[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
*G06T 15/00 (2006.01)*

[12] 发明专利申请公布说明书

[21] 申请号 200580039002.0

[43] 公开日 2007 年 10 月 17 日

[11] 公开号 CN 101057261A

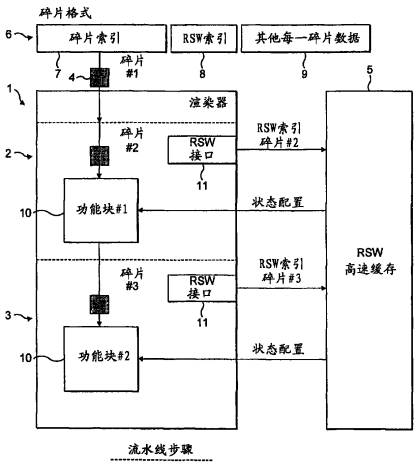
[22] 申请日 2005.11.15
[21] 申请号 200580039002.0
[30] 优先权
[32] 2004.11.15 [33] GB [31] 0425204.5
[86] 国际申请 PCT/GB2005/004385 2005.11.15
[87] 国际公布 WO2006/051330 英 2006.5.18
[85] 进入国家阶段日期 2007.5.15
[71] 申请人 ARM 挪威股份有限公司
地址 挪威特隆赫姆
[72] 发明人 J·尼斯塔德 M·布拉策维克
B·约斯兰德 E·索尔加德
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 张雪梅 王小衡
权利要求书 6 页 说明书 22 页 附图 4 页

[54] 发明名称

# 三维图形处理

[57] 摘要

一种图形渲染流水线(1)具有多个不同的渲染单元(2，3)并接收碎片(4)用于渲染。 渲染器状态字高速缓存(5)用来存储用于当渲染单元渲染碎片时配置渲染单元(2，3)的渲染状态数据。 每个渲染单元(2，3)包括对接收的碎片执行渲染操作的功能块(10)，和渲染状态字接口(11)，其可以用来从渲染状态字高速缓存(5)中查找所需的渲染状态数据。 每个碎片(4)被提供到渲染流水线(1)，其具有碎片数据(6)，其尤其指示碎片索引(7)、渲染器状态字索引(8)，和渲染碎片所必需的其他碎片数据(9)。 当渲染流水线(1)的渲染单元(2，3)接收将要渲染的碎片时，它首先使用与碎片(4)相关联的渲染器状态字索引(8)以利用其渲染器状态字接口(11)从渲染器状态字高速缓存(5)中查找相关的渲染状态数据。 然后它利用该渲染状态数据配置其功能块(10)，并且然后渲染该碎片。



1. 一种图形处理器，包括：

用来渲染图形碎片用于显示的渲染器，该渲染器包括可以被配置成不同渲染状态的一个或多个渲染单元；

用来存储多组数据的装置，每组数据限定用来配置渲染器的渲染单元的不同渲染状态；

用来把将要渲染的图形碎片与存储的渲染状态数据组中的一个相关联的装置；以及

用来将已经与存储的渲染状态数据组关联的图形碎片发送到渲染器的渲染单元以便进行渲染的装置；

其中：

渲染器的渲染单元中的一个或多个包括：

用来确定与将要渲染的图形碎片相关联的渲染状态数据组的装置；以及

用来根据确定的渲染状态数据组配置渲染单元的装置。

2. 如权利要求 1 所述的处理器，其中图形处理的光栅化步骤用来把一个或所述碎片与其相应的渲染状态数据组相关联。

3. 如权利要求 1 或 2 所述的处理器，包括：用来存储与渲染器相关联的一个或多个渲染状态数据组并且可以被渲染器的渲染单元访问的高速缓存存储器，并且其中渲染器的渲染单元被这样配置使得它们将要从高速缓存存储器检索渲染状态数据。

4. 如权利要求 1、2 或 3 所述的处理器，包括：

用来存储限定用于渲染的不同渲染状态的多个数据组的第一存储装置；以及

可以被渲染器的渲染单元访问的、用于存储所选数量的该多个渲染状态数据组的第二存储装置。

5. 一种包括具有可以被配置到特定渲染状态的一个或多个渲染单元的渲染器的图形处理系统，该系统包括：

用来存储限定用于渲染的不同渲染状态的多个数据组的第一存储装置；以及

可以被渲染器的渲染单元访问的、用于存储所选数量的该多个渲染状态数据组的第二存储装置。

6. 如前述权利要求中的任何一个所述的处理器或系统，包括：

用来在基元被接收用于渲染时检查所需的渲染状态数据组是否存储在可以被渲染器的渲染单元访问的存储器中的装置；以及

用于如果所需的渲染状态数据组没有被如此存储时停止基元的处理直到适当的渲染状态数据组在可以被渲染单元访问的存储器中可用时为止的装置。

7. 如前述权利要求中的任何一个所述的处理器或系统，包括：

用来监控渲染状态数据组被渲染器的渲染单元使用的装置。

8. 如前述权利要求中的任何一个所述的处理器或系统，包括用来跟踪与一个或多个特定渲染状态相关联的图形碎片的通过渲染器的行进的装置。

9. 一种图形处理器，包括：

渲染器；

用来把将要渲染的图形碎片和特定渲染状态相关联的装置；以及

用来跟踪与一个或多个特定渲染状态相关联的图形碎片的通过渲染器的行进的装置。

10. 如前述权利要求中的任何一个所述的处理器或系统，包括：用来控制可通向渲染单元的存储器中的渲染状态数据组的存储、并且用来控制用于光栅化的基元和用于渲染的碎片的发出的资源分配器。

11. 如前述权利要求中的任何一个所述的处理器或系统，包括用来在渲染之前根据除了其渲染状态之外的标准对基元分类的装置。

12. 一种在图形处理器中使用的设备，包括：

用来存储多组数据的装置，每组数据限定用来配置图形处理器的渲染单元的不同渲染状态；

用来把将要渲染的图形碎片与存储的渲染状态数据组中的一个相关联的装置；以及

用来将已经与存储的渲染状态数据组相关联的图形碎片发送到图形处理器的渲染单元用于渲染的装置。

13. 如前述权利要求中的任何一个所述的处理器、设备或系统，其中用来把图形碎片和渲染状态数据组相关联的装置包括用来把指示用于碎片的渲染状态数据组的标记分配给该碎片的装置。

14. 如权利要求 13 所述的处理器、设备或系统，其中渲染状态数据标记指示存储渲染状态数据组的存储位置。

15. 一种用在图形处理器中的渲染单元，包括：

用来确定与将要渲染的图形碎片相关联的渲染状态的装置；以及

用来根据确定的渲染状态配置渲染单元的装置。

16. 如前述权利要求中的任何一个所述的处理器、系统、设备或单元，其中渲染单元的一个或多个仅可以选择性地检索渲染状态数据组的一部分。

17. 一种用于图形处理的渲染器，其中用于渲染的图形碎片与相应的存储的渲染状态数据组相联系并且渲染器的渲染单元在接收它们时根据与碎片相联系的渲染状态来配置它们自己。

18. 一种用于图形处理的渲染器，其中渲染器的不同渲染单元可以被同时配置到不同的渲染状态。

19. 一种操作具有渲染器的图形处理器的方法，该渲染器包括用来渲染图形碎片用于显示的一个或多个渲染单元，该方法包括：

存储限定用来配置渲染器的渲染单元的一个或多个渲染状态的数据；

把将要渲染的每一个碎片和存储的渲染状态中的一个相关联；以及

渲染单元，当它接收将要渲染的碎片时，确定与该碎片相关联的渲染状态，根据确定的渲染状态被配置，并且对接收的碎片执行其渲染操作。

20. 如权利要求 19 所述的方法，包括：使用图形处理的光栅化步骤把一个或所述碎片与其相应的渲染状态数据组相关联。

21. 如权利要求 19 或 20 所述的方法，包括：在与渲染器相关联并且可以被渲染器的渲染单元访问的高速缓存存储器中存储一个或多个渲染状态数据组；以及渲染器的渲染单元从该高速缓存存储器中检索渲染状态数据。

22. 如权利要求 19、20 或 21 所述的方法，包括：

在第一存储装置中存储限定用于渲染的不同渲染状态的多个数据组；以及

在可以被渲染器的渲染单元访问的第二存储装置中存储所选数

量的该多个渲染状态数据组。

23.一种操作包括具有可以被配置到特定渲染状态的一个或多个渲染单元的渲染器的图形处理器的方法，该方法包括:

在第一存储装置中存储限定用于渲染的不同渲染状态的多个数据组；以及

在可以被渲染器的渲染单元访问的第二存储装置中存储所选数量的该多个渲染状态数据组。

24. 如权利要求 19 到 23 中的任何一个所述的方法，包括:

在接收基元用于渲染时,检查所需的渲染状态数据组是否存储在可以被渲染器的渲染单元访问的存储器中；以及

如果所需的渲染状态数据组没有被如此存储,则停止基元的处理直到适当的渲染状态数据组在可以被渲染单元访问的存储器中可用时为止。

25. 如权利要求 19 到 24 中的任何一个所述的方法，包括:

监控渲染状态数据组被渲染器的渲染单元的使用。

26. 如权利要求 19 到 25 中的任何一个所述的方法，包括:

跟踪与一个或多个特定渲染状态相关联的图形碎片的通过渲染器的行进。

27. 一种操作图形渲染器的方法，包括:

把将要渲染的图形碎片和特定渲染状态相关联；以及

跟踪与一个或多个特定渲染状态相关联的图形碎片的通过渲染器的行进。

28. 如权利要求 19 到 27 中的任何一个所述的方法，包括:

在渲染之前根据除了其渲染状态之外的标准对基元分类。

29. 一种操作图形处理器的方法，包括:

存储多组数据,每组数据限定用来配置图形处理器的渲染单元的不同渲染状态;

把将要渲染的图形碎片与存储的渲染状态数据组中的一个相关联；以及

将已经与存储的渲染状态数据组相关联的图形碎片发送到图形处理器的渲染单元用于渲染。

30. 如权利要求 19 到 29 中的任何一个所述的方法，其中把图形

碎片和渲染状态数据组相关联的步骤包括:

用来把指示用于碎片的渲染状态数据组的标记分配给该碎片。

31. 一种操作图形处理器的渲染单元的方法，包括:

所述渲染单元:

确定与将要渲染的图形相关联的渲染状态；以及

根据确定的渲染状态配置它自己。

32. 如权利要求 19 到 31 中的任何一个所述的方法，包括:

渲染单元的一个或多个仅选择性地检索渲染状态数据组的一部分。

33. 一种操作用于图形处理的渲染器的方法，包括:

把用于渲染的图形碎片与相应的存储的渲染状态数据组相联系并且渲染器的渲染单元在接收它们时根据与碎片相联系的渲染状态来配置它们自己。

34. 一种操作用于图形处理的渲染器的方法，包括:

将渲染器的不同渲染单元同时配置到不同的渲染状态。

35. 一种包括计算机软件代码部分的计算机程序元件，该计算机软件代码部分用于在该程序元件在数据处理装置上运行时执行如权利要求 19 到 34 中的任何一个所述的方法。

36. 一种 3D 图形处理器或 3D 图形处理平台，包括如权利要求 1 到 18 中的任何一个所述的、或根据如权利要求 19 到 34 中的任何一个所述的方法操作的处理器、系统、设备或渲染单元。

37. 一种基本如上文中参考附图中的任何一个所描述的图形处理器。

38. 一种基本如上文中参考附图中的任何一个所描述的图形处理系统。

39. 一种用在基本如上文中参考附图中的任何一个所描述的图形处理器中的设备。

40. 一种基本如上文中参考附图中的任何一个所描述的渲染单元。

41. 一种基本如上文中参考附图中的任何一个所描述的用于图形处理的渲染器。

42. 一种操作基本如上文中参考附图中的任何一个所描述的图形处

理器、渲染器或渲染单元的方法。

# 三维图形处理

本发明涉及用于例如显示屏上显示的三维（3D）图形处理，并且特别涉及用于显示的三维图形的渲染（rendering）。

正如本领域中已知的，一般通过首先把将要显示的场景分成许多类似的基本分量（所谓的“基元（primitive）”）来执行 3D 图形处理以允许更容易地执行该 3D 图形处理操作。这些“基元”一般采用简单的多边形形式，例如三角形，并且一般通过限定它们的顶点而被描述。

一旦将要显示的场景被分成了多个图形基元，那么正如本领域中已知的，这些图形基元一般将会被进一步分成分立的图形实体或元素，一般称作“碎片（fragment）”，在其上执行实际的图形处理操作（例如渲染操作）。每个这种图形碎片将代表并且对应于基元中的给定位置，并且实际上包括用于所述位置的一组数据（例如颜色和深度值）。

每个图形碎片（数据元）一般对应于最终显示中的单像素（像元）（因为当这些像素是最终将要显示的画面中的奇异点时，在 3D 图形处理器所操作的“碎片”和显示中的像素之间一般将存在一一对应的映射）。然而，可能的情况是：在“碎片”和“像素”之间不存在直接对应，例如在显示最终图像之前对渲染的图像实施后期处理例如按比例缩小的特殊形式的情况下。

因此，一般执行的 3D 图形处理的两个方面是图形“基元”（或多边形）位置数据到图形碎片位置数据的“光栅化（rasterising）”（即确定用来代表将要显示的场景中的每个基元的图形碎片的（x，y）位置），并且然后“渲染”“被光栅化的”碎片（即对这些碎片进行染色（colouring）、着色等），用于在显示屏上显示。

（在 3D 图形文献中，术语“光栅化”有时候用来表示基元转换成碎片以及渲染。然而，在此“光栅化”将仅用来指把基元数据转换成碎片地址。）

渲染过程基本上包括导出显示每个碎片必需的数据。这种数据一般包括用于每个碎片的红、绿和蓝色（RGB）值（其基本上将确定显示上的碎片的颜色），以及用于每个碎片的所谓的“α”（透明度）值。

正如本领域中已知的，一般通过以线性或流水线（pipelined）方式一个接一个地对每个碎片（即用于该碎片的数据）执行单独的渲染过程（步骤）来导出该数据。因此，例如，根据例如记录的用于碎片所属的基元的顶点的颜色和透明度数据，每个碎片首先被分配初始的 RGB 和α值，然后对碎片数据接连地执行例如纹理化、雾化、和混合等操作。这些操作修改为每个碎片设置的初始 RGB 和α值，以便在最后的处理操作之后，每个碎片具有适当设置的 RGB 和α值，以允许该碎片正确地显示在显示屏上。然后存储该最后设置的 RGB 和α值，准备在显示屏上显示该碎片。然后对当前正在渲染的场景区域中的所有碎片重复该过程。

以该方式执行渲染过程，因为单独的渲染操作一般彼此独立，并且可以不参考其他渲染步骤或碎片而执行，因此它们可以在没有损害的情况下以线性、流水线方式来执行。以这种流水线方式（以所谓的“渲染流水线”）执行渲染操作意味着“流水线”的不同渲染单元可以同时对不同碎片进行操作，由此使得渲染过程更有效。

为执行正确的渲染操作，渲染过程的每个阶段需要适当地配置，即将被设置成正确的所谓的“渲染状态”。渲染状态确定渲染过程的这些阶段怎样对每个碎片进行操作，并且一般确定例如渲染图形碎片所使用的反走样模式、混合模式、模板缓存操作、纹理函数（texture function）、纹理映射信息、z 测试模式、RGBa 缓存写模式等中的一个或多个。它也可以用来例如表示将要使用的光栅化模式。

一般通过驱动图形处理器的“驱动器”软件来确定和设置渲染状态。（正如本领域中已知的，一般由“主”设备（例如在图形处理器的主机系统上）控制图形处理器，该主设备借助在该主设备上运行的软件“驱动器”与图形处理器进行通信并且控制图形处理器，该软件“驱动器”被配置以响应于从在该主设备上运行的需要使用图形处理器的（软件）应用程序接收的命令，与图形处理器进行通

信并且控制图形处理器。）

在现有的 3D 图形处理系统中，渲染过程的所有阶段通常首先被配置成期望的渲染状态，然后图形碎片被发出以便进行渲染。换句话说，整个渲染器在图形碎片发送给它之前被配置到单个渲染状态。

然而，这意味着当利用不同的渲染状态将碎片渲染成渲染器的当前状态时（例如当将要渲染来自具有不同渲染状态的另一个基元的第一碎片时），该渲染器必须使其渲染状态在能够渲染新的碎片之前变成新的渲染状态。此外，直到目前在渲染器中的所有碎片的处理都已经结束，该渲染器的渲染状态才能改变。实际上，这意味着当需要渲染器状态改变时，在可以渲染新的碎片之前，需要新的渲染状态的新碎片必须“停止”，已经在渲染器中的碎片“刷新（flush）”它，并且渲染状态改变。这降低了渲染器的性能。

因此公知的是：试图减少渲染状态变化的数量，其需要通过在将它们“发送”给渲染器之前根据它们的渲染状态对将要渲染的基元分类来完成。然而，即使利用这种分类，渲染状态变化仍旧易于相对频繁地发生。

还公知的是：试图增加可以进行渲染状态变化的速度，例如通过在提供给渲染器的基元数据流中包括渲染状态变化命令和数据，以便例如减少实现渲染器状态变化所需的时间。然而，只要需要进行渲染状态变化，这种操作就仍需要渲染器停止和被刷新。

因此本申请人相信仍然存在改善渲染操作的余地，特别是在渲染状态变化的情况下。

根据本发明的第一方面，提供一种图形处理器，包括：

用来渲染图形碎片用于显示的渲染器，该渲染器包括可以被配置成不同渲染状态的一个或多个渲染单元；

用来存储多组数据的装置，每组数据限定用来配置渲染器的渲染单元的不同渲染状态；

用来把将要渲染的图形碎片与存储的渲染状态数据组中的一个相关联的装置；以及

用来将已经与存储的渲染状态数据组关联的图形碎片发送到渲

染器的渲染单元以便进行渲染的装置;

其中:

渲染器的渲染单元中的一个或多个包括:

用来确定与将要渲染的图形碎片相关联的渲染状态数据组的装置;以及

用来根据确定的渲染状态数据组配置渲染单元的装置。

根据本发明的第二方面,提供操作具有渲染器的图形处理器的方法,该渲染器包括用来渲染用于显示的图形碎片的一个或多个渲染单元,该方法包括:

存储限定用来配置渲染器的渲染单元的一个或多个渲染状态的数据;

把将要渲染的每个碎片与存储的渲染状态之一相关联;以及

渲染单元,当它接收将要渲染的碎片时,确定与该碎片相关联的渲染状态,根据确定的渲染状态被配置,并且对接收的碎片执行其渲染操作。

在本发明中,将要渲染的图形碎片与它们的渲染状态相关联,并且然后渲染器的渲染单元利用该关联来确定它们应当使用的用于渲染碎片的渲染状态。换句话说,当渲染单元接收用于渲染的碎片时,它根据与碎片相关联的渲染状态“查找”用于该碎片的渲染状态,并且然后相应地渲染该碎片。这避免了对在渲染任何碎片之前将整个渲染器配置到给定的渲染状态的需要。它也意味着渲染器的渲染单元均可以独立于当前的渲染状态或渲染器中剩余的渲染单元的状态,有效地将它们自己配置成它们的当前碎片所需要的渲染状态。

本发明的这种设置的优点是渲染器的不同单元可以(并且优选)被同时配置到不同的渲染状态,即使得渲染器的渲染单元可以同时对需要不同渲染状态的碎片进行操作。因此这允许具有不同渲染状态的碎片被同时发送到渲染器(并且被该渲染器渲染),而不需要在连续的碎片之间发生渲染状态变化时将渲染器的其他或所有单元重新配置到新的渲染状态。

因此本发明可以用来减少或甚至消除由于渲染状态变化引起的渲染器停止和刷新,由此提高渲染操作的速度和效率,并且便于使用更长(以及更复杂)的渲染器(如果每当发生渲染状态变化时其

都需要停止和刷新，那么它将变得效率太低而不能使用）。使用更长和更复杂的渲染器的能力便于例如实现更复杂的渲染操作，例如补色着色（phong shading），即使在“实时”渲染过程中。

此外，因为本发明可以减少渲染状态变化的有害影响，因此它可以减少或去除对在渲染之前根据它们的渲染状态预分类基元和碎片的需要。于是这将允许在渲染之前根据其他标准来对基元和碎片进行分类，这可能是有利的。例如，代替地，基元可以（并且优选）根据深度来分类。

本发明也延及可以以上述方式工作的渲染单元，以及与渲染器一起使用的、能够适当地准备图形碎片的装置。

因此，根据本发明的第三方面，提供一种在图形处理器中使用的渲染单元，包括：

用来确定与将要渲染的图形碎片相关联的渲染状态的装置；以及

用来根据确定的渲染状态配置渲染单元的装置。

根据本发明的第四方面，提供一种在图形处理器中使用的装置，包括：

用来存储多个数据组的装置，每组数据限定用来配置图形处理器的渲染单元的不同渲染状态；

用来把将要渲染的图形碎片和存储的渲染状态数据组中的一个相关联的装置；以及

用来将已经与存储的渲染状态数据组相关联的图形碎片发送到图形处理器的渲染单元以便进行渲染的装置。

如上所讨论的，用来限定渲染状态的数据组应当包括用于渲染器的渲染单元的必要的配置信息。因此它们优选规定诸如下述的数据：将要使用的有效纹理映射（texture map）（例如将要使用的纹理映射被存储的存储地址以及该纹理映射的尺寸，这是本领域中公知的），将要使用的纹理函数和混合模式，将要使用的 z 缓存和模板缓存模式，和/或是否能够实现反走样，等等。它们也可以（如果需要的话）包括用来规定其他操作的数据，例如光栅化，这是本领域中公知的。

在图形处理器支持例如像素着色程序设计的操作的情况下（正如本领域中已知的，一些 3D 图像处理器支持“像素着色”程序的使用，

并且可以包括例如像素着色处理器，其可以用来执行更复杂的渲染操作，特别是在每个碎片的基础上），那么存储的渲染状态数据组可以并且优选包括指示将要使用的像素着色程序和数据的数据。该数据例如可以是限定存储地址（由该存储地址检索相关着色程序）和该程序的大小的数据形式，和/或是指向存储的像素着色程序的指针形式，其将用来配置渲染器的某一或该像素着色处理器。

优选借助用于图形处理器的驱动器软件产生渲染状态数据组。优选以每一基元为基础来规定它们。

可以按照需要来选择和设置渲染状态数据组。它们例如可以被设置成通常是用于限定渲染器的渲染状态的情况（例如渲染流水线）。在优选实施例中，数据组被设置成预定数据结构，以便每个数据组具有公共格式。最优选地，限定不同渲染状态的数据组被设置成数据字。在特别优选的实施例中，每个数据组，例如渲染器状态字，被再分成多个分立的、更小的部分（例如子字），其每一个例如限定渲染状态的不同方面。

可以按照需要来选择渲染器的渲染单元。它们优选包括通常在渲染器中得到的一个或多个渲染单元。因此它们优选包括例如一个或多个函数发生器、纹理映射器、一个或多个混合器、一个或多个雾化单元、逻辑操作单元、像素着色单元、和一个或多个输出缓冲器中的一个或多个。优选在渲染器中存在多个渲染单元。正如本领域中已知的，每个渲染单元将根据该渲染单元被配置成的渲染状态渲染它接收的图形碎片。

在优选实施例中，渲染器的每个渲染单元能够确定与碎片相关联的渲染状态并且将其本身配置成该渲染状态。然而，这并不是必需的，并且因此在另一优选实施例中，各渲染单元中只有一些能够执行渲染状态“查找”。例如，渲染器的所选单元（例如全部混合单元）能够被集中在一起，并且当在该组中的第一单元处接收碎片时对该组的单元一起执行单一的渲染状态确定和配置。

虽然渲染单元在它执行渲染状态查找时可以对该渲染状态检索（查找）整个渲染状态数据组，但是优选渲染单元仅查找（检索）它们实际需要的能够将它们自己配置成所需渲染状态的渲染状态数据组的一个或多个部分。因此，渲染单元可以优选仅查找或检索渲

染状态数据组的所选部分，它们本希望如此。这减少了在渲染单元接收碎片用于渲染时需要由这些渲染单元检索的数据量。

因此，优选渲染状态数据组的不同部分相应地能够被查找，而与对查找或检索整个数据组的需要无关，例如，以便渲染单元能够例如查找适当的渲染状态数据组的子部分，而不需要从存储器检索整个渲染状态数据组。

可以按照需要来再次选择图形碎片与给定的渲染状态数据组相关联的方式。最优选地，每个碎片被分配标记或索引指针，其与碎片一起“被传递”通过渲染器并且指示将要用于该碎片的渲染器状态数据组。然后渲染单元可以利用碎片的渲染器状态数据组标记或指针来从它被存储的地方检索适当的渲染器状态数据。渲染器状态数据组标记或指针优选与“被传递”通过渲染器的用于碎片的其他数据（例如它的 RGBa 值）相关联。

标记或指针能够标识将要使用的实际渲染状态数据组，但是在优选实施例中，正如以下将要进一步讨论的，其并不这样做，而是代替地简单指示（指向）存储数据组的存储位置。这降低了标记或指针的复杂性，因为它于是不潜在地需要在所有可能的能够使用的渲染状态数据组之间进行区分。

在特别优选的实施例中，将要渲染的基元与将要用来渲染该基元的渲染状态数据组相关联（以及利用所述渲染状态数据组来标记）（因为，一般将利用相同的、单一的渲染状态整体上渲染给定的基元），并且然后，当该基元被光栅化成碎片时，每个碎片被分配分配给该基元的渲染数据组指针或标记。因此，优选地，光栅化步骤用来把这些碎片与其对应的渲染状态数据组相关联。

类似地，可以按照需要来选择由渲染器的渲染单元所使用的渲染状态数据组被存储的方式。优选存储多个这样的数据组，因为其便于渲染器同时渲染具有不同渲染状态的碎片。

然而，本申请人也已经认识到，完全存储很多个渲染状态数据组以便在使用中可以被渲染单元同时访问将需要相对大的、且较慢的存储工具。此外，与每个碎片相关联的渲染数据组指针或标记有必要必须足够大以便区分所有可能的渲染状态。

因此，在特别优选的实施例中，将要用于渲染场景的更受限的、

所选数量的可能的渲染状态数据组被存储，以便由渲染单元访问。这些渲染状态数据组优选被“局部”存储到渲染器，优选在与该渲染器相关联的可更快速访问的局部存储单元中，例如高速缓存存储器。这具有的优点是：存储在局部存储单元中的渲染状态数据组可以被渲染单元更快速地检索，以及与每个碎片相关联的渲染状态数据组指针可以更短（因为有更少的数据组在其间进行区分）。

因此，本发明的设备优选进一步包括与渲染器相关联并且可被渲染器的渲染单元访问的局部存储器，优选高速缓存存储器，其中存储了一个或多个（且优选多个）渲染状态数据组。此外，渲染单元优选被这样配置使得它们将要从该局部存储器检索渲染状态数据组，而不是从可通向渲染装置的另一存储器。

可以按照需要来选择渲染状态数据组的数量，其被存储在渲染单元访问的例如局部高速缓存的存储器中以便被渲染单元访问。如上所讨论的，存储越多这样的数据组，渲染器可以同时处理的不同渲染状态就越多，并且因此渲染器就越不可能因渲染状态变化而停止。另一方面，存储的数据组越多，需要的存储器就越大并且系统就越复杂。

优选根据渲染器的长度选择存储的数据组的数量，即，使得为更大的渲染器存储更多的渲染状态。例如，为确保由于渲染状态变化引起的停止不能发生，应当存储与渲染器中存在的单独可配置的渲染单元（或被共同配置的单元组）一样多的渲染状态数据组。

本申请人已经发现，对于（相对较小的）渲染器的大多数实际的实施方式而言，存储用于四个不同的渲染状态的数据将消除由于渲染状态变化引起的渲染器停止的发生。如果存储了八个或十六个渲染状态，那么渲染器停止的可能性会更进一步减小。因此，优选 4 到 16 个、更优选 4、8 或 16 个、并且最优选 4 个渲染器状态数据组被存储以便被渲染单元访问。

在有限数量的渲染状态数据组被存储以便被渲染器的渲染单元访问的设置中，特别是在它们被存储在渲染器的局部存储器中的情况下，优选在另一存储器例如可通向渲染器的外部存储器中存储更多（并且优选全部）可能或将被需要用于渲染所述场景的渲染状态数据组，使得必要时，例如如果局部的、有限的存储器没有存储必

要的数据组时，那些数据组是可用的。最优选地，当需要它们用于渲染时，数据组可以从存储了（全部）数据组的较大存储器被复制到更受限的（局部）存储器（受其总容量支配）。

令人相信的是，这种设置凭它自身的性能可能是新颖和有优势的。因此，根据本发明的第五方面，提供一种操作图形处理器的方法，该图形处理器包括具有可以被配置到特定渲染状态的一个或多个渲染单元的渲染器，该方法包括：

在第一存储装置中存储限定用于渲染将要渲染的场景的不同渲染状态的多个数据组；以及

在可以被渲染器的渲染单元访问的第二存储装置中存储所选数量的该多个渲染状态数据组。

根据本发明的第六方面，提供一种包括具有可以被配置到特定渲染状态的一个或多个渲染单元的渲染器的图形处理系统，该系统包括：

用来存储限定用于渲染将要渲染的场景的不同渲染状态的多个数据组的第一存储装置；以及

用来存储所选数量的该多个数据组的、可以被渲染器的渲染单元访问的第二存储装置。

如上所讨论的，在本发明的这些方面和实施例中，优选渲染器的渲染单元仅访问存储渲染状态数据组的子集的存储器。

在只有有限数量的渲染状态数据组被存储以便可以被渲染器的渲染单元访问的情形下，如上所讨论的，需要当前不能被渲染单元访问的渲染状态的碎片将有可能需要被渲染。在该情况下，如上所讨论的，优选该新的渲染状态可以被写入到可通向渲染单元的存储器以便使用。然而，本申请人已经认识到，优选碎片不应被传递用于渲染直到适当的渲染状态数据组可用于渲染单元为止。

因此，在特别优选的实施例中，如果与碎片相关联的渲染状态数据组可通向渲染单元（例如存储在渲染器的局部存储器中），那么碎片仅被“发送”用于渲染。如果必需的渲染状态数据组不可用，那么碎片的处理优选被暂停（停止）直到相关的渲染状态数据组可用于渲染单元为止。（在此应当注意的是，尽管所述碎片的处理可以停止，但是必需的渲染状态数据可用的其他碎片的处理可以并且

优选继续进行。换句话说，该渲染过程（流水线）不停止，只有所述的碎片（基元）停止。）

由于实际上，如上所讨论的，用于给定基元的所有碎片都具有相同的渲染状态，因此优选通过确定与该基元相关联的渲染状态数据组是否可通向渲染单元（例如存储在渲染器的局部存储器中）来进行该操作。如果渲染状态数据组是如此可用的，那么该基元可以被发送用于光栅化成碎片并且用于渲染。否则，该基元被阻止直到必需的渲染状态数据组可用为止。

碎片（基元）的这种停止应当与现有技术系统形成对比，在现有技术系统中整个流水线（即所有新碎片的处理）都停止；在本发明中，优选适当的渲染状态数据组可通向渲染单元的随后碎片（基元）的处理继续进行，即使当一个（或多个）碎片（基元）的处理可能停止时。

因此，在特别优选的实施例中，当基元（或碎片）被接收用于渲染时，首先检查看需要的渲染状态数据组是否存储在可通向渲染器的渲染单元的（例如局部的）存储器中。如果是，则基元（碎片）可以被正常地传递到渲染器用于处理。如果渲染状态数据组没有存储在该存储器中，则基元（或碎片）的处理被停止直到可被渲染单元访问的存储器中适当的渲染状态数据组可用时为止，在此时基元（碎片）随后可以被发送用于渲染。

因此还优选能够将新的渲染状态数据组载入到可通向需要它们的渲染器的渲染单元的存储器中，例如，在将要渲染需要还没有在（局部）存储器中的新的渲染状态的基元或碎片时。如上面所讨论的，预期可能需要优选从（全部）渲染状态数据组的现有存储这样载入新的渲染状态数据组。

在这种设置中，在可通向渲染单元的存储器中存在存储新的数据组的备用容量的情况下，可以仅将新的数据组载入存储器。然而，如果用于渲染状态数据组的存储器的容量全部都在使用中，则新的数据组将不得不替换现有存储的渲染状态数据组。

在后者情形中，在将要被载入可通向渲染单元的（例如局部）存储器的新的渲染状态数据组将要替换已经存储的渲染状态数据组的情况下，优选直到现有的渲染状态数据组不再被需要用于渲染时（例

如对于已经在渲染器中的或已知将要被发送到渲染器的碎片）才替换已经存储的渲染状态数据组。

如果在那时不可能替换现有存储的数据组，那么系统优选继续监控该存储器以确定何时现有存储的渲染状态数据组不再被需要并且因此可以被替换。

因此优选本发明的方法和设备包括用来监控存储在可通向渲染单元的存储器中的渲染状态数据组的使用的步骤或装置，以便例如能够确定已经存储在可通向渲染单元的存储器中的渲染状态数据组是否以及何时不再使用或不再被需要（并且因此可以被替换）。最优选通过跟踪利用已存储的数据组的碎片和/或基元的通过渲染器的行进（progress）来监控这种“使用”，以便优选确定例如在渲染器中是否存在需要或将需要所述数据组的任何碎片、或是否存在需要或将需要所述数据组的预计将被发送到渲染器的任何碎片。当确定没有这样的碎片时，那么该渲染状态数据组可以被替换。

再次令人相信的是，这种设置凭它自身的性能可能是新颖的和有优势的。因此，根据本发明的第七方面，提供一种操作图形渲染器的方法，包括：

把将要渲染的图形碎片和特定的渲染状态相关联；以及

跟踪与一个或多个特定渲染状态相关联的图形碎片的通过渲染器的行进。

根据本发明的第八方面，提供一种图形处理器，包括：

渲染器；

用来把将要渲染的图形碎片和特定的渲染状态相关联的装置；以及

用来跟踪与一个或多个特定渲染状态相关联的图形碎片的通过渲染器的行进的装置。

利用每一个存储的数据组的碎片的行进的跟踪优选包括至少确定何时最后一个这样的碎片离开渲染器，并且确定利用该数据组的任何基元当前是否正在被渲染（如其将指示利用该数据组的碎片是否正要被发送到渲染器）。

在特别优选的实施例中，通过保持跟踪碎片和/或基元的行进的一系列计数器来执行利用每一个存储的渲染状态数据组的碎片等的

行进。

最优选的是，保持关于每个存储的渲染状态数据组的两个计数器。每当利用渲染状态数据组的基元被发送以便被渲染时，第一计数器，基元计数器，优选增加（或减少），并且每当光栅化器发出最后一个碎片用于利用特定渲染状态数据组的基元时，该第一计数器减少（或增加，相应地）。第二计数器是“碎片”计数器，并且每当利用渲染状态数据组的碎片被发送到渲染器时增加（或减少），以及每当利用该渲染状态数据组的碎片的渲染完成时减少（或增加，相应地）。当两个计数器都为零时，其表示目前该渲染状态数据组没有被使用并且因此可以被替换，如果期望的话。

在优选实施例中，本发明的设备包括资源分配器，该资源分配器适当地控制在可通向渲染单元的存储器中的渲染状态数据组的存储（和替换）、和用于光栅化的基元以及用于渲染的碎片的发送。

在本发明中可以在渲染基元之前根据渲染器状态对它们分类，这是本领域中公知的。其将进一步降低由于渲染器状态变化发生引起的渲染停止的风险。然而，如上所讨论的，本发明的重要优点是它可以用来消除对在渲染之前根据渲染状态对基元分类的需要。因此，在优选实施例中，在渲染之前根据另一标准（即不根据渲染状态）对基元分类。最优选，在渲染之前根据深度对它们进行分类。

正如将从上述理解的，本发明，至少在其优选实施例中，将给定的渲染状态数据组与将要渲染的每个碎片相联系，并且渲染器的渲染单元在接收它们时根据与碎片相联系的渲染状态来配置它们自己。换句话说，一接收到碎片时，渲染单元就将确定需要的渲染状态并且在渲染碎片之前将它自己配置成该渲染状态。

因此，根据本发明的第九方面，提供一种用于图形处理的渲染器，其中用于渲染的图形碎片与相应的存储的渲染状态数据组相联系并且渲染器的渲染单元在接收它们时根据与碎片相联系的渲染状态来配置它们自己。

此外，本发明允许渲染器的渲染单元被同时配置到不同的渲染状态，并且允许需要不同渲染状态的碎片被同时发送到并且存在于渲染器中。

因此，根据本发明的第十方面，提供一种用于图形处理的渲染

器，其中渲染器的渲染单元可以被同时配置到不同的渲染状态。

本发明可以适用于任何形式或结构的渲染器，尽管正如从上述理解的，本发明特别适用于具有“流水线”配置的渲染器，在该情形下，渲染器将呈渲染流水线的形式。它可以适用于所有形式的渲染，例如立即模式渲染、延期模式渲染、基于瓦片的渲染（tile-based rendering）等。

正如从上述理解的，本发明可特别适用于 3D 图形处理器和处理设备，并且从而延及包括这里描述的本发明的任何一个或多个方面的设备或根据这里描述的本发明的任何一个或多个方面操作的 3D 图形处理器和 3D 图形处理平台。依据（subject to）执行以上所讨论的特定功能所必需的任何硬件，这种 3D 图形处理器另外可以包括 3D 图形处理器包括的普通功能单元等中的任何一个或多个或全部。

本领域的技术人员还将理解的是，本发明的所描述的所有方面和实施例可以适当地包括这里描述的优选和可选特征中的任何一个或多个或全部。

根据本发明的方法可以至少部分地利用软件例如计算机程序来实现。因此将看到，当从另外的方面观察时，本发明提供特别适于在安装到数据处理装置上时执行这里描述的方法的计算机软件、包括计算机软件代码部分的计算机程序元件，该计算机软件代码部分用来在该程序元件在数据处理装置上运行时执行上面描述的方法、以及包括代码装置的计算机程序，该代码装置适于在该程序在数据处理系统上运行时执行这里描述的一种或多种方法的所有步骤。本发明还延及包括这种软件的计算机软件载体，其在用来操作包括数据处理装置的图形处理器、渲染器或微处理器系统时与所述数据处理装置结合使得所述处理器、渲染器或系统执行本发明的方法的步骤。这种计算机软件载体可以是物理存储介质，例如 ROM 芯片、CD ROM 或盘，或可以是信号，例如线上电子信号、诸如到卫星等的光信号或无线电信号。

将进一步理解的是，并不是本发明的方法的所有步骤都需要通过计算机软件来执行，并且因此根据更宽的方面，本发明提供计算机软件和安装在计算机软件载体上用来执行这里所述的方法的步骤中的至少一个的这种软件。

因此本发明可以被适当地具体实施为与计算机系统一起使用的计算机程序产品。这种实施方式可以包括计算机可读指令序列，所述指令被固定在有形介质上，诸如计算机可读介质，例如软盘、CD－ROM、ROM、或硬盘，或者可通过有形介质，包括但不限于光通信线等，或无形地利用无线技术，包括但不限于微波、红外线或其他传输技术，经由调制解调器或其他接口设备被传输到计算机系统。所述计算机可读指令序列具体实施了这里在前描述的功能性的全部或一部分。

本领域的技术人员将理解的是,可以用与多种计算机体系结构或操作系统一起使用的多种编程语言来写入这些计算机可读指令。另外，可以用现在或将来的任何存储技术来存储这些指令，包括但不限于半导体、磁、或光，或者利用现在或将来的任何通信技术来传输这些指令，包括但不限于光、红外线、或微波。还设想了这种计算机程序产品可以被发行为具有附随的印刷或电子文档的可移动介质，例如收缩包装的软件，其利用计算机系统被预加载到例如系统ROM或固定盘上，或从例如因特网或万维网（World Wide Web）的网络上的服务器或电子公告板发行。

现在将仅借助实例并且参考附图来描述本发明的多个优选实施例，其中:

图 1 用示意图示出根据本发明的实施例的 3D 图形处理系统;

图 2 用示意图示出图 1 的实施例的 3D 图形渲染器的操作; 以及

图 3 和图 4 示出根据本发明设置的 3D 图形处理系统的另一实施例。

如上所讨论的，本发明涉及 3D 图形处理器的渲染操作。

正如本领域中已知的，当 3D 图形对象将被渲染时，它们通常首先被限定为一系列基元（多边形），然后这些基元被分成图形碎片，然后这些碎片被依次渲染。在正常的 3D 渲染操作期间，渲染器将修改（例如）与每个碎片相关联的 RGBa 数据以便这些碎片可以被正确地显示。一旦这些碎片已经完全穿过渲染器，那么其相关联的数据值就被存储在存储器中，准备用于显示。

每个基元一般还与渲染状态相关联，其限定基元将怎样被光栅化和渲染（例如将用来渲染基元的碎片的纹理模式、混合模式、z 测试、反走样、模式等）。在正常的过程中，通过在利用 3D 图形处理器的主机系统上运行的软件驱动器设置用于给定基元的渲染状态。该驱动器一般限定限定不同渲染状态的数据组，这些数据组然后以这种方式被存储使得当需要时它们可以用来将渲染器的渲染单元配置到正确的渲染状态。

图 1 和 2 用示意图示出可以根据本发明操作的 3D 图形处理平台的实施例。图 1 示出整个图形处理器系统。图 2 更详细地示出渲染器（渲染流水线）的操作。

在该实施例中，根据本发明，将要渲染的每个碎片与给定的渲染状态相关联，并且然后渲染器的该多个渲染单元在它们接收用于渲染的碎片时“查找”用于该碎片的渲染状态。

如图 1 中所示，图形处理系统 25 包括采用渲染流水线 1 的形式的渲染器，其接收用于渲染的碎片，和高速缓存存储器 5，其存储限定用于被渲染流水线 1 的渲染单元访问的渲染状态的数据以允许那些单元配置它们自己以便渲染碎片（这将在下面进一步讨论）。

该图形处理系统还包括一般存在于 3D 图形处理系统中的多个其他功能单元，例如基元列表读出器 20（用来读出包括指向顶点的指针等的基元列表元素、用于将要渲染的基元的数据（如本领域中已知的））、顶点加载器 21（用来取出被基元列表读出器 20 读出的基元列表中的指针所指向的顶点数据（即在基元即将被光栅化和渲染时确定这些基元的顶点））、三角形设置单元 22（用来根据从顶点加载器 21 接收的基元数据执行三角形设置）、以及光栅化器 23（正如本领域中已知的，用来将基元数据转换成用来渲染基元的碎片位置）。

图 1 中所示的系统还包括资源分配器 24，其可以和系统的各个单元进行通信并且特别用于控制渲染器状态数据组在渲染器状态高速缓存存储器 5 中的存储和基元（并且因此碎片）到系统的顶点加载、三角形设置、光栅化和渲染流水线阶段的发送。

高速缓存存储器 5 存储以渲染器状态“字”形式限定不同渲染状态的数据组。每个渲染器状态字包括用于渲染流水线 1 的配置信息，

规定数据，例如有效纹理映射、使用的纹理函数和混合模式、使用的 z 缓存和模板缓存模式、以及是否能够实现反走样。渲染器状态“字”还包括指示将要被用于一个或多个碎片的像素着色程序（和数据）的数据。

在该实施例中，每个渲染器状态字由 8 个子字构成，每个 32 位宽，存储在连续的存储器中。然而，正如本领域的技术人员将理解的，如果需要的话可以使用不同的设置。

渲染器状态字高速缓存 5 包括 4 个寄存器或线，其每一个可以存储一个渲染器状态字，即使得渲染器状态字高速缓存 5 在任一时刻都可以存储四个不同的渲染器状态字。这使得与该四个当前存储的渲染器状态字中的任何一个相关联的碎片可以前进通过渲染流水线 1。（正如本领域技术人员将理解的，如果需要的话，在渲染器状态字高速缓存 5 中可以使用其他数量的渲染器状态字寄存器。）

可以改变存储在渲染器状态字高速缓存 5 中的渲染器状态字。这在资源分配器 24 的控制下完成。

为了控制渲染器状态字高速缓存 5 中的渲染器状态字的存储和替换，资源分配器 24 为每个渲染器状态字寄存器（线）保持两个计数器：基元计数器和碎片计数器，以使其能够跟踪与存储在渲染器状态字高速缓存 5 中的给定的渲染器状态字相关联的碎片的行进。

每当需要相应的渲染器状态字的基元实际上被载入系统中时基元计数器增加，并且每当光栅化器 23 发送用于利用该渲染器状态字的基元的最后一个碎片时该基元计数器减少。如图 1 中所示，这通过下述来实现：三角形设置单元在它接收基元用于设置时增加用于受该基元束缚的渲染器状态字的基元计数器，并且光栅化器在来自该基元的最后一个碎片离开光栅化器 23 时将用于相关联的渲染器状态字的减量（decrement）多边形计数器信号发送到资源分配器 24。

碎片计数器以类似的方式操作，并且对于发送到渲染流水线 1 的利用相应的渲染器状态字的每个碎片增加 1，以及每当利用渲染状态字的碎片完成时减少 1。这通过下述来实现：渲染流水线在碎片进入流水线时发出用于相关联的渲染器状态字的增量（increment）碎片计数器信号，并且在该碎片离开流水线时发出减量信号，如图 1 中所示。

资源分配器 24 利用这些计数器来确定渲染器状态字高速缓存寄存器线是否可以被“解锁”（即相应的渲染器状态字可以被新的渲染器状态字替换）。当基元计数器和碎片计数器都是零时（并且正要被渲染的新的多边形不使用该特定的渲染器状态字（或者，如果基元列表已经结束，则没有当前基元存在），则资源分配器 24 将“解锁”用于该特定渲染器状态字的渲染器状态字寄存器，以便，如果必要的话，该渲染器状态字可以被可能需要的新的渲染器状态字重写。这样，资源分配器 24 确保已经存储在渲染器状态字高速缓存 5 中的渲染器状态字不能被替换直到所有目前与该渲染器状态字相关联的碎片都已经结束渲染流水线 1 为止。

在本实施例的操作中，当将要渲染场景时，正如本领域中已知的，驱动器软件将准备并且存储必需的顶点数据等，（每一基元数据）用来限定将要渲染的基元。它还将产生并且存储用来渲染整个场景的必需的渲染状态数据组（渲染器状态字）。渲染状态数据组可以以任何适当的方式来产生并被存储。在优选配置中，渲染状态数据组的记录（例如列表）优选被保持在软件中，当渲染状态数据组被产生时，以便允许驱动器软件确定基元需要的渲染状态数据组是否已经被产生用于较早的基元。这有助于减少或避免驱动器软件不必要地产生相同渲染状态数据组的几个副本。

一旦已经产生并存储了必需的数据，那么驱动器软件就将为每个将被渲染用于场景的基元准备包括指向存储的顶点数据的指针等的适当的基元列表（用于被基元列表读出器 20 检索以便允许基元被渲染），这在本领域中是已知的。

驱动器软件在基元列表中还包括指向被存储的用来渲染每个基元的渲染器状态字数据的指针，以便将每个基元与将用来渲染它的适当的渲染器状态字（渲染状态数据）相联系。（尽管在该设置中，在每一基元的基础上相应地规定了渲染状态数据组，但是这不是必需的并且如果需要的话可以使用其他设置。）

优选在基元被发出用于渲染之前准备渲染整个场景（或帧部分（frame-part），例如瓦片（tile））所必需的所有基元数据和基元列表，包括渲染器状态字，并且当渲染场景时其然后会避免对回复到驱动器软件的需要。然而，如果需要的话可以使用其他设置。

一旦已经设置和存储了基元数据和基元列表，那么就可以渲染基元。在该过程中第一步骤是基元列表读出器 20 从存储它们的存储器中读取用于基元的相关的基元列表元素（即顶点数据和渲染器状态字指针等）。然后基元列表读出器 20 向资源分配器 24 发送用于三角形设置操作的请求（即，使得基元可以被渲染）。

当收到这种请求时，资源分配器 24 确定与所述基元相关联的渲染器状态字是否已经存储在渲染器状态字高速缓存 5 中。如果需要的渲染器状态字已经存储在高速缓存 5 中，则资源分配器 24 传递该基元用于三角形设置操作。（因此，尽管以每一基元为基础限定了渲染器状态字，但是在适合这样做的情况下该系统将渲染器状态字从一个基元高速缓存到下一个基元。）

另一方面，如果资源分配器 24 发现当它从基元列表读出器 20 接收用于三角形设置操作的请求时，需要的渲染器状态字仍没有存储在渲染器状态字高速缓存 5 中，它将停止该基元的处理直到渲染器状态字高速缓存 5 寄存器（线）中的一个空闲为止，随后在渲染器状态字高速缓存 5 中存储相关的渲染器状态字，并且继续该基元的处理。

为进行所述后者操作，如上所述，资源分配器 24 利用该基元和用于每一个渲染器状态字寄存器的碎片计数器来确定在渲染器状态字高速缓存 5 中是否以及何时存在用于渲染器状态字的备用容量(即确定是否“解锁”至少一个高速缓存寄存器）。

当它确定在渲染器状态字高速缓存 5 中存在空闲的寄存器时，资源分配器 24 从存储新的渲染器状态字的存储器中将该新的渲染器状态字取出并且将它载入该空闲的渲染器状态字高速缓存寄存器。

一旦资源分配器 24 已经确定用于所述基元的相关的渲染器状态字存储在渲染器状态字高速缓存 5 中（或已经将它存储在渲染器状态字高速缓存 5 中），那么它就给该基元分配在渲染器状态字高速缓存 5 中的适当的渲染器状态字索引（指针），以允许存储在渲染器状态字高速缓存 5 中的用于该基元的渲染器状态字被标识。

在本实施例中，被资源分配器 24 与每个基元相关联的渲染器状态字索引仅仅是指向哪一个渲染器状态字高速缓存寄存器存储了需要的渲染器状态字的指针。因此该索引包括在渲染器状态字高速缓

存 5 中的 2 位索引（因为该高速缓存 5 具有四个渲染器状态字寄存器）。（当然，对于其他大小的渲染器状态字高速缓存 5，可以使用其他的索引设置。如果需要的话，渲染器状态字索引还可以指示使用的实际渲染器状态字，而不是仅指向存储它的存储器位置（寄存器）。）

该基元随后被传递用于三角形设置操作和渲染，并且因而前进通过该系统。

因此顶点加载器 21 然后取出被由基元列表读出器 20 读取的基元列表中的指针所指向的顶点数据，并且三角形设置阶段 22 根据该基元数据执行三角形设置。

然后该基元被传递到光栅化器 23 用于光栅化。正如本领域中已知的，光栅化器 23 将基元数据转换为碎片位置用于渲染。光栅化器 23 还将索引指针与每个碎片相关联，该索引指针指示存储了与碎片相关联的渲染器状态字（其是与和通过多边形列表读出器 20 读取的列表元素中的基元相关联的渲染器状态字相同的渲染器状态字）的渲染器状态字高速缓存 5 寄存器。

被光栅化的碎片然后被传递到渲染流水线 1 用于渲染。如上所讨论的，当来自基元的碎片被渲染时，每个碎片的渲染器状态字索引用来在每个步骤在渲染流水线中执行渲染器状态字查找，以便渲染流水线的渲染单元被配置到适当的渲染状态。

一旦碎片已经被渲染，它就在显示之前被传递到适当的存储器以便存储，例如瓦片式缓冲器。

图 2 更详细地示出渲染流水线 1 的操作。如图 2 中所示，它具有多种不同的功能块或渲染单元 2、3，并且接收碎片 4 用于渲染。这些渲染单元 2、3 可以包括任何适当的这种单元，例如函数发生器、纹理映射器、混合器、雾化单元等，这在本领域中是已知的。

渲染流水线 1 的每个渲染单元 2、3 都包括功能块 10，其对接收的碎片执行渲染操作（并且因此需要被配置到适当的渲染状态），和渲染状态字接口 11，其可以用来从渲染状态字高速缓存 5 中查找适当的渲染状态数据。

如图 2 中所示，每个碎片被提供到渲染流水线 1，其具有碎片数据 6，其尤其指示碎片索引、渲染器状态字索引，和渲染碎片所必需

的任何其他碎片数据（其可以是本领域中已知的任何适当的这种数据，例如 RGBa 值）。

如上所讨论的，渲染器状态字索引 8 是 2 位指针，其指示渲染器状态字高速缓存 5 中的渲染器状态字寄存器中的哪一个存储了应当用来渲染所述碎片的渲染器状态字。

当渲染流水线的渲染单元 2、3 接收到将要渲染的碎片时，它首先使用与碎片 4 相关联的渲染器状态字索引 8 以利用其渲染器状态字接口 11 从渲染器状态字高速缓存 5 中查找相关的渲染状态数据。然后它利用该数据配置其功能块 10，并且然后渲染该碎片。在该实施例中，功能单元 10 仅查找和检索与它们的操作相关的该部分渲染器状态字（尽管如果需要的话它们可以查找整个渲染器状态字）。

当下一个碎片被渲染单元接收到时，它再次检查与该碎片相关联的渲染器状态字索引 8，从渲染器状态字高速缓存 5 中检索相关的配置数据，配置其功能块 10，并且然后渲染该碎片，等等。

这样，避免了只要发生渲染状态改变就需要重新配置整个渲染流水线。此外，渲染流水线 1 可以同时处理具有不同渲染状态的碎片（取决于存储在渲染器状态字高速缓存 5 中的不同的渲染器状态字的数量）。

在此应当注意的是，如图 2 中所示，尽管每当接收碎片时可能需要额外的硬件来执行渲染器状态“查找”，但是与常规的渲染系统相比，在渲染操作的速度或效率方面并没有额外的费用。

图 3 和 4 示出根据本发明设置的 3D 图形处理系统的另一实例。

图 3 特别更详细地示出渲染流水线 1 的设置。如图 3 中所示，该图形系统也包括资源分配器 24，其如同上面相对于图 1 所描述的那样操作、渲染器状态字高速缓存 5，其在该实施例中被示为包括两个渲染器状态字寄存器、光栅化器 23，其如同上面所讨论的那样操作、以及在渲染流水线 1 中的多个渲染单元。

也用示意图示出了渲染器输入级 30，基本上如上所讨论的，其将接收基元列表信息，包括渲染器状态字数据，并且然后提供适当的基元数据给光栅化器 23 用于光栅化。

正如本领域中已知的，光栅化器 23 将提供给渲染器输入级 30 的图形基元转换成图形碎片。光栅化器 23 利用由渲染器输入级 30

确定（或提供给渲染器输入级 30）的线段来产生（x，y）坐标对的序列，对被基元覆盖的所有（x，y）位置而言，每对对应于将用来渲染该基元的图形碎片的位置。

下一级是一个或多个函数发生器 31，正如本领域中已知的，其利用尤其是与正被渲染的基元的每个顶点相关联的数据，连同由光栅化器 23 确定的 x，y 碎片位置一起内插函数用于渲染图形碎片，例如用于设置和修改它们的纹理、颜色等，以确定渲染基元的每个碎片所必需的内插数据。这种被内插的数据通常是下述中的一个或两个：纹理映射坐标，基色，合成色和雾化。

然后渲染流水线 1 包括纹理映射器级 32。正如本领域中已知的，该纹理映射器 32 根据为每一个图形碎片确定的纹理坐标数据（s，t）从存储器（未示出）中检索纹理映射数据并且将它施加到沿渲染流水线 1 向下传递的图形碎片。纹理映射器 32 可以包括纹理映射高速缓存存储器，其中可以存储纹理映射数据以便被纹理映射器 32 更快速地检索。

下一级是混合级 33。正如本领域中已知的，该级将下述值作为其输入：来自渲染流水线 1 的碎片的一个或多个颜色值（在该碎片的着色和纹理映射之后）、和存在于渲染流水线的输出缓冲器 34（见下文）中的用于相应的碎片 x，y 位置的相应颜色值，并且产生这两个碎片的颜色值的混合型式。

渲染流水线 1 的最后一级是一组输出缓冲器 34，其中在渲染的图形碎片数据（即每个碎片的红色、绿色、和蓝色数据（和 α 数据，如果适当的话）值）被提供用于在显示屏上显示之前存储它。正如本领域中已知的，在基于瓦片的渲染的情形下，输出缓冲器 34 将包括一个或多个瓦片式缓冲器。在立即模式渲染的情形下，输出缓冲器 34 将包括在显示之前存储图形碎片数据的一个或多个高速缓存存储器。

该 3D 图形系统以与上面相对于图 1 和 2 所讨论的相同的方式操作。

因此，渲染器输入 30 和光栅化器 23 把指向高速缓存 5 中的适当的渲染器状态字高速缓存入口的索引与将要渲染的每个碎片相关联（并且如上所讨论的，资源分配器 24 检查和确保该适当的渲染器状

态字在发出基元用于渲染之前存储在高速缓存 5 中）。然后，当碎片被渲染时，由渲染流水线 1 的渲染单元使用与每个碎片相关联的渲染器状态字索引来从渲染器状态字高速缓存 5 中查找该适当的渲染器状态字数据并且配置渲染单元，以便碎片可以被正确地渲染。

在图 4 中用示意图更详细地示出了该操作，其中渲染流水线 1 的各个渲染单元被示为普通的功能单元 40。

从以上可见，本发明至少在其优选实施例中提供了一种图形渲染器和渲染流水线，其可以被设置以减少或甚至避免对在发生渲染器状态变化时停止渲染器的需要。

至少在优选实施例中这可以通过把渲染状态索引或指针与渲染器中的每个碎片相关联来实现，其跟随碎片通过渲染器。然后在渲染器中的步骤执行渲染状态查找，以便可以在使用中根据用于该特定碎片的特定渲染状态来配置渲染单元。这样，需要不同渲染状态的碎片可以同时存在于渲染器中并且可以同时被渲染器处理，由此避免了由于渲染状态变化引起的“碎片泡（fragment bubble）”和流水线阻塞的发生。

因此本发明至少在其优选实施例中避免了对在渲染器状态变化发生时停止和刷新渲染器的需要。这进一步意味着不再那么需要根据渲染状态对提供给渲染器的基元分类，使得，如果需要的话，基元可以根据其他的标准例如深度等被分类。

本发明还具有的优点是，至少在其优选实施例中，没有必要通知驱动器软件渲染状态的变化，因此可以降低驱动器软件的复杂性。

此外，本发明便于更长的渲染器（渲染流水线）的使用，因为它可以避免和减少在使用中整个渲染器阻塞和停止的风险。因此这便于使用更复杂的渲染操作，例如补色着色。

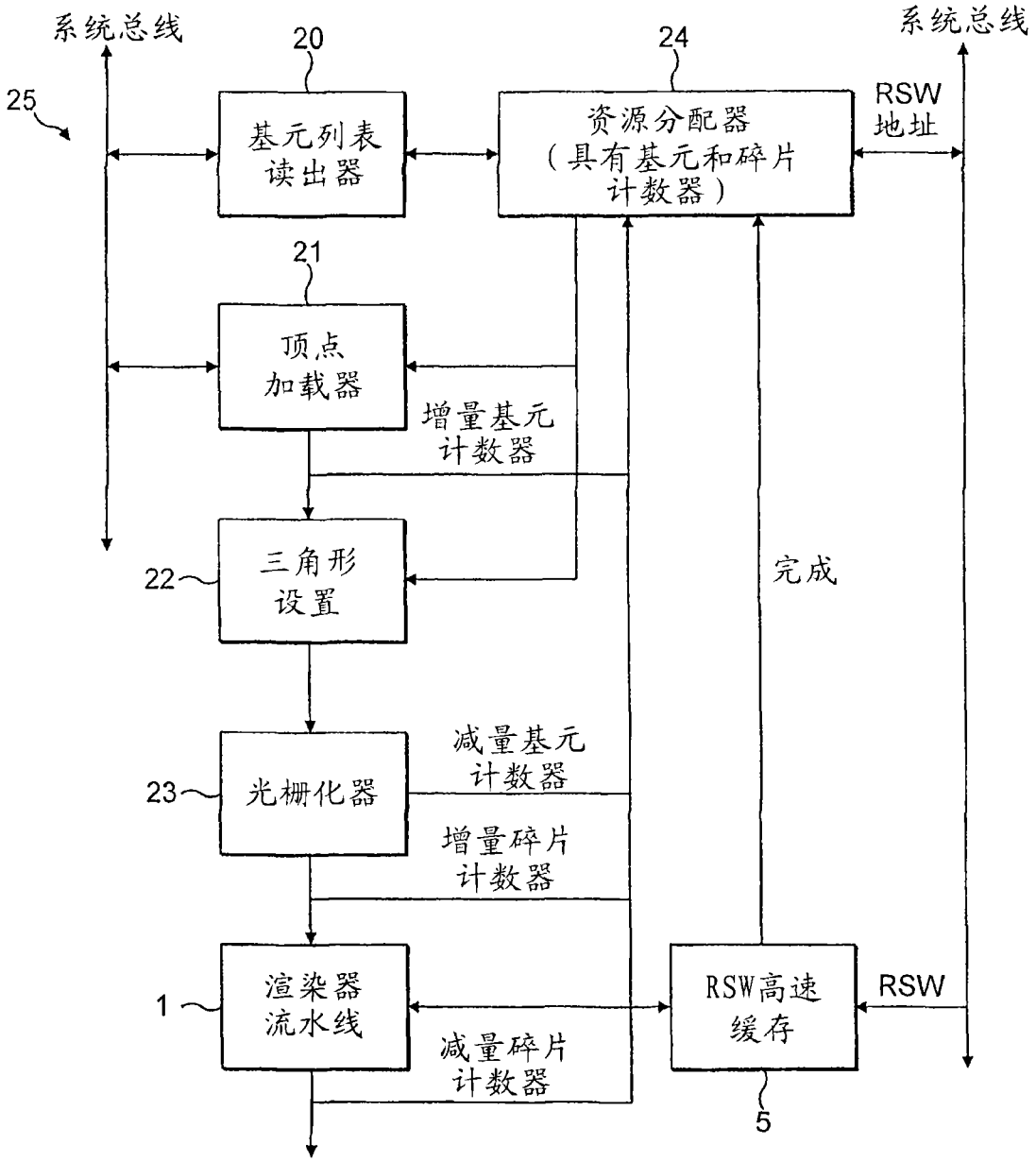

系统总线
20
24
系统总线
25
基元列表
读出器
资源分配器
（具有基元和碎片
计数器）
RSW
地址
21
顶点
加载器
增量基元
计数器
三角形
设置
22
完成
减量基元
计数器
光栅化器
23
增量碎片
计数器
渲染器
流水线
1
RSW高速
缓存
RSW
减量碎片
计数器
5

图 1

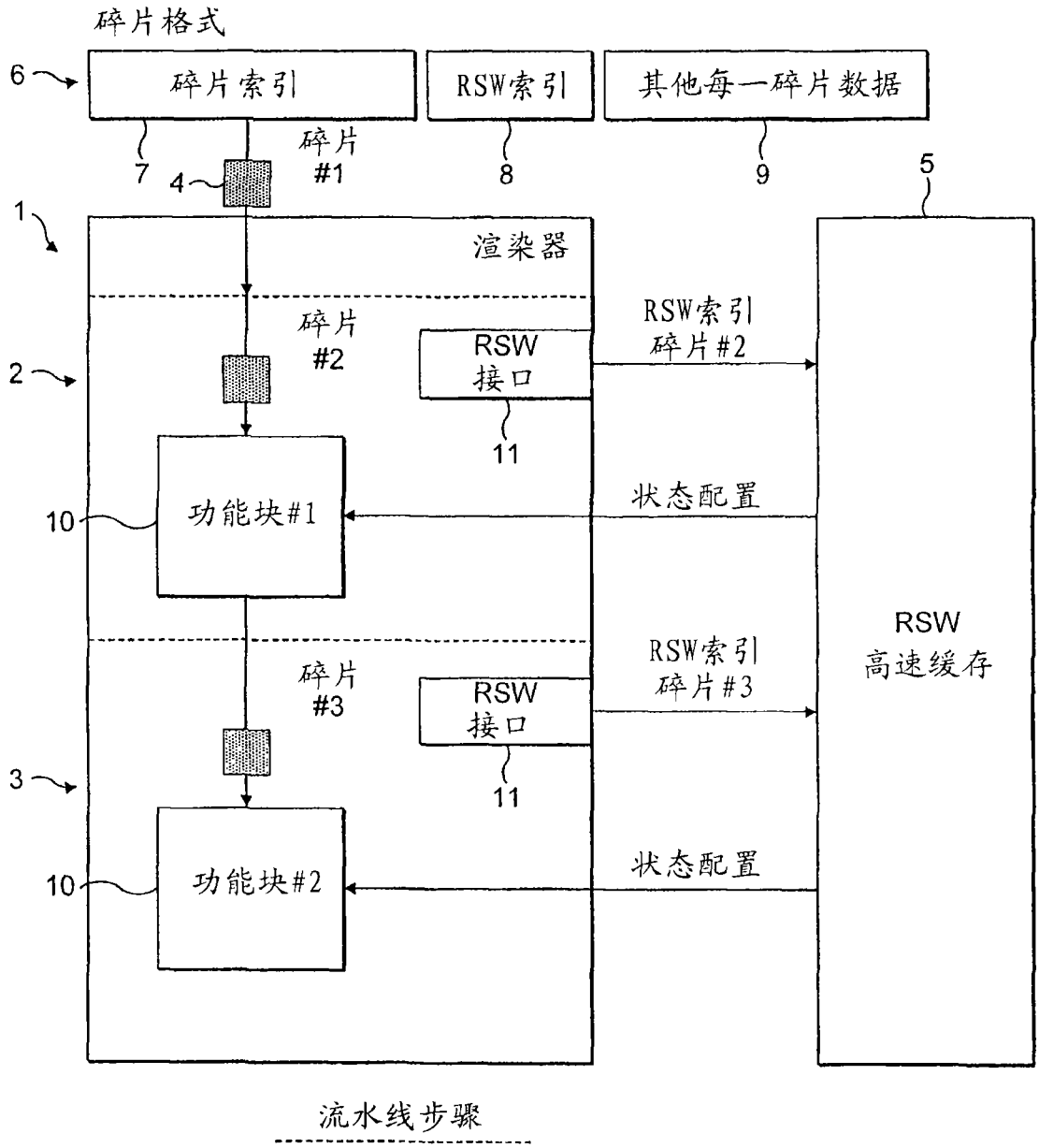

碎片格式
6
碎片索引
RSW索引
其他每一碎片数据
7
碎片
#1
8
9
5
4
1
渲染器
碎片
#2
RSW索引
碎片#2
RSW
接口
2
11
10
功能块#1
状态配置
RSW
高速缓存
RSW索引
碎片#3
碎片
#3
RSW
接口
3
11
10
功能块#2
状态配置
流水线步骤

图 2

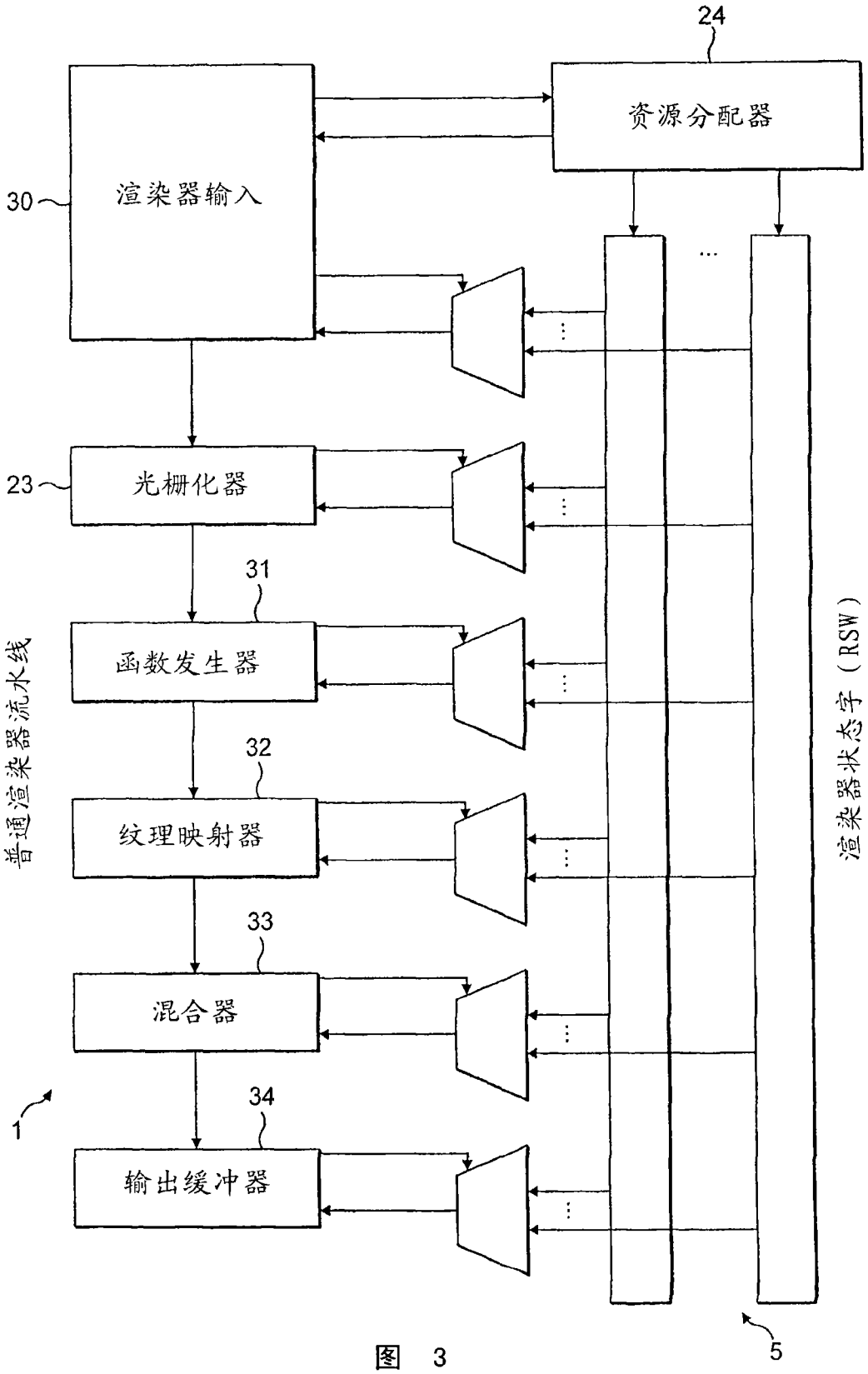

24
资源分配器
30
渲染器输入
23
光栅化器
31
函数发生器
32
纹理映射器
33
混合器
34
输出缓冲器
普通渲染器流水线
1
渲染器状态字（RSW）
5

图 3

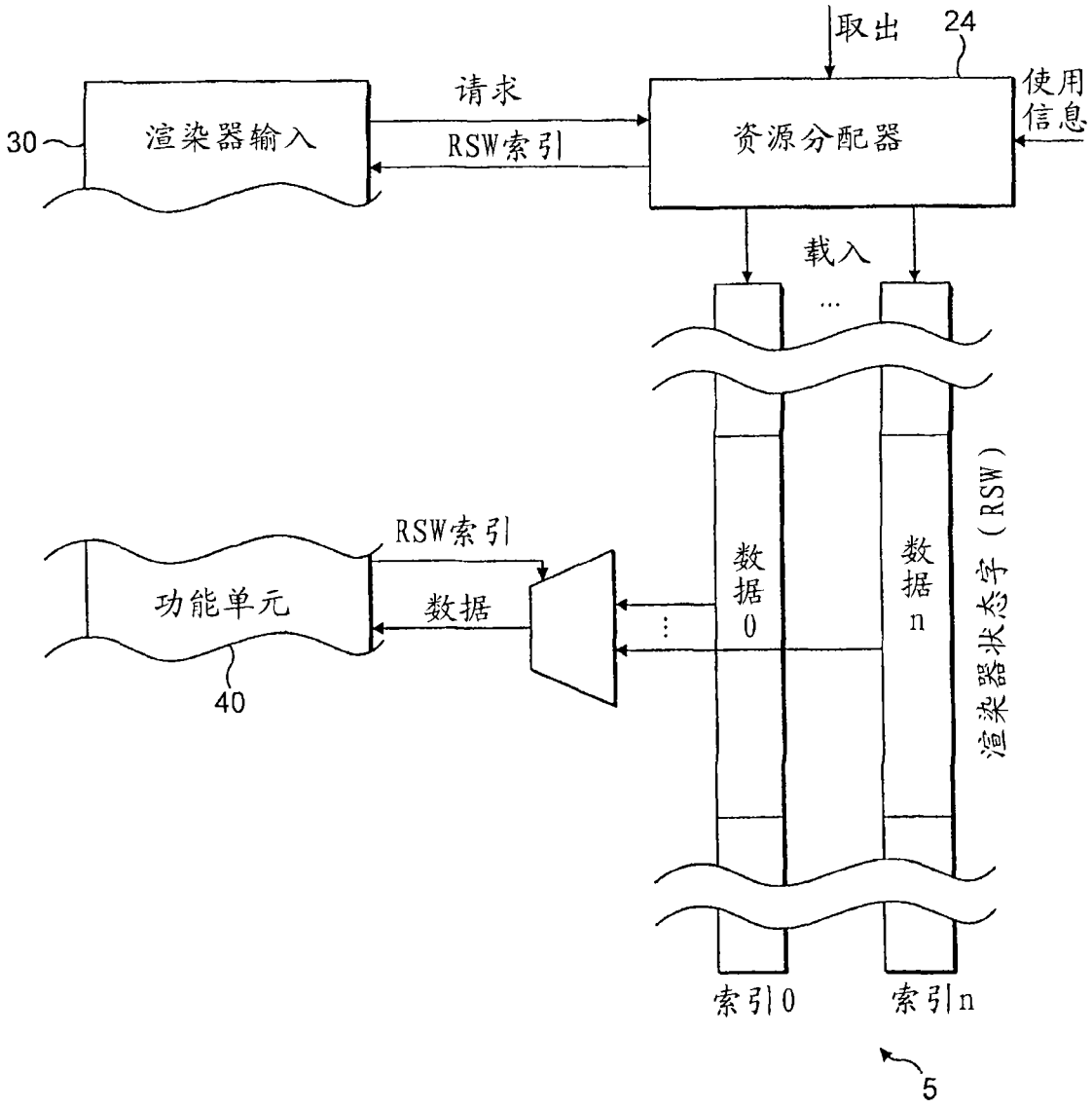

取出
24
使用
信息
请求
RSW索引
30
渲染器输入
资源分配器
载入
...
RSW索引
功能单元
数据
40
数据
0
数据
n
渲染器状态字（RSW）
索引0
索引n
5

图 4