的运动的模拟变成为更为高品质的模拟的游戏用图像处理装置。倘采用再另一具体的构成，则提供即便是不规则形状的区域，也可以准确且容易地执行区域与模型之间的冲突判定肯定后的模型的动作（活动）的计算处理的游戏用图像处理装置。此外，倘采用另一具体的构成，则即便是利用图像处理，即便是反复地进行显示，也可以做成一连串的纹理列，该一连串的纹理列可以再现连续图像而无不协调感。还有，倘采用另外的具体的构成，则可以提供能够在营造效果更为良好的演出感方面的回放速度进行变化的游戏用图像处理装置。

附图的简单说明

图 1 是作为本发明的实施例的图像处理装置的视频游戏机的框图。

图 2 是实施例 1 的主处理的流程图。

图 3 是的斜视图示出了圆周运动中的角色的圆周运动。

图 4 是圆周运动处理的详细流程图。

图 5 是从虚拟空间的上边示出了圆周运动的模式图。

图 6 是相当于圆周运动的结束状态的斜视图。

图 7 是虚拟性的摩擦处理的原理图。

图 8 是虚拟性的摩擦处理的详细流程图。

图 9 是投影显示处理的原理图。

图 10 是交叉判定处理的原理图。

图 11 是交叉判定处理的详细流程图。

图 12 是用内切圆进行近似的构造物的 xz 平面图。

图 13 的原理图示出了交叉判定处理的具体例。

图 14 的原理图示出了交叉判定处理的另一具体例。

图 15 的原理图示出了对壁状的构造物进行近似的形态。

图 16 的斜视图示出了纹理的活动的标准形态。

图 17 的斜视图示出了纹理的内插后的标准形态。

图 18 的概念图示出了用角色的仿样函数进行移动的轨迹。

图 19 是表示纹理的活动处理控制操作的流程图。

图 20 的模式图示出了在两个模型之间存在有高低差的状态。

图 21 的流程图示出了角色的高低差处理。

图 22 是用来说明该处理的时间流的流程图。

图 23 的模式图示出了进行了高低差处理后的状态。

图 24 的照片示出了类似人物的角色的一个例子的正面。

图 25 的照片示出了其第 2 个例子。

图 26 是控制键盘的扩大正视图。

图 27 示出了混合模拟的文件构成。

图 28 是用来对混合模拟进行展开的流程图。

图 29 的图面正视图显示出了在混合模拟处理过程中所显示的画面。

图 30 的图面正视图显示出了在同一处理过程中所显示的另一画面。

图 31 的图面正视图显示出了在同一处理过程中所显示的另一画面。

图 32 的图面正视图显示出了在同一处理过程中所显示的另一画面。

图 33 的图面正视图显示出了在同一处理过程中所显示的另一画面。

图 34 的流程图示出了实施例 2 中的主处理。

图 35 是进行了残像处理后的模型的正视图。

图 36 是残像处理的原理图。

图 37 是残像处理的示意流程图。

图 38 是其详细流程图。

图 39 是图 37 的流程图的详细流程图。

图 40 示出了已飞起来的飞行体的轨迹。

图 41 是说明飞行体的着地情况的说明图。

图 42 是说明飞行体的着地情况的说明图。

图 43 是说明飞行体的着地情况的说明图。

图 44 是说明飞行体的着地情况的说明图。

图 45 是说明飞溅情况的发生的说明图。

图 46 是发生飞溅之际的矢量线图。

图 47 是发飞溅之际的矢量线图。

图 48 是飞行体运动处理的流程图。

图 49 是已应用了飞行体运动处理的角色的背视图。

图 50 是该处理的示意流程图。

图 51 是检查风的发生的详细流程图。

图 52 是检查树叶的飞舞的详细流程图。

图 53 是模拟树叶的运动的详细流程图。

图 54 示出了飞行体的运动形态。

图 55 示出了飞行体的运动形态。

图 56 示出了进行自由下落运动的对象物的移动轨迹。

图 57 是不定形区划与模型之间的冲突判定处理的原理图。

图 58 是说明动作的流程图。

图 59 是纹理列的完成的原理图。

图 60 的正视图示出了用模型产生飞行体的飞溅和着地的状态。

图 61 的侧视图用来说明用于说明损耗回放处理的原理的角色的动作。

图 62 是从图 61 的上方看的平视图。

图 63 的侧视图用来说明处于损耗回放状态中的角色的动作。

图 64 是从图 63 的上方看的平视图。

图 65 的侧视图示出了已返回到本来的位置上的状态的角色。

图 66 是该处理的动作流程图。

图 67 是用不定形区划与模型逐渐的冲突判定处理来表现的模型的活动图。

图 68 说明了图 54 中的飞行体运动处理中的树叶因风使一片叶子的高度所受影响的系数。

优选实施例

以下，参照附图对作为图像处理装置的视频游戏机说明本发明的优

选实施例。

实施例 1

根据图 1～图 33 说明视频游戏机的实施例 1。

在图 1 中示出了本视频游戏机的框图。用该视频游戏机可以分别执行后述各个图像生成和处理 1～6。

该视频游戏机包括对整个装置加以控制的 CPU 块 10，进行游戏屏的显示控制的视频块 11，产生效果声音的音响块 12，读取 CD-ROM 的子系统 13 等构成。

CPU 块 10 由 SCU（System Control Unit，系统控制单元）100，主 CPU101，RAM102，ROM103，子 CPU104，和总线 105 等构成。该框图是本发明的图像生成装置的核心。

主 CPU101 内部具备 DSP（Digital Signal Processor,数字信号处理器），可以高速执行计算机程序。RAM102 存储由子系统 13 从 CD-ROM 读出的各种多边形数据，也可用作主 CPU101 的工作区。

在 ROM103 中，存放有在装置的初始状态中用于进行初始化处理的程序。SCU100 控制在总线 105、106、107 进行的数据传送。此外，SCU101 内部还具备 DMA 控制器，用于在游戏进行期间把必要的图像数据送往视频块 11 内的 VRAM。

控制键盘 2b，起着用户的输入装置的作用，具备操作所需的各种按键。子 CPU104 即所谓的 SMPC（System Manager & Peripheral Controller，系统管理器&外设控制器），具备根据主 CPU101 的请求从控制键盘 2b 收集外设数据的功能。

主 CPU101 根据由子 CPU104 转送来的外设数据进行在显示器上显示的图像的移动处理。子 CPU104 鉴别已连接到插座 2a（主机端子）上的外设机器的种类，并按照相应于鉴别后的外设机器的种类的通信方式收集外设数据。

视频块 11 起图形生成装置的作用，包括用于用多边形生成的图像

VDP（Video display Processor，视频显示处理器）120，和用作图像合成、阴面处理和背景限幅的 VDP130。它被连接到 VDP 和帧缓冲器 122、123 上。

当生成在显示器上显示的虚拟空间的图像时，显示所需要的多边形数据从主 CPU101 经 SCU100 被转送往 VDP120，并写入到 VRAM121 中去。已写入到 VRAM121 中的多边形数据作为含有每一象素 16 位或 8 位的色彩信息描画用数据被存放到描画用的帧缓冲器 122 或 123 中。被存放好了的描画用数据被送往 VDP130。主 CPU101 通过 SCP100 把控制描画的控制信息供给 VDP130。VDP130 根据该控制信息控制描画用数据。

VDP130 已被连接到 VRAM131 上，其具有或者使显示画面整体垂直地或水平地移动或者使之旋转的滚动功能，和决定多边形的显示顺序的优先度的功能（Z 分类或 Z 缓冲）。VDP130 通过存储器 132 把描画用数据向编码器 160 输出。已输出到编码器 160 的描画用数据被变换成视频信号的格式之后，进行 D/A 转换并在监视装置 5 上进行显示。在该监视装置 5 上根据该视频信号显示图像。

音响块 12 包括利用 PCM 方式或 FM 方式进行声音合成的 DSP140 和对 DSP140 进行控制的 CPU141。由 DSP141 生成的声音数据在用 D/A 转换器 170 变换成两声道信号后输出到两个扬声器 5a 和 5b 中去。

子系统 13 有 CD-ROM 驱动器等，并具有读入由记录媒体供给的应用软件。

下面解释用以上所说明的图像生成装置进行的处理。图 2 是一流程图，它示出了由 CPU 块 10 执行的主处理，顺序执行以下处理：用于使角色进行圆周运动的圆周运动处理 S200，虚拟摩擦处理 S202，在具有高低差的地面上的影子的显示（S204），就是说，角色的投影显示处理，交叉判定处理（S206），角色的活动处理控制（S208），角色的高低差处理（S210）。没必要的处理就不必进行而转移到其次的处理。以下详细说明各个处理的内容。

圆周运动处理

这一处理大体上内容如下。当在图示的左或右的方向上操作已经说过的控制键盘 2b 的十字状地形成的方向键 26b（请也参照图 26）时，由

该方向键操作的角色（勇士）30（参照图 3）就在虚拟空间内上下左右方向上移动。在圆周运动处理模式中，在游戏者所操作的勇士 30 与对方勇士 32 之间加上虚拟的向心力（朝向对方的角色的力），就可以实现游戏者一侧的勇士 30 自动地在对方一侧的勇士的周围转圈（圆周运动）的处理 30。

根据图 4 的流程图说明这一处理的内容。在步骤 400 中，判定是否已进入圆周运动处理模式（判定 1）。如果游戏者（比赛者）已执行了规定的操作，即视为提出了圆周运动要求。例如，已经按下了控制器（2b）的规定的按键。其次进行读入图 26 的方向键 26b 的图中所示的左右操作量等，检测游戏者一侧的勇士与对方勇士之间的位置，计算对方勇士相对于游戏者一侧的勇士所成的方向（步骤 402）。

接着，将该方向上的一角度给予游戏者一侧的勇士 30，且把朝向对方勇士的虚拟的向心力给予该勇士（步骤 404）。从物理上来考虑向心力的话，则向心力与作用到两个物体之间的引力等价。

若设对方勇士的 x 坐标和 z 坐标分别为（exp,ezp），设对方勇士的 x 坐标和 z 坐标分别为（mxp,mzp），则上述那样的对方勇士的方向可以用 arctan (exp－mxp, ezp－mzp）计算。此外，在进行该计算时，两方的勇士的高度方向的 y 方向的坐标，为方便起见没有考虑。

其结果是借助于向心力和由切向键产生的左右方向的速度进行在虚拟空间内的圆周运动（步骤 406）。借助于该圆周运动，游戏者的勇士 30 就可以朝向对方一侧的勇士 32 并在其周围转圆圈。图 5 是从虚拟空间的上方看到的该圆周运动的示意图，勇士 30 进行从状态 1 经由状态 2、状态 3 到状态 4 的圆周运动。上述的图 3 是从规定的视点显示的状态 1 的虚拟空间的图像，是将在上述的监视器 5（参照图 1）中显示的图像。图 6 是与状态 4 中所涉及的图 3 相同的图像。

因此，倘采用所说明的圆周运动处理，则由于对一个角色提供了虚拟的向心力，所以对于游戏者来说，只需要在图示的左右方向上操作图 26 的方向键 26b，就可以容易地把圆周运动给予角色 30。例如，就可象上述那样地简单地实现自己所操作的勇士 30 在对方一侧的勇士 32 的周围转圈的动作。

虚拟的摩擦处理

这一处理的内容是，在角色的动作中采用因地面的倾斜而滑动，和改变因地面的性质而加在角色上的摩擦力，且在角色进行移动的时候加上动摩擦，在角色未进行移动的时候加上静摩擦的办法，使角色的动作变成为更加多彩。

首先，图 7 示出了虚拟空间内的地面的剖面。如图 8 所示，在步骤 800 中，因地面的倾斜而滑动的量可以计算如下：

滑动量（dv/dt)=ivg

i：规定的常数

v：表示速度，是地面倾斜的法线单位矢量

g：重力加速度

其中，所谓法线单位矢量，指的是图 7（1）所示的相对于切面切线方向的法线矢量。当该法线单位矢量对于虚拟空间的坐标轴所形成的角度（θ）变大时，对于处于该地面上的角色提供更大使之沿着倾斜的地面向下滑的滑动量（加速度）。

在其次的步骤 802 中，判定角色（勇士）是否正在移动。例如，游戏者判定是否正在操作用于控制勇士的动作的方向键（判定 1）。

在角色正在动作时，就加上动摩擦，在角色尚未动作的时候，就加上静摩擦（步骤 804，步骤 806）。摩擦（= d v/ d t）本身可以计算如下。

$$M\cdot(dv/dt)=\mu\cdot S\cdot M(g\cdot\cos\theta+(dv'/dt)\cdot\sin\theta)$$

$$dv/dt=T\cdot g\cdot\cos\theta+T\cdot(dv'/dt)\cdot\sin\theta$$

其中 M：重量

v：摩擦速度（计算后的物体的速度）

v′：物体的速度

μ：摩擦系数

S：角色与地面接触面积（常数）

g：重力加速度（常数）

θ：地面的角度

$T=\mu\cdot S$

动摩擦和静摩擦采用对用上式得到的摩擦（或摩擦系数）加上规定的处理的办法决定。但是，静态摩擦的值比动态摩擦的值大。这一摩擦因地面角度而变，地面的倾斜越大则摩擦的值越小。

其次，决定地面的状态。就是说，判定角色所接触的地面，例如，是水场还是砂场。这些属性都事先虚拟性地赋予了每一地面。例如，在砂场的情况下，使已计算好了的摩擦量变成 2 倍，在水场的情况下，则使摩擦量变成为 3 倍。接着，使该摩擦力反映到原先计算好了的滑动量上，使之减少（步骤 810）。

由于借助于这一处理，总是把静止摩擦或动摩擦加到角色上，所以停止不动的角色就不容易起动，一旦开始动起来就将变的易于移动。此外，还可以用地面的属性来改变角色的移动特性。采用上述这些事项，就可以更真实更为丰富地表现角色的移动了。

在有高低差的地面上显示影子

如图 9 中所具有的那样，该处理的内容是，在角色所站立的地面 90 上有凹凸 92 的情况下，对地面加上凹凸的同时用更简单的计算来显示角色的影子（投影象）。图 9 示出了该处理的概要。图中的 94 是高度为 0 的虚拟的基准线，而 96 是位于无限远处的平行光源。

该人物由头、胸、腹、腰、足等的集合构成。各要素都由多个多边形构成，图面的 yG 是从基准线到角色所站立的地面处的距离，y1…yn 是每一要素的从地面到基准线为止的距离。这种处理，可以用下述计算式计算。

$$E'(x',y',z') = Ms \cdot E(x,y,z)$$

E′(x′,y′,z′)：影子的坐标点

E(x,y,z)：人物（角色）的坐标点

Ms：是用来把影子的坐标变换成世界坐标系的矩阵（矩阵表达式）。该 Ms 可以用下式给出。

$$Ms = Mu \cdot (-Tyg) \cdot Pn \cdot Tyn \cdot (-Tsn) \cdot Rn \cdot Tsn$$

Mu：用来把人物的坐标变换成图 9 的世界坐标系的变换矩阵。

Tyg：用于使人物从所站立的地面平行移动到基准线为止的平行移动矩阵。

P n：用于把人物投影到基准线上的变换矩阵。

Tyn：用于使斜向投影的的影子平行移动到各个客体的接地点（即，各个客体所处的地面）的矩阵。

Tsn：用于使各个客体对于世界坐标系的原点平行移动的平行移动矩阵。

Rn：用于使影子的多边形与地面的倾斜象吻合地进行旋转的矩阵。

这一处理，用下述过程进行。

处理步骤 1；用来形成影子的人物角色 91 从地面上平行移动到高度为 0 的基准线为止，接着，

处理步骤 2：将此人物多边形集合体 91，用上述平行光源 96 斜向投影到基准线 94 上，接着，

处理步骤 3：使斜向投影后的要素分别对各个客体以 y1…yn 的幅度平行移动到地面，接着，

处理步骤 4：使每个要素从世界坐标系的位置平行移动到其坐标系的原点，接着，

处理步骤 5：根据各自的地面角度，使各个客体旋转，最后，

处理步骤 6：进行平行移动，使用过程 4 而移动到原点的客体返回原来的坐标系。

因此，倘采用该处理，则在地面 90 存在着凹凸的情况下，就可以与该凹凸相吻合地简单地在加上凹凸的地面上显示角色的投影象。

交叉判定处理

图 10 的 xz 平面图示出了捕捉 2 个角色，例如，互相对面的勇士（人物 1，人物 2）的虚拟摄象机的视野区域。用该虚拟摄象机（视点）来显示上述的图 3 所示的那样的图像。在该处理中，除勇士之外的墙壁，房子之类的能将视野区域遮住的虚拟的构造物已与视野重叠起来的情况下，使得该构造物的一部分和全体都不显示。在图 10 中，构造物 100、102 对于从视点开始到人物为止的视野区域进行遮挡，但构造物 104 虽然处于由视点所显示的区域中，但是，对前者的区域却无妨碍。在这里进行消去构造物 102 和 104 的图像生成或图像处理。

图 11 的流程图示出了这一处理。在进行这一处理之际，由于可以更为迅速且容易地进行这些构造物是否已与视野区域重叠的判定，所以，

在 xz 坐标面上进行判定，且已具有厚度的物体，可以用圆来近似。图 12 的 xz 平面图示出了用内切圆近似该构造物的状态。

在图 11 中，步骤 110 是判定该近似圆是否已重叠到上述视野的区域上的处理。图 13 在该步骤中的判定的概念图。对构成相当于从视点到人物 1、2 为止的区域的三角形的区域的全部的边根据以下的处理，进行中心点为 T 的具有半径 d 的某一近似圆是否重叠的判定。此外，在以下的说明中，矢量 L、R、T 是从规定的原点向着 L、R、T 形成的矢量。还有，对于三角形的各边的各个顶点按顺序在顺时针方向上设定 L 点、R 点。

[公式 1]

$$r = R - L$$

$$t = T - L$$

$$p = c + L - T$$

$$cx = rx \cdot |c| / |r|, cy = ry \cdot |c| / |c|$$

其中，

[公式 2]

$$|c| = |t| \cos\theta$$

cx：矢量 c 的 x 分量

cy：矢量 c 的 y 分量

rx：矢量 r 的 x 分量

ry：矢量 r 的 y 分量

根据内积定理

[公式 3]

$$|r| = |t| \cos\theta = rx\,tx + rz\,tz$$

$$|c| / |r| = \{rx\,tx + rz\,tz\} / |r|^2 = \{rx\,tx + rz\,tz\} / (rx^2 - rz^2)$$

其中，

[公式 4]

$$0 \le |c| / |r| \le 1$$

在近似圆的中心 T 位于直线 TR 的正区域的情况下（在 T 位于从三角形的内侧往外的情况下），

[公式 5]

若 $|p| - d > 0$

则圆对于三角形的内侧的区域是独立的，就是说该区域中无重叠，

[公式 6]

在 $|c| / |r| < 0$ 或 $1 < |c| / |r|$

的时候，若点 T 位于直线 LR 的正区域，则圆对区域是独立的。

在步骤 110 中，在近似圆已重叠到三角形的区域内的情况下，就进行不显示该构造物的处理（步骤 111），在已重叠的情况下，就向下一步骤（步骤 112）前进。其次的处理的内容是，对该构造物在一定方向上具有长度的墙壁之类的物体进行重叠判定。该处理如图 14 所示，可如下述进行。

当构造物近似为直线 Ts 时，判定该直线与三角形的区域之间的交叉。对构成三角形的区域的所有的边进行以下的处理。设判定直线 LR 和任意的点 P 之间的位置关系的函数为 F1（P），

[公式 7]

$$F1(P) = a1\,Px + b\,Pz + c1$$

$$a1 = Lz - Rz$$

$$b1 = Rx - Lx$$

$$c1 = LxRz - RxLz$$

设判定直线 ST 和任意的点 P 之间的关系的函数为 F2（P），

[公式 8]

$$F2(P) = a2\,Px + b\,Pz + c2$$

$$a2 = Lz - Tz$$

$$b2 = Tx - Sx$$

$$c2 = Sx\,Tz - Tx\,Sz$$

若 F1（S）· F1（S）为 0 且 F2（S）· F2（S）为 0，则直线 TS 被判定为与直线 LR 已交叉。

在该交叉被否定了的情况下，就转移到步骤 114 判定近似圆的重叠的有无。其理由如下。在上述的直线已完全地进入到三角形的区域中去了的情况下，重叠判定被否定而显示壁状的构造物。于是，就把相当于图 15 的那样的壁的厚度的小圆 150 给予该直线 TS，并对该小圆进行与步骤 110 相同的重叠判定。其结果是，即便是在直线 TS 完全进入到三角形的区域中去，在步骤 112 的判定中已判定为没有重叠的情况下，在该近似圆已进入了三角形的区域内的情况下，也将被判定为有重叠而转移到步骤 111 中去。

借助于这些处理，就可以在向着已成为显示对象的视野区域上不用

虚拟的摄象机显示墙壁、栅和围墙等的构造物。接着，如图 10 所示，在该区域上已经显示出了该墙壁等的构造物的 104 之类的情况下，就保持原样不变地显示该构造物。倘采用以上的处理，就可以满意地显示角色而不会使多余的构造物遮挡角色。当然，在未设定作为显示对象的角色的情况下，这些构造物则可以照原样进行显示，此外，作为近似圆也可以不用图 12 的情况下的内切圆而改用外切圆。

角色的活动处理控制

该处理的内容是，当希望连续地显示一个动作的一连串效果时，对之进行内插使得一个动作的结束位置与目标点相吻合。

图 16 示出了一个角色的活动的一连串的过程，类似人物的角色（勇士）160 借助于游戏者对键盘 2b 的操作，向着对方勇士反复地用胳膊 160A 进行的殴打的过程。

该胳膊的动作，用众所周知的仿样（spline)函数（特别是 3 次仿样函数）进行计算并显示在监视器上。现在，设该胳膊 160A 正向着对方一侧的勇士的胸部移动的情况（例如，胳膊快速弯曲的情况）下，对游戏者来说，要把正在移动着的胳膊的轨迹快速地如图 17 所示改变到下方，是非常困难的，另外，若企图仅仅在 CPU 块 10 的一侧进行上述动作，则胳膊的动作就易于变成为不自然。

于是，在现在正在执行中的活动的仿样轨迹变化过程的终点附近设以连接区域，在该连接区域上，进行给予新的仿样函数等的内插，使得当初的仿样轨迹的终点变成为另一位置的目标点。

用附图对此进行说明。如图 18（1）所示，位于图 16 的勇士的胳膊的顶端的拳头 108A，将沿着由仿样函数决定的轨迹 180 移动。图 18（2）示出了用该仿样函数得到的轨迹 180，拳头 180A 就从该估计的起点进行移动直到终点，之后回到起点。

现在，假定手指 180A 正在从仿样函数轨迹 180 的起点向着终点进行移动。在该过程中，假定，例如，因对方一侧的勇士的举动变化等产生了把当初的终点变更为目标点的必要。

这样一来，拳头位置 180A 位于连接区内时，CPU 块 10 校正该区域

的函数，以使拳头停在目标点。这一内插，例如，可以如后述那样地进行。当然，也可以采用使连接区域的仿样函数或系数等进行适当的变化。其结果是，如图 16 所示的活动自动的内插为图 17 的活动，连接区域中的活动也得以更为圆滑地表现。此外，为便于进行图示，在图 18（2）中，纵轴仅仅示出了 y 轴方向的移动量。就象 3 维视频游戏的一般情况那样，x 轴定为面对监视器的画面的横方向，y 轴则定为其高度方向，z 轴定为其纵深方向，就是说与画面相垂直的方向。

连接区域（连接时间）由具备结束帧的多个帧群适当地设定。此外，对于拳头以外的胳膊上部和胳膊下部的各个部分，也可以用同样的处理以完成整个胳膊的内插后的活动。连接区域中的内插后的坐标可用下述的特性式子来设定。

当前坐标+（当前坐标－目标坐标）•（当前时间/连接时间）

图 19 是由于说明上述的处理的流程图，在步骤 190 中，游戏者操作键盘 26b 并判定是否已发生了活动移动的指令。在步骤 192 中，判定目标点是否已设定于与终点不同的坐标上。在步骤 194 中，再现被选择的活动。在步骤 196 中，在该判定是肯定的情况下，就在计算连接时间的同时，转移到步骤 194，再现该连接时间的内插后的活动。在步骤 198 中，判定现在的帧是否一连串的活动的结束帧，在是结束帧的情况下，就返回主程序，在尚未到结束帧的情况下，就返回步骤 192。

在这里，构成连接时间的连接帧，是活动的最大帧数（从图 18（2）的起始帧到结束帧为止的帧数）的 0．1，在连接帧超过 10 的情况下，就固定为 10。连接帧只要在大于 1 且在最大帧数的 1/2 的范围的话就可以。若连接帧过长，则活动所具有的原来的动作就会丢掉，另一方面，若过短，则内插将变得陡峻，动作将失去圆滑性。

倘采用该处理，则可以得到如下效果。当象视频游戏装置那样，为了迅速地操作在游戏画面上登台的勇士等的角色而操作控制器时，就可以连续地再现角色的多彩的动作。这时，在正在再现一连串的活动的时刻，即便是企图与对方的角色的快速的动作相协调地变化其活动，只要活动本身是固定的活动，那就是困难的。但是，如已经说过的那样，倘对活动进行内插，则结果极变成为可以多彩地再现与例如对方一侧的角色的动作相协调的活动，使活动变成为现实感更为丰富的活动。通常，

用户进行这样的内插是很困难的，最重要的是，要是用现有的视频游戏装置的话就不能构成为使之进行这样的内插。

此外，本处理虽然是以用仿样函数对活动进行计算的情况为例进行的说明，但该处理也可以应用到可以顺次再现预定的图形的图形改变的方式中去。在这种情况下，在连接区域中，只要使之再现修正后的图形即可。

角色的高低差处理

本处理的内容是，在角色（勇士）和地面之间有高低差 (level difference)时，对该高低差进行修正以显示所希望的图像。例如，如图 20 所示，在互相面对面的两个勇士 200 和 202 之间存在着高低差的情况下，从勇士 200 向位于高的位置处的勇士水平地反复出拳是不现实的。所以，本处理使得对该高低差进行修正，使勇士 200 的攻击朝向较高的方向（参照图 23）。

本处理，如图 21 所示，在步骤 2100 中，判定是否有必要修正该高低差。在该判定中，定为“有必要”的是这样的情况，例如如图所示，对于面对面的勇士来说，从一方的勇士对另一方的勇士，发生了拳打脚踢的攻击要求（指令）。

其次，在该判定是否定的情况下，就返回主程序。另一方面，在该判定是肯定的情况下，再在步骤 2102 中进行对方一侧的勇士是否站在地面上，即，角色与地面之间的冲突的判定，在这也是肯定的情况下，因需要进行角色的高低差处理而转移到下一步骤。在该判定为否定的情况下，程序结束。

在下一步骤 2104 中，计算从游戏者本身所操作的角色所站立的地面到该角色为止的距离。对于对方一侧的角色也进行同样的计算。其次，在步骤 2106 中，将此结果与规定值进行比较，在超过了规定值的范围的情况下，就在步骤 2108 中，进行使有关上述距离的差收缩于该范围之内的处理。就是说，设一方的游戏者所操作的勇士的高度为(my)与另一方的游戏者所操作的或图像处理装置本身用规定的程序自动地操作的勇士的高度为(ey)，则两者的差(d iff1)将变成为（my -ey），例如对(d iff1)进行是否处于下述范围内的判定：$-0.15 \leq d\ iff\ 1 \leq 0.15$，在(d iff1)已超过了该范围的情况下，比该范围小的 diff1，就把它定为-0.15，比该范

围大的，则将之定为 0．15，在该范围内的情况下，就把 diff1 照原样不动地确定为(d iff2)。至于象这样地对 diff1 进行修正的理由将在下边说明。

在下一步骤 2110 中，用发生攻击力的时间（帧数）除 d iff2，以其结果作为“结果 1”，用从发生攻击力到解除对抗为止的时间除 diff2，把其结果定为“结果 2”。此外，用图 22 进行说明。图 22 示出了攻击动作（勇士伸脚踢，出拳，跳跃等）的动作，对于 1，2…5 的每一时间的帧的流动（时间的流逝）如下所述。

1：动作的开始（第 0 帧）

2：攻击力的发生（第 1～3 帧）

3：攻击力的消灭（第 3～4 帧）

4：对抗时间（这是不能接受其它的攻击动作的指令的时间，可以是第 0～4 帧）

5：动作的结束（第 0～5 帧）

因此，上述的“结果 1”将变为(d iff2/2)，而“结果 2”则变为(d iffi2 / (4－2))。这些值当然要设定和在规定的工作 RAM 区域中。示于图 22 的特性对于每一动作都已预先定好了。结果 1 是相当于攻击力发生之前的每一帧的变化率的值，结果 2 是相当于对抗被解除为止的每一帧的变化率的值。

接下来，在步骤 2112 中，决定在对每一所要求的动作进行时的手和脚等的移动的坐标。该坐标值已经以对对方一侧的角色没有高度之差为前提而预先定好了。把该坐标（调整垫，leaf）定为“结果 3”。当用一函数 fat 执行一个动作以返回到调整垫 leaf 时，可根据动作时的手或脚的空间坐标位置来决定调整垫。在下一步骤之后的步骤是说明执行攻击动作时的处理的步骤。

在步骤 2114 中，检测从动作发生之后当前所显示的帧相当于第多少帧。在其次的步骤 2116 中，判定该帧是否发生攻击力之前的帧。在图中，到帧 1 为止，判定为攻击力发生之前的帧。

其次，在步骤 2116 中的判定结果是肯定的情况下，就在步骤 2118 中，使上述“结果 1”与帧数相乘，把其运算结果作为差加到上述步骤 2112 的调整垫上。接着，用其结果，用众所周知的逆运动学（inverse kinematic)

的技术（例如，‘机器人的力学和控制（ロボットの力学と制御）’，系统控制信息学会编，有本卓著，朝仓书店发行，5•2 逆运动学（132 页～）），根据对调整垫加上的差的值等重新计算勇士的身体的形状。

在上述的步骤 2116 中，在已判定为是已发生了攻击力之后的情况下，就在步骤 2200 中，从解除对抗的帧数（图中的“4”）中减去现在的帧数（图的“2 或 3”），该处理进行到给结果 2 乘上这一帧数的步骤中去。

其结果是，用步骤 2118，如图 23 所示，从自己所操作的勇士 200 向对方的勇士 202，向已对勇士的高低差进行了补偿的高度伸出拳头，在拳头命中后或即便是没有命中，之后也要边对作为拳头而伸出的胳膊进行高低差补偿边返回原来的位置。

倘采用在这里所说明的处理，则由于执行反映了 2 个勇士的高低差的处理，故从一个勇士发出的攻击（拳头，脚踢）向着处于高的位置上的勇士进行，所以可以提供符合实际的攻击的现状的图像。因此，即便是在存储器中事前没存储下对处于高的位置的勇士的活动，也可以在已反映了角色件的高低差的状态下，实现一个角色与另一角色之间的图像处理。

此外，之所以对 d iff1 进行修正，是因为如下的理由。若不进行这样的修正，假如在两方的勇士间有大的高低差，则从一方的勇士向对方的勇士的次击，说得极端一点的话，将变成为向着对地面呈直角的方向，反而会变成为不自然的缘故。

此外，在已说明过的各个处理的说明中，虽然是以视频游戏为例进行的说明，但并不受限于此。在图 1 中，也可以不用 CD-ROM 而代之以用 ROM 盒式磁盘。这些物品起着已存储下本发明的图像生成装置的动作程序的存储媒体的作用。 此外，在交叉判定处理中，定为除完全不显示已说过的构造物的情况之外，用网状多边形或多孔物体（polyline）等显示该构造物也包含于“不显示”的形态之内。还有，图 24 和图 25，分别是作为本发明的模型的类似人物的角色（勇士）的一个例子。

至于用 CPU 块 10 进行的辉光黑点（glow shading）（其它的黑点也一样），它不仅可以实施在角色的所有多边形上，也可以仅仅对角色的一部分，特别是对颜色进行线性内插表现出立体性的部分（例如，相当于

已露了出来的皮肤的多边形）进行处理。借助于这样的处理，降低给予CPU 块 10 上的处理执行上的负荷就成为可能。在图 24 的角色中，仅对从衣服中露了出来的手指、脸、以及脖子下边的胸部进行了辉光黑点处理，在图 25 中，也可以对几乎上半身和腿部进行辉光黑点处理。

此外，在对各个多边形加上了 Z 分类的情况下，在四角形的多边形中，虽然以往利用 4 个顶点，但是也可以仅仅用分别连接两个点的对角线的交点（中点）进行判定。其结果是，可以减少主 CPU 存取存储器的频率以改善处理速度。再有，即便是利用中点加上 Z 分类，由于该中点是 4 个顶点的平均值，故可以不加变动的维持用了 4 个顶点的 Z 分类的精度。在三角形的多边形的情况下，可以利用重心。

其次，对上述的图像生成装置的处理动作的另一实施例进行说明。图 26 的正视图示出了上述控制键盘 2b 的细节，该控制键盘具有已经说过的方向键 26b 和 A、B、C、X、Y、Z、L、R 等各个按键。这些方向键或按键的按下，与图 24 或图 25 的角色（勇士）的动作相对应，例如，A 按键相当于“对来自对方的勇士的攻击的防御行动”，B 按键相当于“对方的勇士伸出的拳头”，C 按键则相当于“向对方的勇士伸出拳头”。本身为用户的游戏者，虽然想多彩的操作这些键和按键来操作角色使之满足自己的希望，但是要想正确地操作极其迅速而多彩地动作的勇士，还需要非常的熟练才行。

所以，在这里要说明的处理形态中，如图 27 所示，把本身为按下多个按键功能的组合的指令动作的集合作为混合动作，并以此作为文件设定于工作 RAM 中。然后，用按下上述按键之一的办法对之进行读出，使勇士顺次作出指令动作。在图 27 中，象 PPK 这样地产生的，与顺次按下了 B 键，B 键，C 键的情况相等，而象 P+K 这样的产生的，则与同时按下了 B 键和 C 键的情况相等。此外，对于各个动作已变成为可以设定该动作所占的帧数。因此，由于游戏者可以自由地设定混合动作中的各个指令动作的帧数，所以可以多彩地作出自己所希望的混合动作。作为该混合动作的具体例子，有“先把对方一侧的勇士摔倒，待他刚起来时就踢”等等。

如果说得再详细一点，“P、K、P”表示连续地按下拳击键、脚踢键、拳击键，而且，在已判定这些已经连续时，若假定进行摔跤，则在已按下 P 键时角色就进入拳击的活动。从该活动的开始到例如活动变成为返

回拳击为止的期间，如果假定按下了 K 键，则在从已伸出了拳头的姿势到变成为收回拳头的活动为止的时间内开始进行脚踢，在脚踢的活动已作完的期间再次输入 P， 则判断为连续动作，连续地进行摔跤的活动。如果把这时的定时和在先前的脚踢中摔倒的角色的定时合在一起，则即便是相同的摔跤，也可以作成为易于发挥动作等非常有效的摔跤。

游戏者可以制作多个这样的混合动作文件并给各个文件分配一个按键。当然，若按下没有分配该混合动作文件的按键，就将出现原来已分配给该按键的单独的动作。把混合动作登录到何种范围的按键上去，应根据混合动作的必要性或单独动作的必要性恰当地决定。各个文件中所分配的按键，也可以是多个。

根据示于图 28 的流程图说明其次的本处理的动作。首先，在步骤 S280 中，判定上述的按键是否已按下。在按键已被按下的情况下，在对于被按下的按键已登录下混合动作的情况下（步骤 S282，肯定），就在步骤 S284 中，按顺序读出已登录在工作 RAM 中的混合动作中的各个指令（与多个或单独的按见的按下相等）。接着，在步骤 S286 中，判定文件中的指令是否执行到最后，在这一判定是肯定的情况下，就直到结束为止反复地执行该流程。

另一方面，在步骤 S280 中，在没有按键被按下的情况下，就返回，还有，在步骤 S282 中，在已按下的按键未登录在混合动作的文件中的情况下，就执行已分配给该按键的单独动作。

倘采用这里所说明的实施例，则把对角色（模型）的动作指令的组合（例：按下动作开关的组合、一个开关的按下、多个开关的按下、顺次按下这些开关、或同时按下这些开关）预先存储到存储器中，并把它做成为用简单的开关操作，例如，按下一个按键的办法，使图像处理装置进行读出，并根据所读出的指令群连续地控制模型的动作（活动）。因此，如上所述，用户就可以表现无须进行复杂的键操作且可以适合自己的爱好的更为多彩的模型的动作（活动）。

还有，图 29 是用来在这里所说明的混合处理时在监视器上显示的模式选择的初始画面，为要进行混合动作的选择要选择上方右端的画面。图 30 是示出了已把按键（键）分配给了动作的画面。图 31 是用来作成上述混合动作文件的画面，图 32 的画面则示出了彼此的勇士出于格斗状

态的一个状态。图 33 是在该一个状态的画面上进一步在后边展开的画面，是勇士“あきら（阿基拉）”把勇士“ウラ（乌拉）”摔倒了的状态的画面。从图 33 可知，在一方的勇士把另一方的勇士摔倒了的时候，（在 CPU 块一侧这样判定的时候），画面中的角色 330 就变成为对此进行显示。在图 33 中，采用使“带花蕾的类似植物的角色”变更为“已开了花的类似植物的角色”的办法，来显示这一情况。

实施例 2

接着，根据图 34～图 68 说明作为图像处理装置的视频游戏机的实施例 2。该实施例 2 中的视频游戏机的硬件构成与上述的实施例 1 相同，故免予说明。

图 34 的流程图示出了由 CPU 块 10 执行的主处理，执行后述的各个处理。S200 是残像处理程序，S202 是飞扬处理的程序（飞行体的处理），S204 是模型的自由下落的处理，S206 是不规则区域与模型的冲突判定处理，S208 进行回放速度的变更处理。各个处理都反复执行。以下，对各个处理进行说明。

残像处理

图 35 示出了对于作为模型的勇士适用的残像处理的原理。在图 35 中，30 是勇士，对于勇士的脚 30b 进行残像显示。就是说，在显示现在的帧（一个场面）的实像的同时，同时显示以前的帧的残像 1～4。

图 36 示出了该残像处理的原理。多边形数据文件（或参数文件）存储了规定该勇士的脚的“环绕式脚踢”活动的参数。多边形数据文件#1～#n 示出的是距残像开始点的帧数。

在多边形数据文件＃1 中，存储有与残像 4 相当的模型＃1。因此，在该时刻，勇士的模型 #1 作为实像被显示在画面上。在下一时刻，把与残像 3 相当的模型 #2 存储到多边形数据文件#2 中。以后用同样的处理，在画面上显示勇士的脚模型 #3…# n。在图 4 的第 4 帧的情况下（ # 4），作为脚的实像模型顺次显示 #1～# 4。

作为残像用缓冲器，设定有 B#1、B#2、B#3 和 B#4。在这些缓冲器中，顺次记录有已记录到多边形数据文件中的勇士的活动数据。在图 36 的下栏里边示出了这一情况。已记录在多边形数据文件#1 中的模型#1 的

数据，在残像开始后记录在残像用的缓冲器#1 中，在其次的时刻已记录在多边形数据文件#2 中的模型#2 的数据被记录在残像用缓冲器#1 中，残像用缓冲器#1 的内容被送往残像用缓冲器#2。就是说，残像用缓冲器的数据被以#1→#2→#3→#4 的顺序送出，从多边形数据文件中把前一帧中的实像的活动数据送往残像用缓冲器#1 中。残像缓冲器#4 顺次照原样更新。

例如，在第 5 帧的时刻，把模型#5 的活动数据存放在多边形数据文件中，而且，作为实像显示该模型。另一方面，把模型#4 的活动数据存储到残像用缓冲器#1 中，把模型#3 的活动数据存储到残像用缓冲器#2 中，把模型#2 的活动数据存储到残像用缓冲器#3 中，把模型#1 的活动数据存储到残像用缓冲器#4 中。

多边形数据文件的活动数据与残像用缓冲器的数据（与过去计算出来的活动数据相等）同时进行显示，所以在画面上结果就变成为同时进行显示。图 35 示出了这一情况。对于实像来说，施以通常的粉刷（rendering）进行显示，对此，对于残像来说，为了提高演出效果，换句话说，为了使之“看起来象是残像”要边施行半透明的处理边进行粉刷。可以把该半透明的程度提高到更往前的时刻的帧数中的残层那么高。象前一时刻的残像那种程度透明地进行显示。残像用缓冲器被设定于 RAM102 中。

其次，对本实施例中的残像处理的动作进行说明。图 37 是进行这一处理的示意流程图。首先，判断残像是否是必须的（S500）。该判断，例如，对每一活动进行。例如，在伴随着勇士的大的动作（环绕式脚踢，背摔等）中残像处理认为有必要。请参照图 35。除此之外，与动作量超过规定值是一样的。

图 38 是该残像必要性处理的详细流程图。首先，检查动作的发生（S600）。这种检查由上述控制器 2b 的操作按键或操作杆的操作状况来判定。在已判定为已经发生时，就读入动作的属性或数据（S602）。所谓动作的属性，指的是对于每一动作所赋予的与性质有关的数据，例如，“攻击的动作”，“利用脚的动作”，“利用手的动作”等等。

其次，判定该属性数据是否是应该发生残像的数据（S604）。例如，在是“攻击的动作”而且是“利用脚的动作”的情况下，作为必须进行

残像表现，建立规定的标志“F”并设为“1”（S606），在其它的情况下，把标志设为“0”（S608）。

在示意流程图的下一个步骤中，指定需要残像的部位（S502）。这一指定用于本身为详细流程图的图 39 的处理来进行。首先，在 S700 中，检查是否已发生了攻击指令。所谓攻击指令是操作按键或操作杆的操作，在游戏者所操作的勇士对交战对方的勇士达到了进行攻击之类的命令形态的状态下发生的指令。

在图 39 中，例如，在已发生了一次脚踢的指令的情况下（图 35 的情况相当于此），参照攻击部位是脚脖子的部分（图 35 的 30A），接着，为了形成脚全体的残像，脚脖子 30A、下肢 30B、大腿部分 30A、就是说左侧的整个脚（参照图 35）被特定为需要残像的部位（S704）。此外，如这里的说明所示，勇士为多要素（part）构成，而各个要素又分别由多边形构成。

当然，如本实施例的情况所示，在没有发生攻击指令的情况下，也可以使之显示残像（S706）。在这种情况下，当然，在图 37 的 S500 中的“需要残像吗？”的判定中，对残像的必要性进行判定，使得即便不是“现在正在攻击中”也可以进行残像显示。例如，有这样的情况：边伴以残像边显示勇士被施以动作而倒下的过程。在这种情况下，由活动数据，象 S702、S704 那样地决定需要残像的部分。

接下来，在图 37 的 S504 中，执行在图 35 中已说明过的残像显示，到 S506，判定残像显示是否已结束。例如，判定一连串的活动是否已结束，在图 35 的例子中，判定一次脚踢是否已结束，在判定为已结束了的情况下，残像显示就结束了。

倘采用本实施例，则把具有勇士的一连串的动作的数据的多边形文件的数据不加改动地顺次记录于残像用缓冲器中。在该多边形文件中，把进行了坐标变换，进行了限幅后的多边形的各个顶点的坐标加上在各个点上的法线矢量进行记录。由于仅仅把该多边形文件的数据照原样不变地记录于残像缓冲器中，并使之与现在的实像一起输出，所以可以减轻给予 CPU 块的计算负荷，而不必象现有技术那样每次都对残像用的多边形进行坐标变换（标本化）。而且，在脚（部分）的过去的位置上显示残像，且加上了半透明的演出，所以可以演出感丰富地显示残像表现。

还有，也可以不用半明的演出而代之以用网眼状或者用线来表现脚多边形。借助于半透明的处理，得以同时显示脚的残像和背景。

倘采用本实施例，由于借助于残像的有无，并介以游戏者的视觉，可以区分带残像的那样的大的动作和不带残像的中小动作，所以将提高游戏的演出感（作为游戏的图像表现的多样化、好玩劲）。

飞行体运动处理

其次，对飞行体（设置物）的飞舞处理进行说明。该处理，如后所述，具有使地面的砂子，或者水，或落在地面上的树叶飞舞的形态。首先，对前两者进行说明。其概要如下。

如图 35 所示，在采用勇士 30 用脚向上踢的活动的情况下，或者在勇士从空中着地的情况下，要进行水，砂飞舞的处理。为了提高游戏的演出效果，就要执行这样的处理。特别是在格斗游戏的游戏者想激烈地操作按键以迅速地操作角色的游戏中，由于角色的动作多彩而迅速，所以需要使水等的飞舞与角色的动作吻合起来迅速而准确地计算后进行表现。以下要说明的处理使这样的事情成为可能。

首先，对砂、水的飞舞处理进行说明。在这里，所谓飞舞处理，指的是，如图 40 所示，进行图形改变的多边形（总共 32 个图形）顺次在用计算所求到的轨道上运动。该图形模拟水或砂子顺次进行扩散的状态。如果对砂子或水的一粒一粒进行计算，则要花时间，所以把数粒汇集在一起边进行图形改变边使之运动。

在该处理中，进行勇士的要素检查。上述勇士由头、右手、（上胳膊、下胳膊，手腕）、左手、胸、腹、腰、左脚（大腿部分、下肢、脚脖子）、右脚的各个要素构成。要素检查对脚、头、腰进行。因此，例如，模拟用脚往上踢，从头或腰进行的下落。在要素检查中，包含要素移动量的检查。

此外，还检查虚拟地面的属性标志、要素位置标志。虚拟地面属性标志(b water)被设定为“1”（水的属性）或“0（砂子的属性）”，和要素位置标志（b air）被设定为“1（着地状态）”或“0（空中状态）”。在（b water）为“1（水的属性)”的情况下，水将进行飞溅，在“0（砂子的属性）”的情况下，则进行砂子的飞溅。水或砂子的飞溅中有以下的形态。

着地情况发生：这是要素已在地面和水面上着地（着水）的情况，砂子或水从发生点向四方扩展开来。

上踢发生：在要素正在地面（水面）上着地（着水）的状态下产生，由于脚的往上踢，使砂子或水在脚的移动方向上飞溅起来。此外，在图 60 模式性地示出了飞溅情况发生和着地情况发生。图中 280 是角色（勇士），角色的脚往上踢，使砂子或水 282 相应于脚的特性而往上飞溅。或者，着地以后水或砂子相应于着地动作向四方跳起来。

在该处理中，执行称作继续处理，其持续发生着地或上踢的数个中断（数个场面或数帧）。通常，在这些的发生时，决定飞溅数并设定其 1/4。设飞溅数为 N，结果就变成为每进行一次中断飞溅数就如下所述那样地减少下去。

若设飞溅数为 N，则

第 1 次：1/4N，N′=N-（1/4）• N

第 2 次：1/4N′，N″=N′-（1/4）• N′

第 3 次：1/4N″

即，每一次中断，就每次设定飞溅数的余数的 1/4。飞溅的形态如上所述，结果变成为改变图形。

对着地情况发生的处理进行说明。检查要素的前一次（前一次中断）的位置设为（OX,Oy,OZ），并设本次的位置为（PX,Py,PZ)。设地面或水面的高度（Y 坐标）为 epox。

要素的基本速度：SPDX=PX-OX　X 方向

SPDy=Oy-Py　Y 方向

SPDZ=PZ-OZ　Z 方向

飞溅总数（设定数）等于基本设定数 • 要素移动量

飞溅原点：图 41 是用虚拟空间的高度方向表示的模式图。在检查要素正在向箭头方向移动时，如图 42 所示，水、砂子的飞溅原点被设定为（OX,epos,OZ)。

砂子（水滴）的发散处理：抽出检查要素的基本速度的 X，Z 分量。

图 43 是已虚拟地设定于图像生成装置内的模型空间的 X-Z 平面。此外，该图示出的是从斜向上方的位置看的状态。使这一 X，Z 分量的矢量在 XZ 平面上在+135°到－135°的范围内随机地旋转。在图中，被圆包围起来的范围（阴影部分除外）是砂子（或水滴）进行扩散的方向。

现在，设随机旋转的矢量为 A。其次，使该矢量在矢量的方向上移动 L 并设为 A′。其中，‘L’是要素的平均半径。例如，在脚的情况下，被设定为 10cm，在腰的情况下，被设定为 20cm，在头的情况下，被设定为 15cm。对该 A′加上基本速度的 Y 分量以最后决定砂子、水滴的飞溅速度。在该速度被决定之后，用图 40 的轨迹边改变砂子，水滴等的图形边使之移动。这样一来，就如图 44 所示，就模拟了从要素的周围（半径为 L）砂子（水滴）向四方飞散的状态。

其次，对飞溅情况发生处理进行说明。如与图 41 对应的图 45 所示，设处于水中或砂子中的脚往上踢。在进行这样的动作的情况下，设检查要素的基本速度为（PX-OX,Py-Oy,PZ-OZ）。该矢量的飞溅角度（cosθ），如图所示，使用矢量的内积如图 46 那样地进行计算。

在这里，求 cosθ理由是因为飞溅的角度越大，越接近 90°，飞溅量就越减小。就是说，水，砂子的设定数将变成为：基本设定数・移动量・cosθ。

砂子，水的速度将变成为对基本速度乘上随机数（0～255）/2550 后的值。这种事情即便是在前边的着地发生的情况中也是一样的。因为要乘上随机数，所以就可以多彩地显示随机发生的砂子，水的飞散。

给该速度再加上（OX,Oy,OZ）后的点将变成砂子，水的飞溅发生点。若参照图 47，则在随机数为 1/10（=255/2550）的情况下，发生点就可以用图 47 的坐标来表示。此外，砂子，水如图 40 所示顺次进行图形改变（32 图形）。

上述的继续处理可以使用这样决定的着地发生点，飞溅发生点继续地显示砂子，水的飞散。

图 48 是以上所说明的本实施例（飞行体的运动处理）的流程图。在 S1600 中，判定检查要素的下边是否是水面。这一判定针对于预定要进行

水或砂子的飞溅的场所进行。换句话说，在勇士所站立的地面为岩石，土或其它的固形物之类的场所的情况下，则不进行这种判定。这样一来，即便是存在着种种的场所，也可以仅仅对需要进行这样的飞行体的处理的场所迅速地进行处理。

在要素的下边不是地面而是水的情况下，就转移到下一步骤。把在该场所中上述的设置面的属性标志（b water）设定为“1”。在场所是在砂的情况下，就转移到砂子地面的处理（S1602）。由于该处理与水面上的处理是一样的，故略去其说明。

其次的处理，在 S1604 中，角色（勇士）的各个要素是否已着地（着水），用对角色的当前位置的 Y 坐标(Py)和水面位置的 Y 坐标(epos)进行比较的办法来判定。利用该判定，在检查要素处于水面下边的情况下，就转移到着地情况发生处理，判定为与此相反的情况下，就转移到飞溅情况发生的处理。

在 S1606 中，采用检查着地判定标志的办法判定在上次的中断中是否是着地状态。在该标志（bair）为‘0’的情况下，就要在数个中断间继续这一着地状态（S1608）。另一方面，在该标志为‘1’的情况下，可以判定在本次的中断中着地刚刚发生，并执行着地情况发生处理，使该标志复位为‘0’（S1602）后返回图 34 的主程序中去。

另一方面，在检查要素的 Y 坐标比水面位置还高的情况下，就转移到 S1614 采用检查着地判定标志的办法执行在上次的中断中是否是着地状态的判定。在上次是着地状态的时候，就是说，检查要素处于水面下边的情况下，例如，如图所示，本次就假定为有脚的上踢，并显示使水与脚的动作相一致地飞溅的图形（S1606）。接着，把该标志置位为‘1’后（S1607）返回。另一方面，在上次未处于着地状态时，则继续进行飞溅情况处理（S1618）。

如以上所说明的那样，倘采用本实施例，则可以把角色的移动量、移动方向与角色的动作相吻合地反映到移动的设置物（水、砂子等）上去。

其次，对模拟因游戏中的角色（勇士）的运动而发生风使落到地面上的树叶（飞行体之一）飞舞起来的运动（飞行体运动之一种）的处理

进行说明。

参照图 49。170 表示进行正在动作的勇士，172 表示因勇士的动作而飞舞的树叶。这里所说的‘风’，不是持续数个中断（所谓中断，指的是用于进行显示的一个场面，在该情况下，为 1/60 秒，换言之，指的是同步信号之间）的风，而说的是在一个中断内对叶子的举动有影响的矢量。

这一处理用图 50 的示意流程图来执行。该流程图进行是否由于角色的运动而产生了风的检查（S1800，1802）。接着，在模拟发生风的情况下，就转移到检查树叶的飞舞的程序（S1804），接着，模拟树叶的运动（S1806）。上述游戏机最初进行树叶在空中飞舞然后落下来的状态的模拟，在树叶到达地面之前继续模拟自由落下运动，在树叶从空中落到地面之后，在树叶停止之前继续模拟在地面上爬行的运动。在不产生风的情况下，则树叶的运动继续着而无须进行树叶是否飞舞的检查（S1806）。

首先，根据图 51 说明风的发生的检查的流程。在 S1900 中，计算要素位置。该计算在存在一对勇士的情况下，对勇士 1、2 每一中断交互进行。进行检查的要素，对右脚、左脚和头合计 3 个部位进行。对各个要素的移动量进行检查以判定风的发生的有无。该判定如下进行。

如图所示，设要进行检查的要素的上次（一个中断之前）的位置为(OX,Oy,OZ)，设本次的位置为(PZ,Py,PZ)。这时的移动量 M 如图 51 的式（1）所示。在这里，使用 XZ 坐标系中的移动量 M′(式(2))。

在 S1902 中，在 M′ 比 0．2 大时，就定为要素的移动量大到使之发生风那么大，就转移到下一步骤。在 M′比该值小时，就视为不发生风，结束该程序。

在 S1904 中，在 Y 方向的变化量(Py-Oy) 为 (Py-Oy) ≤ 0．8 的情况下，就视为风的方向是直接向下；从而不产生风。在 S1906 中，把风的发生标志建立为‘有风发生（=1）’，并决定风的发生位置和风的大小（矢量）。

风的发生位置和该 X，Y，Z 方向的矢量，如图所示。M′被修正为 0．12 ≤ M′ ≤ 1．0。该 M′是风的 Y 方向的矢量。X，Z 方向的矢量定为对 X 方

向，Y 方向的变化量乘上 0．6 的值。之所以在这里乘上 0．6，是因为把 X、Z 方向的矢量设定为比 Y 方向的矢量相对地小，主要为了模拟在模型空间的上方飞舞的状态的缘故。

风设定为在一个中断之间只发生一个矢量。在多个部位发生了风的情况下，把其中的值最大的算作应发生的风。

其次，对树叶的飞舞检查的流程图进行说明（图 52）。从风的发生位置和树叶的位置对每一树叶逐一进行树叶是否受到风的影响的检查，如果在该影响的范围之内，则因受到风的影响，树叶的运动（速度、矢量）会发生变化。

在步骤 S2000 中，读入风的数据和树叶的数据。这些数据（X 分量，Y 分量，Z 分量）如图所示。在 S2000 中，进行有关树叶的高度的运算。该运算如下所述。设地面的高度为 epos，则地上的 1m 的高度将变成 epos + 1．0(m)。设从该高度减去树叶的高度 lypos 后的值为 C1。

C1=epos+1．0 - l ypos(m)

在 C1≤0 的时候，就判定树叶的位置（处于距地面 1m 以上）高到不受因角色的动作而发生的风的影响的程度（影响 0%）。另一方面，在判定有风的影响时（S2004，NO），即 C1>0 时，就决定与 C1 的值对应的影响系数（S2006）。例如，在树叶处于地上时，(l ypos =epos ，C1=1)，树叶将受风 100%的影响。在 C1=0.5 时，受 50%的影响。C1 取 0≤C1≤1．0 的值。这些情况，将在图 68 中说明。该影响系数用于指明取决于树叶的高度树叶将会受到风的多大的影响。其次的 S2008 计算风的发生位置和树叶在 XZ 平面上的距离 L′。图中所示的计算式中的 w yspd，就象已经说过的那样是风的 Y 轴方向的矢量，而且，还兼作对树叶的影响距离（半径）。

在 w yspd- L′为 0 或小于 0 时，就当作是树叶在风的影响范围之外，作为没有风的影响的情况返回（S2010）。另一方面，在大于 0 时，就计算 C2（距离的影响系数）。此外，还要修正该 C2 计算 C2′。该影响系数的例子示于图 68 中。

在 S2014 中，如图所示，决定风对树叶的影响（矢量：X 方向，Y

方向，Z 方向），即决定树叶的速度（矢量），对树叶的现在位置加上该影响来模拟树叶的运动（图 50 的 S1806）。树叶的运动的处理程序紧接着就要说明。树叶的 X 方向，Z 方向的速度总要重新设定，Y 速度要加到上次（上一中断）的速度上去。

其次，对模拟树叶的运动的详细流程图进行说明（图 53）。树叶的运动，如图 54 所示，有两种形态：树叶的中心位置的运动和树叶的模型的 X，Yrot 的旋转运动。

（1）示出的是因风的影响或重力树叶的中心位置进行移动的运动，（2）示出的是用 Xrot，Yrot 模拟（表现）树叶摆动的运动的情况。这些运动是用示意流程图说明过的自由落下运动的一个环节。

前者的运动（位于空中的情况，从空中着地的情况）的特性，如图的 S2100 所示。在树叶位于空中的情况下，对 X 方向的速度（1 xspd)乘上 0．8 作为新的 X 方向的速度的理由，是因为考虑到了空气阻力。对于 Z 方向也是同样的。Y 方向的速度减去 0．002 作为新的 Y 方向的速度。这样一来，就可以模拟叶的重力方向的速度渐渐变化下去的情况。其中，设 Y 方向的落下速度（1 yspd）为-0．006。关于树叶正在着地的情况将在后边描述。

另一方面，如图 55 所示，树叶 230‘哗啦哗啦’飞舞的状态可以用上述的 Xrot，Yrot 模拟（S2102）但，在本实施例中已准备好了 32 个图形的 Xrot，顺次进行图形改变。使用 Yrot 进行的旋转如下。

Y 旋转速度=1 yspd×1200×256

用 0×0000～0×FFFF（0～65536）来表现一个旋转（360°）。这是 1°(实黑线框)² 182（0×b6)。此外，若使 Yrot 也进行图形变化，则由于有对树叶的动作模拟得粗糙的危险，所以使一方的旋转要素（Xrot）进行图形变化，其余的要素（Yrot）用计算来求。

树叶到达地面之后的运动，如图 53 所示，用中心运动特性式和旋转运动特性式模拟（S2100，S2102）。中心运动特性式示于图 53。在该运动特性式中，树叶的 Y 轴方向的速度（1 yspd）由于已被设定为 0，故可以模拟树叶在地面上边旋转边动作的状态。在该特性式中，X 方向的速

度和 Y 方向的速度都乘上 0．5，是因为考虑到了地面的摩擦，乘上一个比空中有树叶的情况还小的值，模拟地面的摩擦阻力比空气阻力还大的情况。此外，判定 X 轴周围的树叶的旋转角度（Xrot）是否为‘0’，在是‘0’的情况下，则作为树叶已对地面平行的相接而使树叶停止动作。因此，在因为角色活动而发生的风使树叶在空气中飞舞后该树叶落到地面上变成为与地面平行的时刻树叶的动作停止。

在空中飞舞的树叶着地与否，比较地面的 Y 坐标和树叶的 Y 坐标来判定。树叶的 Y 坐标≤（或‘=’）地面的 Y 坐标时，就判定树叶处于着地状态。此外，树叶采用树叶的模样（纹理）贴到一个多边形上去（映象）的办法使树叶变成形状模型。

以上所说明的飞行体（水、砂子、树叶）的移动方向由角色（模型）的动作的特性决定，所以可以把模型的动作正确地反映到飞行体上。这样的飞行体是借助于本来模型的动作而飞翔，或者落下，借助于使模型的动作反映到这些上的办法就可以高品质地模拟飞行体的动作。倘采用以上说明的飞行体的运动处理，则就可以以丰富的演出感，且以高品质再现在虚拟空间内的模型（飞行体）的（自由）落下运动。例如生成树叶‘哗啦哗啦’的飞舞的图像。

自由落下运动处理

其次，说明在虚拟空间内落下的物体的运动处理。这里的实施例，把雪花作为对象物，对模拟把‘飘舞’运动赋予雪花的演出进行说明。在这里要说明的‘飘舞’指的是雪花边随风飘动边落下来的状态，或由于勇士的动作而飞舞起来的落叶飘落的状态等不规则的运动。本实施例企图逼真地模拟这种‘飘舞’的状态。

对雪花飘舞的实施例进行说明。飘落的雪花由一个或多个多边形构成，受到已设定于模型空间中的风的矢量（使本矢量进行适当的变化）的影响。雪花的多边形的运动由重力形成的自由落下，风矢量引起的影响（以上，权利要求的第 1 装置），虚拟空间的 Z 平面（水平面，即与地表面平行的坐标系）的圆周运动（同上权利要求的第 2 装置）来进行模拟。圆周运动的角速度、振幅已在存储器内做成了表，可以选择其一使用。

下面描述赋予飘舞雪花以圆周运动的处理。设角速度为ω，旋转角

为 q，振幅为 a，则飘舞运动的 X 分量(x off)和 Z 分量(z off)，即对雪花的基本举动（由自由落下和风的矢量决定）的偏移（offset）量如下。

$$x\,off = \sin(q+\omega)\cdot a$$
$$y\,off = \cos(q+\omega)\cdot a$$

该偏移数据在每一中断期间计算 10 次，并加到雪花的基本位置上去。

雪花的运动方程式如下：

x pos ：雪花的位置（X 方向）、

y pos：雪花的位置（Y 方向）、

z pos：雪花的位置（Z 方向）。

x pos = x pos + x off + 风 x（风矢量速度的 x 分量）

y pos = y pos + 下落（雪花的落下速度，例如－1cm/int）

z pos = z pos + z off + 风 z（风矢量速度的 Z 分量）

倘采用本实施例则可以用上述运动方程式表现雪花随风飘摇落下的举动（参看图 56）。因此，可以用简单的计算式得到复杂的雪花的飘舞运动，且给予 CPU 块的计算负荷也不大。

图 56 示出了该雪花的移动轨迹，即便是在规定的方向上给自由落下的雪花加上方向或大小变化的矢量，也不可能准确地模拟‘飘舞’的状态。所以，采用给该雪花加上圆周运动（圆周运动的一种，也可以是其他的椭圆周运动之类的运动）的办法，就可以再现这种‘飘舞’。

倘采用本处理，由于可以提高模拟虚拟空间内的模型自由落下运动时的品质，所以可以丰富地营造出落下图像处理的演出感。

不规则区域中的冲突判定处理

其次，对不规则区域与模型之间的冲突处理进行说明。该处理具有如下的内容。在这里，所谓区域，指的是限制勇士的活动的区域，例如若想象为用不规则的墙壁围起来的圆圈是易于理解的。勇士虽然可以在该区域内自由地移动，但却不能越过墙壁移动到圆圈之外，这种墙壁就相当于这里所说的‘区域“。

图 57（A）中示出了不规则的区域 250。这一区域被设定于虚拟空间的 XZ 平面（Y 轴是高度方向）上。该区域被设定为 7 角形。作为该区域可以设定为种种形态的区域。作为这样的区域在想象为周围是岩石的圆圈的情况下说的就是该岩石。

图 57 还示出了该处理的原理。角色的位置必须加以校正，以便角色不会移出墙壁之外，从而可以示范出墙壁(区域)的确存在。本实施例对大小和形状皆可任意的设定的竞技场的墙壁和角色之间的冲突进行判定，在用于修正位置的墙壁的法线方向上计算排出矢量(expulsion vector)。

图 57（A）是该冲突判定的模式图，（B）是矢量计算的原理图，（C）是角色（模型）和区域之间的冲突判定模式图的局部扩大图。此外，图 57 中还同时记载有矢量计算式。

如（A）所示，在进行冲突判定时，角色的形状用球近似，再把墙壁和该球体平行地投影到与地面平行的平面（XZ 平面）上，研究形成区域的各边与进行球投影后的结果所得到的圆 252 之间的位置关系。其中，球的位置是角色的计算上的位置。在该图中，L 是该区域的任意的边的一个顶点，R 是该边的另一顶点。对某一任意的边，进行设定使得 L 在反时针方向上。

图中的 T 是角色投影后的圆的中心点。矢量 R，矢量 T，矢量 L 是从该坐标系的某一任意的点对这些各个点设定的矢量。此外，矢量 P 是从圆的中心坐标对区域的各边垂直的引出的矢量。矢量 V 是从对该圆的边 LR 平行的切线与圆之间的交点，对边 LR 成直角地引出的矢量，相当于该处理中的赶出矢量。

在该处理中，目的是计算排出矢量 V。在圆的中心线处于直线 RL 的外侧（以此为正区域）的情况下，如图所示，把赶出矢量设定为如图的式（1）所示。

另一方面，在圆的中心线处于直线 RL 的内侧（以此为负区域）的情况下，就设定为如式（2）那样。即，圆的中心于直线 RL 之间的距离，在大于半径（d）的情况下，就不能设定排出矢量。在超过该距离的情况下，则可以设定排出矢量。这时，由于矢量向着与墙壁的方向，所以把单位矢量 Pe 设定于和圆的中心位于直线 RL 的外侧时相反的方向上。

图 58 是对本实施例的动作进行说明的流程图。首先，读入控制器的按键或方向键的操作量（S2600）。用该操作量决定所计算的角色的位置（S2602）。即这是在假定无区域的情况下的所计算的角色的位置借助于控制器的操作来求该位置。

其次，在 S2604 中，进行该角色的位置和某一边 RL 之间的比较。即检查图 57 的式（3）的关系。在该判定为否定的情况（>1）下，对下一条边进行检查。在该情况下，设从点 T 不能向该边垂直的引出矢量 P（即不能当做将成为冲突判定对象的边），就转移到 S2608，而不进行 S2606 的矢量 P 的计算。

在该关系为肯定的情况下，就在 S2606，从图 57 所示的地方，计算矢量 P。其次，在 S2608 中，对所有的边判定 S2604 的处理是否已结束，在否定的情况下，就转移 S2610 执行与其它的边的比较（S2604）。另一方面，在该判定为肯定的情况下，就转移到 S2612，选择矢量 P 的最小点。把已设定了该矢量 P 的边，作为用于在法线方向上设定排出 P 的墙壁（边，区域）来决定。就是说，判定角色已接触到设定该矢量的边的方向上。

接着，在下一步骤（S2614）中，判定圆的中心是否在区域之内，是在区域外时，就用图中式（1）设定排出矢量，是在区域内时，则用式（2）设定排出矢量。在式（1）的情况下和在（2）中，在矢量 P 的大小比圆的半径小的情况下，就看做是角色与墙壁之间有冲突，设定上述的排出矢量。用该排出矢量，在实际的图像中，执行限制从该区域（墙壁）向目的地移动图像处理（S2616）。

当然，也可以根据该墙壁用排出矢量进行处理，使得从该墙壁的该边的法线方向上定义的矢量 V 拉住模型。图 67 示出了这时的图像处理的一个例子，标号 350 是勇士，标号 352 是不规则的墙壁。所示出的是即便是勇士 352 企图向图中区域一侧移动（图中的左方向）也将被区域挡住而不得移动的状态（图的（1）的状态）。但，如图的（2）所示，虽然勇士 350 的整体是因区域而不能移动的状态，但是右脚 354（角色的一部分的要素）却可以如双向箭头所示那样的进退。这样一来，游戏者就可以知道自己所操作的模型被区域所挡不能在该方向上再进行移动。

倘采用本实施例所示的处理，则由于使得对每一条边都计算出排出矢量，所以即便是不规则区域也可以确实地进行该区域和角色之间的冲突（即，在给出了排出矢量的值的时候冲突判定就被肯定），就可以把该判定结果反映到图像处理上去。

纹理列的作成

其次，对作成重复的纹理列的方案进行说明。图 59 是其说明图。在下段（bottom）示出了第 1 纹理列，在中段（Top）示出了第 2 纹理列，在上段（Together）示出了第 3 纹理列，第 3 纹理列是第 1 纹理列和第 2 纹理列重叠后的结果。

第 3 纹理列的目的是形成以规定的周期进行反复的图像‘海上的反射图形’的图像。第 3 纹理列由 0～29 这 30 枚图画构成，把这些图画按顺序映象到多边形上，即，把第 0～29 号的图画进行映象后，再次进行第 0～29 号的映象，采用反复这样作的办法，这些纹理列的连接部分（例如，第 28～2 号）处的图画的变化就是自然的（连续性的，换句话说，没有漏画）。这样一来，就可以作成重复海上的反射图形的图像。

在该处理中，把透明度顺次变化（0～100%）的参数赋予第 2 纹理列。第 2 纹理列的上边所记载的数值是与该透明度有关的值。具有 100%，表示完全无透明度的状态，此外，有 7%的，表明是有较好的透明度的状态（透明度 93%）。在完全透明的情况下，则给予 100%的参数。

第 3 纹理列是如图所示顺次把第 1 纹理列与第 2 纹理列重叠起来作成的。就是说，第 1 纹理列的第 0 号与第 2 纹理列的第 30 号重叠形成第 3 纹理列的第 0 号，第 1 纹理列的第 1 号与第 2 纹理列的第 31 号重叠形成第 3 纹理列的第 1 号，…，第 1 纹理列的第 29 号与第 2 纹理列的第 59 号重叠形成第 3 纹理列的第 29 号。

这时，在第 1 纹理列的第 0 号与第 2 纹理列的第 30 号重叠的时候，由于第 2 纹理列的第 30 号的透明度是 0（完全不透明的状态），所以第 1 纹理列的第 0 号就被第 2 纹理列的第 30 号完全给遮了起来，第 3 纹理列的第 0 号的图像就变得与第 2 纹理列的第 30 号的图像相等。

另一方面，在第 1 纹理列的第 29 号与第 2 纹理列的第 59 号重叠时，由于第 2 纹理列的第 59 号的透明度是 97%，（几乎近于透明），所以第 3

纹理列的第 29 号的图像就变得几乎于第 1 纹理列的第 29 号的图像相等（换句话说，第 2 纹理列的第 59 号的图像是几乎看不见的状态）。

在这里，在发生第 2 纹理列的时候，参数的提供方法由于已做成为使得其第 30 号的图画变成为与第 1 纹理列的第 29 号的图画连续起来，所以，在看第 3 纹理列的情况下，在纹理列的连接部分处（在图 59 中，第 28～2 号），就可以如上所述使之产生使得连接变成为自然的图画。

采用给每一纹理列提供参数，使得第 1 纹理列的结束部分（例如，第 25～29 号）和第 2 纹理列的最初的部分（例如，第 30～34 号）连接起来的办法（或者采用从结果上看将变成这样的办法），如图的箭头所示，当考虑透明度时，采用使第 1 纹理列的第 29 号和第 2 纹理列的第 30 号连接的办法，则如图所示，第 3 纹理列将变成自然地连接起来的形态的图画。

第 3 纹理已存储到存储器的规定的区域中，并顺次映象到表示海面的多边形中，海面的白浪（在画面上是辉度高的部分，在图 27 中是显示为白的部分）的波形和形态可以周期性地表现反复的图像。在图示的例子中，定为每 1/30 秒显示一枚图画，如果设为 30 枚图画，则每一秒就可以回放周期性地重复海面的白浪的部分的样子的图像。

回放波浪的形态的速度和所施用的纹理数都可以适当地变更。设有勇士站在画面的中心的舞台，在舞台近旁（即在画面的中心附近），设定已说过的回放速度（每个场面 1/30 秒，30 个场面回放），比此处更远的的波浪，从减轻计算机机的计算负荷的观点看，可以降低每一场面的回放速度，且可以减少合计使用的场面数。例如，使用第 3 纹理列的 0 号、10 号、20 号、29 号的场面。之所以这样作，是因为即便是使位于画面的周边附近的图像的回放多少有些变粗，但是对于减少给予游戏者的坏印象且对于减低 CG 装置的计算负荷是有利的。在上述的游戏装置的存储器中已存储好第 3 纹理列。因此，在使用了 CG 的 2 的领域中，就可以利用上述的纹理列。其中，已说明过的纹理列的发生方法，不仅仅在 CG 的技术领域，在动画片的动画作成的技术中也可以利用。

慢回放处理

图 61～图 64 是用于说明慢回放处理（回放速度变更处理）的原理的角色的动作图。示出的是 2 个勇士 290、292 互相面对面，正在比赛两

者的技术的优劣的状态。在图 61 中，290 是受攻击的勇士，292 是进行攻击的勇士。标号 294 是勇士 290 受到来自勇士 292 的进攻动作（攻击；出拳）的攻击部位。图 61 和本身为从上方看图 61 的平面图 62 示出了勇士 290 正在遭受攻击的状态。

另一方面，如图 63 及本身为其平面图的图 64 所示，勇士 290 用右手的要素化解勇士 292 的攻击（即进行防御），若防御成功则将回放勇士 292 向箭头所示的方向后仰（移动）的活动。这时，就使勇士 292 的回放速度降低进行慢回放。

如上所述，假定一方的角色（勇士）向另一勇士作出了动作‘出拳’。另一方面，另一勇士几乎同时作出了化解该出拳的动作。此外，这些出拳或躲闪动作采用游戏者适当地操作控制键盘的操作按键或方向键的办法发生。

在该出拳的躲闪已成立时，就要再现攻击一侧的勇士向后方大后仰的活动。这时，要使得该活动的回放速度比防御一侧的勇士的图像的回放速度小。这样，由于该慢回放对攻击一侧的勇士将产生所谓‘间隙’，易于由防御一侧的勇士向攻击一而的勇士进行攻击。勇士 292 一旦后仰之后，如图 65 所示，就将慢慢的返回原来的体形（后仰之前的）。

图 65 是该处理的流程图。在 S3400 中，防御一侧的勇士作出上述的躲闪动作。在 S3402 中，判定对方一侧的攻击动作（在图的情况下，，是‘出拳’是否已碰到躲闪动作（就是说，判定处于躲闪动作的动作指令下的手的要素与攻击一侧的勇士的手的要素的冲突）。

在该判定为否定的情况下，就返回而无须进行慢回放。另一方面，在 1 判定是肯定的情况下，在攻击一侧的角色未位于防御一侧的角色的正面一情况下，就执行返回程序；在正面的情况下，就转移到 S3406（S3404）。在 S3406 中，判定是否是能够解除攻击动作的动作。例如，若为出拳则可以解除，若为脚踢则不能解除等。这种判定对每一动作都设定专用的标志采用参照标志寄存器的内容的办法就可比较容易地实现。

在是不能躲闪的动作的情况下，就结束处理，是可以躲闪的动作时则转移到 S3408。在该步骤中，设定应进行慢回放的帧数，并把它存放在

定时器中。在其次的步骤 S3410 中，在在取决于动作设定被躲闪量（在图的例子中，在对攻击一侧的勇士后仰的活动进行回放的情况下，各个要素的后仰量）。帧数和躲闪量用规定的特性式计算，或者作成表。

在 S3412 中，所设定的帧数被除以躲闪量，并计算出相应一帧的调整垫量（空间坐标系中的移动量和移动方向）。接着，在 S3414 中，使攻击一侧的勇士的回放速度（显示速度）变成防御一侧的勇士的例如一半。例如，设攻击一侧的角色的回放速度成为 1/60 秒的一半的 1/30 秒。

接着，在 S3416 中，判定上述定时器的值，若为‘0’，则结束慢回放处理，使攻击一侧的勇士的回放速度返回原来的状态（S3417）后，返回。另一方面，在非‘0’的情况下，把定时器的值乘到一个帧的调整垫量 21 攻击一侧的移动量。接着，在 S3420 中，把这一移动量加到攻击一侧的勇士的坐标值上。就是说，给图 51 的攻击一侧的勇士 292 的位置加上图 63 的躲闪动作成立时的移动量（后仰量：本来的位置（图 61 的勇士 292 的位置））。

接下来，在 S3422 中，使定时器的值（T）减 1，在 S3424 中，进行攻击一侧的勇士 292（图 63）的活动位置的再计算。接着，返回主程序。

倘采用本处理，则在躲闪动作已成功的时候，要回放下述图像：攻击一侧的勇士最初要边大大地后仰边在慢回放状态下慢慢后仰下去。这时，防御一侧的勇士变成为易于对攻击一侧的勇士作出攻击动作。

倘采用本处理，由于在降低一方的勇士的活动的回放速度的同时进行另一方的勇士的活动的回放，所以结果就变成为使游戏者自己所操作的防御一侧的勇士动作得以有效地发挥，这将使得作为游戏的要素的慢回放成为可能。因此，可以以丰富的演出感表现慢回放。

此外，本发明并不限定于上边所说过的各种实施例的构成，如果是本专业的人员，则作权利要求范围所述的构成的范围内还可以有各种各样的变形。还有，作为已存储了游戏机的动作用程序的存储（记录）媒体，除已讲和的盒式磁盘 ROM、CD-ROM 之外，也可以是互连网、PC 机网上的通信媒体。

图 1

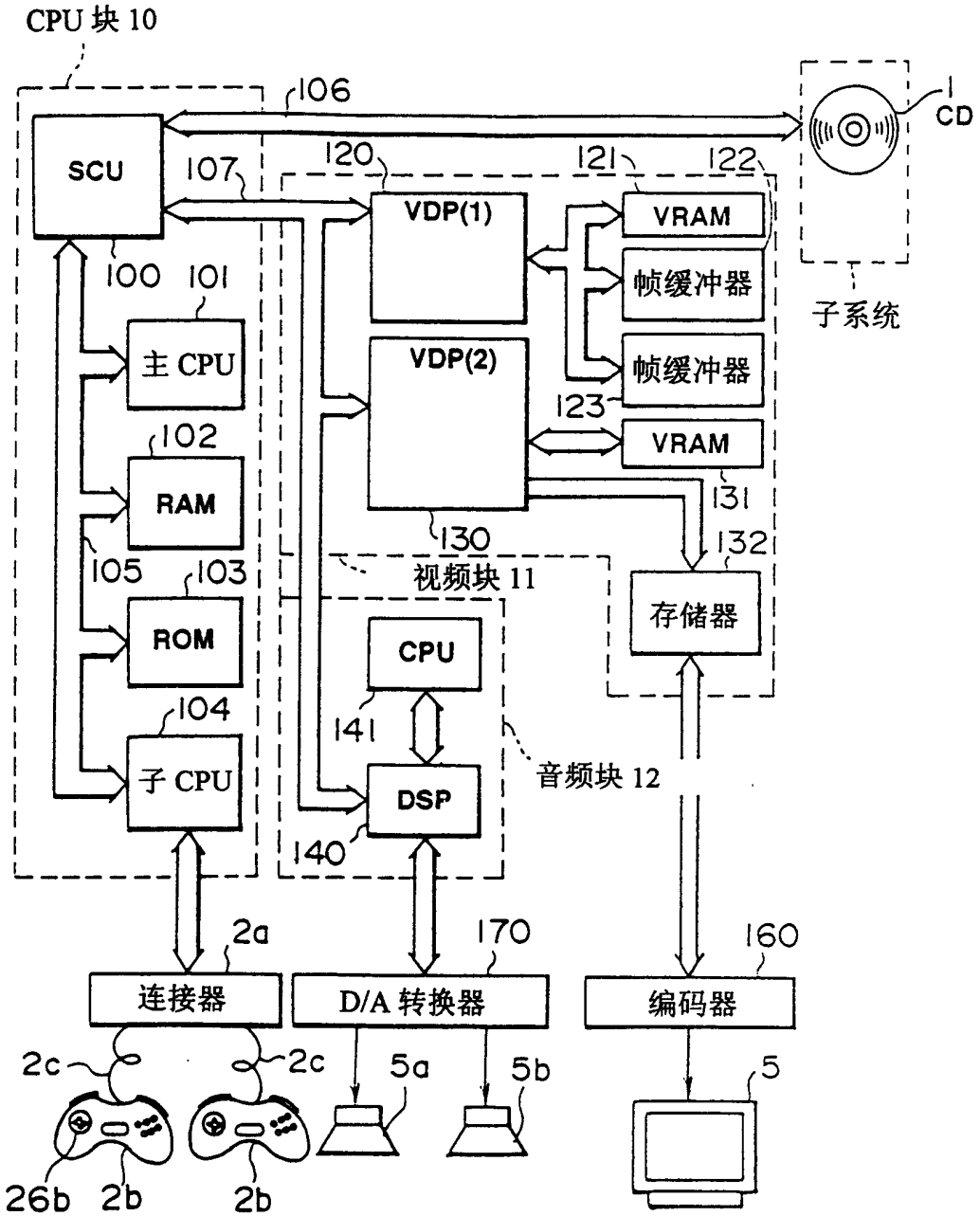

CPU 块 10
106
1
CD
SCU
107
120
121
122
VDP(1)
VRAM
帧缓冲器
子系统
100
101
主 CPU
VDP(2)
帧缓冲器
123
VRAM
102
RAM
131
130
132
105
103
视频块 11
存储器
ROM
CPU
104
141
子 CPU
音频块 12
DSP
140
2a
170
160
连接器
D/A 转换器
编码器
2c
2c
5a
5b
5
26b
2b
2b

图 2

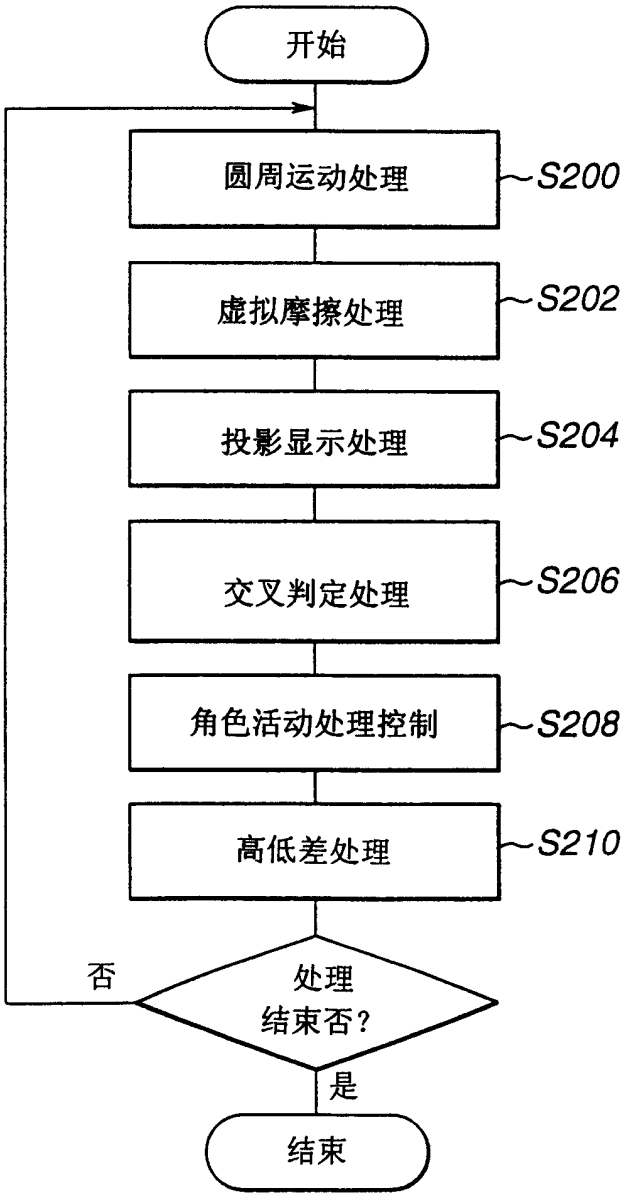

开始
圆周运动处理
S200
虚拟摩擦处理
S202
投影显示处理
S204
交叉判定处理
S206
角色活动处理控制
S208
高低差处理
S210
处理
结束否？
否
是
结束

图 3

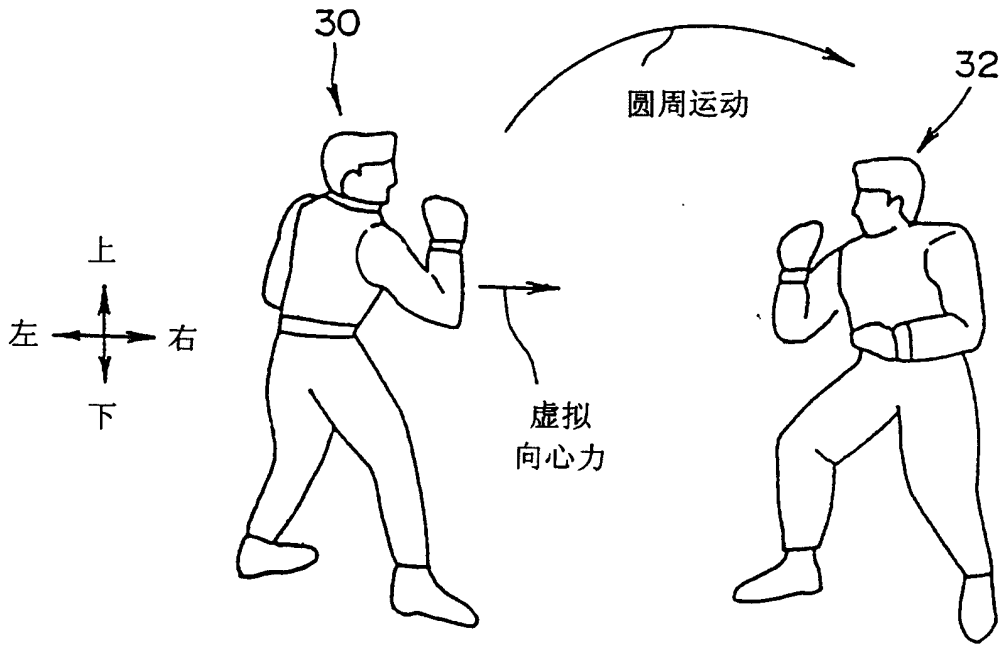


图 4

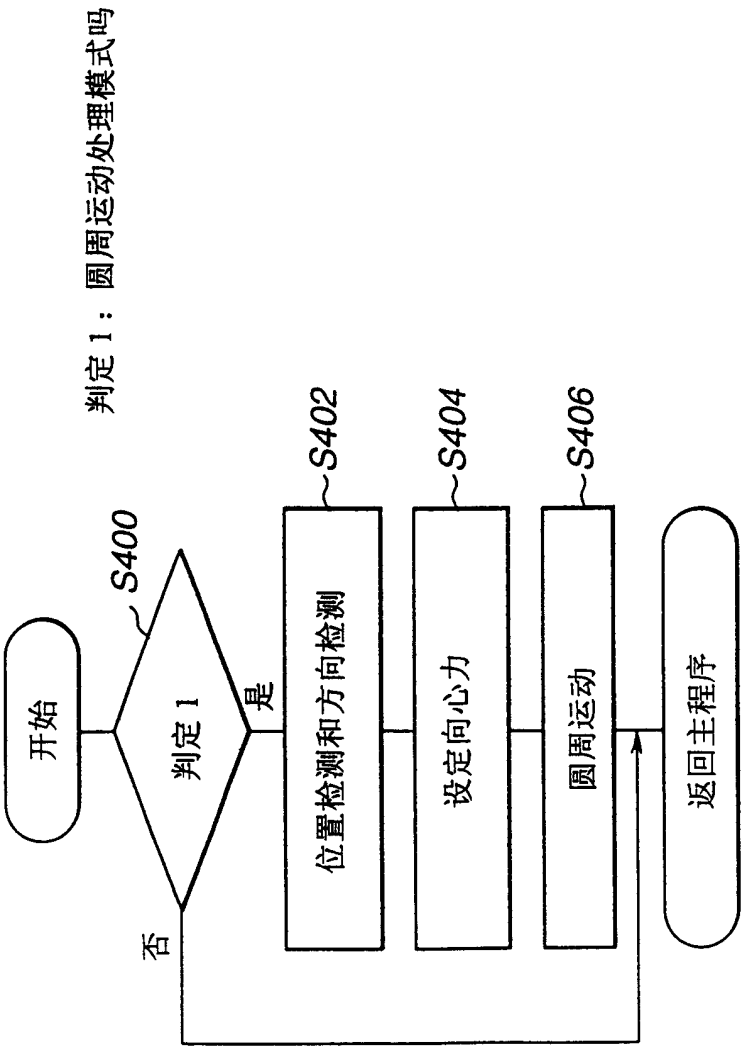

判定 1：圆周运动处理模式吗
开始
S400
判定 1
是
否
S402
位置检测和方向检测
S404
设定向心力
S406
圆周运动
返回主程序

图 5

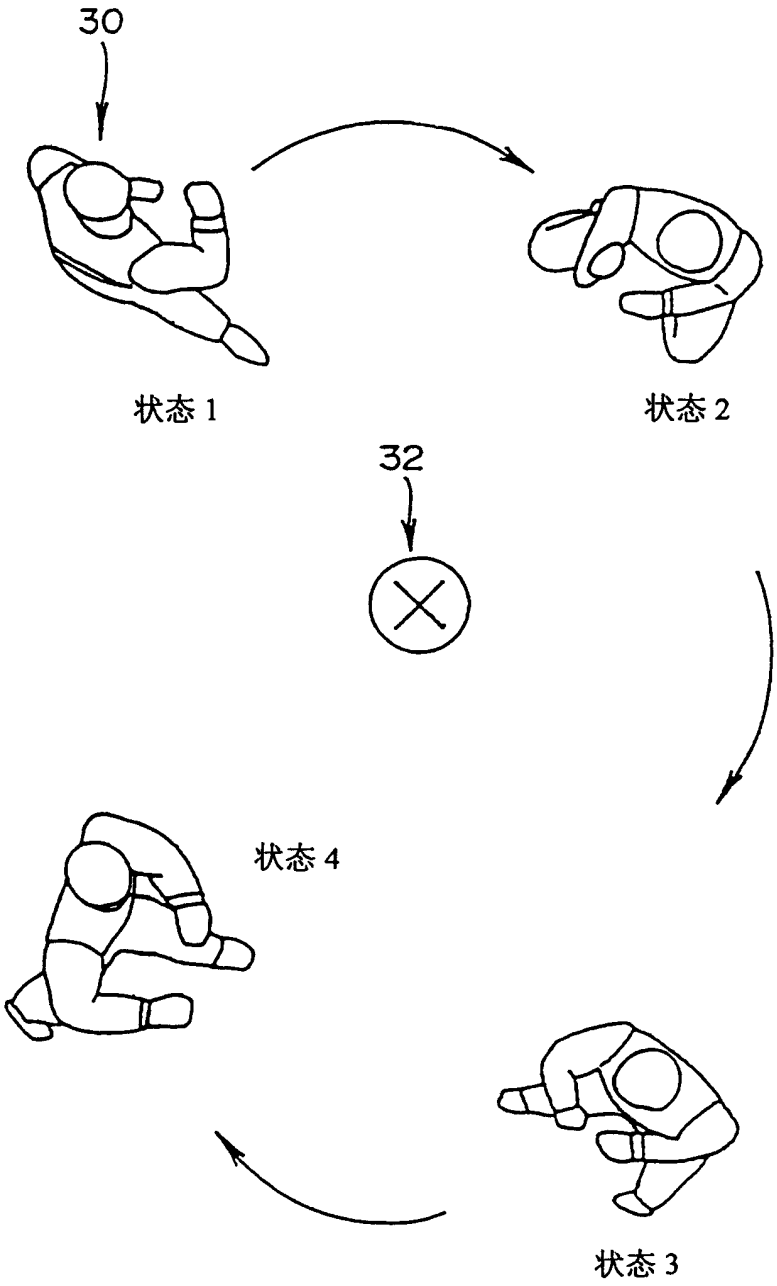


图 6

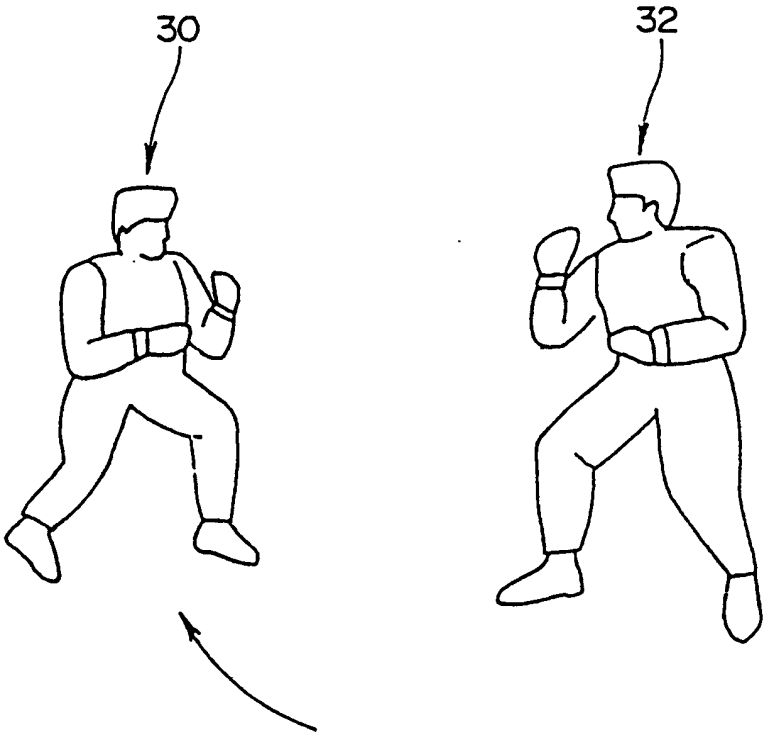


图 7

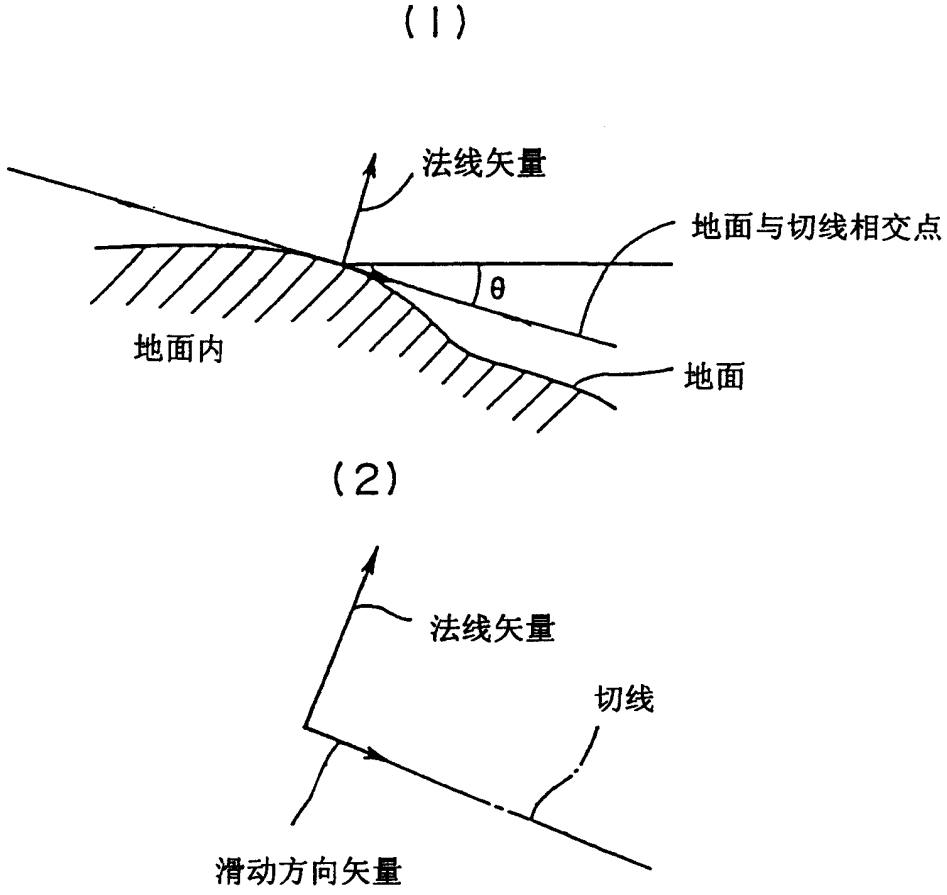


图 8

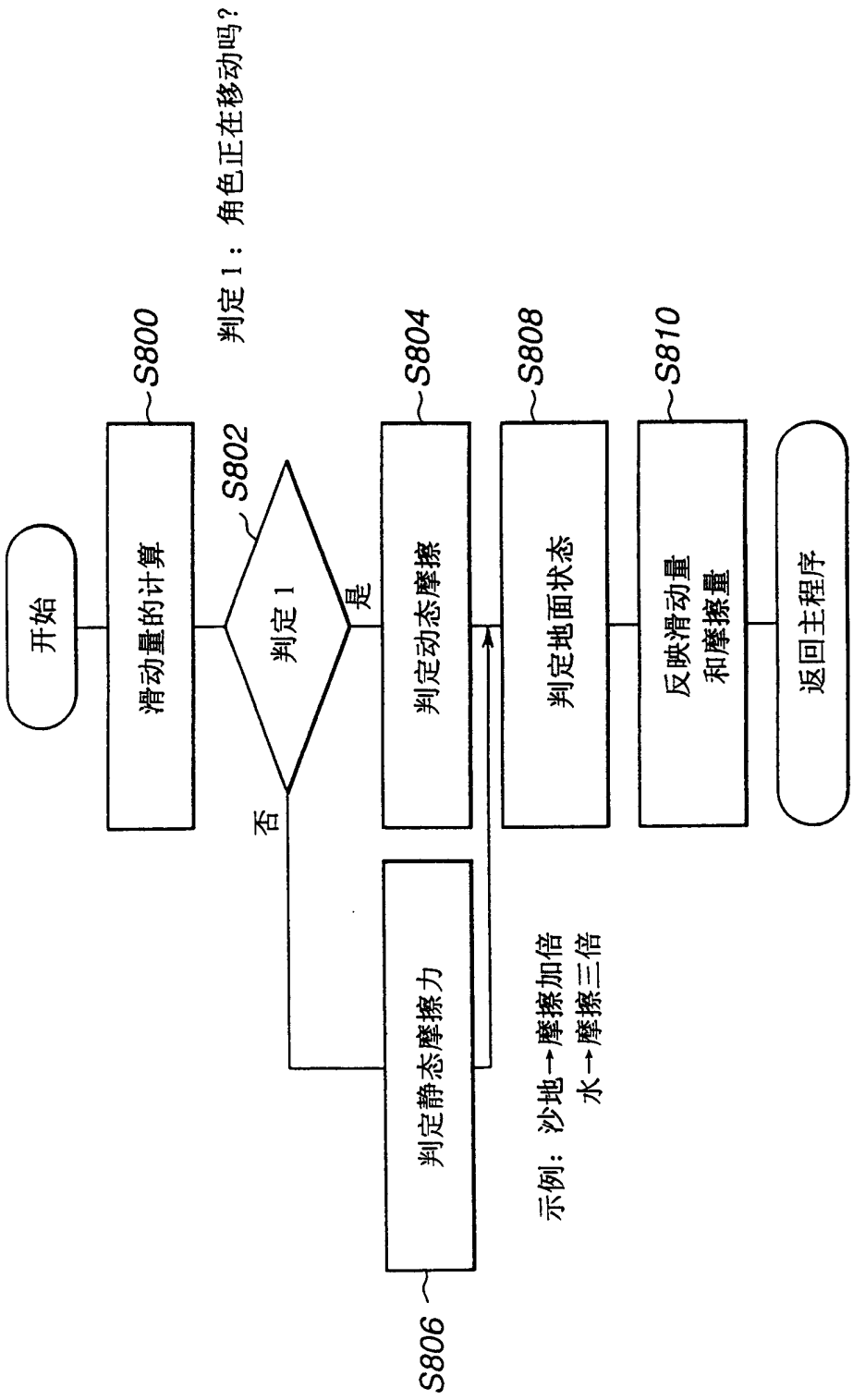

开始
S800
滑动量的计算
S802
判定 1
判定 1：角色正在移动吗？
是
否
S804
判定动态摩擦
S806
判定静态摩擦力
S808
判定地面状态
S810
反映滑动量
和摩擦量
返回主程序
示例：沙地→摩擦加倍
水→摩擦三倍

图 9

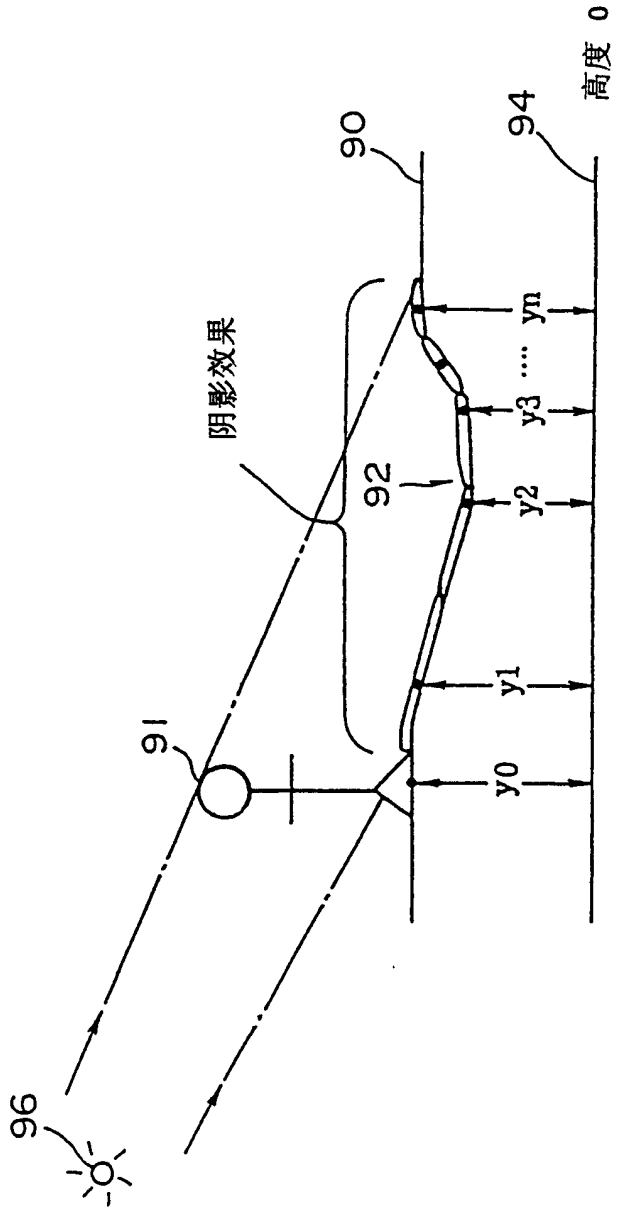


图 10

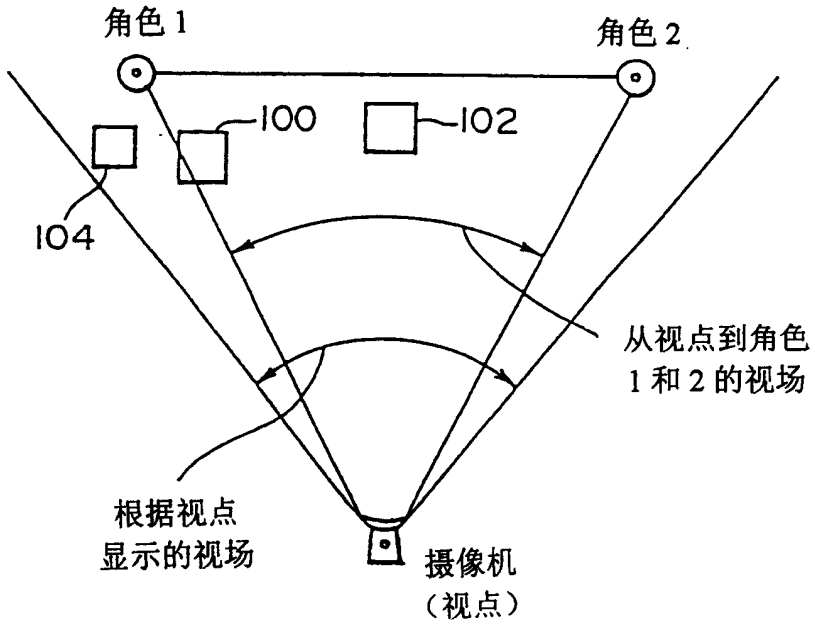


图 11

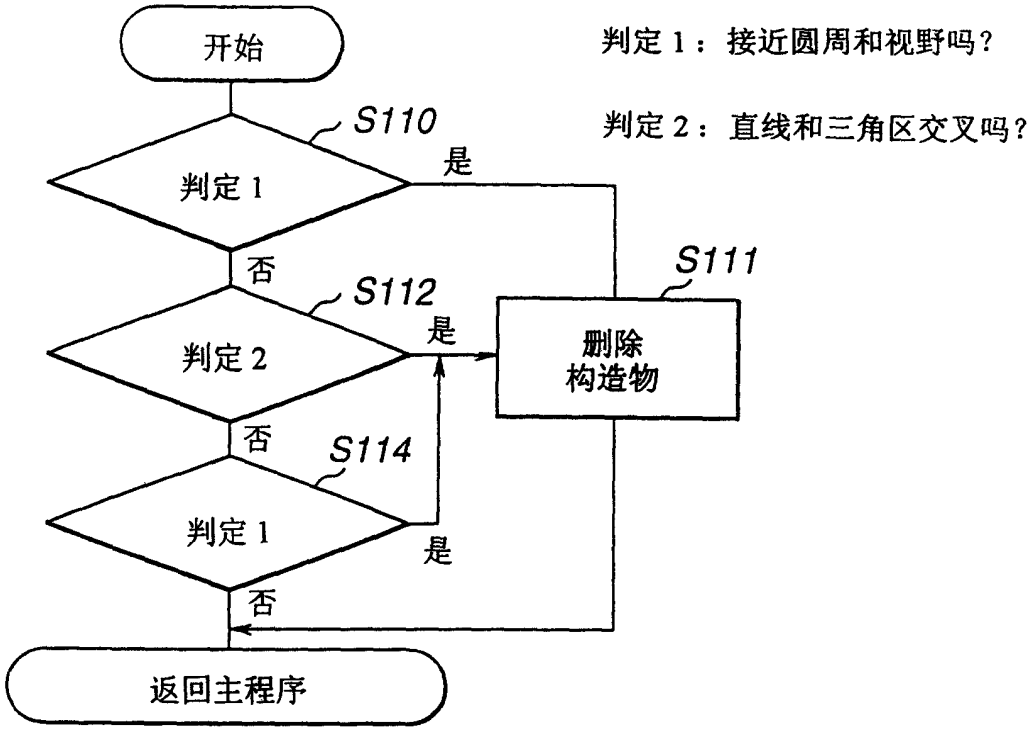

开始
S110
判定 1
是
否
S111
S112
判定 2
是
删除
构造物
否
S114
判定 1
是
否
返回主程序
判定 1：接近圆周和视野吗？
判定 2：直线和三角区交叉吗？

图 12

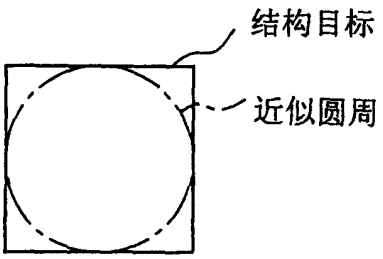
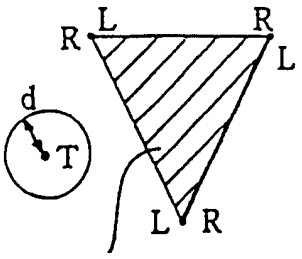

结构目标
近似圆周

图 13



从视点到角色 2 的视野

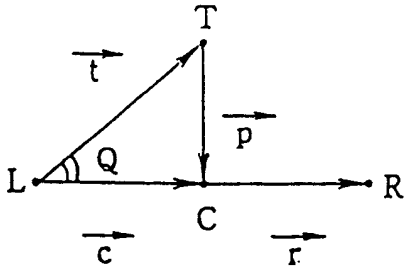
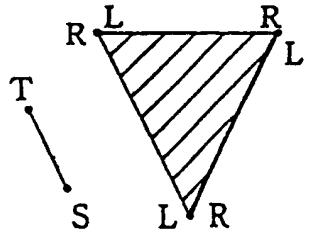


图 14



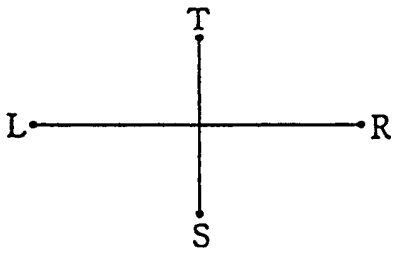


图 15

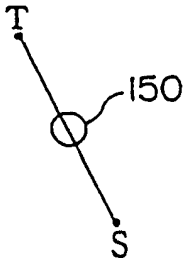


图 16

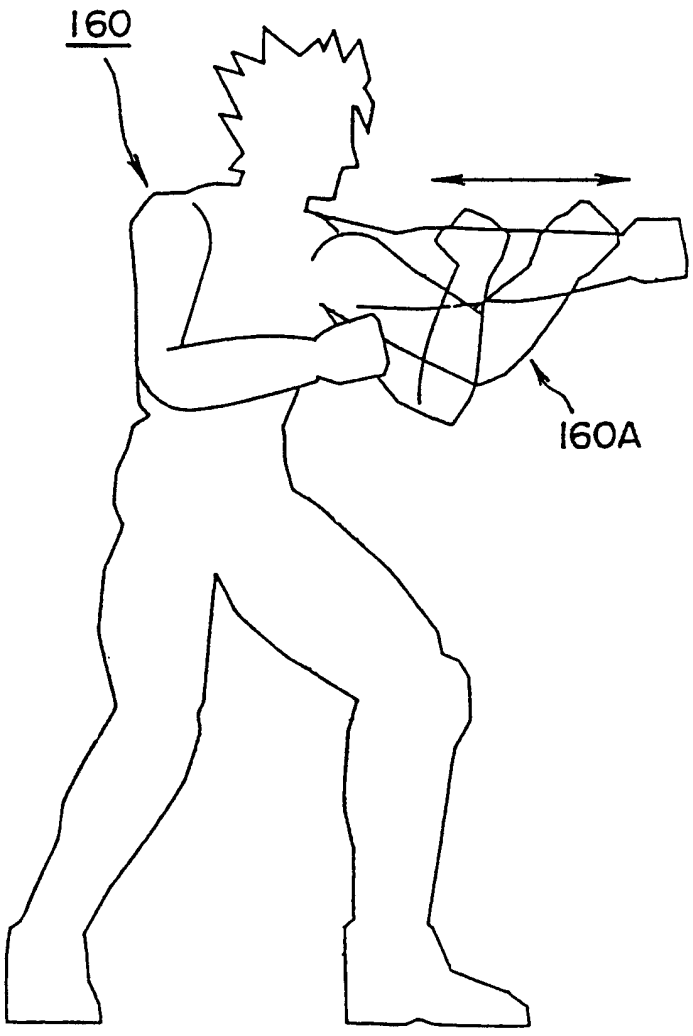


图 17

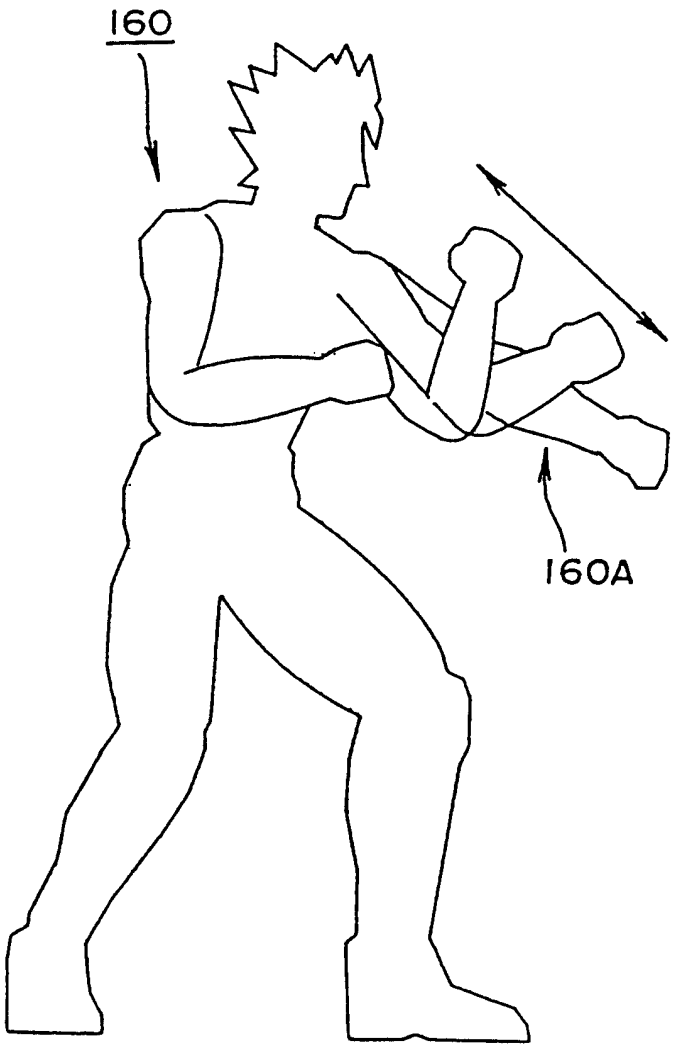


图 18

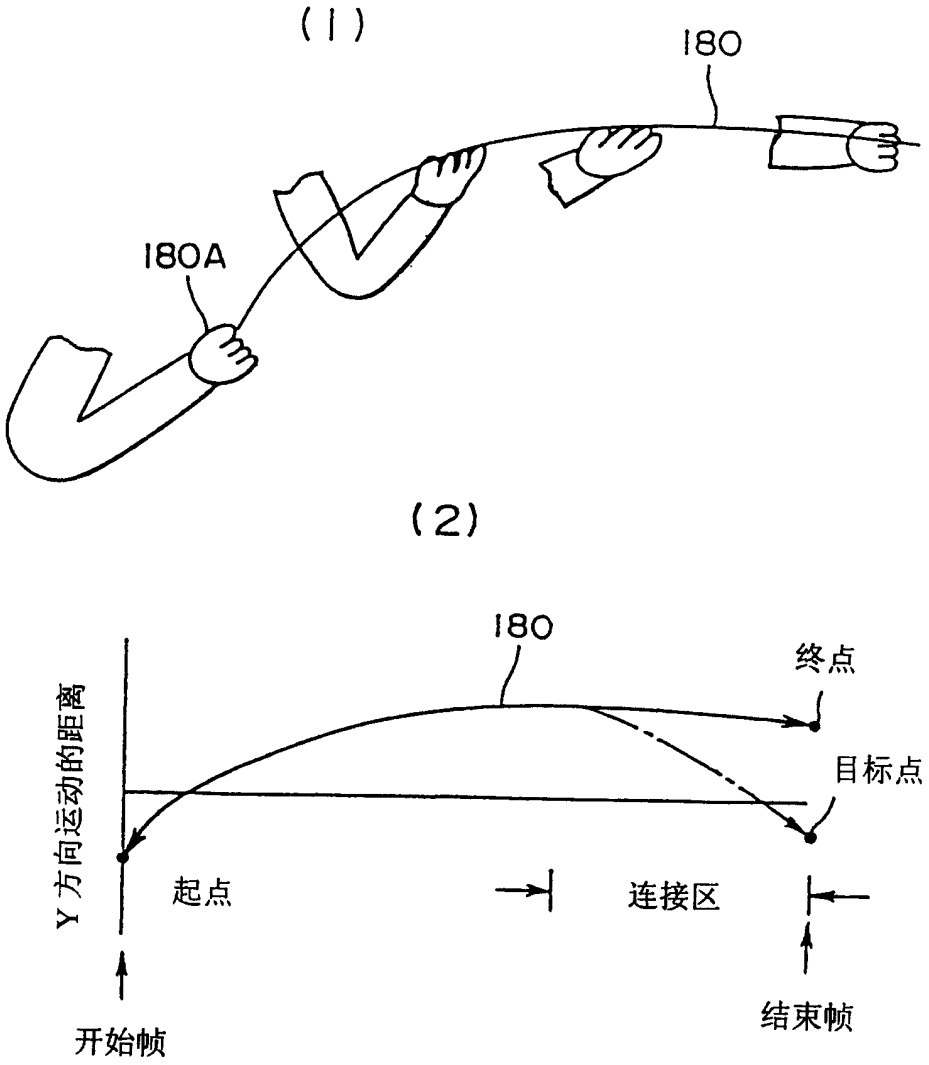
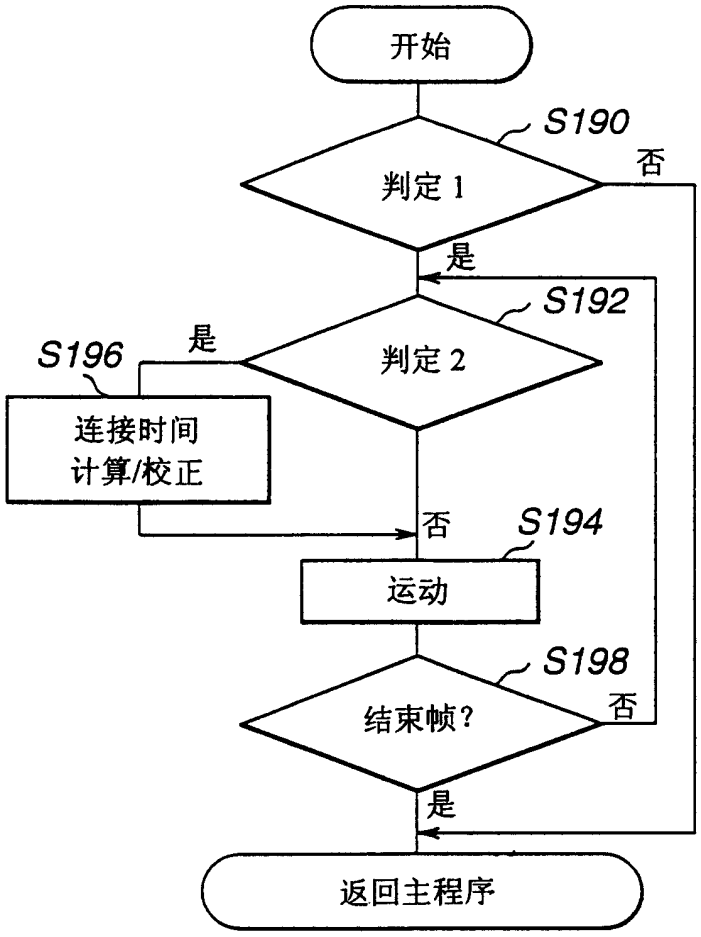


图 19



判定 1：有运动命令吗？

判定 2：终点≠图标点

图 20

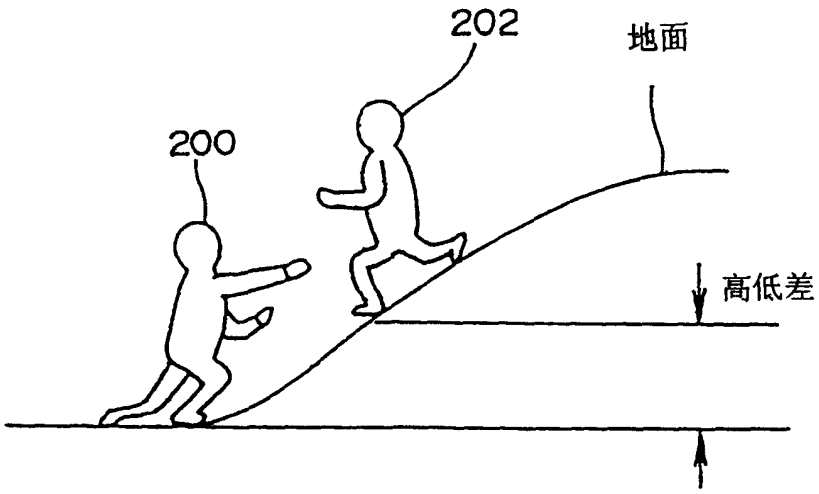


图 21

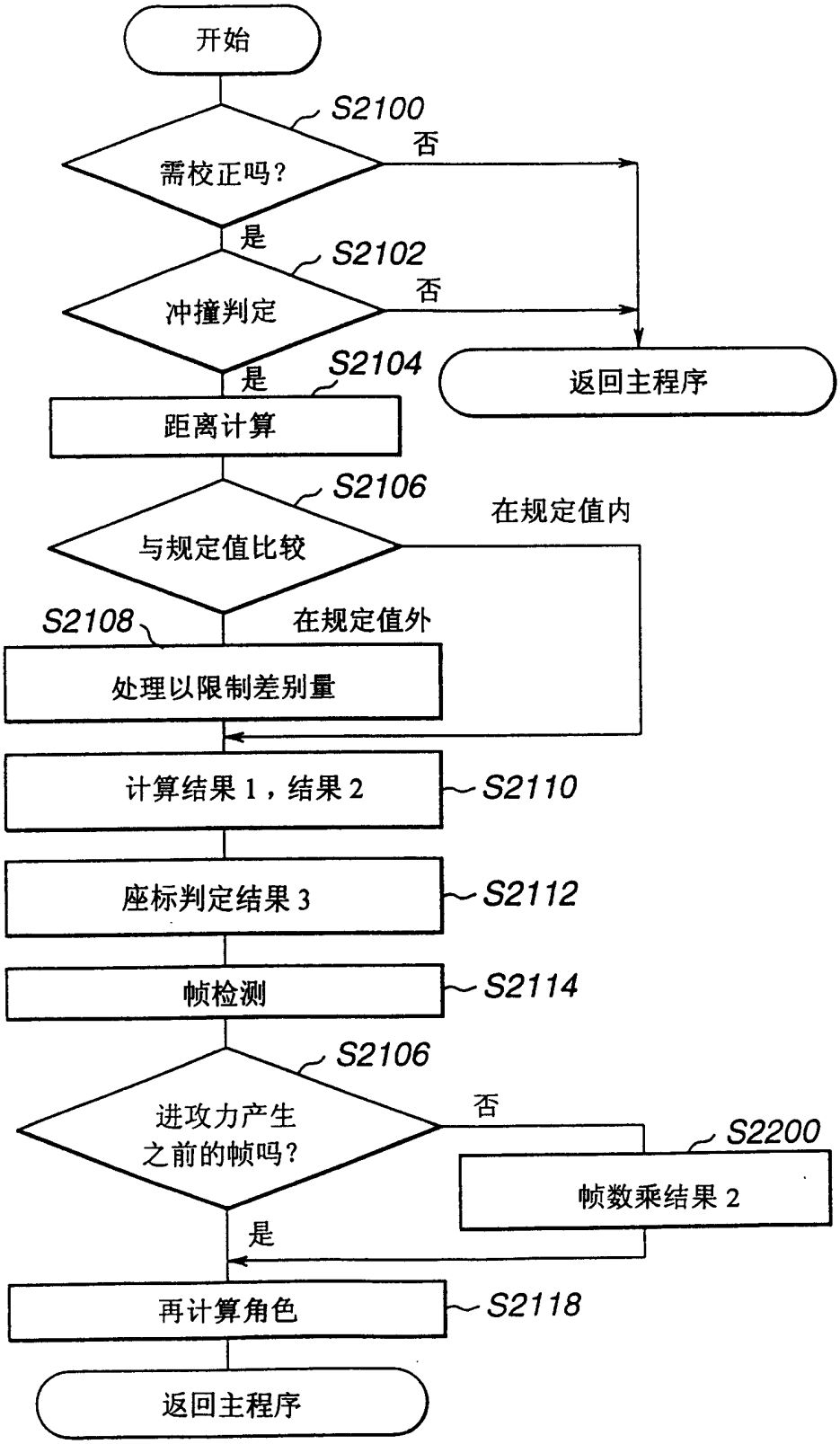

开始
S2100
需校正吗?
否
是
S2102
冲撞判定
否
是
S2104
返回主程序
距离计算
S2106
与规定值比较
在规定值内
S2108
在规定值外
处理以限制差别量
计算结果 1，结果 2
S2110
座标判定结果 3
S2112
帧检测
S2114
S2106
进攻力产生
之前的帧吗?
否
S2200
帧数乘结果 2
是
再计算角色
S2118
返回主程序

图 22

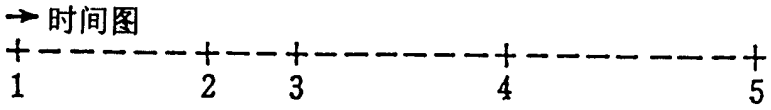


图 23

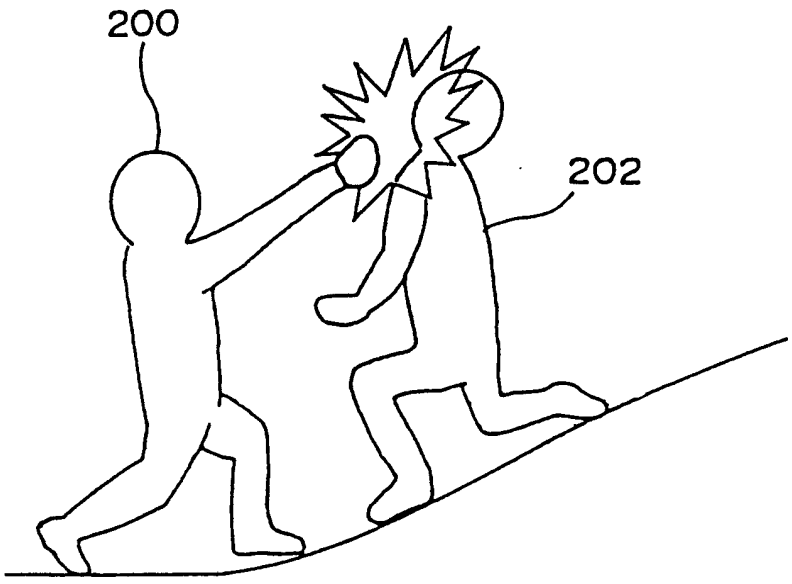


图 24

图25

图 26

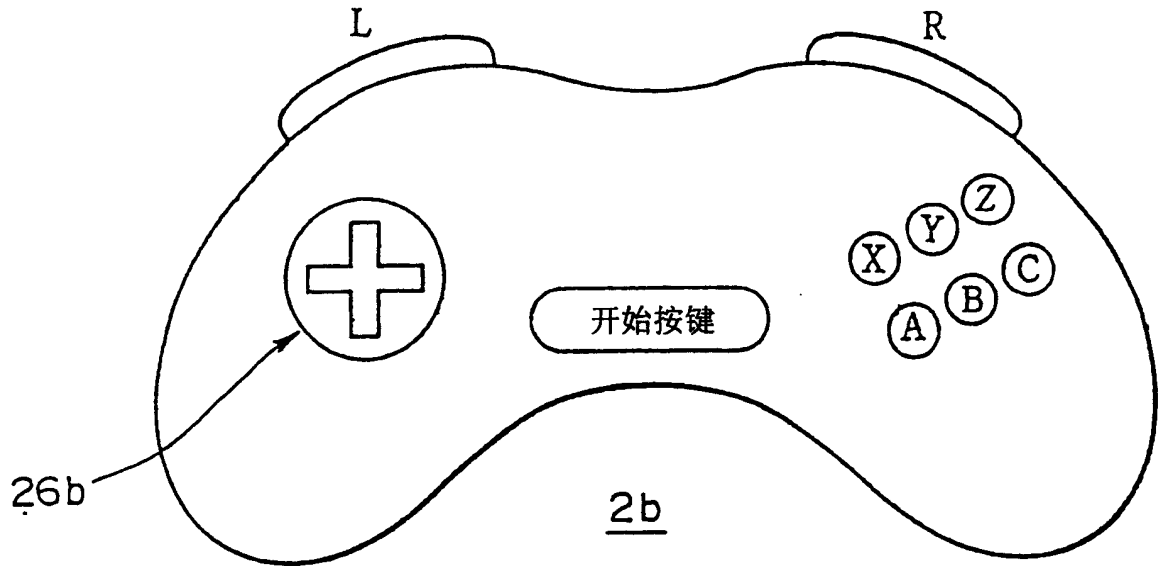
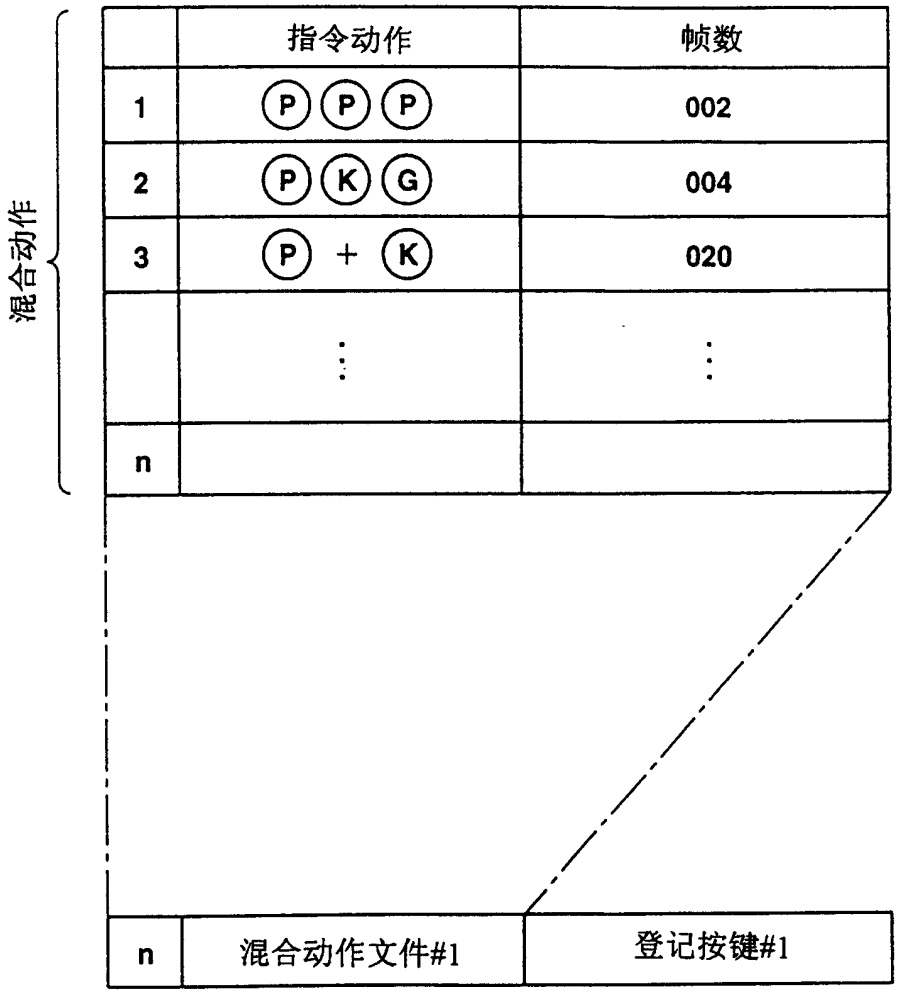


图27



Ⓟ (打击)：键 B

Ⓚ (踢)：键 C

Ⓖ (防卫)：键 A

+：同时推

图 28

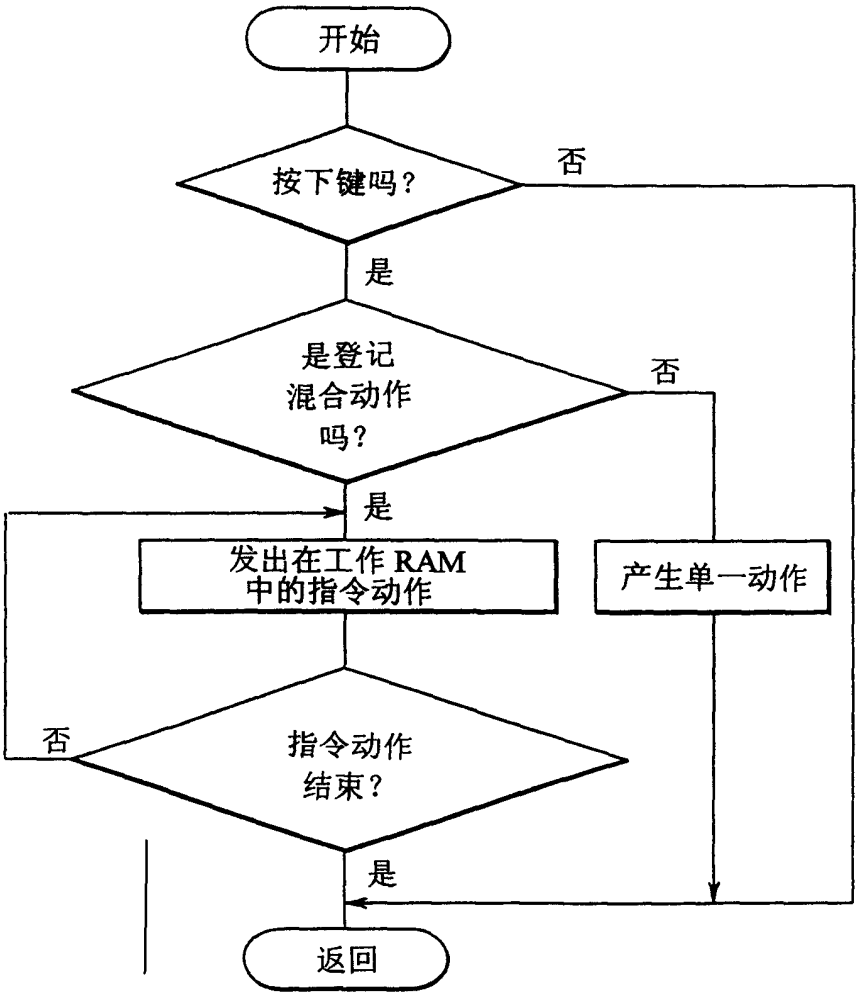


图 29

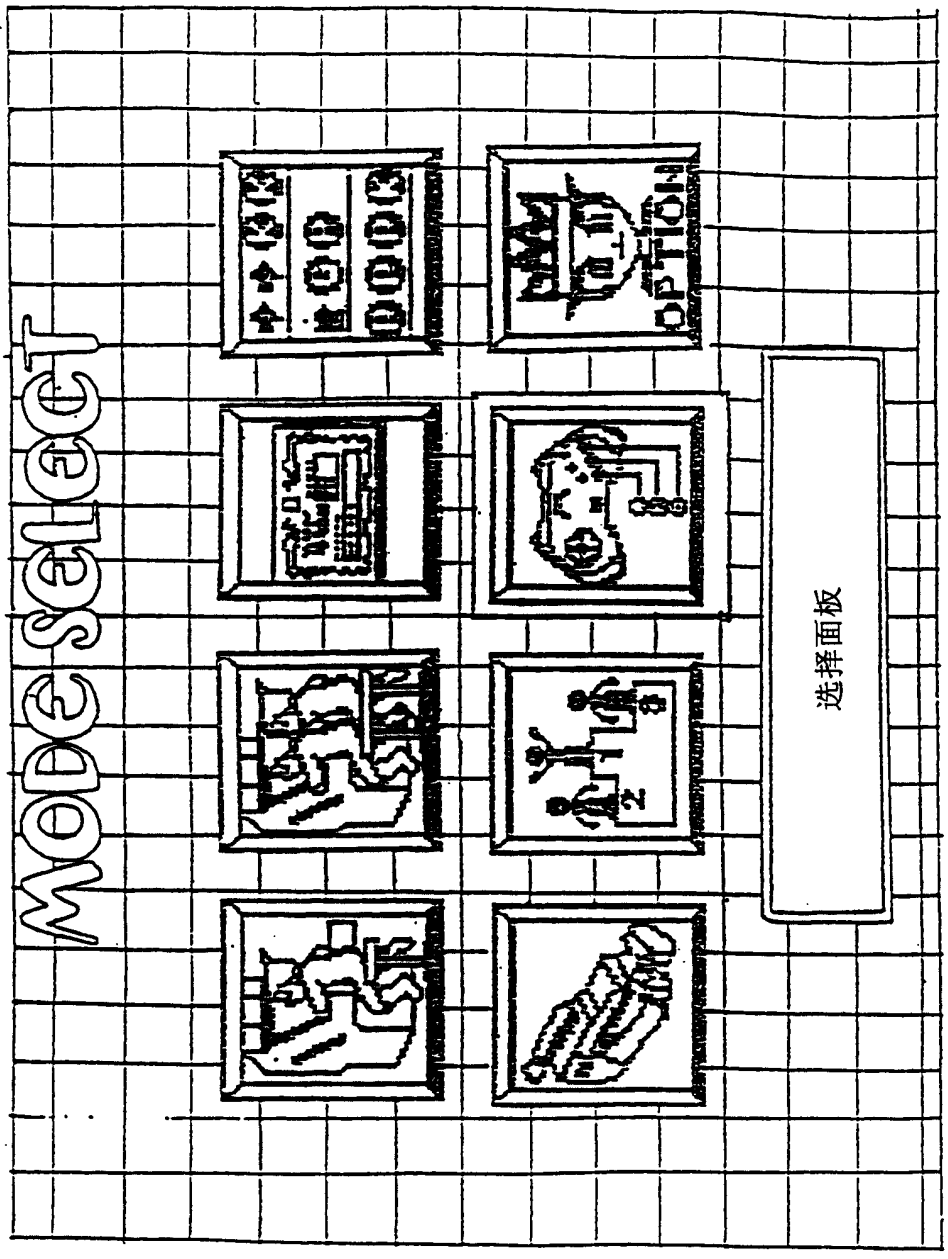


图30

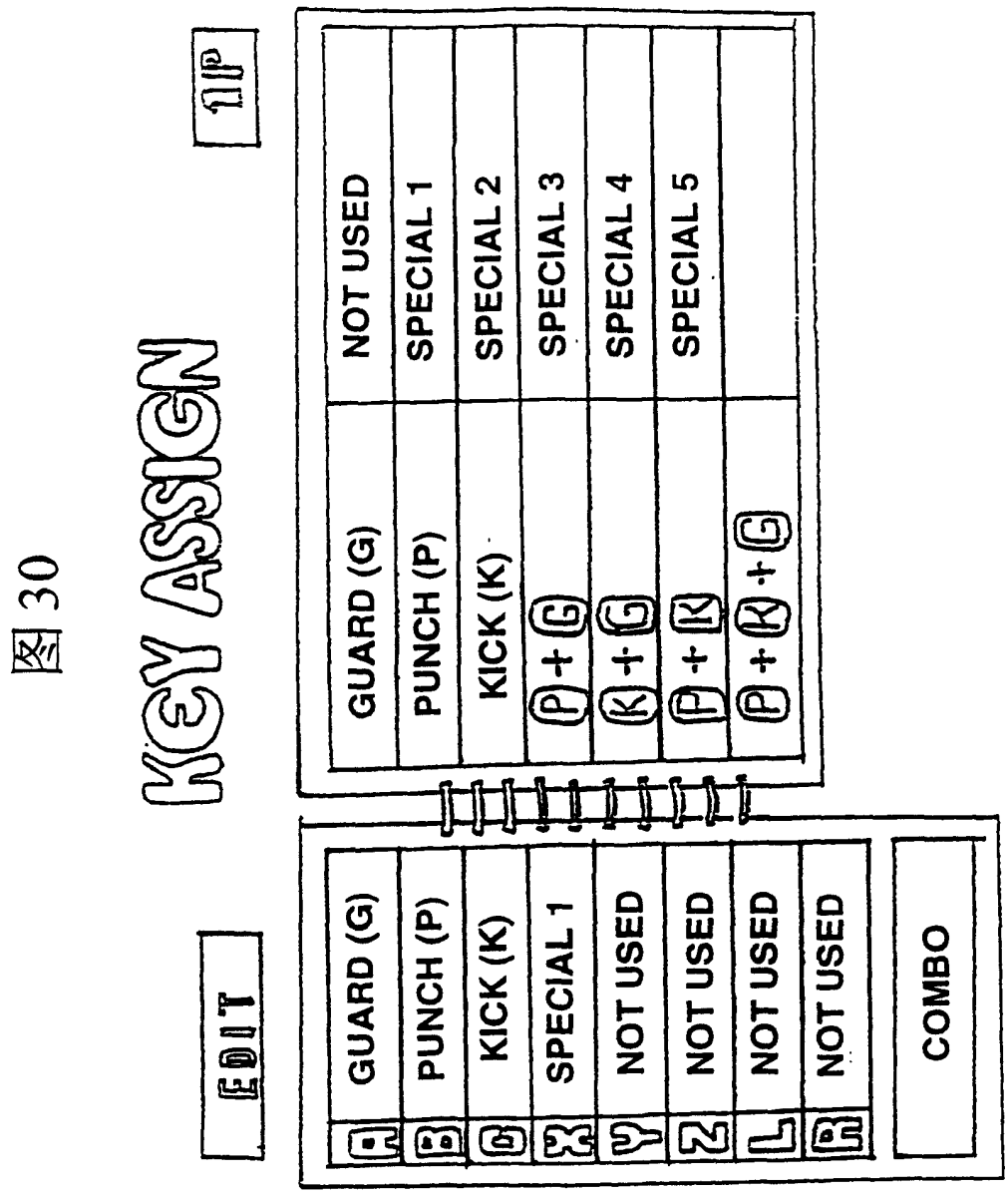

KEY ASSIGN
EDIT
1P
A GUARD (G)
B PUNCH (P)
C KICK (K)
X SPECIAL 1
Y NOT USED
Z NOT USED
L NOT USED
R NOT USED
COMBO
GUARD (G)
PUNCH (P)
KICK (K)
P + G
K + G
P + K
P + K + G
NOT USED
SPECIAL 1
SPECIAL 2
SPECIAL 3
SPECIAL 4
SPECIAL 5

图 31

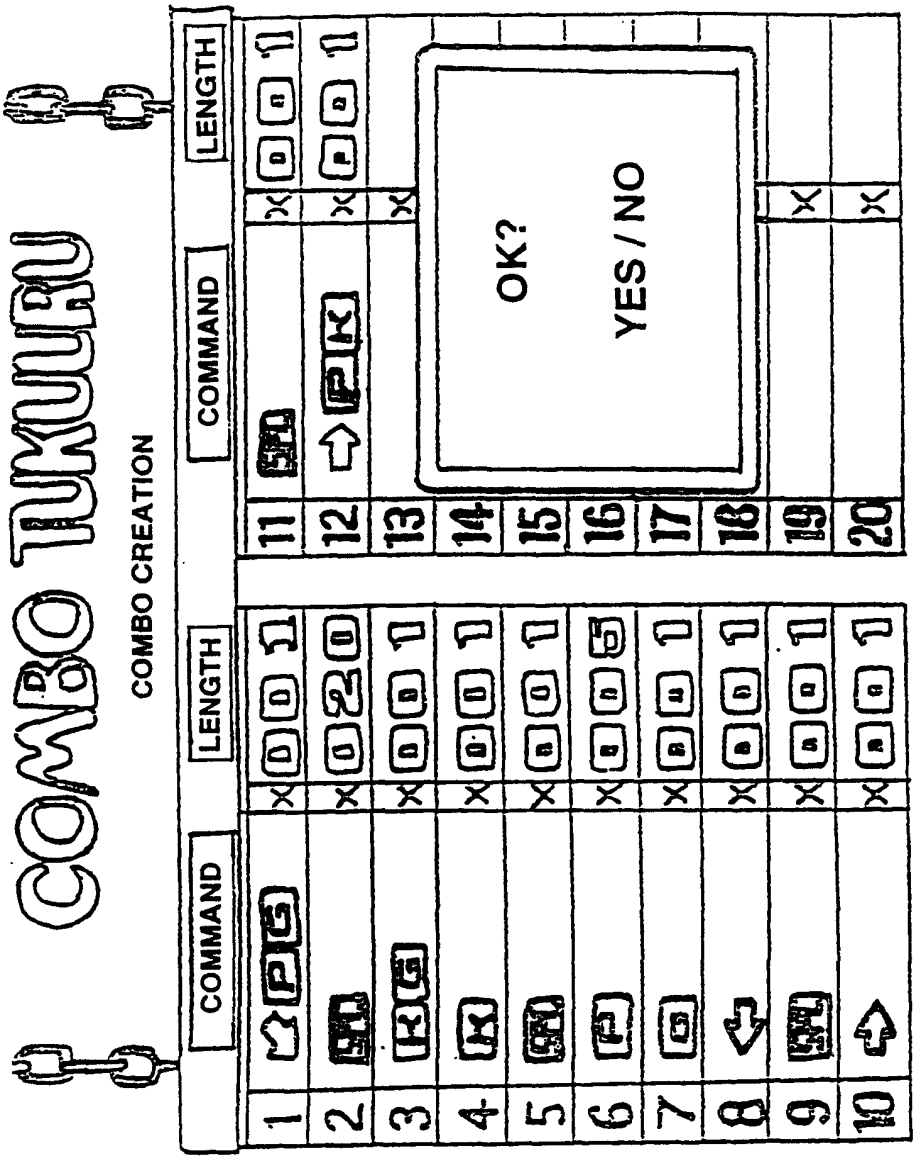

COMBO TUKURU
COMBO CREATION
COMMAND
LENGTH
COMMAND
LENGTH
OK?
YES / NO

图 32

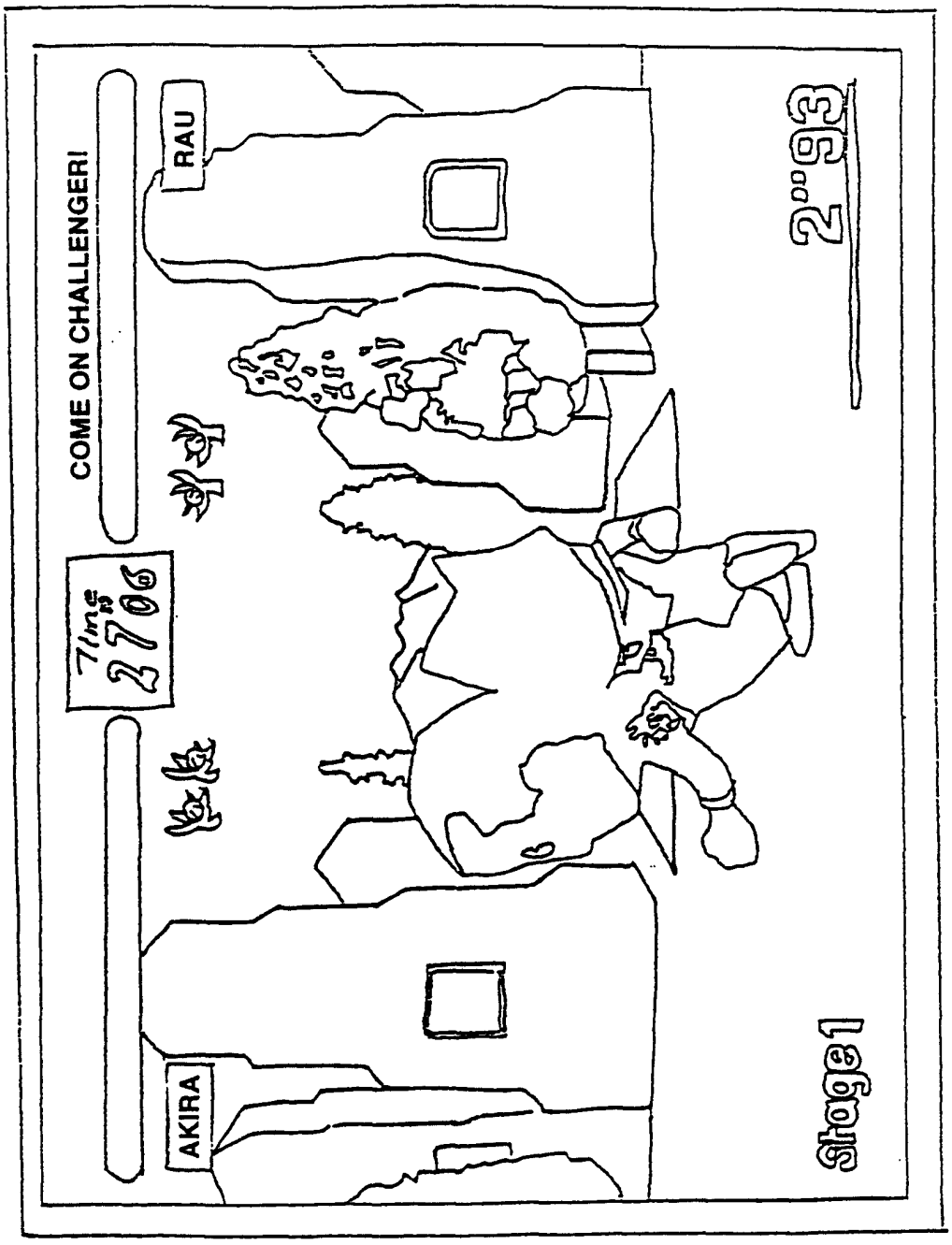


图 33

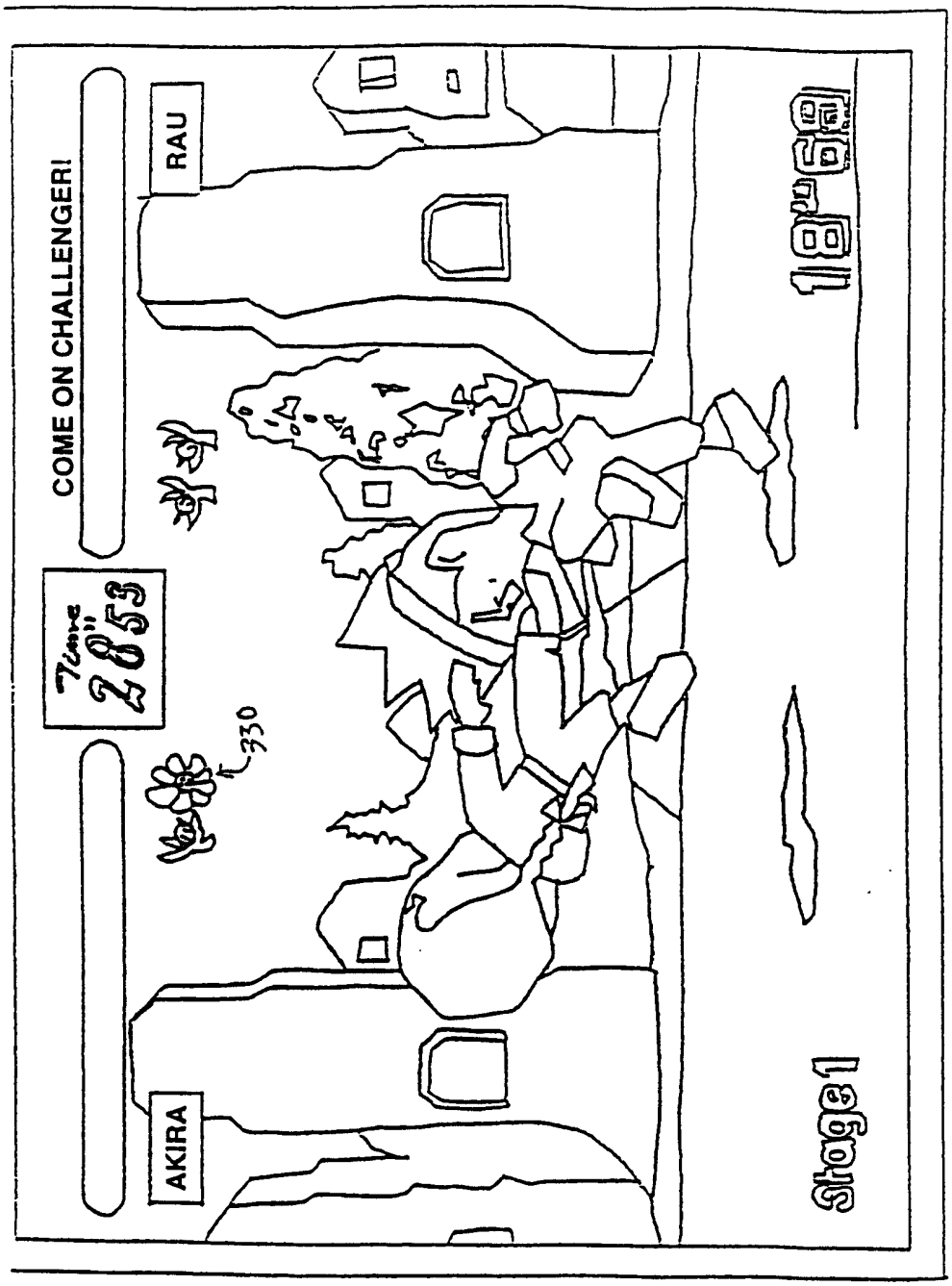


图 34

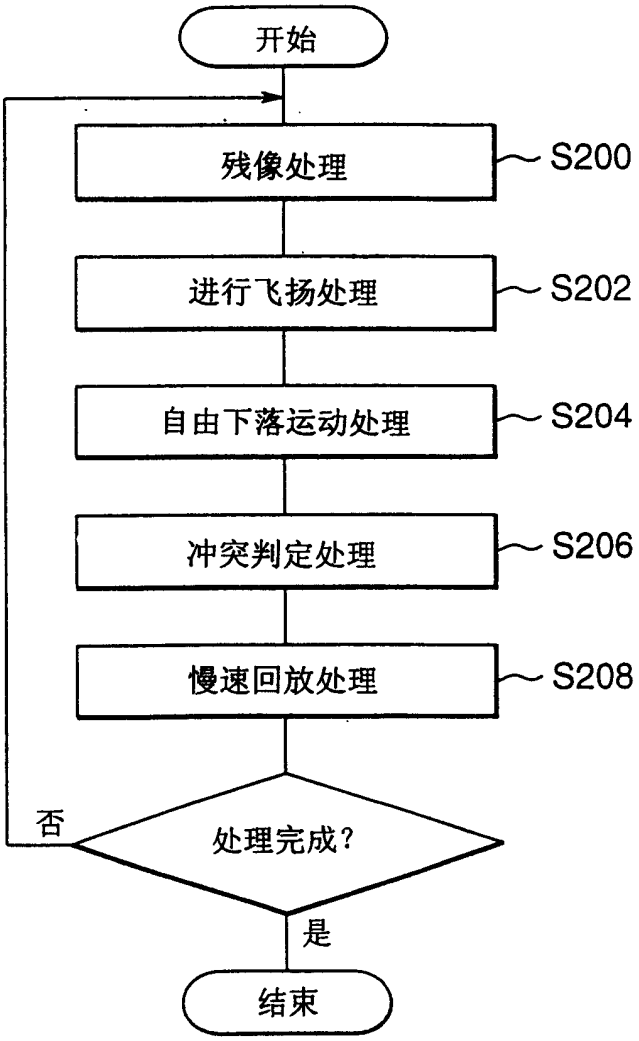

开始
残像处理
S200
进行飞扬处理
S202
自由下落运动处理
S204
冲突判定处理
S206
慢速回放处理
S208
处理完成?
否
是
结束

图 35

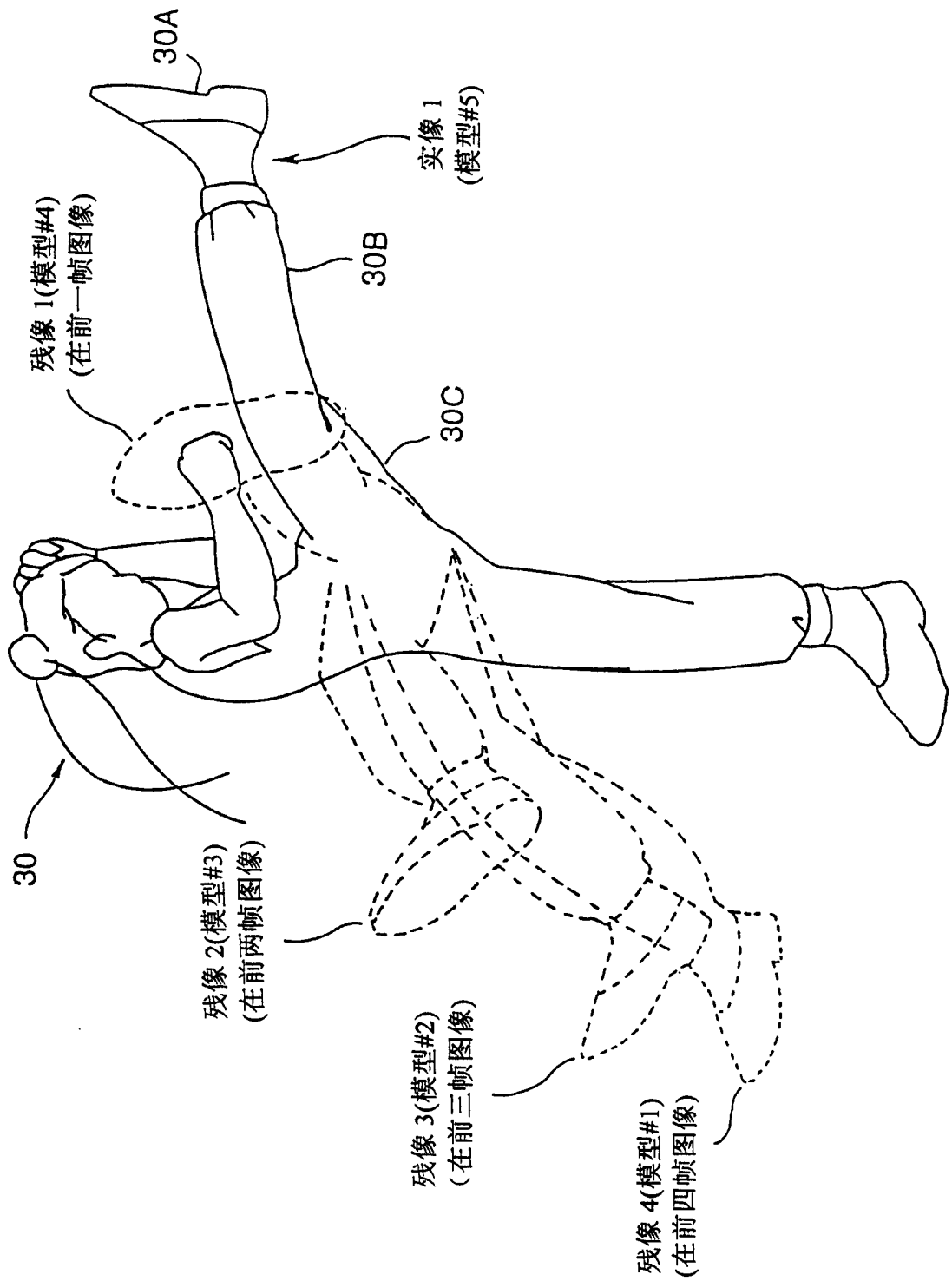

30A
实像 1
(模型#5)
30B
30C
残像 1(模型#4)
(在前一帧图像)
30
残像 2(模型#3)
(在前两帧图像)
残像 3(模型#2)
(在前三帧图像)
残像 4(模型#1)
(在前四帧图像)

图36

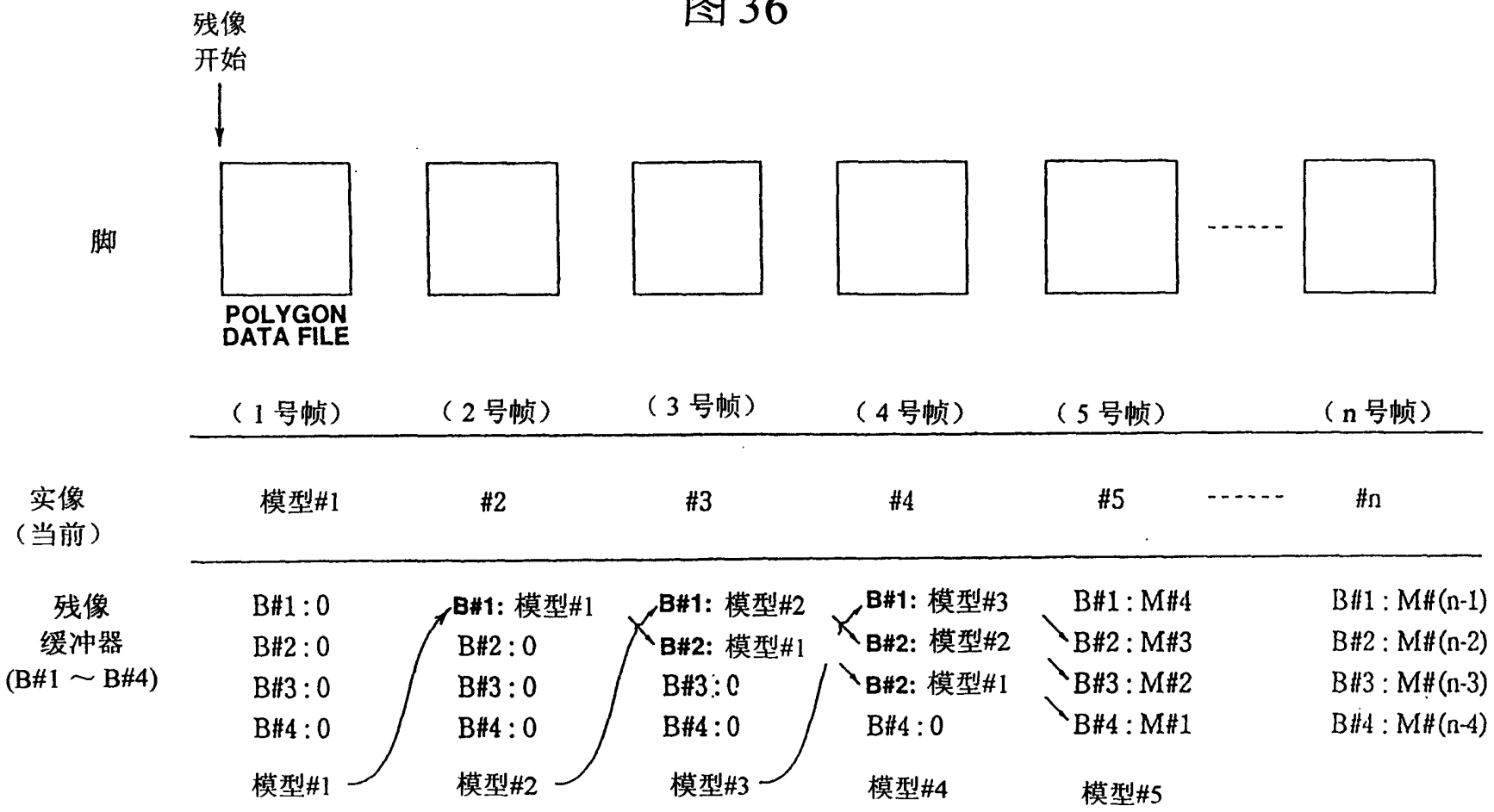

残像
开始
脚
POLYGON
DATA FILE
（1号帧）
（2号帧）
（3号帧）
（4号帧）
（5号帧）
（n号帧）
实像
（当前）
模型#1
#2
#3
#4
#5
#n
残像
缓冲器
(B#1 ～ B#4)
B#1 : 0
B#2 : 0
B#3 : 0
B#4 : 0
模型#1
B#1: 模型#1
B#2 : 0
B#3 : 0
B#4 : 0
模型#2
B#1: 模型#2
B#2: 模型#1
B#3 : 0
B#4 : 0
模型#3
B#1: 模型#3
B#2: 模型#2
B#2: 模型#1
B#4 : 0
模型#4
B#1 : M#4
B#2 : M#3
B#3 : M#2
B#4 : M#1
模型#5
B#1 : M#(n-1)
B#2 : M#(n-2)
B#3 : M#(n-3)
B#4 : M#(n-4)

图37

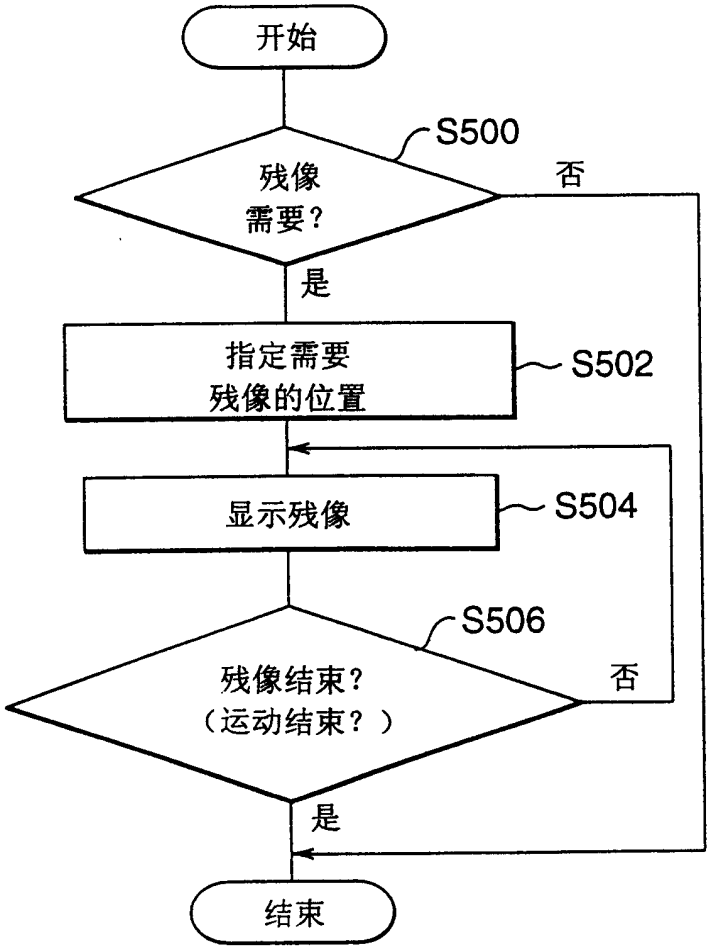


图 38

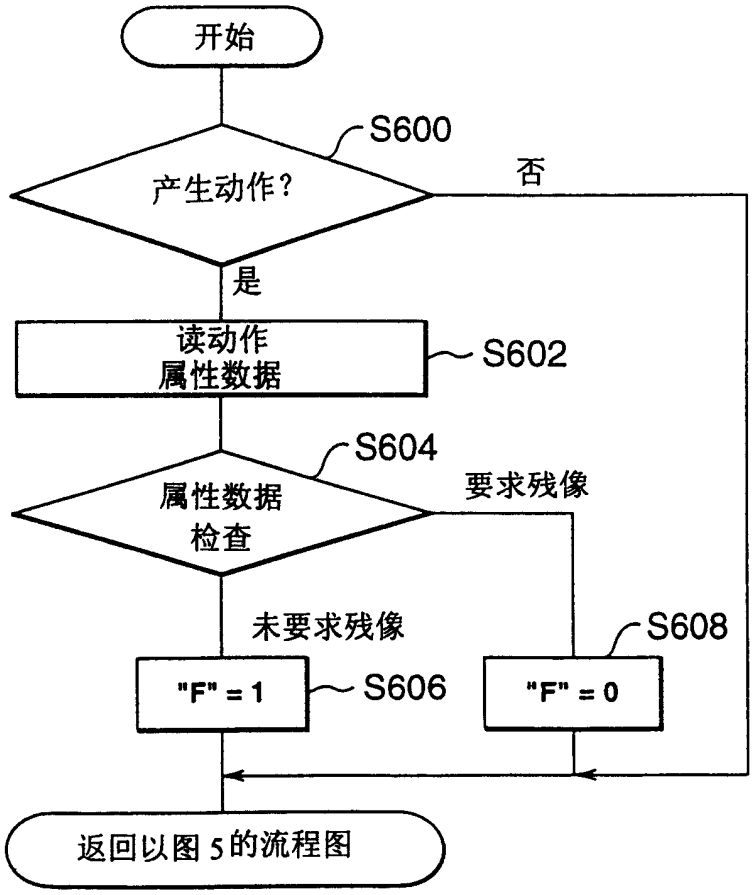

开始
S600
产生动作?
否
是
读动作
属性数据
S602
S604
属性数据
检查
要求残像
未要求残像
S608
"F" = 1
S606
"F" = 0
返回以图 5 的流程图

图 39

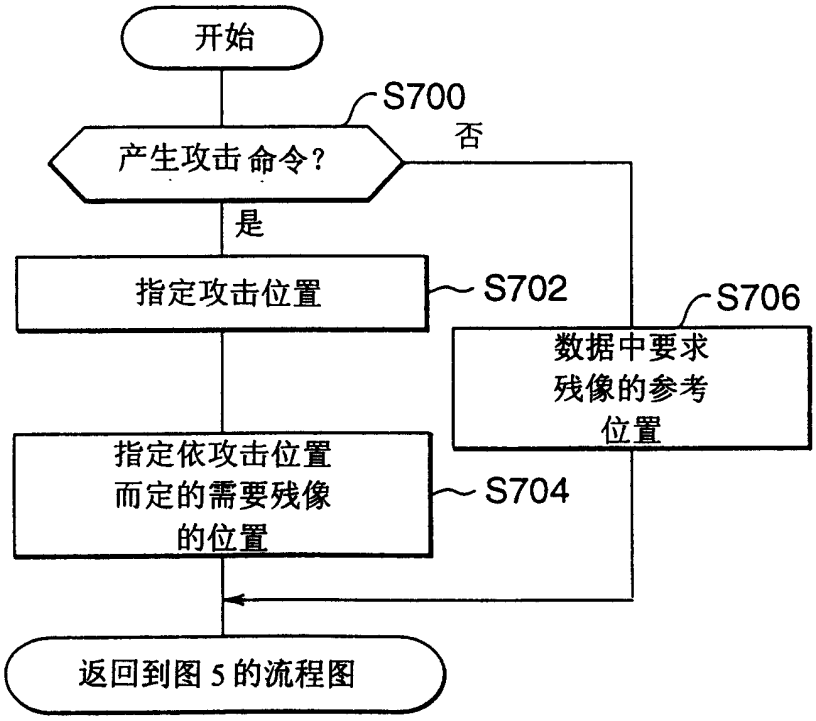

开始
S700
产生攻击命令?
否
是
指定攻击位置
S702
S706
数据中要求
残像的参考
位置
指定依攻击位置
而定的需要残像
的位置
S704
返回到图 5 的流程图

图 40

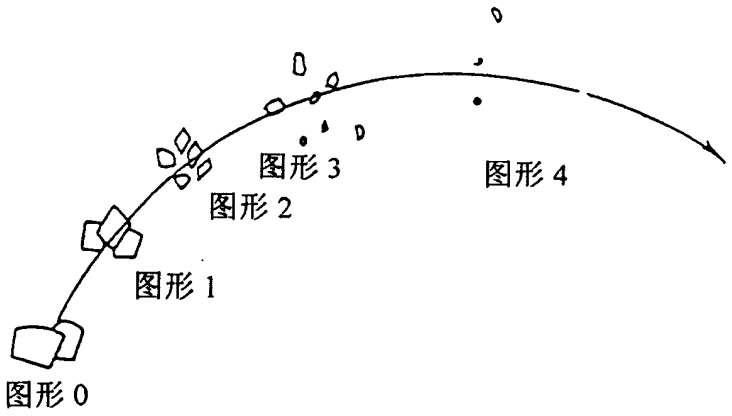

图形 3
图形 4
图形 2
图形 1
图形 0

图 41

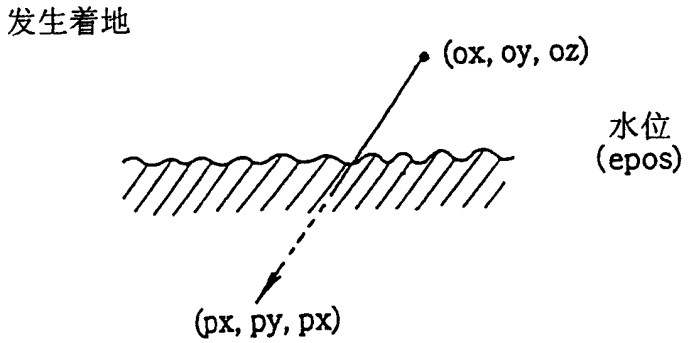

发生着地
(ox, oy, oz)
水位
(epos)
(px, py, px)

图 42

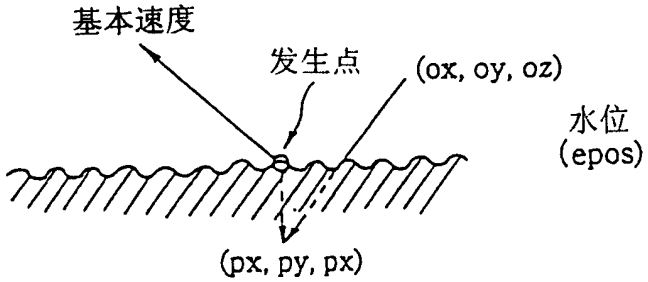

基本速度
发生点
(ox, oy, oz)
水位
(epos)
(px, py, px)

图 43

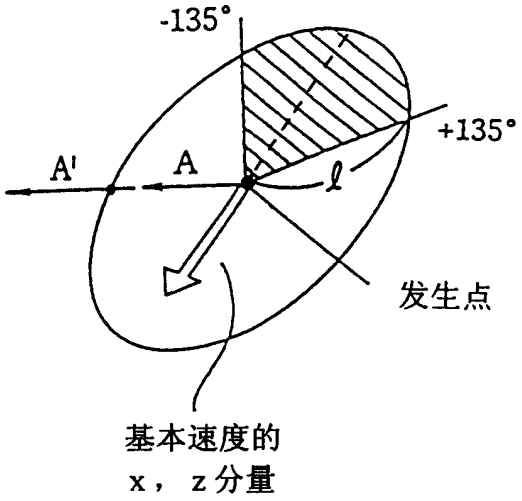


图 44

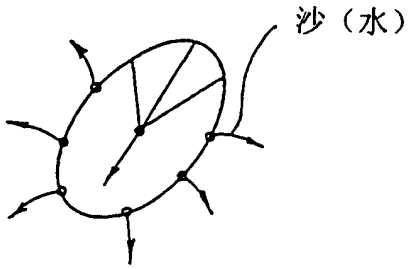


图 45

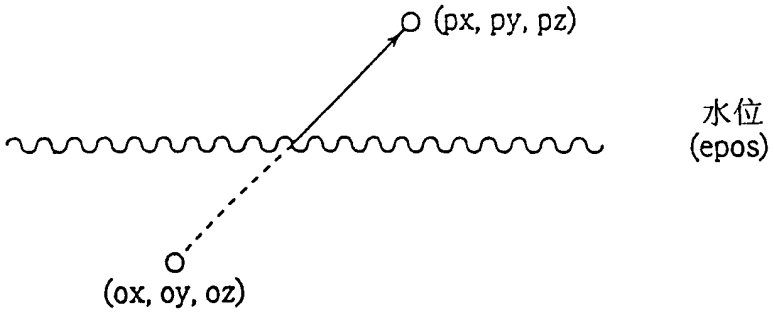
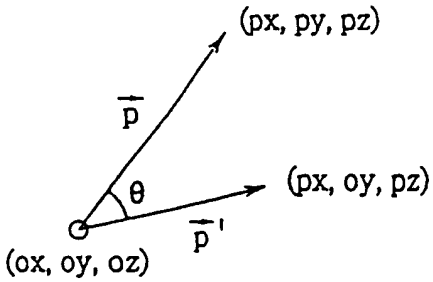
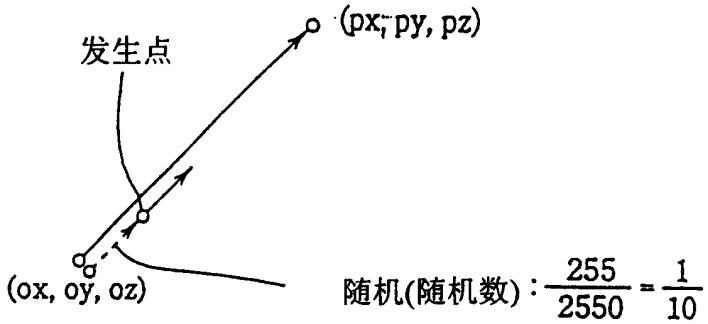


图 46



$$\cos\theta = \frac{\vec{p}\ \vec{p}'}{|\vec{p}|\ |\vec{p}'|}$$

图47



发生点 ：$(ox + spdx \times \frac{1}{10},\ oy + spdy \times \frac{1}{10},\ oz + spdz \times \frac{1}{10})$

图 48

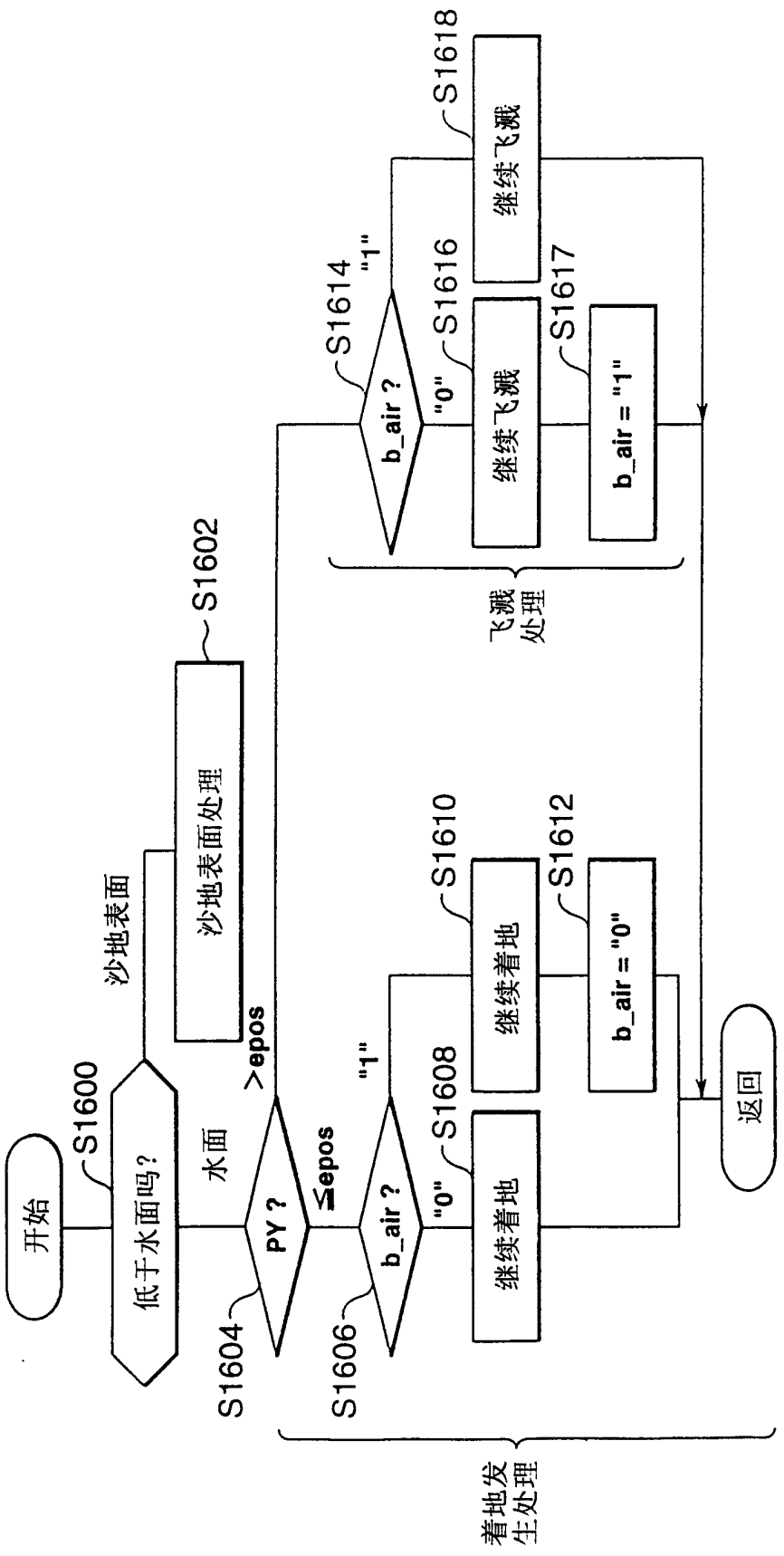

开始
S1600
低于水面吗?
沙地表面
水面
S1602
沙地表面处理
S1604
PY ?
>epos
≤epos
S1606
b_air ?
"1"
"0"
S1608
继续着地
S1610
继续着地
S1612
b_air = "0"
S1614
b_air ?
"1"
"0"
S1616
继续飞溅
S1617
b_air = "1"
S1618
继续飞溅
飞溅处理
着地发生处理
返回

图 49

图 50

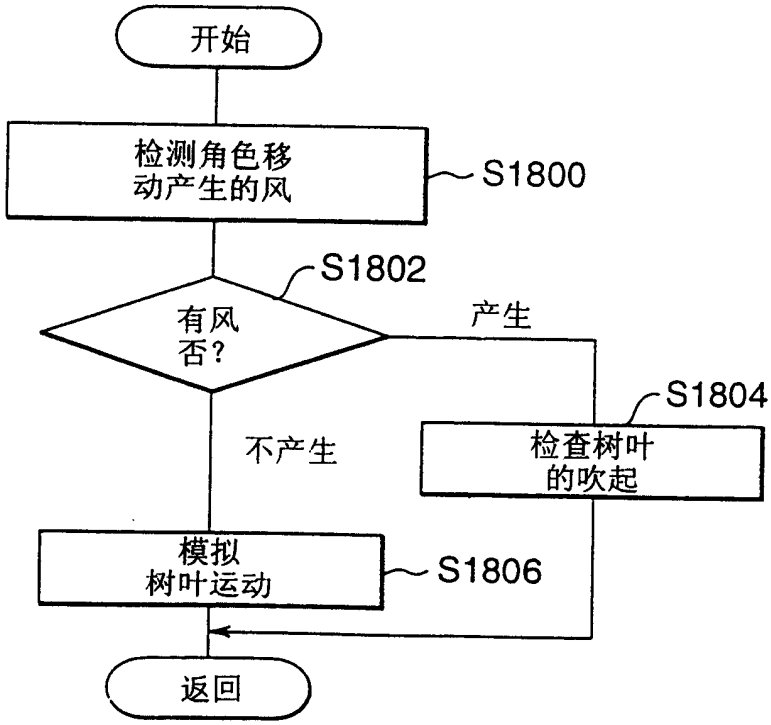

开始
检测角色移动产生的风
S1800
S1802
有风否?
产生
不产生
S1804
检查树叶的吹起
模拟树叶运动
S1806
返回

图 51

(1) $M = \sqrt{(px-ox)^2+(py-oy)^2+(pz-oz)^2}$

(2) $M' = \sqrt{(px-ox)^2+(pz-oz)^2}$

风产生位置（ox,oy,oz）
风矢量
w_xspd = (px-ox)·0.6
w_yspd = M'
w_zspd = (pz-oz)·0.6

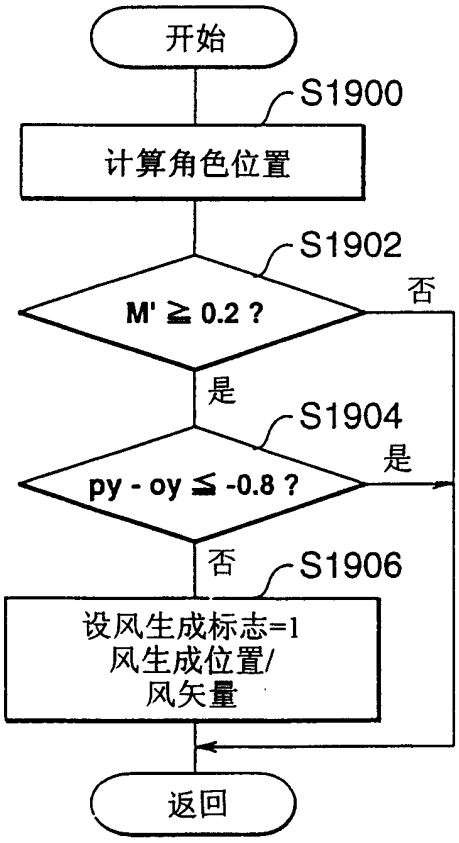


图 52

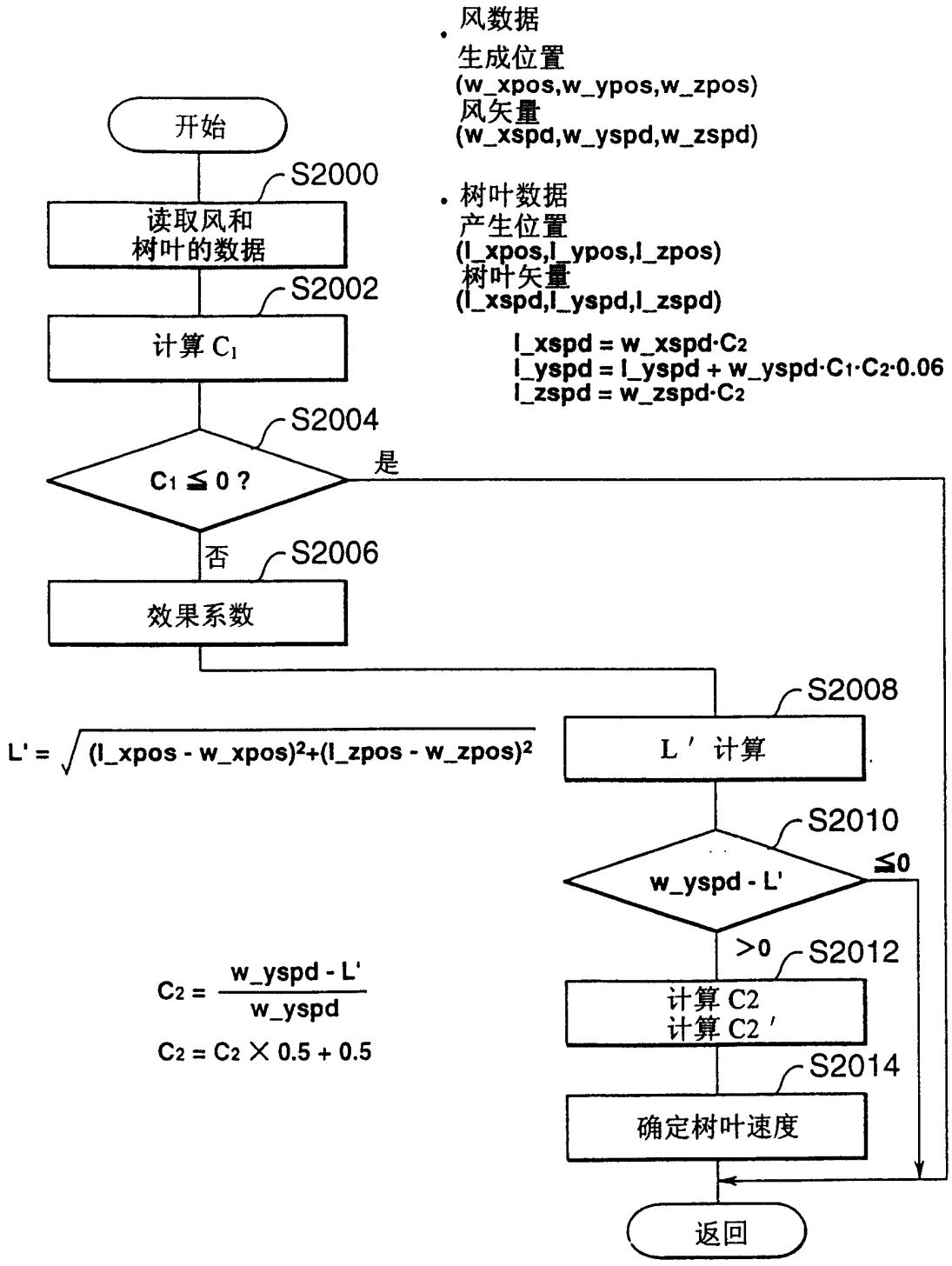
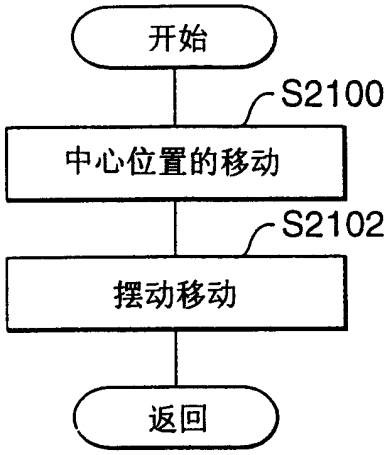

• 风数据
生成位置
(w_xpos,w_ypos,w_zpos)
风矢量
(w_xspd,w_yspd,w_zspd)
• 树叶数据
产生位置
(l_xpos,l_ypos,l_zpos)
树叶矢量
(l_xspd,l_yspd,l_zspd)
l_xspd = w_xspd·C2
l_yspd = l_yspd + w_yspd·C1·C2·0.06
l_zspd = w_zspd·C2
开始
S2000
读取风和树叶的数据
S2002
计算 C1
S2004
C1 ≦ 0 ?
是
否
S2006
效果系数
S2008
L' 计算
L' = √(l_xpos - w_xpos)²+(l_zpos - w_zpos)²
S2010
w_yspd - L'
≦0
>0
S2012
计算 C2
计算 C2'
C2 = (w_yspd - L') / w_yspd
C2 = C2 × 0.5 + 0.5
S2014
确定树叶速度
返回

图 53



S2100

在空中时

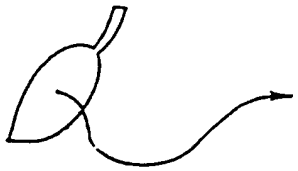
l_xspd = l_xspd × 0.87
l_yspd = l_yspd × 0.02
l_zspd = l_zspd × 0.87

已着地

l_xspd = l_xspd × 0.50
l_yspd = 0
l_zspd = l_zspd × 0.50

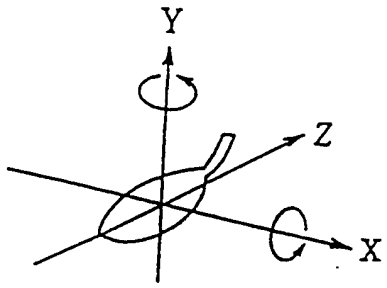
图 54

(1)



在风和重力的影响
下树叶中心位置
移动的运动

(2)



树叶摆动

图 55

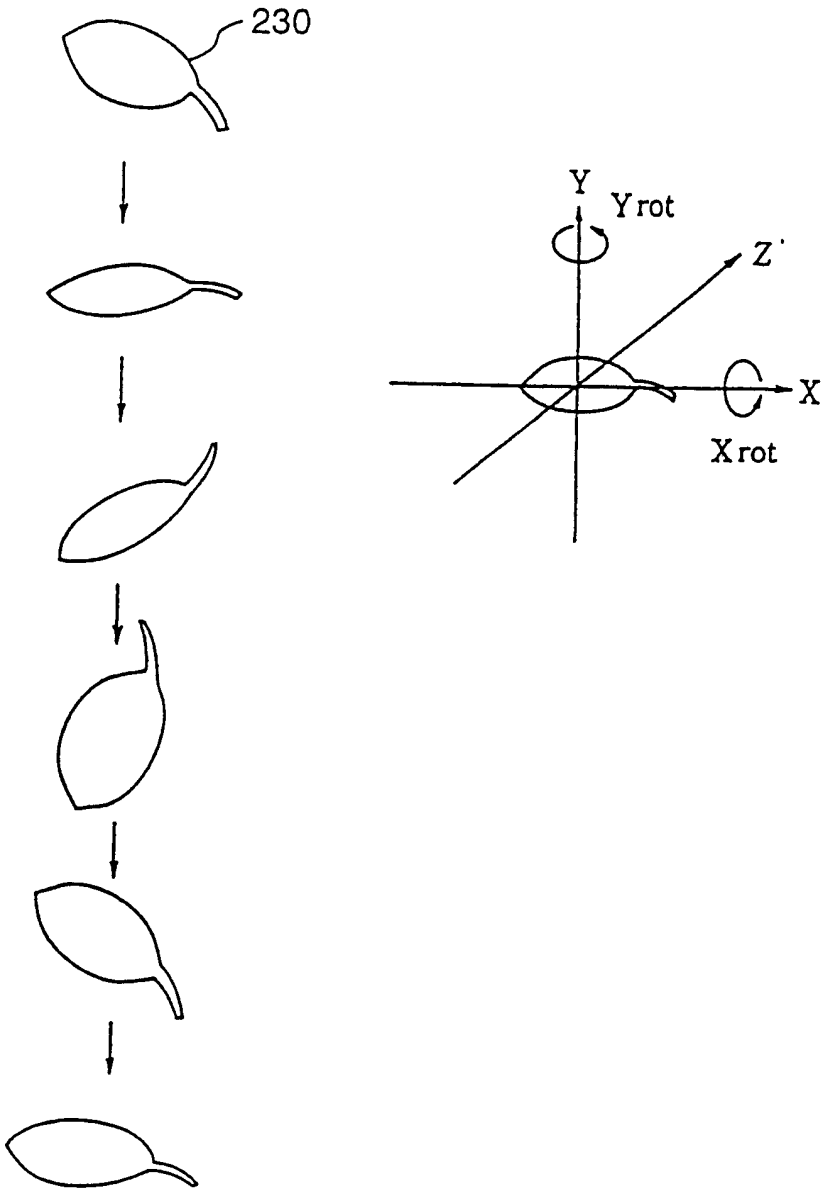


图 56

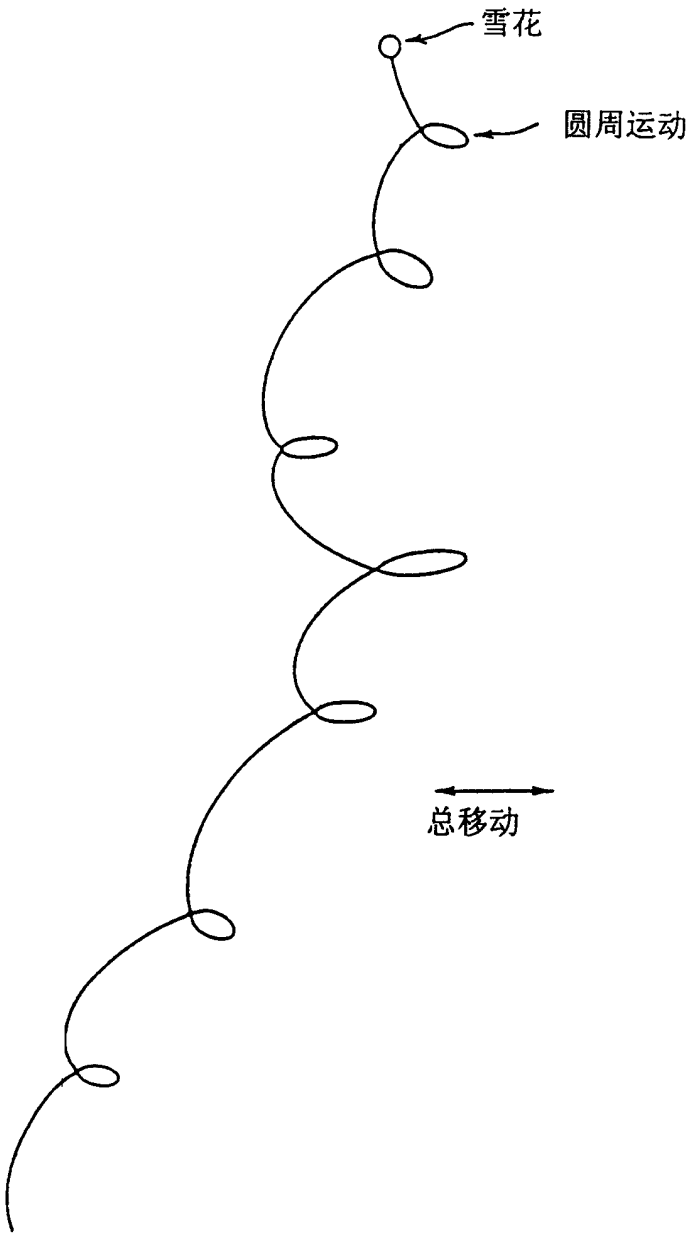
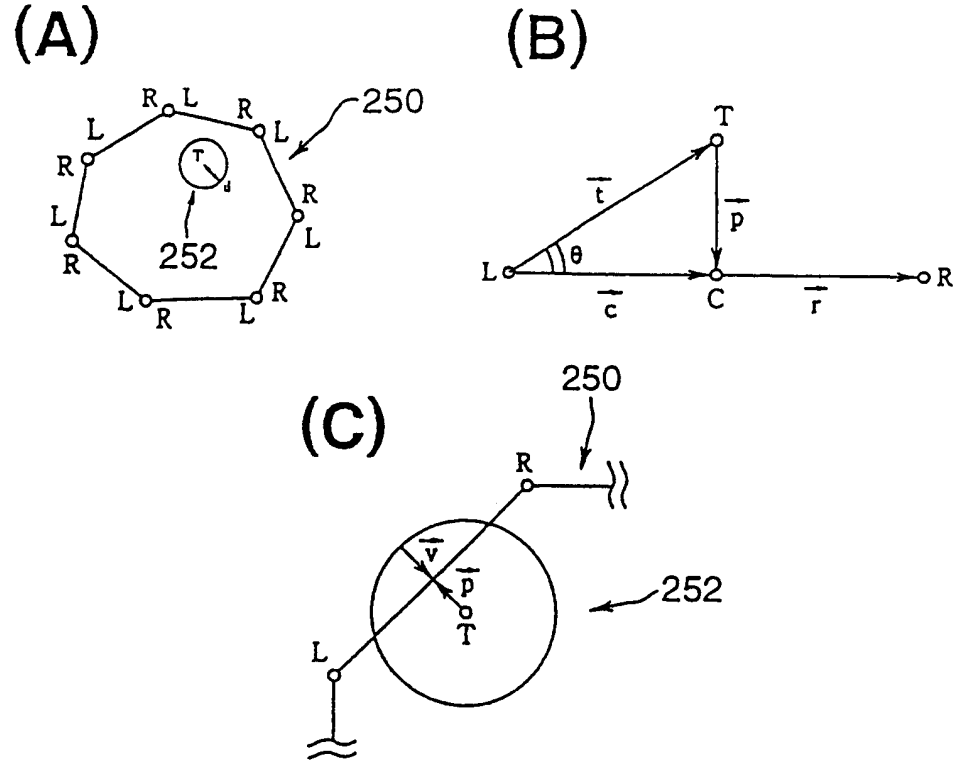


图 57



$$\vec{r} = \vec{R} - \vec{L}$$

$$\vec{t} = \vec{T} - \vec{L}$$

$$\vec{p} = \vec{c} + \vec{L} - \vec{T}$$

$$cx = rx \cdot \frac{|\vec{c}|}{|\vec{r}|} \quad , \quad cz = rz \cdot \frac{|\vec{c}|}{|\vec{r}|}$$

其中

$$|\vec{c}| = |\vec{t}| \cos\theta$$

根据内积的定义

$$|\vec{r}| \cdot |\vec{t}| \cos\theta = rx\,tx + rz\,tz$$

$$\frac{|\vec{c}|}{|\vec{r}|} = \frac{rx\,tx + rz\,tz}{|\vec{r}|^2} = \frac{rx\,tx + rz\,tz}{rx^2 + rz^2}$$

其中

$$0 < \frac{|\vec{c}|}{|\vec{r}|} < 1 \quad \text{——} \quad (3)$$

如果点 T 位于直线 LR 的正区，取
单位矢量作为排除矢量 V， P

$$\vec{v} = (|\vec{p}| + d) \cdot \vec{pe} \quad \text{——} \quad (1)$$

如果点 T 在直线 LR 的负区

$$\left.\begin{array}{ll} |\vec{p}| - d < 0: & \vec{v} = (|\vec{p}| - d) \cdot (-\vec{pe}) \\ |\vec{p}| - d \geqq 0: & \vec{v} = \vec{0} \end{array}\right\} (2)$$

图 58

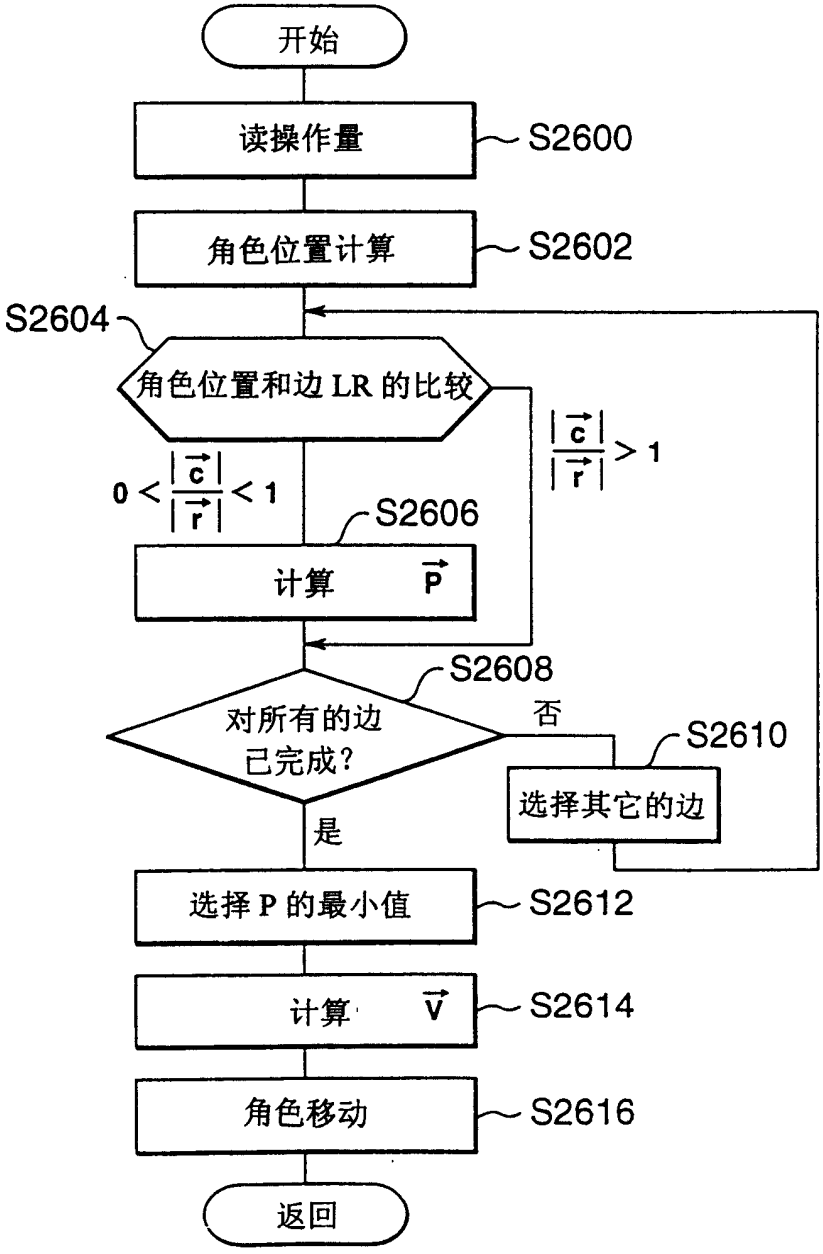

开始
读操作量
S2600
角色位置计算
S2602
S2604
角色位置和边 LR 的比较
$\frac{|\vec{c}|}{|\vec{r}|} > 1$
$0 < \frac{|\vec{c}|}{|\vec{r}|} < 1$
S2606
计算 $\vec{P}$
S2608
对所有的边已完成?
否
S2610
选择其它的边
是
选择 P 的最小值
S2612
计算 $\vec{V}$
S2614
角色移动
S2616
返回

图 59

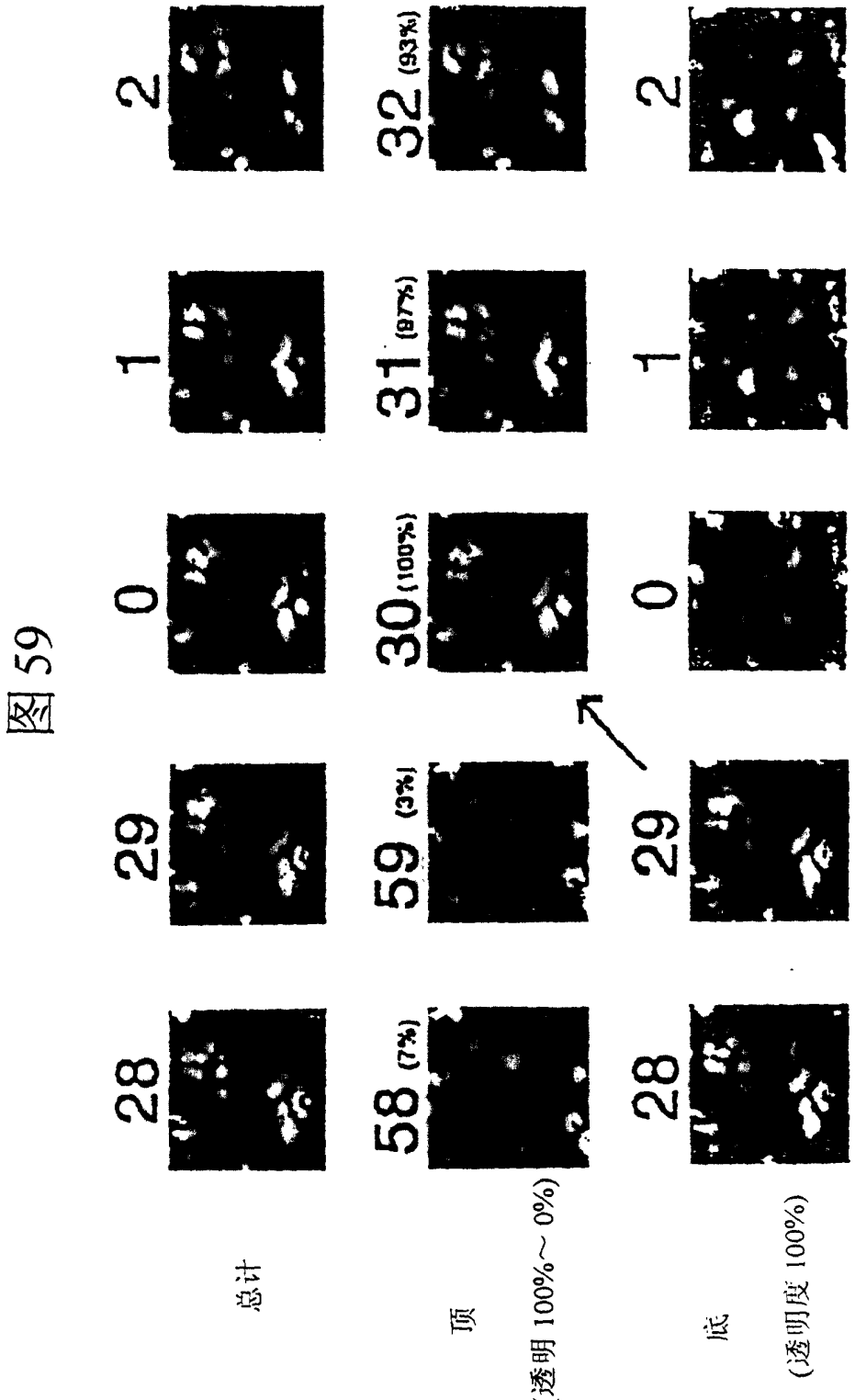


图 60

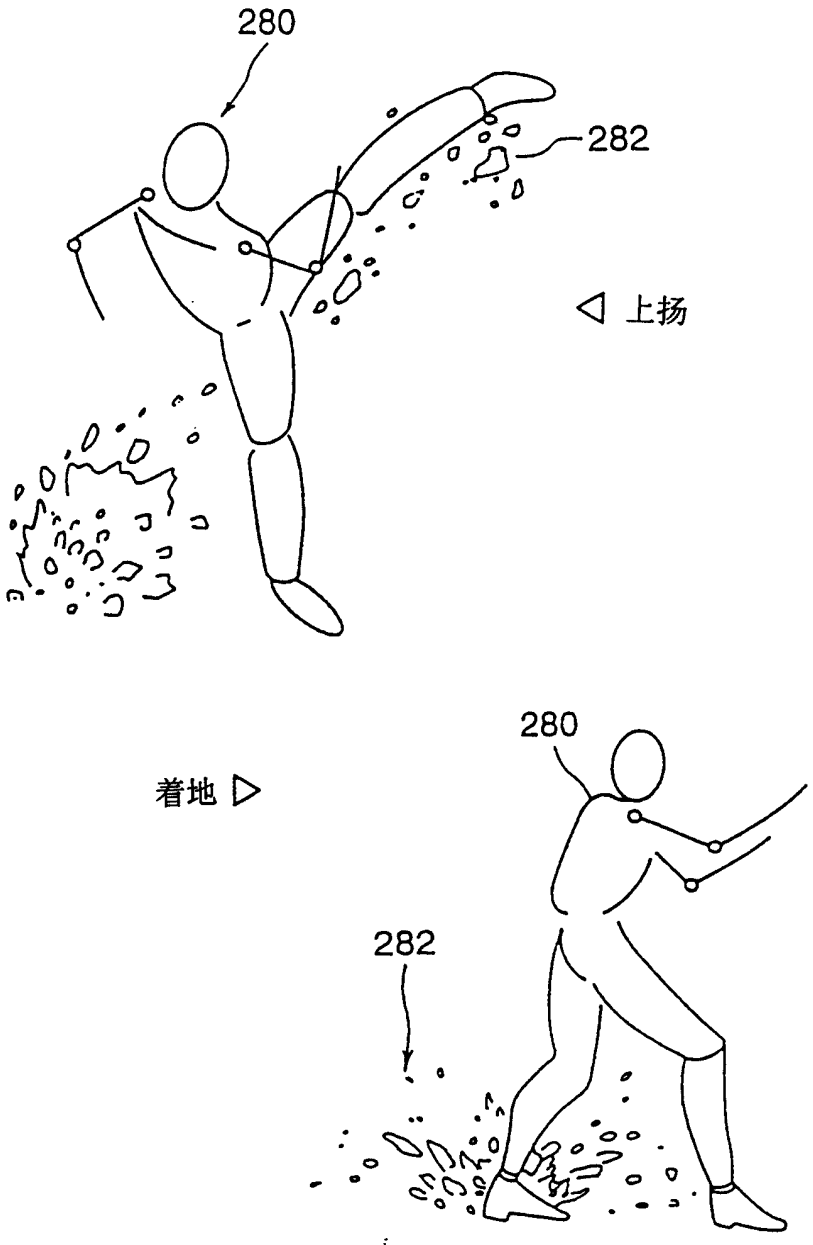


图61

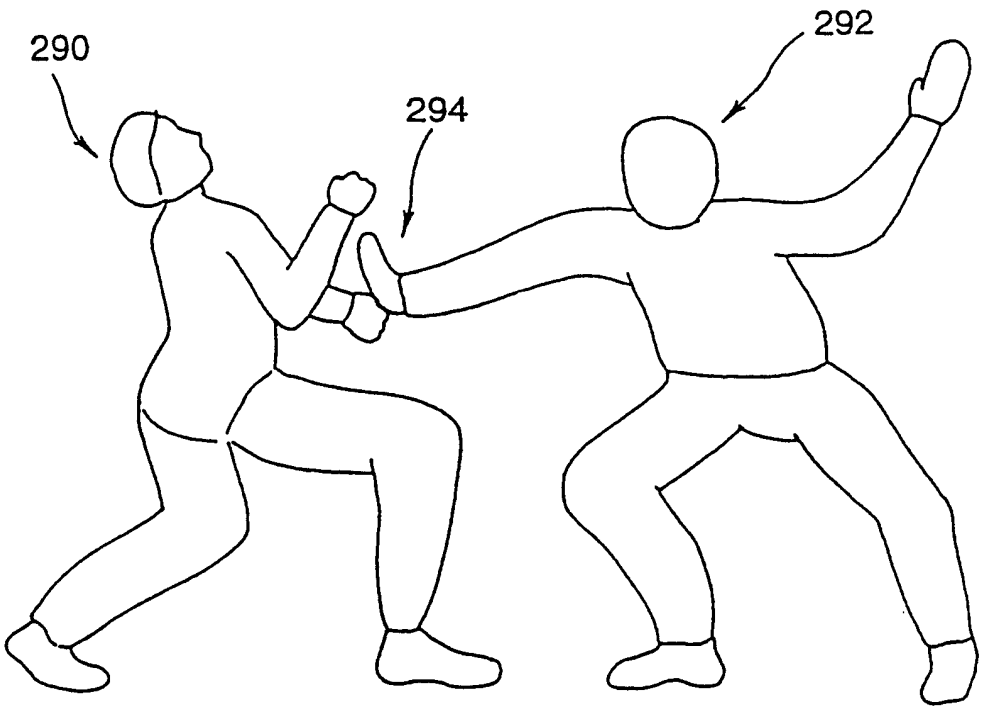


图 62

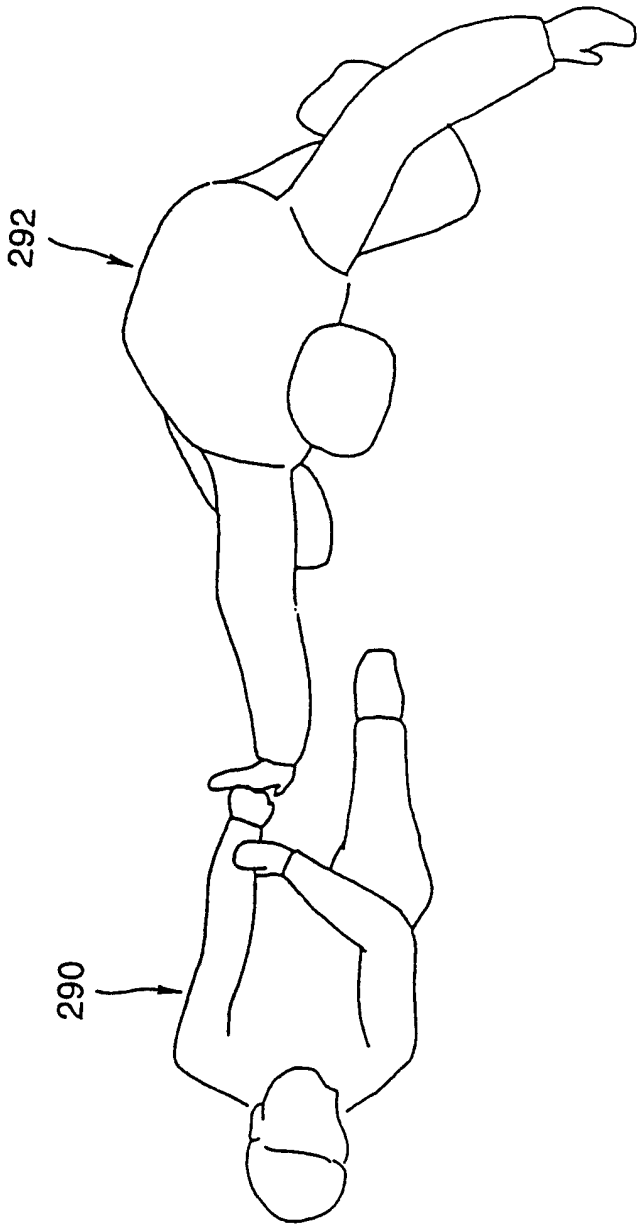


图 63

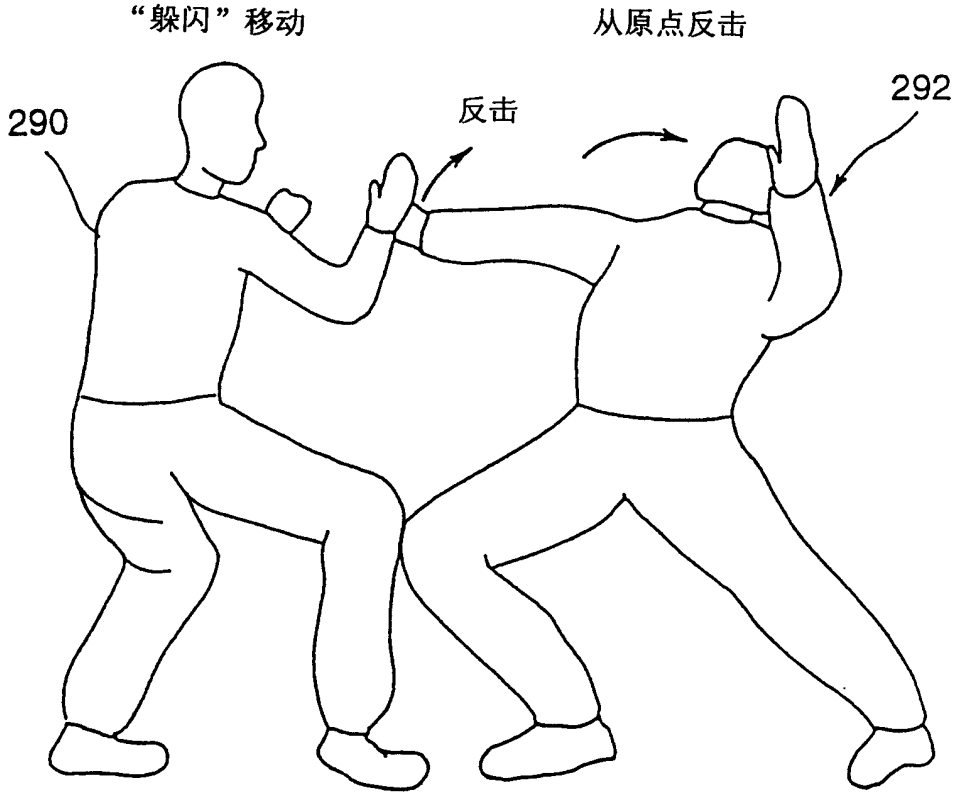


图 64

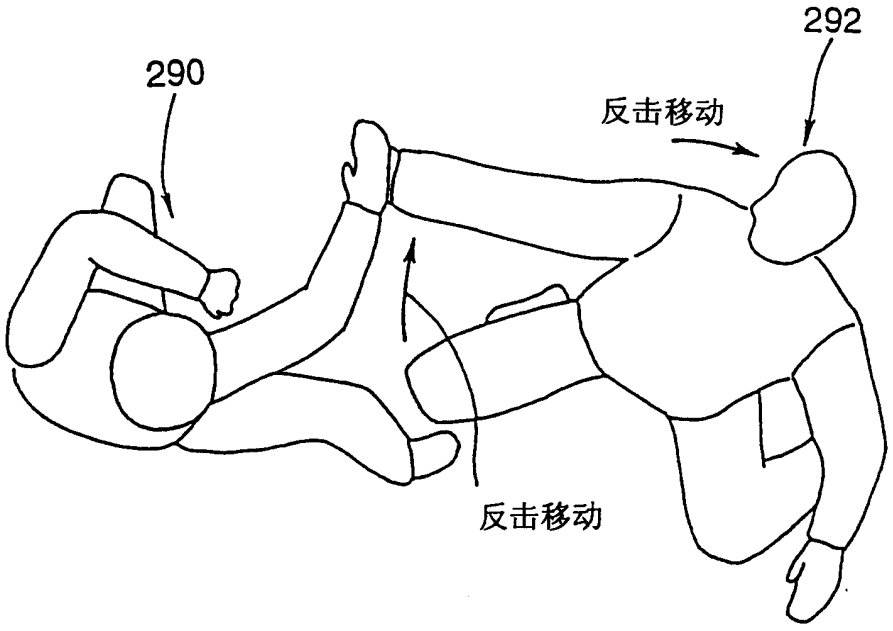


图 65

图 66

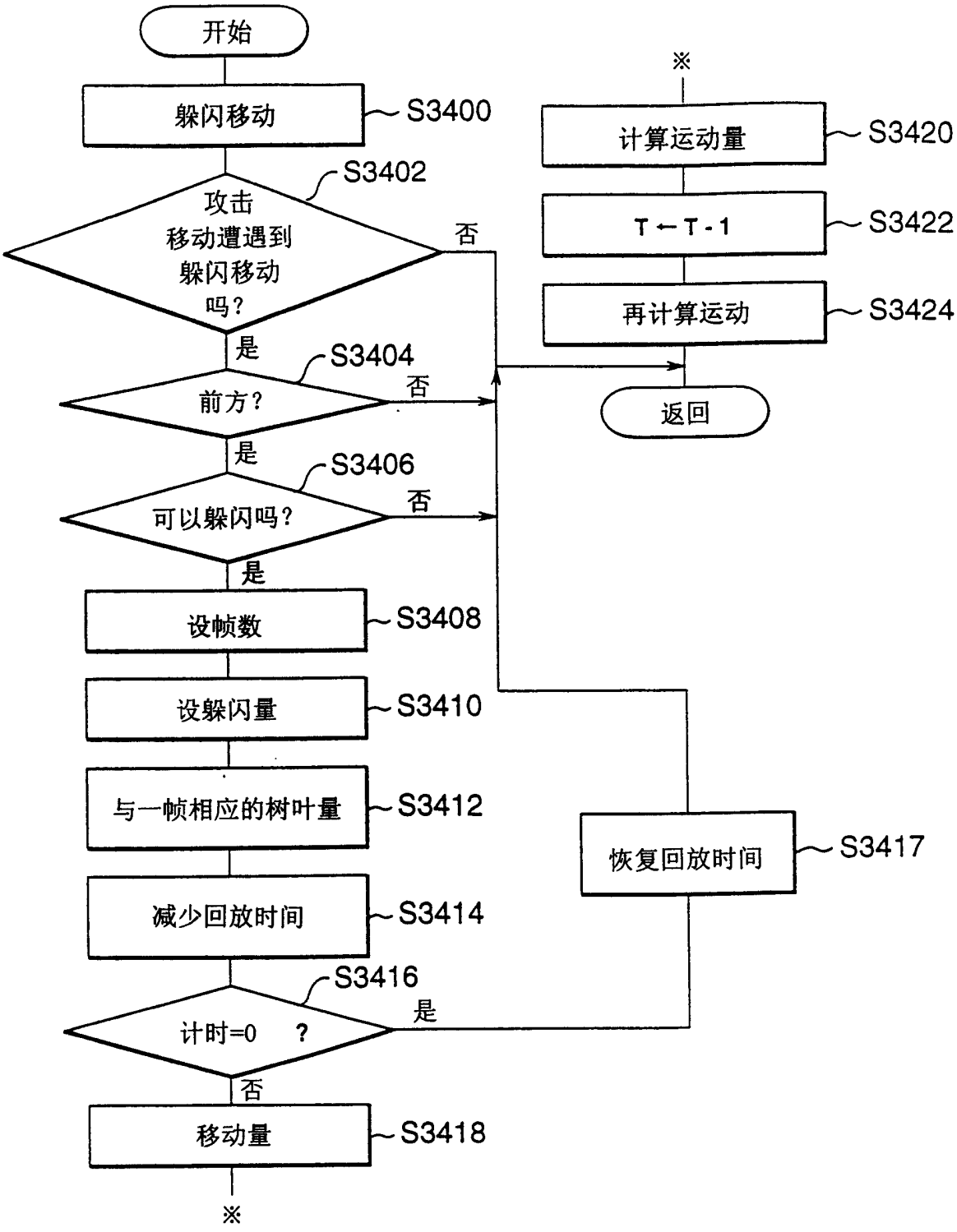

开始
躲闪移动
S3400
攻击
移动遭遇到
躲闪移动
吗？
S3402
否
是
前方？
S3404
否
是
可以躲闪吗？
S3406
否
是
设帧数
S3408
设躲闪量
S3410
与一帧相应的树叶量
S3412
减少回放时间
S3414
计时=0 ？
S3416
是
否
移动量
S3418
※
※
计算运动量
S3420
T←T-1
S3422
再计算运动
S3424
返回
恢复回放时间
S3417

图 67

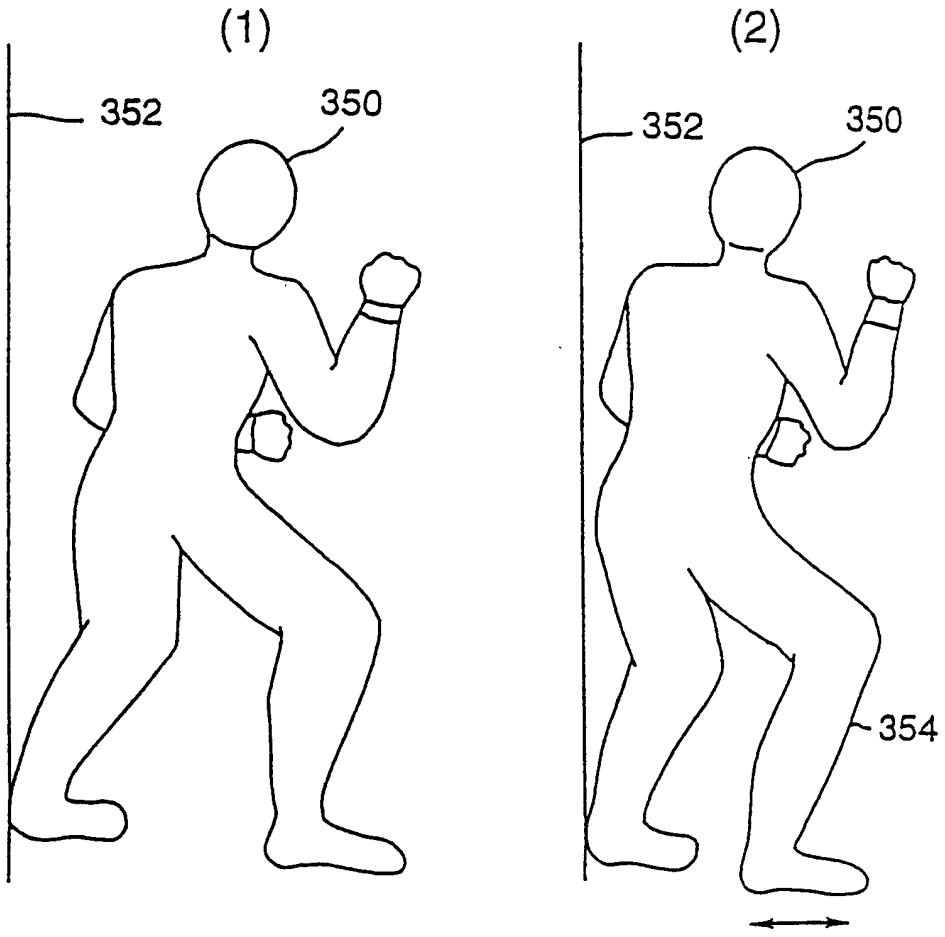


图 68