(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 106164839 B
(45)授权公告日 2019.10.22

(21)申请号 201580017186.4
(22)申请日 2015.02.04
(65)同一申请的已公布的文献号
申请公布号 CN 106164839 A
(43)申请公布日 2016.11.23
(30)优先权数据
61/935,674 2014.02.04 US
(85)PCT国际申请进入国家阶段日
2016.09.28
(86)PCT国际申请的申请数据
PCT/US2015/014494 2015.02.04
(87)PCT国际申请的公布数据
WO2015/120073 EN 2015.08.13
(73)专利权人 触觉实验室股份有限公司
地址 美国纽约州
(72)发明人 D·威格多 S·L·桑德斯
R·J·J·柯斯塔 C·福林斯
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 侯颖媛
(51)Int.Cl.
G06F 3/0488(2006.01)
(56)对比文件
CN 102414653 A,2012.04.11,
CN 101989175 A,2011.03.23,
WO 2008008267 A2,2008.01.17,
Albert Ng etc..Designing for Low-Latency Direct-Touch Input.《Acm Symposium on User Interface Software & Technology》.2012,
审查员 李剑炜

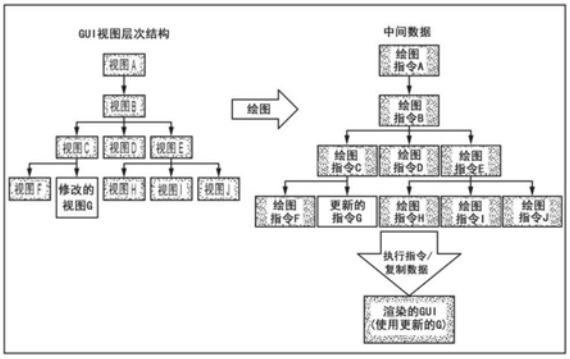
权利要求书8页 说明书20页 附图20页

(54)发明名称

以减小的等待时间提供对输入的视觉响应的方法

(57)摘要

一种用于在计算设备中以减小的等待时间提供对输入的视觉响应的方法,该方法包括计算针对第一图形用户界面元件的替代的中间数据集,每一个替代的中间数据集包括对于产生图形用户界面元件的视觉表示有用的数据。多个替代的中间数据集和针对第二图形用户界面元件的中间数据集存储在存储器中。该方法创建索引,该索引标识针对第一图形用户界面元件的该多个替代的中间数据集中的第一个替代的中间数据集,以便在形成最终像素图像中使用。该索引、针对图形用户界面元件的第一个替代的中间数据集、和针对第二图形用户界面元件的中间数据用来创建用于向用户显示的第一最终像素图像,该第一最终像素图像包括第一和第二图形用户界面元件。响应于用户输入,索引被修改来包括针对第一图形用户界面元件的多个替代的中间数据集的第二个的标识,并且所修改的索引用于创建用于向用户显示的最终像素图像。



CN 106164839 B

1.一种用于在计算设备中以减小的等待时间提供对输入的视觉响应的方法,所述方法包括以下步骤:

计算针对第一图形用户界面元件的多个替代的中间指令集,每一个替代的中间指令集包括对于产生所述第一图形用户界面元件的视觉表示有用的数据;

将针对所述第一图形用户界面元件的所述多个替代的中间指令集存储在存储器中;

将针对第二图形用户界面元件的至少一个中间指令集存储在所述存储器中;

创建索引,所述索引标识针对所述第一图形用户界面元件的所述多个替代的中间指令集中的第一个替代的中间指令集,以便在形成最终像素图像中使用;

使用所述索引,针对所述第一图形用户界面元件的所述第一个替代的中间指令集和针对第二图形用户界面元件的中间指令来创建用于向用户显示的第一最终像素图像,所述第一最终像素图像包括所述第一和第二图形用户界面元件;

接收来自用户输入设备的用户输入;

响应于所述用户输入,修改所述索引以包括针对所述第一图形用户界面元件的所述多个替代的中间指令集中的第二个替代的中间指令集的标识;

使用所修改的索引,针对所述第一图形用户界面元件的所述第二个替代的中间指令集和针对所述第二图形用户界面元件的中间指令来创建用于向用户显示的最终像素图像,所述最终像素图像包括所述第一和第二图形用户界面元件。

2.如权利要求1所述的方法,其特征在于,所述多个替代的中间指令集包括多个替代的绘图指令集。

3.如权利要求2所述的方法,其特征在于,使用所述第一个替代的中间指令集来创建用于向用户显示的第一最终像素图像的步骤包括执行第一个替代的绘图指令集。

4.如权利要求1所述的方法,其特征在于,所述多个替代的中间指令集包括多个替代的像素数据集。

5.如权利要求4所述的方法,其特征在于,使用所述第一个替代的中间指令集来创建用于向用户显示的第一最终像素图像的步骤包括将渲染的像素表示集复制到像素缓冲区。

6.如权利要求1所述的方法,其特征在于,所述多个替代的中间指令集包括多个替代的所述第一图形用户界面元件的视图的属性集,所述视图的属性影响所述第一图形用户界面元件的视觉外观。

7.如权利要求1所述的方法,其特征在于,所述多个替代的中间指令集包括多个替代的向量数据集。

8.如权利要求1所述的方法,其特征在于,所述多个替代的中间指令集包括多个替代的栅格数据集。

9.如权利要求1所述的方法,其特征在于,所述多个替代的中间指令集包括多个替代的显示列表。

10.如权利要求1所述的方法,其特征在于,所述第一图形用户界面元件是按钮,所述第一个替代的中间指令集包括处于非按压状态的所述按钮的表示,且所述第二个替代的中间指令集包括处于按压状态的所述按钮的表示。

11.如权利要求1所述的方法,其特征在于,所述第一图形用户界面元件是窗口,所述第一个替代的中间指令集包括处于非最大化状态的所述窗口的表示,且所述第二个替代的中

间指令集包括处于最大化状态的所述窗口的表示。

12.如权利要求1所述的方法，其特征在于，所述第一个替代的中间指令集包括当没有被另一个用户界面元件影响时的所述第一图形用户界面元件的表示，并且所述第二个替代的中间指令集包括当被其他用户界面元件影响时的所述第一图形用户界面元件的表示。

13.如权利要求1所述的方法，其特征在于，所述第一图形用户界面元件有多个替代的视觉状态，并且所述第二图形用户界面元件有单一的视觉状态。

14.如权利要求1所述的方法，其特征在于，所述第一和第二图形用户界面元件各自拥有多个替代的视觉状态。

15.如权利要求1所述的方法，其特征在于，所述第一图形用户界面元件是按钮，所述第一个替代的中间指令集包括处于禁用状态的所述按钮的表示，且所述第二个替代的中间指令集包括处于启用状态的所述按钮的表示。

16.如权利要求1所述的方法，其特征在于，进一步包括多于两个的替代的中间指令集。

17.如权利要求1所述的方法，其特征在于，所述第二个替代的中间指令集包括对于第三图形用户界面元件的替代的指令的指针。

18.如权利要求17所述的方法，其特征在于，所述第三图形用户界面元件是所述第一图形用户界面元件的子代。

19.如权利要求1所述的方法，其特征在于，所述第一个替代的中间指令集和所述第二个替代的中间指令集中的至少一个包括当所述用户进行滚动时的所述第一图形用户界面元件的表示。

20.如权利要求1所述的方法，其特征在于，所述第一个替代的中间指令集和所述第二个替代的中间指令集中的至少一个包括当所述用户进行摇摄时的所述第一图形用户界面元件的表示。

21.如权利要求1所述的方法，其特征在于，所述第一个替代的中间指令集和所述第二个替代的中间指令集中的至少一个包括当所述第一图形用户界面元件被拖动时的所述第一图形用户界面元件的表示。

22.如权利要求1所述的方法，其特征在于，所述计算多个替代的中间指令集的步骤由图形处理单元执行。

23.如权利要求1所述的方法，其特征在于，所述计算多个替代的中间指令集的步骤由中央处理单元执行。

24.如权利要求1所述的方法，其特征在于，所述创建索引的步骤由图形处理单元执行。

25.如权利要求1所述的方法，其特征在于，所述创建索引的步骤由中央处理单元执行。

26.如权利要求1所述的方法，其特征在于，所述使用所述索引的步骤由图形处理单元执行。

27.如权利要求1所述的方法，其特征在于，所述使用所述索引的步骤由中央处理单元执行。

28.如权利要求1所述的方法，其特征在于，所述第一图形用户界面元件是滚动视图的可见区域，所述第一个替代的中间指令集包括所述滚动视图的下一区域的表示，并且所述第二个替代的中间指令集包括所述滚动视图的先前区域的表示。

29.如权利要求1所述的方法，其特征在于，所述替代的中间指令集包括从由下列各项

组成的组中选择的至少一项的表示:处于非按压状态的按钮、处于按压状态的按钮、处于选中状态的控件、处于未选中状态的控件、处于启用状态的按钮、处于禁用状态的按钮、处于启用状态的元件、处于非启用状态的元件、处于悬停状态的元件、处于未悬停状态的元件、处于展开状态的元件、处于非展开状态的元件、具有焦点的元件、不具有焦点的元件、处于可见状态的元件、和处于不可见状态的元件。

30.如权利要求1所述的方法,其特征在于,所述替代的中间指令集包括在显示器的不同区域中的表示。

31.如权利要求1所述的方法,其特征在于,所述替代的中间指令集包括不同形状或尺寸的所述用户界面元件的表示。

32.如权利要求1所述的方法,其特征在于,所述替代的中间指令集包括所述用户界面元件的先前状态。

33.如权利要求1所述的方法,其特征在于,所述替代的中间指令集包括所述用户界面元件的可能的未来状态。

34.一种用于在计算设备中以减小的等待时间提供对输入的视觉响应的方法,所述方法包括以下步骤:

渲染针对第一图形用户界面元件的多个替代的中间指令集,每一个替代的中间指令集表示所述第一图形用户界面元件的替代的视觉表示;

将针对所述第一图形用户界面元件的所述多个替代的中间指令集存储在存储器中;

将针对第二图形用户界面元件的至少一个中间指令集存储在所述存储器中;

创建索引,所述索引标识针对所述第一图形用户界面元件的所述多个替代的中间指令集中的第一个替代的中间指令集,以便在形成最终像素图像中使用;

使用所述索引,针对所述第一图形用户界面元件的所述第一个替代的中间指令集和针对第二图形用户界面元件的中间指令来创建用于向用户显示的第一最终像素图像,所述第一最终像素图像包括所述第一和第二图形用户界面元件;

接收来自用户输入设备的用户输入;

响应于所述用户输入,修改所述索引以包括针对所述第一图形用户界面元件的所述多个替代的中间指令集中的第二个替代的中间指令集的标识;

使用所修改的索引,针对所述第一图形用户界面元件的所述第二个替代的中间指令集和针对所述第二图形用户界面元件的中间指令来创建用于向用户显示的最终像素图像,所述最终像素图像包括所述第一和第二图形用户界面元件。

35.如权利要求34所述的方法,其特征在于,使用所述第一个替代的中间指令集来创建用于向用户显示的第一最终像素图像的步骤包括将渲染的像素表示集复制到像素缓冲区。

36.如权利要求34所述的方法,其特征在于,所述多个替代的中间指令集包括多个替代的向量数据集。

37.如权利要求34所述的方法,其特征在于,所述多个替代的中间指令集包括多个替代的栅格数据集。

38.如权利要求34所述的方法,其特征在于,所述第一图形用户界面元件是按钮,所述第一个替代的中间指令集包括处于非按压状态的所述按钮的表示,且所述第二个替代的中间指令集包括处于按压状态的所述按钮的表示。

39.如权利要求34所述的方法，其特征在于，所述第一图形用户界面元件是窗口，所述第一个替代的中间指令集包括处于非最大化状态的所述窗口的表示，且所述第二个替代的中间指令集包括处于最大化状态的所述窗口的表示。

40.如权利要求34所述的方法，其特征在于，所述第一个替代的中间指令集包括当没有被另一个用户界面元件影响时的所述第一图形用户界面元件的表示，并且所述第二个替代的中间指令集包括当被其他用户界面元件影响时的所述第一图形用户界面元件的表示。

41.如权利要求34所述的方法，其特征在于，所述第一图形用户界面元件有多个替代的视觉状态，并且所述第二图形用户界面元件有单一的视觉状态。

42.如权利要求34所述的方法，其特征在于，所述第一和第二图形用户界面元件各自拥有多个替代的视觉状态。

43.如权利要求34所述的方法，其特征在于，所述第一图形用户界面元件是按钮，所述第一个替代的中间指令集包括处于禁用状态的所述按钮的表示，且所述第二个替代的中间指令集包括处于启用状态的所述按钮的表示。

44.如权利要求34所述的方法，其特征在于，进一步包括多于两个的替代的中间指令集。

45.如权利要求34所述的方法，其特征在于，所述第二个替代的中间指令集包括对于第三图形用户界面元件的替代的指令的指针。

46.如权利要求45所述的方法，其特征在于，所述第三图形用户界面元件是所述第一图形用户界面元件的子代。

47.如权利要求34所述的方法，其特征在于，所述第一个替代的中间指令集和所述第二个替代的中间指令集中的至少一个包括当所述用户进行滚动时的所述第一图形用户界面元件的表示。

48.如权利要求34所述的方法，其特征在于，所述第一个替代的中间指令集和所述第二个替代的中间指令集中的至少一个包括当所述用户进行摇摄时的所述第一图形用户界面元件的表示。

49.如权利要求34所述的方法，其特征在于，所述第一个替代的中间指令集和所述第二个替代的中间指令集中的至少一个包括当所述第一图形用户界面元件被拖动时的所述第一图形用户界面元件的表示。

50.如权利要求34所述的方法，其特征在于，所述计算多个替代的中间指令集的步骤由图形处理单元执行。

51.如权利要求34所述的方法，其特征在于，所述计算多个替代的中间指令集的步骤由中央处理单元执行。

52.如权利要求34所述的方法，其特征在于，所述创建索引的步骤由图形处理单元执行。

53.如权利要求34所述的方法，其特征在于，所述创建索引的步骤由中央处理单元执行。

54.如权利要求34所述的方法，其特征在于，所述使用所述索引的步骤由图形处理单元执行。

55.如权利要求34所述的方法，其特征在于，所述使用所述索引的步骤由中央处理单元

执行。

56.如权利要求34所述的方法，其特征在于，所述第一图形用户界面元件是滚动视图的可见区域，所述第一个替代的中间指令集包括所述滚动视图的下一区域的表示，并且所述第二个替代的中间指令集包括所述滚动视图的先前区域的表示。

57.如权利要求34所述的方法，其特征在于，所述替代的中间指令集包括从由下列各项组成的组中选择的至少一项的表示：处于非按压状态的按钮、处于按压状态的按钮、处于选中状态的控件、处于未选中状态的控件、处于启用状态的按钮、处于禁用状态的按钮、处于激活状态的元件、处于非激活状态的元件、处于悬停状态的元件、处于未悬停状态的元件、处于展开状态的元件、处于未展开状态的元件、具有焦点的元件、不具有焦点的元件、处于可见状态的元件、和处于不可见状态的元件。

58.如权利要求34所述的方法，其特征在于，所述替代的中间指令集包括在显示器的不同区域中的表示。

59.如权利要求34所述的方法，其特征在于，所述替代的中间指令集包括不同形状或尺寸的所述用户界面元件的表示。

60.如权利要求34所述的方法，其特征在于，所述替代的中间指令集包括所述用户界面元件的先前状态。

61.如权利要求34所述的方法，其特征在于，所述替代的中间指令集包括所述用户界面元件的可能的未来状态。

62.一种用于在计算设备中以减小的等待时间提供对输入的视觉响应的方法，所述方法包括以下步骤：

计算针对第一图形用户界面元件的多个替代的中间指令集，每一个替代的中间指令集包括绘图指令，所述绘图指令用于渲染所述第一图形用户界面元件的替代的视觉表示；

将针对所述第一图形用户界面元件的所述多个替代的中间指令集存储在存储器中；

将针对第二图形用户界面元件的至少一个中间指令集存储在所述存储器中；

创建索引，所述索引标识针对所述第一图形用户界面元件的所述多个替代的中间指令集中的第一个替代的中间指令集，以便在形成最终像素图像中使用；

使用所述索引，针对所述第一图形用户界面元件的所述第一个替代的中间指令集和针对第二图形用户界面元件的中间指令来创建用于向用户显示的第一最终像素图像，所述第一最终像素图像包括所述第一和第二图形用户界面元件；

接收来自用户输入设备的用户输入；

响应于所述用户输入，修改所述索引以包括针对所述第一图形用户界面元件的所述多个替代的中间指令集的第二个替代的中间指令集的标识；

使用所修改的索引，针对所述第一图形用户界面元件的所述第二个替代的中间指令集和针对所述第二图形用户界面元件的中间指令，来创建用于向用户显示的最终像素图像，所述最终像素图像包括所述第一和第二图形用户界面元件。

63.如权利要求62所述的方法，其特征在于，使用所述第一个替代的中间指令集来创建用于向用户显示的第一最终像素图像的步骤包括执行第一个替代的绘图指令集。

64.如权利要求62所述的方法，其特征在于，所述多个替代的中间指令集包括多个替代的向量数据集。

65.如权利要求62所述的方法，其特征在于，所述多个替代的中间指令集包括多个替代的栅格数据集。

66.如权利要求62所述的方法，其特征在于，所述多个替代的中间指令集包括多个替代的显示列表集。

67.如权利要求62所述的方法，其特征在于，所述第一图形用户界面元件是按钮，所述第一个替代的中间指令集包括处于非按压状态的所述按钮的表示，且所述第二个替代的中间指令集包括处于按压状态的所述按钮的表示。

68.如权利要求62所述的方法，其特征在于，所述第一图形用户界面元件是窗口，所述第一个替代的中间指令集包括处于非最大化状态的所述窗口的表示，且所述第二个替代的中间指令集包括处于最大化状态的所述窗口的表示。

69.如权利要求62所述的方法，其特征在于，所述第一个替代的中间指令集包括当没有被另一个用户界面元件影响时的所述第一图形用户界面元件的表示，并且所述第二个替代的中间指令集包括当被其他用户界面元件影响时的所述第一图形用户界面元件的表示。

70.如权利要求62所述的方法，其特征在于，所述第一图形用户界面元件有多个替代的视觉状态，并且所述第二图形用户界面元件有单一的视觉状态。

71.如权利要求62所述的方法，其特征在于，所述第一和第二图形用户界面元件各自拥有多个替代的视觉状态。

72.如权利要求62所述的方法，其特征在于，所述第一图形用户界面元件是按钮，所述第一个替代的中间指令集包括处于禁用状态的所述按钮的表示，且所述第二个替代的中间指令集包括处于启用状态的所述按钮的表示。

73.如权利要求62所述的方法，其特征在于，进一步包括多于两个的替代的中间指令集。

74.如权利要求62所述的方法，其特征在于，所述第二个替代的中间指令集包括对于第三图形用户界面元件的替代的指令的指针。

75.如权利要求74所述的方法，其特征在于，所述第三图形用户界面元件是所述第一图形用户界面元件的子代。

76.如权利要求62所述的方法，其特征在于，所述第一个替代的中间指令集和所述第二个替代的中间指令集中的至少一个包括当所述用户进行滚动时的所述第一图形用户界面元件的表示。

77.如权利要求62所述的方法，其特征在于，所述第一个替代的中间指令集和所述第二个替代的中间指令集中的至少一个包括当所述用户进行摇摄时的所述第一图形用户界面元件的表示。

78.如权利要求62所述的方法，其特征在于，所述第一个替代的中间指令集和所述第二个替代的中间指令集中的至少一个包括当所述第一图形用户界面元件被拖动时的所述第一图形用户界面元件的表示。

79.如权利要求62所述的方法，其特征在于，所述计算多个替代的中间指令集的步骤由图形处理单元执行。

80.如权利要求62所述的方法，其特征在于，所述计算多个替代的中间指令集的步骤由中央处理单元执行。

81.如权利要求62所述的方法，其特征在于，所述创建索引的步骤由图形处理单元执行。

82.如权利要求62所述的方法，其特征在于，所述创建索引的步骤由中央处理单元执行。

83.如权利要求62所述的方法，其特征在于，所述使用所述索引的步骤由图形处理单元执行。

84.如权利要求62所述的方法，其特征在于，所述使用所述索引的步骤由中央处理单元执行。

85.如权利要求62所述的方法，其特征在于，所述第一图形用户界面元件是滚动视图的可见区域，所述第一个替代的中间指令集包括所述滚动视图的下一区域的表示，并且所述第二个替代的中间指令集包括所述滚动视图的先前区域的表示。

86.如权利要求62所述的方法，其特征在于，所述替代的中间指令集包括从由下列各项组成的组中选择的至少一项的表示：处于非按压状态的按钮、处于按压状态的按、处于选中状态的控件、处于未选中状态的控件、处于启用状态的按钮、处于禁用状态的按钮、处于激活状态的元件、处于非激活状态的元件、处于悬停状态的元件、处于未悬停状态的元件、处于展开状态的元件、处于非展开状态的元件、具有焦点的元件、不具有焦点的元件、处于可见状态的元件、和处于不可见状态的元件。

87.如权利要求62所述的方法，其特征在于，所述替代的中间指令集包括在显示器的不同区域中的表示。

88.如权利要求62所述的方法，其特征在于，所述替代的中间指令集包括不同形状或尺寸的所述用户界面元件的表示。

89.如权利要求62所述的方法，其特征在于，所述替代的中间指令集包括所述用户界面元件的先前状态。

90.如权利要求62所述的方法，其特征在于，所述替代的中间指令集包括所述用户界面元件的可能的未来状态。

91.一种用于在计算设备中以减小的等待时间提供对输入的听觉响应的方法，包括：

计算针对第一用户界面元件的多个替代的中间指令集，每一个替代的中间指令集包括对于产生与用户界面元件相关的听觉输出有用的数据；

将针对所述第一用户界面元件的所述多个替代的中间指令集存储在存储器中；

将针对第二用户界面元件的至少一个中间指令集存储在所述存储器中；

创建索引，所述索引标识针对所述第一用户界面元件的所述多个替代的中间指令集中的第一个替代的中间指令集，以便在形成最终声音中使用；

使用所述索引，针对所述用户界面元件的所述第一个替代的中间指令集和针对第二用户界面元件的中间指令来创建用于向用户显示的第一最终声音，所述第一最终声音包括所述第一和第二用户界面元件的声音；

接收来自用户输入设备的用户输入；

响应于所述用户输入，修改所述索引以包括针对所述第一用户界面元件的所述多个替代的中间指令集中的第二个替代的中间指令集的标识；

使用所修改的索引，针对所述第一用户界面元件的所述第二个替代的中间指令集和针

对所述第二用户界面元件的中间指令来创建用于向用户输出的最终声音,所述最终声音包括来自所述第一和第二用户界面元件的声音。

以减小的等待时间提供对输入的视觉响应的方法

[0001] 本申请要求于2014年2月4日申请的美国临时专利申请No.61/935,674的优先权，且是该临时专利申请的非临时申请，其全部公开通过引用纳入本申请。

[0002] 本申请涉及诸如在2013年10月4日提交的题为“Hybrid Systems And Methods For Low-Latency User Input Processing And Feedback”的美国专利申请No.14/046,823、2013年3月15日提交的题为“Low-Latency Touch Sensitive Device”的美国专利申请No.13/841,436、2013年10月4日提交的题为“Hybrid Systems And Methods For Low-Latency User Input Processing And Feedback”的美国专利申请No.14/046,819、2013年3月15日提交的题为“Fast Multi-Touch Stylus”的美国专利申请No.61/798,948、2013年3月15日提交的题为“Fast Multi-Touch Sensor With User-Identification Techniques”的美国专利申请No.61/799,035、2013年3月15日提交的题为“Fast Multi-Touch Noise Reduction”的美国专利申请No.61/798,828、2013年3月15日提交的题为“Active Optical Stylus”的美国专利申请No.61/798,708、2012年10月5日提交的题为“Hybrid Systems And Methods For Low-Latency User Input Processing And Feedback”的美国专利申请No.61/710,256、2013年7月12日提交的题为“Fast Multi-Touch Post Processing”的美国专利申请No.61/845,892、2013年7月12日提交的题为“Reducing Control Response Latency With Defined Cross-Control Behavior”的美国专利申请No.61/845,879、2013年9月18日提交的题为“Systems And Methods For Providing Response To User Input Using Information About State Changes And Predicting Future User Input”的美国专利申请No.61/879,245、2013年9月21日提交的题为“Systems And Methods For Providing Response To User Input Using Information About State Changes And Predicting Future User Input”的美国专利申请No.61/880,887、2013年11月1日提交的题为“Fast Multi-Touch Post Processing”的美国专利申请No.14/069,609、2013年10月7日提交的题为“Touch And Stylus Latency Testing Apparatus”的美国专利申请No.61/887,615、2014年1月16日提交的题为“Fast Multi-Touch Update Rate Throttling”的美国专利申请No.61/928,069、2014年1月22日提交的题为“Dynamic Assignment Of Possible Channels In A Touch Sensor”的美国专利申请No.61/930,159和2014年1月27日提交的题为“Decimation Strategies For Input Event Processing”的美国专利申请No.61/932,047中披露的快速多触摸传感器以及其他界面之类的用户界面。那些申请的完整公开内容以引用的方式并入本文中。

技术领域

[0004] 本发明总体涉及用户输入的领域，尤其涉及给予低等待时间用户体验的用户输入系统。

附图说明

[0005] 下列对如附图所示的各实施例的更加较具体的描述，本公开的前述的及其他目标、特征，和优点将变得显而易见，在附图中，各个图中的附图标记表示相同部分。附图不一定按比例绘制，而是着重于所公开实施例的原理。

[0006] 图1示出触摸用户界面中的拖动等待时间在100ms、50ms、10ms和1ms时的效果的演示。

[0007] 图2示出收件箱的用户界面元件的示例，其中该元件具有对触摸用户交互的低等待时间、低保真响应以及对触摸用户交互的高等待时间、高保真响应。

[0008] 图3示出滑动双态元件(toggle element)的用户界面的示例。

[0009] 光标310(由含有“十字”字符的框所表示)可被拖动到目标320(第二个空框，在右边)以激活UI元件。

[0010] 使用低等待时间和高等待时间系统两者来启用此元件以便提供触摸交互，其中加速了移动的元件310，因此提供低等待时间体验。

[0011] 图4示出用于等待时间感知研究的原型(prototype)高性能触摸系统的基本架构的说明性实施例。

[0012] 图5示出使用图4的原型设备的等待时间感知研究的结果。

[0013] 图6示出按钮的用户界面元件的示例，其中该元件具有对触摸用户交互的低等待时间、低保真响应以及对触摸用户交互的高等待时间、高保真响应。

[0014] 图7示出可变尺寸框的用户界面元件的示例，其中该元件具有对触摸用户交互的低等待时间、低保真响应以及对触摸用户交互的高等待时间、高保真响应。

[0015] 图8示出滚动列表的用户界面元件的示例，其中该元件具有对触摸用户交互的低等待时间、低保真响应以及对触摸用户交互的高等待时间、高保真响应。

[0016] 图9示出低等待时间输入设备的基本构架和信息流的说明性实施例。

[0017] 图10示出用于音量控制的UI。当拖动滑动件时，工具提示(tooltip)出现，显示当前设置的数字表示。使用提供触摸交互的低等待时间和高等待时间系统两者来启用此元件，其中加速了移动的元件，因此提供低等待时间体验。

[0018] 图11示出与本混合反馈用户界面系统中的笔输入的UI的实施例相比的现有技术系统中的笔输入的系统的响应。在此混合系统中，墨水笔划对笔输入有低等待时间响应，以及对笔用户输入有高等待时间响应。

[0019] 图12示出本系统的实施例，其中数据流过经过系统的组件的两个重叠路径以支持高与低等待时间反馈。

[0020] 图13示出本领域公知的编程范式，称为模型视图控制器(Model View Controller)。

[0021] 图14示出系统的架构的实施例，该系统的架构支持开发和运行对用户输入有混合的高与低等待时间响应的应用。

[0022] 图15是示出根据现有技术的GUI视图和中间数据的分级结构的框图。

[0023] 图16是示出中间数据的执行的时间线视图。

[0024] 图17是示出GUI视图和中间数据的分级结构的框图。

[0025] 图18-19是示出根据本公开系统和方法的实施例的GUI视图和中间数据的分级结

构的框图。

[0026] 图20-23是示出根据本公开系统和方法的GPU、CPU、输入设备控制器和显示器的操作的框图。

具体实施方式

[0027] 以下描述和附图是说明性的，并且不解释为限制性的。描述了众多特定的细节以提供透彻的理解。然而，在某些实例中，不描述公知的或常规的细节以避免使描述变得模糊。在本公开中，对一个实施例或实施例的引用不一定是对同一个实施例的引用；此类引用意味着至少一个实施例。

[0028] 在本说明书中对“一个实施例”或“一实施例”的引用表示结合该实施例描述的特定特征、结构或特性被包括在本公开的至少一个实施例中。在本说明书中的不同位置出现短语“在一个实施例中”不一定都是指同一个实施例，也不是指与其他实施例互相排斥的单独的或备选实施例。此外，还描述了可由一些实施例呈现而不可由其他实施例呈现的各种特征。类似地，还描述了可能是对于一些实施例但不是对于其他实施例的要求的各种要求。

[0029] 本申请涉及诸如在2013年3月15日提交的题为“Low-Latency Touch Sensitive Device”的美国专利申请No.13/841,436、2013年3月15日提交的题为“Fast Multi-Touch Stylus”的美国专利申请No.61/798,948、2013年3月15日提交的题为“Fast Multi-Touch Sensor With User-Identification Techniques”的美国专利申请No.61/799,035、2013年3月15日提交的题为“Fast Multi-Touch Noise Reduction”的美国专利申请No.61/798,828、2013年3月15日提交的题为“Active Optical Stylus”的美国专利申请No.61/798,708、2012年10月5日提交的题为“Hybrid Systems And Methods For Low-Latency User Input Processing And Feedback”的美国专利申请No.61/710,256、2013年7月12日提交的题为“Fast Multi-Touch Post Processing”的美国专利申请No.61/845,892、2013年7月12日提交的题为“Reducing Control Response Latency With Defined Cross-Control Behavior”的美国专利申请No.61/845,879和2013年9月18日提交的题为“Systems And Methods For Providing Response To User Input Using Information About State Changes And Predicting Future User Input”的美国专利申请No.61/879,245中披露的快速多触摸传感器以及其他界面之类的用户界面。那些申请的完整公开内容以引用的方式并入本文中。

[0030] 在各种实施例中，本公开针对提供具有低等待时间的直接操控用户界面的系统和方法。伪“真实世界”物体的直接物理操控是针对很多类型的输入设备(诸如实现直接触摸输入、触笔输入、空中手势输入(in-air gesture input)的那些输入设备)以及间接设备(包括鼠标、触摸板、手写平板(pen tablet)等)所使用的普通用户界面隐喻。出于本公开的目的，用户界面中的等待时间涉及为用户呈现出对物理输入动作的响应所花费的时间。测试已表明用户偏爱低等待时间且用户确实能感知到低至5-10ms的等待时间，如将在下文更加详细地描述。

[0031] 图1示出示例性触摸用户界面中的等待时间分别在100ms(附图标记110)、50ms(附图标记120)、10ms(附图标记130)和1ms(附图标记140)时的效果的演示。当拖动对象时，增加的等待时间被反映为用户的手指与正在被拖动的物体(在此情况中为正方形用户界面元

件)之间增加的距离。如可观察到,等待时间的效果在100ms(附图标记110)和50ms(附图标记120)时显著,而在10ms(附图标记130)时变得逐渐较不显著,且实际上在1ms(附图标记140)时消失。图11示出等待时间在示例性触笔或笔用户界面(1110、1120)中的效果。在此示例中,滞后1120是可见的,作为触笔1100尖与所计算笔划1110之间的增加的距离。利用低等待时间系统的引入,触笔1100尖与所计算笔划1130之间的距离将显著减小。

[0032] 在以实施例中,本公开的系统和方法提供了混合触摸用户界面,其提供了具有少于10ms的等待时间的即时视觉反馈,交织或叠加有在更高水平的等待时间上的附加视觉响应。在一些实施例中,这两组响应的设计可被设计成在视觉上统一的,使得用户不能够区分它们。在一些实施例中,"低等待时间"响应可在等待时间上超过10ms。

[0033] 等待时间的原因

[0034] 在不同实施例中,用户输入设备和处理其输入的等待时间可以有许多来源,包括

[0035] (1)捕捉触摸事件的物理传感器;

[0036] (2)处理触摸事件并为显示器产生输出的软件;

[0037] (3)显示器本身;

[0038] (4)包括总线的组件之间的数据传输;

[0039] (5)存储器存储或短暂缓冲区器中的数据内部存储;

[0040] (6)系统资源的中断与竞争;

[0041] (7)可引入等待时间的电路的其他来源;

[0042] (8)物理限制,诸如光速及其在电路构架中的余波(repercussion);

[0043] (9)机械限制,诸如阻性触摸传感器弯曲回到其"中性"状态所需的时间。

[0044] 在不同实施例中,可通过改进这些组件中的一个或多个中的等待时间来降低系统等待时间。在实施例中,当前公开的系统和方法提供一种输入设备,该输入设备可通过将低等待时间输入传感器与具有专用处理系统的显示器结合来获得1ms或更低的等待时间。在实施例中,当前公开的系统和方法提供一种输入设备,该输入设备可通过将低等待时间输入传感器与具有专用处理系统的显示器结合来获得5ms或更低的等待时间。在进一步实施例中,当前公开的系统和方法提供一种输入设备,该输入设备可通过将低等待时间输入传感器与具有专用处理系统的显示器结合来获得0.1ms或更低的等待时间。在进一步实施例中,当前公开的系统和方法提供一种输入设备,该输入设备可通过将这种低等待时间输入传感器与具有专用处理系统的显示器结合来获得10ms或更低的等待时间。在实施例中,为了获得这种极低的等待时间,当前公开的系统和方法可用专用的定制编程的现场可编程门阵列(FPGA)或专用集成电路(ASIC)来替换常规的操作系统(OS)和计算硬件。在实施例中,FPGA或ASIC替换常规的OS和计算硬件以提供低等待时间响应,同时将传统的OS和计算硬件保留在位以提供较高的等待时间响应(在除了低等待时间响应的另外情况中使用)。在另一实施例中,可通过将附加的逻辑集成到现有组件(诸如但不限于图形处理单元(GPU)、输入设备控制器、中央处理器(CPU)或芯片上的系统(SoC))中来替换所描述的FPGA或ASIC的功能中的一些或全部。可将低等待时间逻辑编码在硬件中或在由那些组件或其他组件所储存和/或执行的软件中。在需要多种组件的实施例中,可通过使用共享的存储器来帮助通信和/或同步化。在这些实施例的任何一个中,以高或低等待时间提供的响应可被混合在一起,或者响应于任何给定输入事件,可仅提供高等待时间或低等待时间。

[0045] 在不同的实施例中，所公开的系统和方法提供的在本文中被称为“混合反馈”。在混合反馈系统中，对输入的基本系统响应中的一些在逻辑上与较宽泛的应用逻辑分开。此结果提供具有灵敏输入处理器的系统，能够对用户输入事件提供几乎立即的系统反馈，基于以传统水平的等待时间所提供的应用逻辑具有更多的反馈。在一些实施例中，在视觉上提供了这些系统响应。在不同的实施例中，可通过音频或震动触觉反馈来提供混合反馈系统的低等待时间组件。在一些实施例中，可在与应用逻辑反馈相同的模态中提供几乎立即的反馈。在一些实施例中，可在不同的模态或多个模态中提供低等待时间反馈。图2中示出了全视觉实施例的示例，在此情况中示出触摸输入设备的使用。具体地，图2示出在用户触摸然后拖动表示收件箱的图标210后的结果。当用户触摸图标210时，可显示边框220或其他适当的基元(primitive)。在实施例中，在全视觉低等待时间反馈中，由于其容易呈现，可选择适当的低等待时间表示。在实施例中，可使用可提供适当低等待时间表示的一个或多个基元来提供低等待时间反馈。在实施例中，如果用户在触摸显示器200上将图标拖动到另一个地方，则显示低保真边界230且可用例如1ms的低等待时间来操控(例如，移动)低保真边界230。同时地，可用较高的等待时间来示出图标210的移动。在实施例中，用户可感知到几乎立即的低等待时间响应与可能较慢的应用逻辑反馈之间在响应中的差异。在另一实施例中，低等待时间与传统响应之间在响应中的此差异被混合且对于用户是较不容易看见的或不可容易看见的。在实施例中，与传统路径应用逻辑反馈相比，可以以较低的等待时间提供几乎立即的反馈。在实施例中，至少在一些情况中，与应用逻辑反馈相比，可以以类似的或甚至较高的保真度提供低等待时间响应。在实施例中，低等待时间几乎立即的反馈的形式由应用逻辑所规定，或者由存在于系统软件(诸如用户界面工具箱)中的逻辑所规定。例如，在实施例中，应用逻辑可预渲染多种图形基元，然后低等待时间子系统可使用该图形基元。类似地，在实施例中，软件工具箱可提供开发图形基元的工具，可在低等待时间系统需要之前渲染图形基元。在实施例中，可预先确定低等待时间响应，或者在其他情况下不考虑应用和/或系统软件逻辑来确定低等待时间响应。在实施例中，单个预渲染或部分渲染的低等待时间响应，或预渲染或部分渲染的低等待时间响应可被预加载到存储器中以便被需要用来响应于用户输入事件之前可被低等待时间子系统访问。

[0046] 在实施例中，低等待时间输出的模态可以是音频的。在实施例中，例如，可使用低等待时间系统以向音频输出系统快速发送麦克风输入，可为用户提供说出到系统的用户自己声音的“回声”。这样的低等待时间输出可提供具有与传统模拟电话相同类型的回声特征的效果，允许用户听见它们自己的声音。在实施例中，响应于用户输入事件(例如，触摸、手势、笔输入或口头输入)，可提供低等待时间音频反馈，如果在视觉上则具有较高的等待时间响应。

[0047] 在图3中示出采用本方法和系统的系统的另一说明性实施例。在该说明性实施例中，光标310(由含有“十字”字符的框表示)可被拖动到设备的屏幕300上的任何地方。当光标310被拖动到目标框320时，接受UI动作。如果光标310被拖动到屏幕300上的其他地方，则拒绝该动作。在实施例中，当被拖动时，用低等待时间拖拽光标310，因此该光标310跟踪用户的手指而没有可感知的等待时间。在实施例中，可用较高的等待时间拖拽目标320而不影响用户感知。类似地，在实施例中，“拒绝”或“接受”的响应330随后可容易感知地发生，因此可以以较高的等待时间来拖拽(例如，不使用低等待时间子系统)而不影响用户感知。

[0048] 应该理解到，图示的实施例是示例性的。图3中示出的原理可被应用到任意种类的UI元件，包括本领域中现今已知的或以后开发的所有UI元件。类似地，基本上可在不同类型的输入设备和/或输出设备上与任意种类的输入事件一起使用图3中示出的原理。例如，在实施例中，除了如上所述的“触摸”事件之外，输入事件可包括而不限于，空中或表面上手势、语言、故意的(或非故意的)眼睛运动、以及笔。在实施例中，一旦发生手势，任何UI元件的响应就可分为两部分，其中，低等待时间响应(例如，UI元件的低等待时间表示被表示且快速地响应，例如，在0.01ms内)，以及用由系统所普通呈现的等待时间提供非低等待时间响应(例如，UI元件的进一步细化表示)，该系统不提供加速的输入。在实施例中，在混合系统中可不分离响应，响应代替地可以是完全低等待时间，而另外用较高的等待时间执行不负责低等待时间响应的应用逻辑。

[0049] 在实施例中，可使用多种技术来获得触摸和/或手势输入事件，包括而不限于：阻式、直接照明、受抑(frustrated)全内反射、漫射照明、投射电容、电容耦合、声波、以及像素传感器。在实施例中，可使用阻式、视觉的、电容的、磁的、红外的、光成像、色散信号、声脉冲或其他技术来实现笔输入。在实施例中，也可使用视觉传感器或手持物体(包括含有传感器的那些和仅用于跟踪的那些)或者诸如用2D和3D传感器之类的不需要手持的物体来实现手势输入。也预期用于识别输入事件的传感器或技术的组合，如作为事件类型(即，触摸、笔、手势、视网膜运动等)的组合。识别或捕捉输入事件的技术共有的一个属性是它们有助于用户动作与系统对动作的响应之间的等待时间。此贡献的衡量随着技术和实施而变化。

[0050] 在典型的多触摸系统中，在输入设备与可包括通信的显示器、操作系统、UI工具箱、应用层和/或最终的音频或图形控制器之间存在信息流的路径，这些中的每一个可增加等待时间。此外，由操作系统，尤其是非实时操作系统所引入的等待时间是可变的。Windows、iOS、OSX、Android等不是实时操作系统，因此使用这些操作系统，不保证响应将在某一时期内发生。例如，如果大量地加载处理器，那么等待时间可急剧增加。此外，一些操作在软件栈中非常低的级别处被处置且具有高优先级。例如，通常高度优化鼠标指针从而即使当处理器处于大量负载下时，感知的负载也相对较低。相比之下，诸如在触摸或手势系统上用两个手指调制照片大小之类的操作通常是更加计算密集的，因为它需要在应用和/或UI工具箱等级处持续改变图像尺寸。因此，当处理器处于大量负载下时，这样的操作几乎不能够具有低感知的等待时间。

[0051] 在典型的多触摸系统中，显示器系统(包括图像系统以及显示器本身)也可贡献等待时间。具有高帧速率的系统可使经过该系统的实际等待时间不明显。例如，60Hz监视器可包括一帧或多帧的缓冲器以允许复杂的图像处理效果。类似地，诸如投影仪之类的一些显示器设备在电子设备中包含双缓冲，实际上加倍显示器等待时间。对3D电视和降低的运动伪像的期望推动较快LCD的发展，然而，液晶本身的物理性质不大可能使传统LCD的性能超过480Hz。在实施例中，本文中所述的低等待时间系统可使用LCD显示器。与LCD显示器的性能相反，OLED或AMOLED显示器能够良好地有1ms以下的响应时间。因此，在实施例中，本文中所述的高性能触摸(或手势)系统可被实现在具有快速响应时间的显示器上，包括而不限于，基于以下技术中的一个或多个的显示器：OLED、AMOLED、等离子体、电湿润、彩色场序(color-field-sequential)LCD、光学补偿弯曲模式(OCB或Pi-Cell)LCD、电子墨水等。

[0052] 等待时间感知研究

[0053] 着手一些研究以确定用户将直接触摸界面中什么程度的等待时间感知为基本上同时发生的。图4中的框图所表示的原型设备示出原型高性能触摸系统400的基本构架的说明性实施例。在实施例中，高速输入设备420是多触摸阻式触摸传感器，其具有24cm x 16cm的有效区域和允许非常高速操作的电子设备。经过此传感器的延迟微微小于1ms。在实施例中，可在光链路上连续地发送触摸数据。

[0054] 在说明性测试系统中，显示器460是基于德州仪器的数字光处理技术的DLP显像4100套件(DLP Discovery 4100kit)。说明性测试系统在触摸传感器上利用前投影，因此消除可扰乱手指与图像对准的用户感知的视差。所采用DLP投影仪使用数字微镜设备(DMD)，即以非常高的速度有效地接通和断开像素的镜子矩阵。镜子的高速可被用于改变接通相对于断开的时间百分比以创建连续的彩色图像的表现。在实施例中，其中仅使用了简单的二值图像，可以以更高的速率来产生这些图像。在说明性测试系统中，投影仪开发系统用低于40性测的等待时间以1024x768分辨率显示32,000个二值帧/秒。在为了获得此速度的说明性测试系统中，视频数据以25.6Gbps流向DMD。

[0055] 在说明性测试系统中，为了获得最小的等待时间，所有触摸处理被执行在专用的FPGA 440上，在触摸输入与低等待时间输出的显示器之间没有采用PC或操作系统。DLP套件的板载XC5VLX50应用FPGA可被用于处理触摸数据并渲染视频输出。与FPGA串联的USB允许参数被动态地改变。在说明性测试系统中，可用1ms分辨率将等待时间从1ms调节到数百ms。可获得不同的测试模式，且端口允许收集触摸数据以供分析。

[0056] 在说明性测试系统中，为了从传感器420接收触摸数据，系统通过定制高速UART通信。为了最小化等待时间，可使用2Mbps的波特率，这表示可使用的高波特率而不会由于通信信道上的高频噪声损失信号完整性。在说明性测试系统中，然后由在FPGA 440上实现的触摸检测有限状态机来处理压缩的触摸数据的各个字节。有限状态机(FSM)同时地对数据解码并执行质心团块检测(blob-detection)算法以识别触摸的坐标。在说明性测试系统中，系统是流水线式的使得FSM的每次迭代运行在最终接收的字节上从而不发生触摸数据的缓冲。

[0057] 在说明性测试系统中，触摸坐标然后被发送到10级可变延迟块。每个延迟阶段是具有计数器的简单FSM并采用指示时钟周期的数量的控制信号以使触摸坐标延迟，允许不同等级的等待时间。延迟块在迭代的开始处闭锁触摸样本，且在发送样本并锁闭下一个样本之前等待适当数量的周期。延迟块因此通过延迟计数的系数降低了样本速率。在实施例中，为了将样本速率保持在合理的等级处，可使用10个延迟阶段，从而使得，例如，为了获得100ms的等待时间，对于100Hz的样本速率，块在样本之间等待10ms。在说明性测试系统中，为了运行基本的应用，MicroBlaze软处理器被用于渲染显示器。

[0058] 在实施例中，测试系统可使用硬编码控制FSM来代替MicroBlaze以便改进性能。在实施例中，可使用另一个软处理器。在说明性测试系统中，MicroBlaze是32位哈佛构架RISC处理器，其被优化为在Xilinx FPGA中合成。MicroBlaze软处理器实例化允许仅所需的核、外围设备和存储器结构的选择。在说明性测试系统中，除了基础MicroBlaze配置之外，可使用中断控制器，例如用于触摸数据的多个GPIO、设置可变等待时间的GPIO、用于图像缓冲的BRAM存储器控制器、以及与PC通信的UART单元。在说明性测试系统中，以100MHz来时钟控制MicroBlaze。MicroBlaze使用中断系统来检测有效的触摸坐标。当触摸数据从延迟块到达

GPIO时，产生触摸准备中断事件，并且对应的图像被写入图像缓冲器。由于基于中断的系统的非一致性质，不可计算准确的等待时间，但通过设计，与归因于输入设备的1ms等待时间相比，这是无关紧要的。

[0059] 在说明性测试系统中，图像缓冲器被合成在芯片上的BRAM块中。这些块可提供双端口高速可配置的存储器缓冲器，其具有足够的带宽以支持高帧速率显示。在说明性测试系统中，对于25.6Gbps的总带宽，用128位的总线宽度以200MHz来时钟控制图像缓冲器，如DLP所需。最后，DMD控制器从图像缓冲器连续地读出帧并产生具有适当定时的信号以控制DMD。

[0060] 在说明性测试系统中，用户输入被同时地发送到传统的PC并被处理以产生传统的较高等待时间响应。此较高等待时间响应被传统的数据投影仪输出，且被对齐以与投射的较低等待时间响应重叠。

[0061] 进行了一些研究以确定当在触摸屏界面上执行普通任务时用户能够感知的表现的精确等级。为此目的，进行了一些研究以确定不同表现等级的刚好可注意到的差异(JND)。JND是可由观察者检测到的刺激的两个等级之间差异的度量。在此情况下，JND被定义为参与者能够在两个不等的刺激之间做出区分的阈值等级，一个刺激始终表现在同一等级，称为参考，而一个刺激的值在整个实验期间动态变化，称为探查(probe)。针对某一任意参考值处的JND的一般接收值是参与者可在75％的时间正确地识别参考的探查。不能与参考区分的探查值且具有此准确度等级的探查值被认为是与参考没有显著差异的。

[0062] 进行了一些研究以当与用作参考的1ms的等待时间的最大表现相比时确定探查等待时间的JND等级。尽管这种确定不提供最大可感知表现的绝对值，但它可用作我们的“最好情况”下限条件，假定它是我们的原型可获得的最快速度，相对于该条件可测量等待时间的其他等级。发现参与者能够分辨显著较低(<20ms)的等待时间值，典型的电流产生硬件(例如，电流平板和触摸计算机)提供显著较低的等待时间值(～50-200ms)。

[0063] 从本地社区招募十位用右手的参与者(3位女性)。年龄范围在24和40之间(平均27.80，标准差4.73)。所有的参与者有过触摸屏设备的在先经验，且所有的参与者拥有一个或多个触摸设备(诸如基于iOS或Android的电话或平板)。参与者被重复地呈现多对等待时间条件：参考值(1ms)和探查(在1和65ms的等待时间之间)。参与者在触摸屏显示器上从左向右拖动他们的手指，然后从右向左。尽管任何拖动任务是适当的，但左/右运动降低高等待时间情况中的阻塞。参与者被请求在两个方向上移动以确保他们不“匆忙做完”此研究。在用户的触点之下，系统渲染了实线白色2cm×2cm正方形，如图1所示。留给参与者来决定运动的速度。对于每对随机化条件的顺序。此研究被设计成双项必选实验；指示参与者在每次试验内选择哪个情况是参考(1ms)值且不允许做出“不知道”或“不确定”选择。在每对之后，参与者告知实验者两个中的哪一个“更快”。

[0064] 为了使每次试验向75％的期望JND等级收敛，根据自适应阶梯算法来控制增加的等待时间的量。对参考值的每一次正确识别导致探查中的等待时间的量减小，而每次不正确的响应导致探查的等待时间增加。为了达到75％置信度等级，增加与减小遵循由Kaernbach所描述的简单加权的自上而下(up-down)方法(Kaernbach，C.1991.Perception& Psychophysics(知觉和心理物理学)49，227-229)，其中，增加具有施加到基础步长的三倍乘法器，而减小是基础步长(最初8ms)。

[0065] 当参与者在正确响应之后不正确地响应时，或者在不正确响应之后正确地响应时，这被称为反转，因为它导致阶梯(增加或减小)的方向反转。最初8ms的步长在每次反转时二等分，直到1ms的最小步长。这继续直到总共10个反转发生为止，结果是收敛到75％正确性。每个参与者完成八个阶梯“运行”。这些中的四个起始于最小探查等待时间(1ms)而另外四个于最大(65ms)。选择阶梯的较高起始值，因为它与商业提供的粗略地相符，且因为引导(pilot)测试使得该值与1ms参考以接近100％准确度来清楚地区分，避免天花板效应。可在交织的对中一次运行两个阶梯以防止响应偏差，该响应偏差是由参与者在连续刺激之间跟踪他们的进展的能力导致的。为这些对中的每一对随机地选择阶梯条件而没有来自可能性的替代(2起始等级×4重复)。每个参与者在单个1小时时间内完成整个实验，包括阶梯之间的休息。

[0066] 此研究被设计成针对大于1ms的等待时间值找到刚好可注意到的差异(JND)等级。此JND等级一般被约定为参与者能够在75％的时间正确识别参考的等级。参与者JND等级范围从2.38ms到11.36ms，对于所有参与者有6.04ms的平均JND(标准差4.33ms)。对于每个参与者，JND等级在阶梯的8次运行上没有显著地变化。图5中呈现每个参与者的结果。

[0067] 此结果示出参与者能够分辨远远低于消费者设备的典型阈值(50-200ms)的等待时间的差异。注意到，参与者很可能经常通过当手指在触摸屏上移动时估计屏幕上物体与他们的手指之间的距离来确定等待时间；这是在UI中使用的输入基元的伪像(具体地，拖动)。测试不同的输入基元(轻触(tapping)，例如)将呈现等待时间的不同感知。结果证实触摸设备的用户将注意到并感激等待时间的数量级改进。

[0068] 用于低等待时间直接触摸输入设备的构架

[0069] 在实施例中，给定低等待时间系统的存在，软件界面可被设计成使应用开发者能够继续使用基于工具箱的应用设计过程，而且使那些工具箱能够以极低的等待时间提供反馈。在实施例中，本公开所概括的系统和方法可被实现在UI开发的模型视图控制器(“MVC模)模型上，很多UI工具箱是基于该模型的。MVC允许应用逻辑与应用的视觉表示分开。在实施例中，MVC可包括该应用的第二个重叠的实际视图。具体地，在实施例中，触摸输入从UI控件接收立即响应，该立即响应部分地基于做出触摸的时刻处该应用的状态。目标是提供与底层应用前后联系的几乎立即的响应。

[0070] 在对触摸的应用无关的视觉响应上的先前工作是甚至与UI的视觉元件完全分开的，增加视觉复杂性。在实施例中，根据本文中所概括的系统和方法，一组视觉响应更加完整地集成到UI元件本身中以减少视觉复杂性。因此，在实施例中，其中示出的特定视件(visual)为触摸提供实际“鼠标指针”，目标是将高性能响应集成到控件本身中，提供更加统一的可视化。依然在实施例中，系统和方法通过低等待时间子系统允许前后无关的响应的渲染，该前后无关的响应随后与来自高等待时间子系统的响应融合。在实施例中，不需要在与剩余的系统的响应相同的渲染管道中表示视件。相反，除了由传统系统所产生的较高等待时间响应之外，利用如本文中所讨论的混合反馈的系统或方法可对用户输入表现较低的等待时间响应。

[0071] 因此，在实施例中，设计加速的输入交互从而使得传统的直接触摸软件如同它一般具有高等待时间响应那样运行，同时以较低的等待时间提供针对UI元件定制的一组附加的反馈；具有用户感知不到的等待时间的目标。在实施例中，通过叠加两个或多个图像来组

合这两层。在实施例中，两个组合的图像可包括来自低等待时间触摸设备的一个投射图像，和来自与桌面计算机连接的传统投影仪的第二图像，该桌面计算机运行定制的触摸软件并从低等待时间子系统接收输入。

[0072] 以上所述的两个投影仪解决方案仅仅表示用作将低等待时间响应与传统响应组合的更通用思想的一个特定实施例。在实施例中，来自低等待时间与高等待时间子系统的视觉输出在被发送到显示器之前在系统中的显示器缓冲器或其他地方中逻辑地组合。在实施例中，透明的重叠的显示向用户呈现低等待时间与高等待时间输出。在实施例中，使显示器的像素交错(interlace)从而使得一些像素由低等待时间子系统来控制，一些像素由高等待时间子系统来控制；通过交错，这些显示可对用户显得重叠。在实施例中，使显示器上呈现的帧交错从而使得一些帧由低等待时间子系统来控制，一些帧由高等待时间子系统来控制；由于帧交错，显示可对用户表现为含有组合的图像。在实施例中，可在硬件中主要地或完全地产生低等待时间响应。在实施例中，可从输入传感器直接接收的输入传感器数据来产生低等待时间响应。在实施例中，由至显示器硬件的高带宽链路来显示低等待时间响应。

[0073] 在为低等待时间子系统设计用户界面中，可考虑以下约束中的一个或多个：

[0074] • 信息：高等待时间子系统用来形成系统对输入的响应所需要的任何信息或处理将必然具有高等待时间，除非例如预先渲染或预先供给了这种信息或处理。

[0075] • 性能：以低等待时间形成响应所允许的时间必然受限。即使利用硬件加速，必须谨慎地性能驱动响应的设计以保证响应满足期望的低等待时间。

[0076] • 保真度：所渲染的低等待时间图像的保真度可与较高等待时间渲染(实际上，它被高等待时间系统预渲染)难以区分；附加的约束可被设置在保真度上以改善性能，诸如例如，视件仅仅是单色的，和/或对视觉基元进行限制，和/或限制音频或触觉响应的持续时间或特性。可由系统的不同元件来引入此类型的约束，包括加速硬件或由输出硬件(诸如显示器、触摸输出设备、或扬声器)。

[0077] • 无干扰：在响应是混杂组合的实施例中，可在低等待时间层中产生应用的响应中的一些，在高等待时间层中产生一些，考量可以是怎样混合这两者，例如，为了向用户的输入提供无缝响应。在实施例中，低等待时间响应不干扰任何可能的应用响应，这必然将在随后发生。在实施例中，在低等待时间响应与传统响应之间可发生干扰，但此干扰可通过设计或通过响应的混合来处置。

[0078] 在实施例中，进行设计过程以创建一组视觉UI控件，具有对触摸有区别的低等待时间与高等待时间视觉响应。寻求将实现两层响应之间的无缝转变的隐喻。这些可视化包括诸如对象位置和状态之类的信息。使用上述约束基于可行性来精选设计。类似于在军用飞行器中使用的可视化，这种实施例的最终设计基于抬头显示器(HUD)隐喻。HUD是合适的，因为传统的HUD在几何学上是简单的，并且它相对容易地以真实保真度实现几何上简单的显示器。HUD表示组合的两个视觉层的仅一个示例，然而在很多HUD中，计算机化的显示被叠加在视频或“真实世界”本身上。因此，HUD通常被设计成无干扰的。

[0079] 基于HUD隐喻，为在很多直接触摸系统中发现的一组UI元件开发了一组示例性的触摸事件和UI元件特定的低等待时间层可视化。

[0080] 这些示例性元件是既普通又有代表性的；他们的交互(轻触、拖动、两手指收聚

(pinching))覆盖在电流直接触摸设备中使用的大多数交互空间。在这样的实施例中开发的低等待时间响应在表1中描述,并且它们在图6-8中示出。

[0081]

元件	触下	触动	抬起
按钮（图 6）	有轮廓的边界 610	（无）	如果在边界内，则第 2 轮廓 620，否则无
可拖动/可变尺寸件（图 7）	有轮廓的边界 710	轮廓随着输入位置 720 改变并运动和/或随着输入手势 730 按比例缩放	当高等待时间层追上时轮廓 710 减弱
滚动列表（图 8）	有轮廓的列表项 810	如果滚动手势，则关于滚动距离的列表边缘高亮 830。如果在滚动手势期间，则当高等待时间层追上时边缘高亮 840 减弱	如果列表项选择，轮廓 820 缩小且减弱

[0082] 表1:针对每个元件和触摸事件的加速视件,补偿对触摸输入的标准高等待时间响应。

[0083] 这三个元件表示针对触摸输入的标准UI工具箱的广泛使用范围。最高阶UI元件由这些较简单的元件组成(例如单选按钮(radio button)和复选框(checkbox)都是“按钮”,滚动条是具有受限平移和旋转的“滚动/可变尺寸件”)。本文中所述的加速的输入系统和方法依赖于以两个明显不同的等待时间等级工作的视件的结合;此等待时间差异已被纳入低等待时间可视化的设计中。在实施例中,可通知用户两个系统的状态,当各视觉层取得对齐时具有相关的同步性。在实施例中,用户能够在系统反馈的高与低等待时间部分之间做出区分。在实施例中,以在低等待时间响应与传统响应之间提供不明显差别的方式来混合视觉元件。

[0084] 在实施例中,应用开发者利用工具箱经由组建GUI控件的一般过程来建立他们的应用。在执行后,UI元件将它们的可视化分为在单个显示器上渲染且重叠的高与低等待时间可视化两部分。如在图9中示出通过这种系统的信息流的实施例。信息从输入设备910流到系统中且由输入处理单元(IPU)920首先处理,经由IPU软件工具箱930来编程。然后由两个子系统(低等待时间低保真度子系统940和高等待时间子系统950(诸如例如在常规软件栈中运行的常规软件))平行地处理UI事件。在实施例中,可在硬件(诸如图4的FPGA 440)中实现低等待时间低保真度子系统940。

[0085] 在此实施例中描述的分为两部分创建了基本的通信问题,其中必须在用户开始给予输入之前限定由应用逻辑所需的低等待时间子系统940所提供的初始响应的参数化。需要在由应用进行表示的时刻处理的任何响应将引入低等待时间系统940对高等待时间系统950的依赖性,因此可将滞后引入回到系统中。在实施例中,低等待时间系统940对输入的响应的随后阶段可依赖于高等待时间子系统950。在实施例中,管理低等待时间系统940对输入的响应的随后阶段对于高等待时间子系统950的依赖性从而使得该依赖性不引入附加的

等待时间。在实施例中，将完全避免该依赖性。

[0086] 在实施例中，UI元件逻辑可被建立到低等待时间子系统中。在用户输入之间，在高等待时间子系统950中执行的应用有机会为UI元件的低等待时间子系统940的模型提供参数。因此，在实施例中，可通过提供为低等待时间反馈负责的单独控制器来扩展UI软件设计的MVC模型。在实施例中，在软件设计中，可针对每种控件指定以下中的一个或多个：

[0087] • 元件类型(例如，按钮、可拖动对象、滚动列表等)。

[0088] • 边界尺寸(例如，x位置、y位置、宽度、高度等)。

[0089] • 有条件的：附加的基元信息(例如，滚动列表的情况中的列表项的尺寸等)。

[0090] 在实施例中，给定元件类型对触摸输入的响应的逻辑被储存在低等待时间子系统940中。可用同样的方式来传递低等待时间子系统对用户输入的响应的进一步参数化，允许更大程度的定制。在实施例中，处理传感器数据以产生事件(或输入流的其他处理形式)，该事件然后被单独分配到低等待时间子系统940和高等待时间子系统950。可对低等待时间子系统940和高等待时间子系统950以不同速率产生事件，因为低等待时间子系统能够比高等待时间子系统更快地处理事件，且以高速率向高等待时间子系统发送事件可压垮该子系统。低与高等待时间子系统对用户输入的响应因此是独立的且并列的。在实施例中，一个子系统担当主控器摂，在用户输入之间设置另一个子系统的状态。在实施例中，低等待时间与高等待时间子系统之间的关系包括两个子系统之间的同步性。在实施例中，低等待时间与高等待时间子系统之间的关系包括高等待时间子系统为低等待时间子系统940卸载处理的能力。在实施例中，低等待时间与高等待时间子系统之间的关系包括低等待时间子系统940减少其处理负载和/或利用高等待时间子系统950来预处理或预渲染的能力。在实施例中，第二图形处理和输出系统的响应依赖于第一图形处理和输出系统，并且状态信息从第一图形处理和输出系统传递到第二图形处理和输出系统。在这种实施例中，从第一图形处理和输出系统传递到第二图形处理和输出系统的信息由一片或多片数据组成，该一片或多片数据描述了用户界面中的一个或多个图形元件。例如，此数据可以是用户界面中的图形元件的尺寸、位置、外观、可选的外观、对用户输入的响应、以及类型。从第一图形处理和输出系统传递到第二图形处理和输出系统的数据可被储存在第二图形处理和输出系统可用的高速存储器中。传递的数据可描述按钮、滑动件、可拖动和/或可变尺寸GUI元件、滚动列表、旋转件、下拉列表、菜单、工具条、组合框、可移动图标、固定图标、树状图、网格视图、滚动条、滚动窗或用户界面元件的外观和/或行为。

[0091] 在实施例中，在用户输入信号被第一或第二图形处理和输出系统中的一个或两者接收之前，输入处理系统在用户输入信号上执行抽取(decimation)。基于从第一图形处理和输出系统发送的关于用户界面的信息从全部输入信号组中选择抽取的输入信号或未抽取的信号。可通过将输入信号组在逻辑上组合成较小的输入信号组来执行输入信号的抽取。可通过窗口的求平均值来执行输入信号的逻辑组合。当减小输入信号组的尺寸时，该抽取考虑用户输入信号的时间。可通过加权求平均值来执行输入信号的逻辑组合。在实施例中，已经有区别地处理了由第一和第二图形处理和输出系统接收的用户输入信号。

[0092] 在实施例中，高等待时间和低等待时间层之间的通信可以是重要的。以下说明在确定高与低等待时间子系统怎样保持同步化中要考虑的一些重点：

[0093] • 等待时间差异：低等待时间响应可使用关于高与低等待时间层之间的等待时间

差异的信息以使响应同步。在实施例中，这些等待时间值是静态的，且因此被预编程到FPGA中。在等待时间等级可在任一子系统中变化的实施例中，可有利地将等待时间等级固定在始终可达到的常数，而不是具有可变得不同步的动态值，或者有利地提供明确的同步化机制。在等待时间等级可在任一子系统中变化的实施例中，可使用动态值，然而应该当心避免变得不同步。在等待等级可在任一子系统中变化的实施例中，可在子系统940、950之间提供明确的同步化机制。

[0094] • 点击测试：点击测试决定通常以关于可见UI元件的视觉层次和属性的数据为条件。在实施例中，通过不允许边界矩形重叠、要求UI的平面的“点击测试友好的”映射。在实施例中，单独的点击测试可为低等待时间子系统提供必要的信息(物体状态、z顺序、和收听站)。在实施例中，低等待时间与高等待时间子系统可并列地进行点击测试。在实施例中，低等待时间子系统进行点击测试，并且向高等待时间子系统提供结果。

[0095] • 条件响应：很多界面可视化不仅以立即用户输入为条件，而且以应用逻辑中所限定的进一步做出决定逻辑为条件。

[0096] 条件响应逻辑的两个说明性示例如下：考虑信用卡购买提交按钮，其被编程为当按压时禁用(为了防止双重记账)，而且仅基于输入表单的数据的有效性。在这种情况中，按钮的行为不仅依赖于立即用户交互，而且进一步以附加的信息和处理为条件。还考虑关联的可视化，诸如图10中示出的一个。在此情况中，不仅由用户操控的UI元件1010，而且还由第二UI元件1020来为用户提供反馈。这些示例可被直接编程到低等待时间子系统中。

[0097] 在实施例中，高与低等待时间子系统之间的划分可与任意用户界面元素无关。实际上，子系统之间责任的划分可基于任意数量的因素来定制，且仍有可能存在于缺乏用户界面工具箱的系统中，或者包括在使用和不使用可能获得的UI工具箱两者来开发应用的机制的系统中。在实施例中，可在子系统运行时动态地改变两个子系统之间的责任的划分。在实施例中，UI工具箱自身可被包括在低等待时间子系统内。可用数个方式来向应用开发者提供定制响应的能力而不脱离本文所述的系统和方法。在实施例中，响应可被定制为要在UI控件中调节的参数。在实施例中，可在低等待子系统中本身执行的代码中或者在其他高或低等待时间组件中，通过允许向低等待时间子系统直接提供指令的能力来定制响应。在实施例中，可使用由应用代码(例如，在运行时间)所产生的数据来设置低等待时间子系统的状态。

[0098] 尽管以上所述的示例是在触摸输入的情况下提供的，但预期其他的实施例，包括但不限于，笔输入、鼠标输入、间接触摸输入(例如，触摸板)、空中手势输入、口头输入和/或其他输入模态。所述的架构同等地可应用到任何种类的用户输入事件，包括但不限于混合输入事件(即支持来自超过一个模态的输入)。在实施例中，混合输入设备可导致相同数量的用于低和高等待时间子系统中的每一个来处理的被产生的事件。在实施例中，将以所产生的事件的数量来区别混合输入设备，如此例如，触摸输入可比笔输入具有较少的事件。在实施例中，每个输入模态包含其自己的低等待时间子系统。在实施例中，在包含针对多个输入模态的多个低等待时间子系统的系统中，各子系统可通信以配合他们的响应。在实施例中，在包含针对多个输入模态的多个低等待时间子系统的系统中，多个子系统可共享公共存储器区域以实现协作。

[0099] 输入处理

[0100] 在本发明的实施例中，来自输入硬件的低等待时间输入数据被最小程度地处理为快速的输入事件流。此事件流被直接发送到低等待时间子系统以便进一步处理。然后在事件流被发送到高等待时间子系统之前，可删除来自这同一流的事件，或者可缩减或过滤该流。因为低等待时间子系统能够比高等待时间子系统更快地处理事件且以高速率向高等待时间子系统发送事件，这可能压垮该子系统，所以对于低等待时间子系统940和高等待时间子系统950可以以不同速率产生事件。低等待时间与高等待时间子系统对用户输入的响应因此可以是独立的且协调的。

[0101] 可优化事件的缩减。在实施例中，可基于与应用、UI元件、输入设备等中的一个或多个相关联的标准在候选事件之中选择代表性事件。对于当用户绘制数字墨水笔划的笔输入的这个示例可包括选择最适于用户所绘笔划的事件。对于语言输入的另一个示例是偏向于其中输出流中的随后事件将具有类似音量的事件，由此“均匀化”来自麦克风的声音。对于触摸输入的另一个示例是偏向于将产生具有一致速度的输出事件流的事件，从而提供更加“光滑的”输出。这种形式的智能缩减担当智能过滤器，而不会降低高等待时间子系统的性能。在实施例中，可产生新的事件(例如，合并的事件或虚事件)，代表输入流中其他事件的聚集。在实施例中，可产生新的事件(例如，校正的事件、合并的事件或虚事件)，代表更加期望的输入流，例如校正或平滑化。

[0102] 例如，对于空中手势输入，对于来自高速输入设备的每10次事件，可对高等待时间子系统发送相同数量或更少的事件，这些事件提供实际输入事件的“平均”，因此平滑化输入并去除抖动。

[0103] 还可产生新的事件，作为输入设备的不同参数的多个“期望”等级的混合(amalgam)。例如，如果笔的倾斜和压力属性的智能缩减将导致不同事件的选择，则可创建单个新的事件对象(或者一个或多个现有事件对象被修改)以包括这些属性中的每一个的期望值。

[0104] 在实施例中，IPU或低等待时间子系统系统可被用于向高等待时间系统提供经处理的输入信息。一个或多个方法可被用于协调两个子系统的活动。这些方法包括：

[0105] a.在实施例中，低等待时间子系统可立即响应于所有用户输入，但在向高等待时间系统提供输入之前等待用户停止输入(例如抬起手指或笔，终止手势)。这在用户交互期间具有避免阻塞系统同时仍处理全体数据的优点。

[0106] b.在实施例中，低等待时间系统可几乎实时提供输入的缩减估计；并且可任选地储存完整的输入队列，高等待时间系统可在请求时获得该完整的输入队列。

[0107] c.在实施例中，用户反馈可被分成两个步骤。第一，低等待时间反馈将提供图11中用户输入1130的粗略的立即的表示。第二，只要高等待时间系统能够计算细化的响应，例如在笔1150尖提起之后，高等待时间系统响应1140就可替代第一响应1130。可选地，高等待时间反馈可连续地“追赶”(且有可能包含)低等待时间反馈。

[0108] d.在实施例中，低等待时间系统可从输入流中推断简单的手势动作，且因此产生不同于或者代替原始事件的手势事件，该手势事件被包括在输入队列中。

[0109] e.在实施例中，IPU或低等待时间子系统可使用多个输入位置来预测进一步输入位置。此预测可被传递到高等待时间子系统以减小其有效等待时间。

[0110] f.在实施例中，在IPU或低等待时间子系统中执行可受益于附加样本或早期检测

的算法。在实施例中，可在时间上限制这些事件的执行。例如，可使用最初的50个事件以将输入分类为特定手指，或在手指与笔输入之间做出区分。在实施例中，这些算法可连续运行。

[0111] g.在实施例中，将事件流传递到高等待时间子系统的低等待时间子系统的过程可被延迟以接收并处理附加的连续或同时的相关输入，不然此相关输入可能被不正确地视为不相关输入。例如，字母“t如通常被绘制为两个单独的，但相关的笔划。在一般进程中，从低等待时间系统传递到高等待时间系统的输入流的一部分将包括绘制第一条线的端处的“抬笔”信号。在实施例中，缩减过程等待样本窗口内的输入的最后帧以传递到“抬起”事件，以免在窗口内的显示器上再次检测到笔，因此排除对事件的需要。

[0112] 硬件架构

[0113] 在实施例中，数据流过经过系统的组件的两个重叠路径以同时支持高等待时间与低等待时间反馈。图12示出一个这样的系统，包括输入设备1210、IPU 1220、系统总线1230、CPU 1240和连接到显示器1290的GPU 1280。用户1200使用输入设备1210执行输入。此输入由IPU 1220来感测，在不同的实施例中，IPU 1220可以是FPGA、ASIC或者集成到GPU 1280、MPU或SoC的附加软件和硬件逻辑。此时，控制流分成且跟随经过系统的两个单独的路径。对于输入的低等待时间响应，IPU 1220将输入事件经由系统总线1230发送到GPU 1280，绕过CPU 1240。GPU 1280然后快速地显示对用户1200的反馈。对于输入的高等待时间响应，IPU 1220将输入事件经由系统总线1230发送到CPU 1240，CPU 1230运行图形应用且可与其他系统组件交互。CPU 1240然后将指令经由系统总线1230发送到GPU 1280以向用户1200提供图形反馈。从输入设备1210到IPU 1220到系统总线1230到GPU 1280的低等待时间路径主要是硬件且用低等待时间来运行。从输入设备1210到IPU 1220到系统总线1230到CPU 1240再回到系统总线1230到GPU 1280的高等待时间路径是高等待时间的，由于在此说明书早先所述的因素。在相关实施例中，输入设备1210与GPU 1280之间通信且绕过系统总线1230。

[0114] 图13示出惯用的编程范例，称为模型视图控制器。在此范例中，用户1300在控制器1310上执行输入，控制器1310进而基于此输入来操控模型1320。模型1320中的变化导致视图1330变化，这被用户1300观察到。本发明所解决的等待时间中的一些是由于输入、这些组件之间的通信、以及由视图1300组件所产生的图形的显示中的等待时间。

[0115] 图14示出支持在系统上开发并运行应用的架构的实施例，此系统对用户输入有混合的高等待时间与低等待时间响应。用户1400用输入设备1410执行输入。此输入由IPU 1420接收。IPU 1420同时将输入事件经由传统机制发送到高等待时间子系统中运行的控制器1430并发送到低等待时间中运行的视图模型(L)1490。输入由控制器1430来处置，该控制器1430操控高等待时间子系统中运行的模型1440，该模型1440可与易失性存储器1450、固定存储器1470、网络资源1460等中的数据交互(所有交互引入滞后)。由视图模型(L)1490接收的输入事件导致视图模型(L)变化，该变化体现在视图(L)1491的变化中，这被用户1400观察到。模型1440的变化导致高等待时间子系统的视图(H)1480变化，这也被用户1400观察到。在实施例中，在同一显示器上示出用户所观察到这两种类型的变化。在实施例中，这两种类型的变化经由其他输出模态(诸如例如，声音或振动)体现到用户。在实施例中，在输入之间，模型1440更新视图模型(L)1490和视图(L)1491的状态，从而使得视图模型(L)1490包含需要的数据以在系统的显示器上的正确位置中呈现GUI的组件且使得视图模型(L)1490

可以以模型1440的当前状态的情况来正确地解释来自IPU 1420的输入;且使得视图(L)1491可以以模型1440的当前状态的情况来正确地产生用于显示的图形。

[0116] 借助示例,考虑具有按钮的触摸敏感的应用,在按钮的功能之中,通过改变其外观对用户的触摸进行响应,指示它已被激活。当应用运行时,应用从存储器和编译的应用代码中读取按钮的位置、尺寸和外观的细节。视图(H)1480代码产生必要的图形,该必要的图形被呈现给用户以显示按钮。模型1440更新视图模型(L)1490的状态以记录此图形元件是按钮,以及当它被触摸时应该将外观从“正常”外观改变到“按压”外观。模型1440还更新视图(L)1491的状态以记录针对视图模型(L)1490中的“正常”和“按压”状态的正确外观。此外观可以是低保真度图形元件的描述或完整的栅格(raster)以用于显示。在此示例中,通过在按钮的位置周围显示白框来表示“按压”状态。

[0117] 用户触摸触摸屏显示器,且描述该触摸的输入数据在小于1ms之后由IPU 1420接收。IPU 1420从输入数据创建代表触下事件的输入事件并将此输入发送到应用控制器1430。控制器1430操控模型1440。在此情况下,控制器1430对模型1440指示按钮已被触摸且应用应该执行与此按钮有关联的任何命令。在IPU 1420向控制器1430发送事件的同一时刻,它向视图模型(L)1490发送事件,指示按钮已被触摸。先前模型1440对视图模型(L)1490下在触摸的情况下做什么的指令,且在此情况中它通过将其状态改变为“按压”来响应触摸事件。视图(L)1491通过在按钮周围显示白框(对应于其“按压”外观的反馈)来响应此变化。触摸按钮给模型1440带来的改变引起视图(H)1480的更新,从而它也反映了现在按钮被触摸。观察到视图(H)1480和视图(L)1491两者的输出的用户,通过视图(L)1491观察到他们的触摸的立即反馈,在一秒内跟随着来自视图(H)1480的反馈。

[0118] 贯穿此申请的文本,词语“事件”被用于描述说明用户输入的属性的信息。一般使用此项,且此项包括采用事件驱动架构的实施例(实际事件对象在软件元件之间传递),以及正在描述的“事件”的更加基本的输入流仅存在于信息流中。例如,这种事件可以是非面向对象类型的事件或面向对象型事件。

[0119] 通过预生成应用元件的替代的图形表示来对输入做出低等待时间视觉响应以及在图形处理单元上的输入处置

[0120] 背景

[0121] 图15是示出根据现有技术的图形用户界面(GUI)视图和中间数据的分级结构的框图。在CPU上运行的应用包括多个GUI元件,通常但不是必须地,以树状排布。这些“视图”(也被称作小部件、组件、元件等)可以包括诸如滑块、按钮、面板等熟悉的元件,其中的每一个拥有当前状态和相关的应用代码,该应用代码在用户输入作用在该元件上时运行。

[0122] 当应用更新其任何状态并且对应用的视觉外观的改变需要向用户显示的时候,应用执行“绘制(paint)”命令(在某些系统中也被称作绘图(draw)、渲染(render)等),该“绘制”命令行走此树(或其他数据结构:例如‘场景图(scene graph)’)并从应用的GUI元件中产生中间绘图数据。此中间数据可以包括用于应用中的每个元件的单独的位图(又名,栅格、像素数据),可以包括绘图指令来产生应用中的每个元件的最终像素(被渲染的)外观,或可以包括允许计算机在显示器上产生像素(或对于显示技术适合的基础图形基元)的任何表示(像素数据、显示列表、绘图指令、向量数据等),所述像素在存储器中表示应用的视觉外观。在图15中示出的示例中,此中间数据包括绘图指令,该绘图指令被执行来产生GUI

元件的最终像素外观。此中间数据驻留在存储器中，并且可以被计算机的CPU或专用图形处理单元(GPU)中的任一或二者访问。

[0123] 为了产生最终渲染的GUI，此中间数据被执行或被复制到将被发送到显示器的像素缓冲区并对于用户可见。见图16。

[0124] 在包括用来渲染的CPU和GPU二者的系统中，处置用户输入并产生/更新中间数据的过程(由CPU执行)通常比执行中间指令以产生最终像素缓冲区的过程(由GPU执行)花费显著更长的时间。

[0125] 图17示出如何使得GUI元件的视觉外观的变化对于用户可见。在此示例中，在CPU上运行的应用接收来自用户的要求“视图G”的视觉外观变化的输入。对于此示例，假设视图G是一按钮并且用户按压该按钮，要求在显示器上该按钮呈现“被按压”。用户输入修改了应用中视图G的状态，其触发了产生G的更新的中间数据的“绘制”命令。为了产生最终渲染的GUI(包括G的新的视觉外观)，此中间数据被执行或被复制到将被发送到显示器的像素缓冲区并对于用户可见。此显示器包括修改的视图G的外观。

[0126] 虽然现代操作系统执行许多步骤来有效地仅仅更新要求更新的中间数据，但接收用户输入、修改应用状态并产生中间数据的过程依然是耗时的并且将等待时间引入对用户输入的视觉响应中。因此，期望创建一种在对GUI的用户显示对输入的视觉响应所要求的时间上做出改进的系统。

[0127] 通过预生成应用元件的替代的图形表示来对输入做出低等待时间视觉响应

[0128] 本文描述了一个发明，其中GUI中的元件用来产生一个或多个中间数据，所述一个或多个中间数据对应于GUI元件的一个或多个可能的视觉状态。这些多个视觉表示与控制逻辑配对，该控制逻辑选择恰当的中间数据以便在渲染最终像素图像来向用户显示时使用。

[0129] 图18示出本发明的实施例，其中应用的GUI中的每个GUI元件根据元件的类型和元件的可能的视觉状态的数目来产生一个或多个中间数据。在此示例中，视图G和视图H是此GUI中仅有的拥有多个可能的视觉外观的视图，并且视图G产生对应于两个替代的视觉外观的两个替代的中间数据，视图H产生用于三个可能的视觉外观的三个中间数据。在优选实施例中，本发明记录对应于当前替代物的索引以便在执行中间数据以产生向用户显示的最终像素缓冲区时使用。

[0130] 图19示出了拥有对于视图G的更新索引的中间数据。在此示例中，假设视图G是一按钮，并且“绘图指令G”给出绘制其未被按压的外观的指令以及“替代的绘图指令G”给出绘制其被按压的外观的指令。在此示例中，当用户按压按钮时，系统更新对于G的索引以便“替代的绘图指令G”被选择。此选择确保：当中间数据之后被执行或复制到将被发送到显示器的像素缓冲区并对于用户可见时，视图G正确地呈现。由于GUI中的元件的绘图指令被预先计算，对用户输入的视觉响应可以非常快速地发生且具有低的等待时间，因为此时不需要执行耗时的“绘制”操作。

[0131] 可能与替代的绘图指令绑定的UI视图状态的其他示例包括：窗口的当前/最大化的状态、任何UI元件的按压-非按压状态、当被其他元件影响时所呈现的UI元件(例如：如果一个视图在另一个视图上方通过并且示出落影(drop-shadow)，在该视图‘下面’的视图上的阴影的外观)、或可能在环境中应用的替代的‘皮肤’(例如：在游戏中可能被‘击中’的未

损坏和损坏版本的UI对象)。实际上,可以影响视图视觉外观的视图的任何属性可以被绑定和预先计算。更进一步,那些值相互影响的属性可以又提供更多的替代的渲染(例如,禁用且未被按压、禁用且被按压等)。拥有大量可能值的属性可以用可能已知的值来预先计算,例如基于给定的替代的外观是否表示可以从当前状态直接转换成的状态、基于过去的用户行为、基于其他用户的行为、或由应用的开发者明确地指示。

[0132] 在这些示例中,利用替代的元件绘制的视图是树中的"叶片"节点。在本发明的某些实施例中,相关的视图可以是非叶片节点,诸如图18和图19中的视图E。在这种情况中,子节点(或实际上,当理解此示例的递归属性时的所有后代)可能有或可能没有与它们母体的替代的绘图指令绑定的替代的绘图指令。例如,如果视图E是包含视图H、I和J的集合的UI面板,则对于E的一个替代的绘图指令可以包括给予其"被禁用"的外观。虽然不是一贯,一般地在GUI中,如果母视图设为禁用,那么子视图也是。因此,用于视图E的给予其'禁用'外观的替代的绘图指令可以包含对于视图H、I和J的类似替代的指令的指针以便给予它们类似的被禁用的外观。此指针(或其他指示器)可以存储在某些中央登记表中,或对读取器已知的任何数目的其他地方。在某些实施例中,视图树的根(通常但不是一贯地,其被包含在内的窗口)可以被预先计算,并且因此在树内的视图的某些子集(或全部)也可以被预先计算。这将帮助前景窗口的快速切换。

[0133] 应该理解的是,维护替代的绘图指令可能在某些点上变得艰巨。这些指令可以被存储用于之后的恢复,例如在应用被编译/准备以便分配时、其被载入到设备上时、程序被首次执行时、视图被首次置入场景中时、或空闲计算周期可用的空闲时刻。

[0134] 同样应该理解的是,替代的绘图指令可以包括(或被包括于其中)动画。已知UI的变化的动画可以帮助用户来理解状态之间的转换。在某些实施例中,全部的替代的指令集可以被预先确定来加速动画。

[0135] 在某些实施例中,特定的视图可以被渲染而不将任何其他视图(例如,母或子)重新绘制。这可以要求系统仅渲染视图的不被其他视图遮挡的部分。将被绘制的元件的部分的限制可以被包含在相关的绘图指令中(和/或替代的绘图指令)。在某些实施例中,取决于不同的遮挡区域,全部的替代的指令可以被包含。

[0136] 在图形处理单元中的输入处置

[0137] 虽然描述的发明显著地减少了对于用户输入的视觉响应中的等待时间,计算机的CPU依然负责接收来自输入设备的用户输入事件、将这些事件分派到正确的应用、执行点击测试来将事件发送到GUI中正确的元件、运行可以执行任何量的代码以及改变GUI元件的视觉外观的回调等。

[0138] 图20示出了概括现有技术中显示对用户输入的视觉响应所采取的步骤的概念图。虽然现有技术中单独的系统与图20中概括的具体步骤有所不同,但它们都遵循基本的模式:在CPU中接收并处置输入、在CPU中更新图形元件的属性、在CPU中产生中间数据、并且然后将最终的渲染交给GPU。为了本公开的目的,应该假设我们的发明应用于所有这些变型。

[0139] 图21示出实施例,其中输入事件不仅发送给CPU,而且发送给GPU,其中它们用来对用户输入执行低等待时间响应。在GPU中,"点击测试"操作首先被执行来判定哪个图形元件需要在显示器上更新。对于拥有多个外观的图形元件(在图21中,我们看到拥有三个替代的外观的元件,每个替代的外观用单独的中间数据集来表示),"临时数据采集器"操作然后判

定将在绘制GUI并且将像素传送到对于用户可见的显示器时使用哪个中间数据集。

[0140] 因为GPU和CPU并列运行，由于CPU工作来“追赶”并执行与GUI元件的视觉外观中的变化无关的用户输入的编程的副作用，GPU中的这些步骤可以非常快速地执行。最终结果是对于用户的低等待时间视觉响应。

[0141] 虽然附图示出CPU和GPU之间某些重复(例如，点击测试在两个地方均被执行)，在某些实施例中，通过在GPU中执行这些操作并且将它们的结果传送回CPU而在不降低性能的情况下消除此重复。例如，输入可以仅传送给GPU，并且点击测试可以仅在GPU中完成，将其结果传送到CPU用于进一步处理。

[0142] 我们已经描述了使用本发明来响应于用户输入，在GUI元件的替代的外观之间快速地切换。图22示出通过直接修改GPU中的其中间数据结构对GUI元件的视觉外观进行快速改变。GUI元件的外观的许多普通改变通过它们的位置(例如，滚动、拖动、摇摄(panning))、尺寸(例如，将元件改变尺寸或缩放)、旋转、扭曲或其他视觉属性的改变来发生。同样的，在此发明的实施例中，GPU接收用户输入，并且执行点击测试来判定用户正在输入哪个GUI元件。接下来，正在被作用的元件的临时数据在GPU中被直接地修改。例如，在垂直滚动的情况中，元件的Y位置可以在此步骤中直接地更新，消除重新产生该元件的整个临时数据的需要。在更新后，临时数据的执行可以继续，并且GUI可以被渲染并在为用户观看的屏幕上显示。

[0143] 在某些实施例中，更新受限于图形变换。在某些实施例中，这些变换可以取决于应用逻辑。在某些实施例中，此逻辑可以仅对CPU可用，因此要求使得交互放缓的‘报到’，或执行曾经在GPU中的操作，但然后之后将其用CPU中的操作的绘图的结果替换。在其他实施例中，机制可以存在来让应用的开发者将应用逻辑置于GPU中。此机制可以包括在UI元件上的属性集合(例如，转换的最大范围、或诸如允许在一个方向但不在其他方向上的转换的条件操作)、从一组预先定义的方法之间的选择、或实际上提供指令，以GPU的自身编程语言或被翻译成‘本地’指令的其他语言。由应用开发者指定的这些指令可以跟随点击测试被执行。事实上，这些将相当于在GPU内执行的事件处置的形式。

[0144] 在某些实施例中，由GPU和CPU处置的输入可以导致冲突。例如，如果GPU不了解其范围，用户可能滚动超过列表的末尾，CPU将在事件处置中将其捕获并防止。然而，因为GPU节点比CPU执行得更快，此防止将在滚动已经发生后到来。在某些实施例中，关于公共UI视图的基本逻辑将作为对GPU的指令被编码，从而在最初就防止许多此类冲突发生。然而，在应用开发者能够写入CPU代码来改变视图的外观和/或行为的实施例中，冲突可能无法避免。在这种实施例中，可以包含机制来将其缓解。这些机制可以包括将‘事件’回调提供给CPU和GPU部分中的一个或两者，以允许开发者指定应该如何处置冲突。这些机制也可以包括关于其被处置的策略(规定的或开发者可选择的)。这些策略可以包括从‘非法的’GPU创建状态到‘适合的’CPU创建状态(或反之)的转换的动画或其他图形效果的调用。

[0145] 冲突的另一示例可以包括输入流的处理。例如，某些交互系统包括用于处理输入的机制来判定手势是否已经发生。在某些实施例中，手势检测机制可以驻留在GPU中，在其他实施例中驻留在CPU中，在其他实施例中同时驻留在两个地方，在其他实施例中驻留在系统内的另一位置。在该实例中的冲突解决可以使用与上文描述的那些类似的机制。如果手势被检测到，该事实被编码为状态信息，并且传播到CPU和GPU表示中的一个或两者。在某些

实施例中，此状态信息可以直接地或通过复制存储器的其他途径传送。在其他实施例中，可以通过传送用于CPU和GPU之一或两者执行的指令来传播。

[0146] 由于点击测试和中间数据的修改是可以在GPU上被极端快速地执行的操作，本发明的结果是对于用户输入的低等待时间视觉响应来修改GUI元件的视觉属性。

[0147] 图23示出替代的实施例，其中GPU响应于运行的动画而不是响应于用户输入来修改临时数据。在此情况中，过程“属性动画”以规律的时间间隔对临时数据执行更新，以便随时间影响GUI元件的外观的视觉变化。在每次更新之后，GPU执行临时数据，并且更新显示器以供用户观看。此实施例将CPU从运行动画中解放出来，并且因此当CPU的资源被操作系统的其他方面消耗时动画不被妨碍。

[0148] 一般地，在CPU/GPU的一“侧”的视图的实际的或表现的状态(即，向用户示出的状态)的任何修改将要求两侧之间的某种程度的协作。在某些实施例中，此协作通过以下方式来发生：在两者之间传送状态信息；可能使用冲突解决机制(诸如上文描述的那些)来判定和设置‘正确的’状态，并且将屏幕上所示的转换为正确的状态(如果需要的话)。在其他实施例中，状态冲突通过将指令从一侧传送到另一侧(或通过某种冲突解决单元)来解决。在其他实施例中，可简单地将状态全部从一侧复制到另一侧。在又一实施例中，应用的多个实例可以被实例化，每个实例拥有不同的状态，拥有用来覆写当前状态的‘经选择的’那些实例之一(例如根据上文描述的策略)。

[0149] 当前系统和方法已在前面参照包含能够接收用户输入并对其做出响应的计算机系统的方法和设备的框图和操作示图进行了描述。应该理解，框图或操作说明中的每一个框，以及框图或操作说明中的方框的组合，可以通过模拟或数字硬件和计算机程序指令来实现。这些计算机程序指令可以被提供到通用计算机、专用计算机、ASIC，或其他可编程数据处理设备的处理器，以便通过计算机或其他可编程数据处理设备的处理器执行的指令实现在框图或操作框所指定的功能/动作。在某些可选实施方式中，在框中所指出的功能/动作可以不按照操作说明中所说明的顺序发生。例如，取决于所涉及的功能/动作，连续示出的两个框实际上可以基本上同时执行，或者这些框有时可以按相反的次序来执行。

[0150] 虽然已经参照优选实施例具体示出和描述了本发明，但本领域内技术人员可在形式上和细节上对其做出多种改变，而不背离本发明的精神和范围。

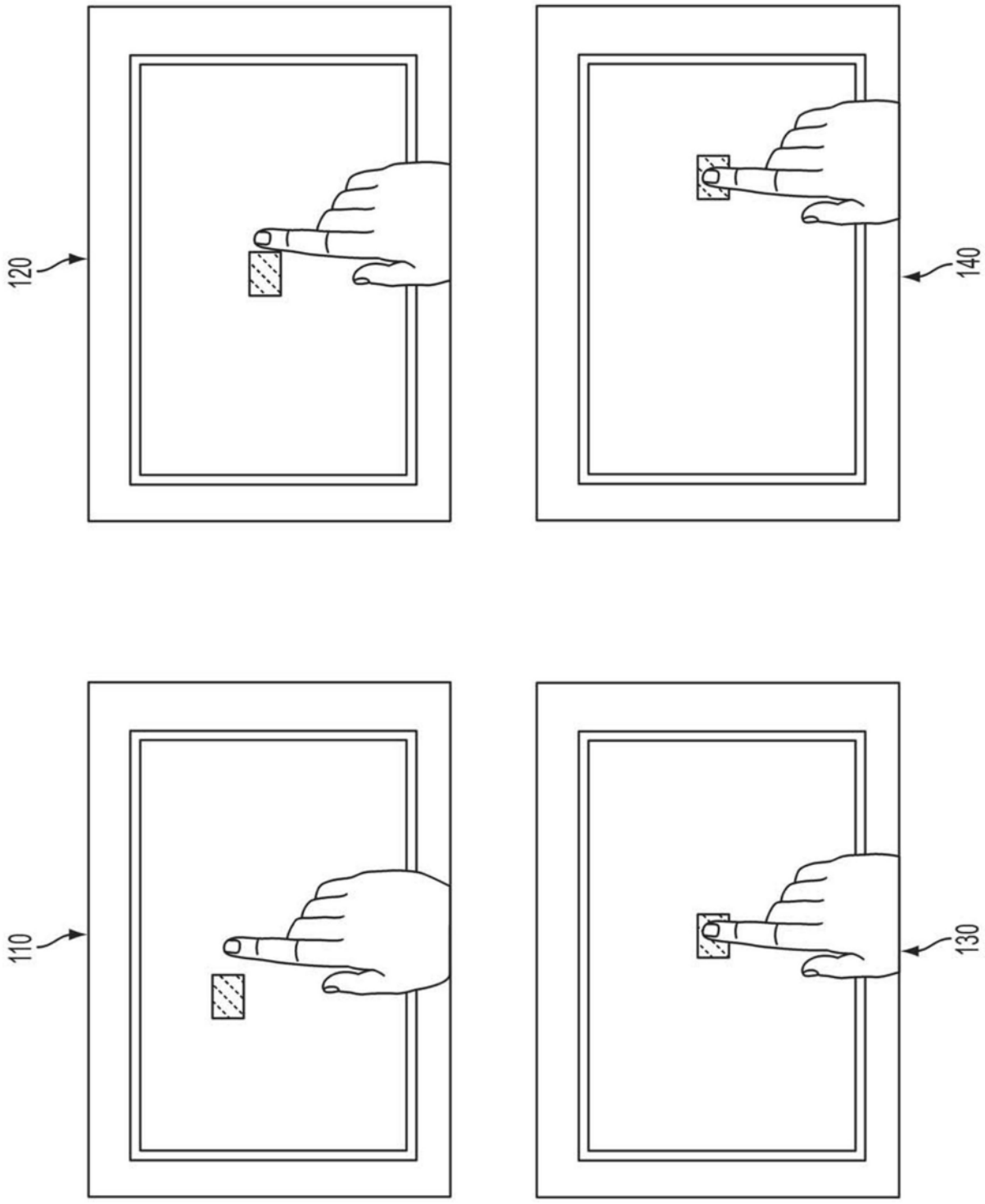


图1

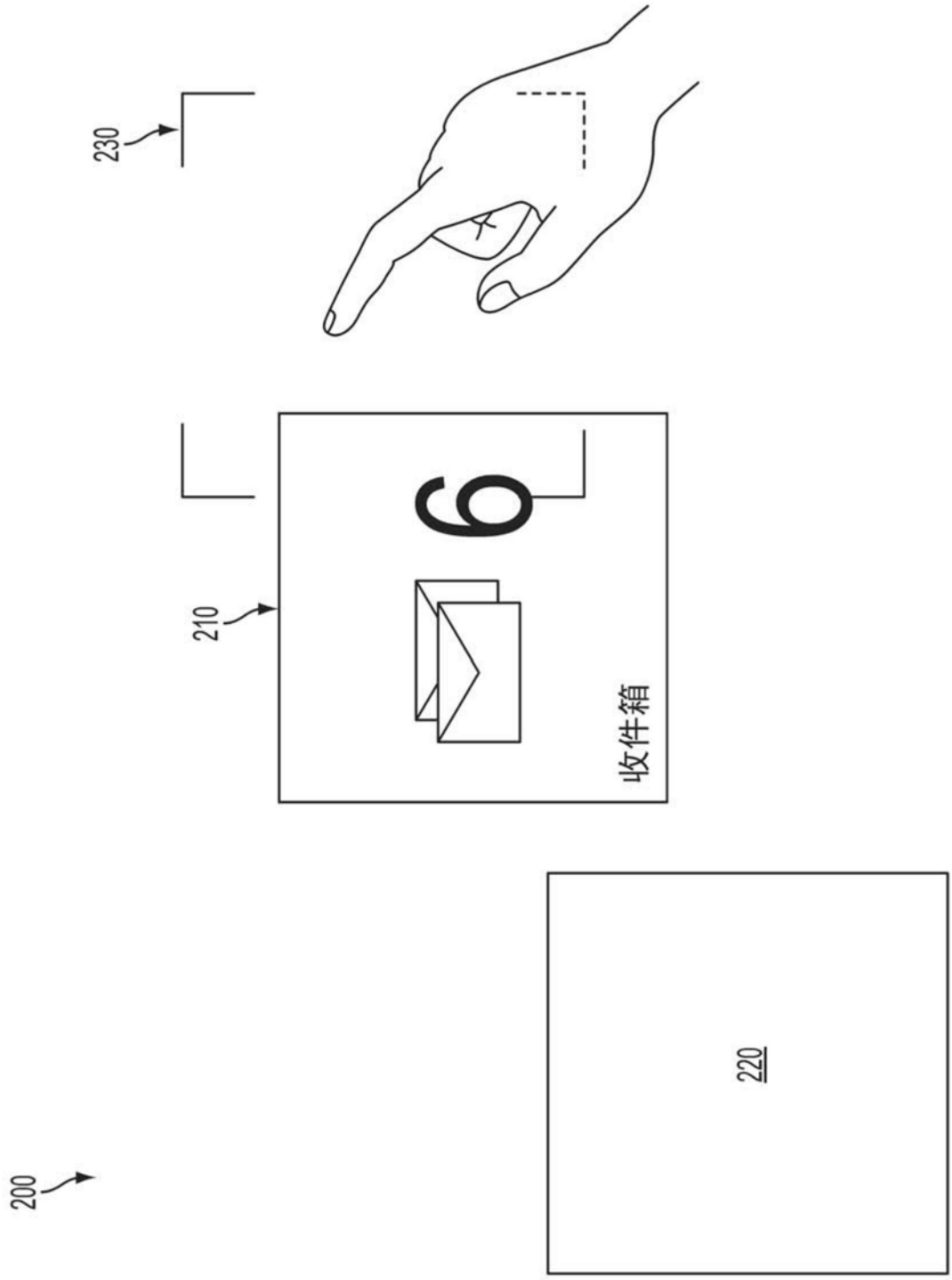


图2

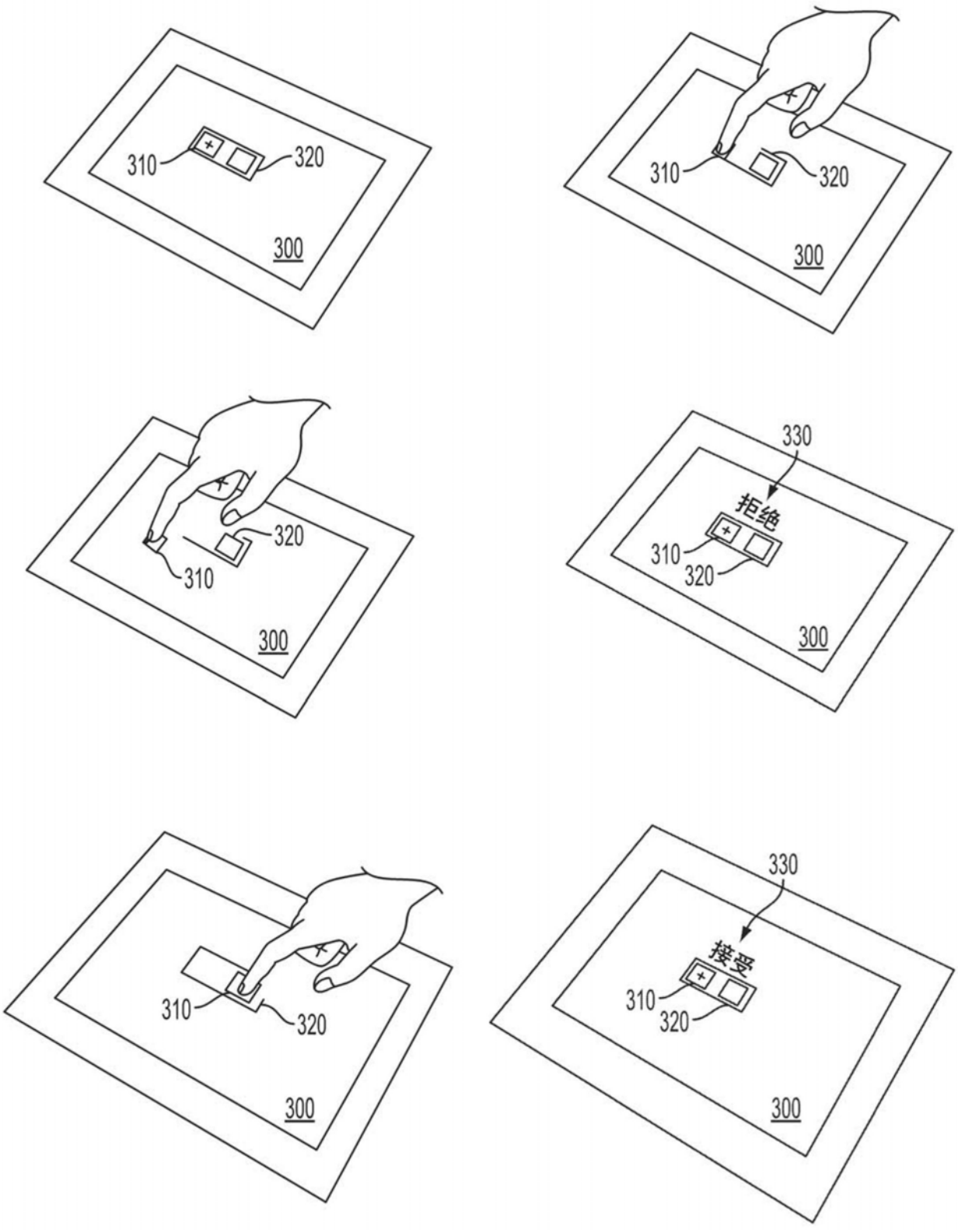


图3

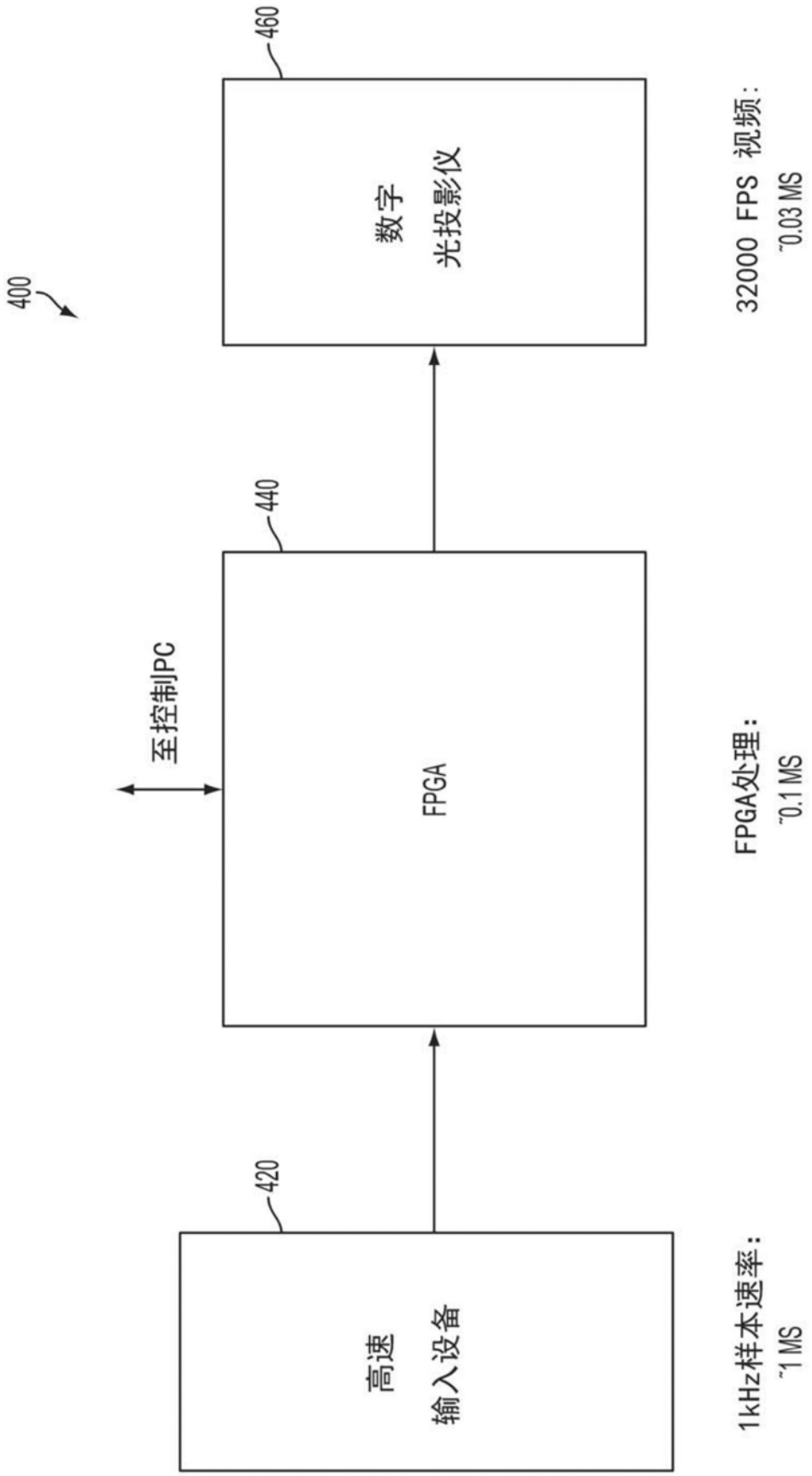

400
460
数字
光投影仪
32000 FPS 视频:
~0.03 MS
440
至控制PC
FPGA
FPGA处理:
~0.1 MS
420
高速
输入设备
1kHz样本速率:
~1 MS

图4

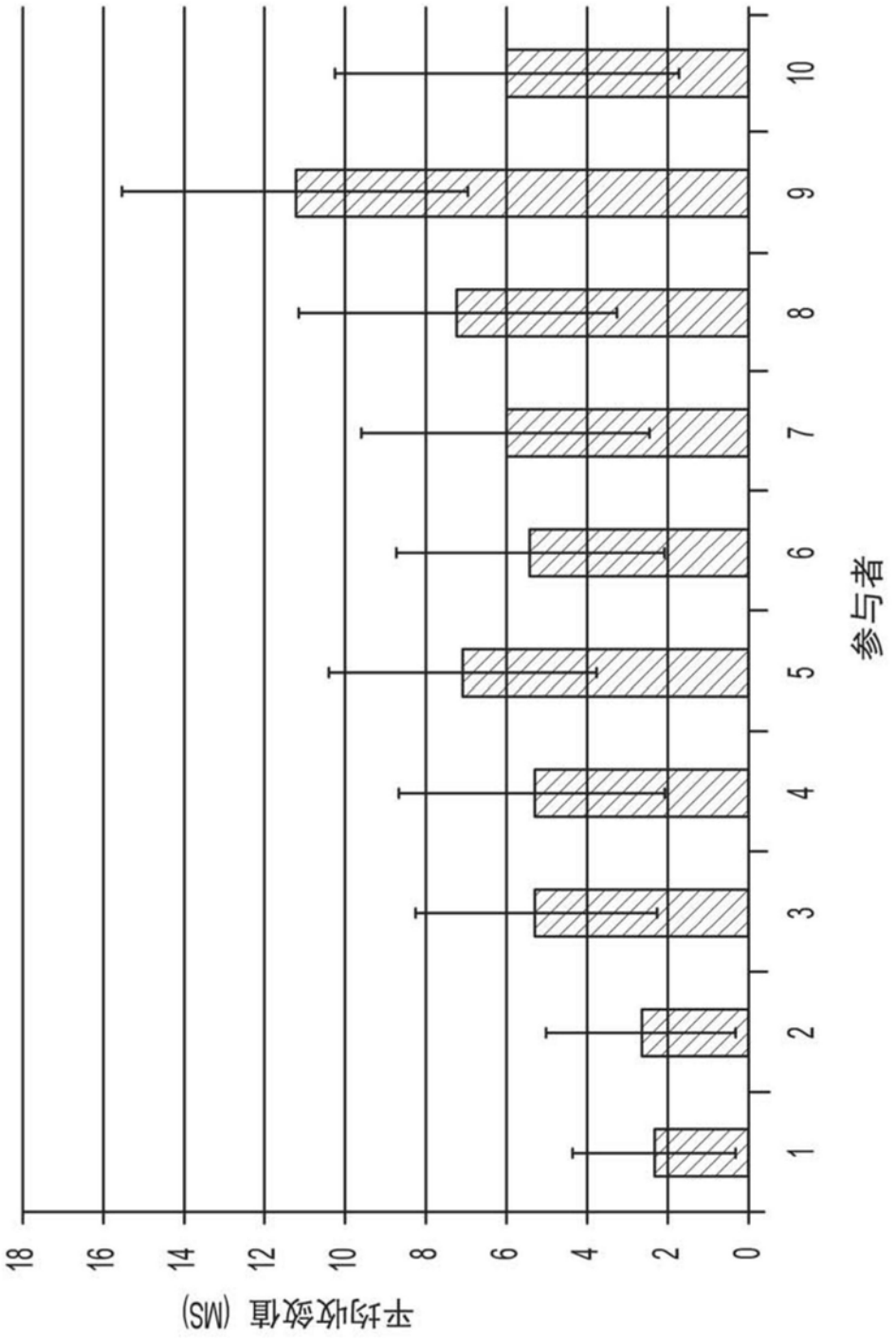


图5

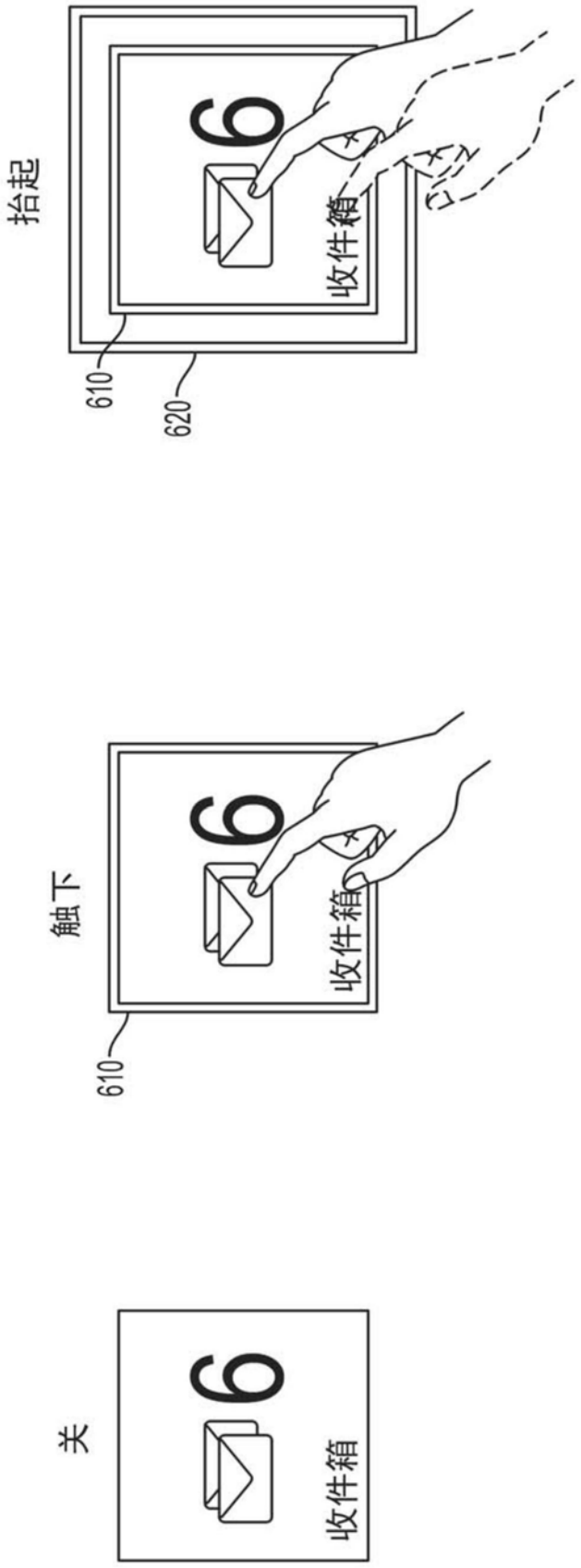


图6

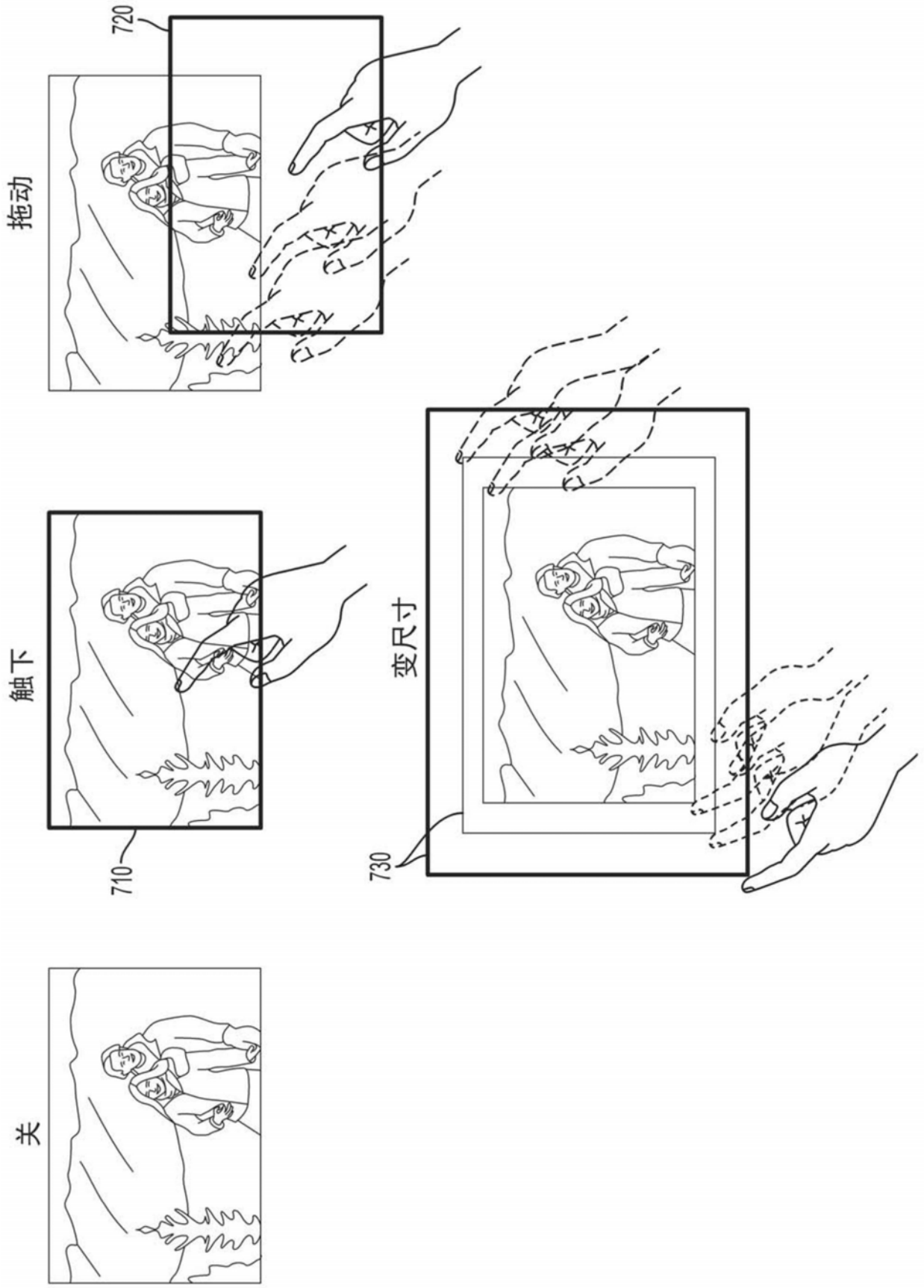


图7

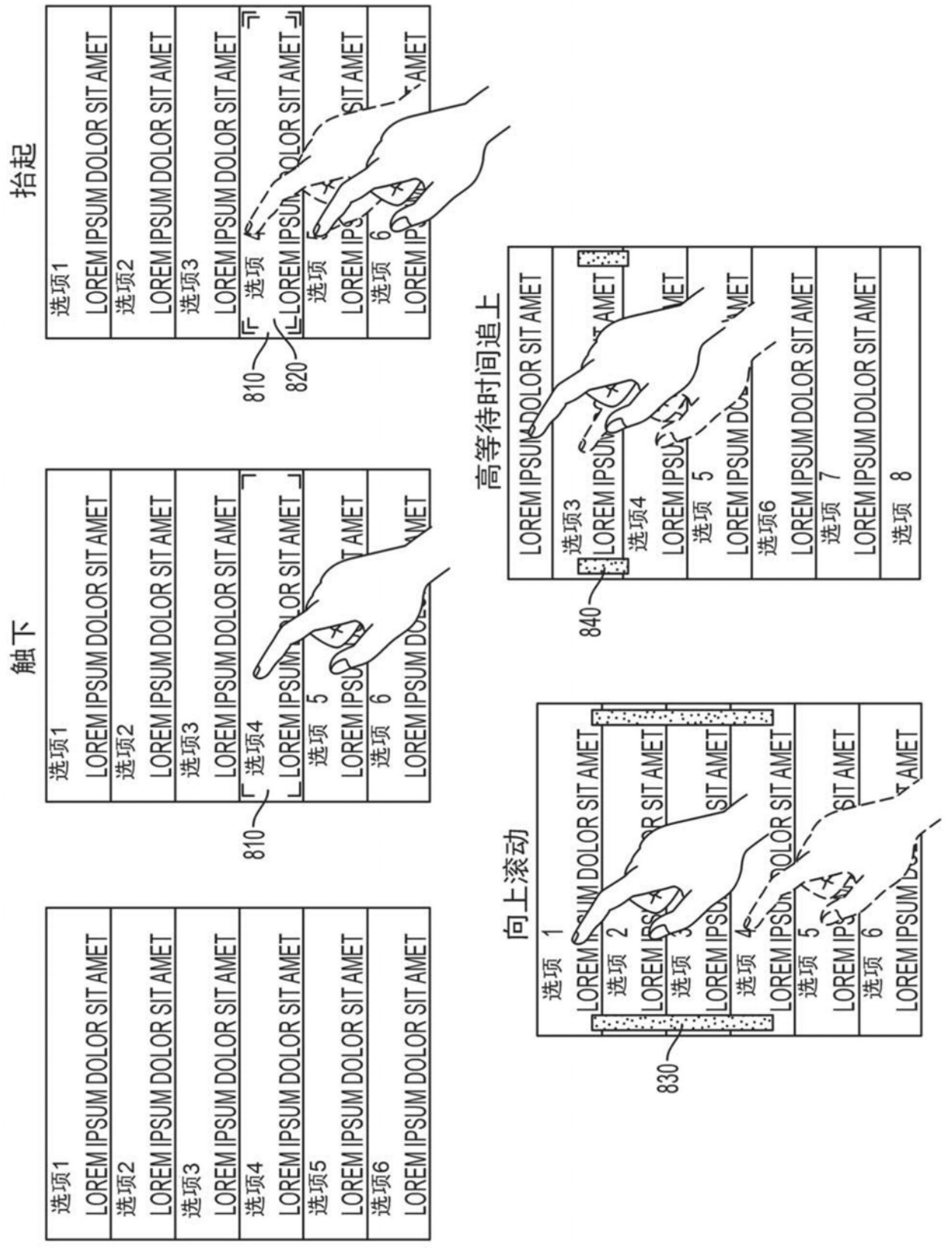

抬起
触下
高等待时间追上
向上滚动
选项1
选项2
选项3
选项4
选项5
选项6
选项 7
选项 8
LOREM IPSUM DOLOR SIT AMET
810
820
830
840

图8

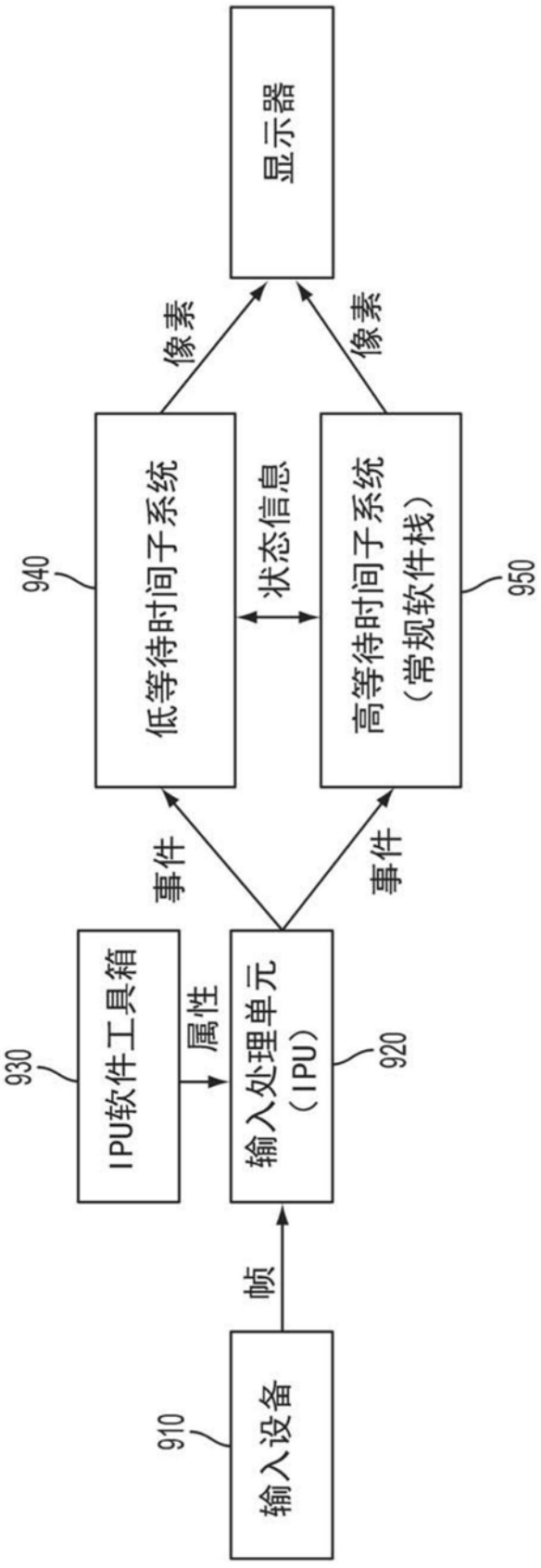

显示器
像素
像素
940
低等待时间子系统
状态信息
高等待时间子系统
（常规软件栈）
950
事件
事件
930
IPU软件工具箱
属性
输入处理单元
（IPU）
920
帧
910
输入设备

图9

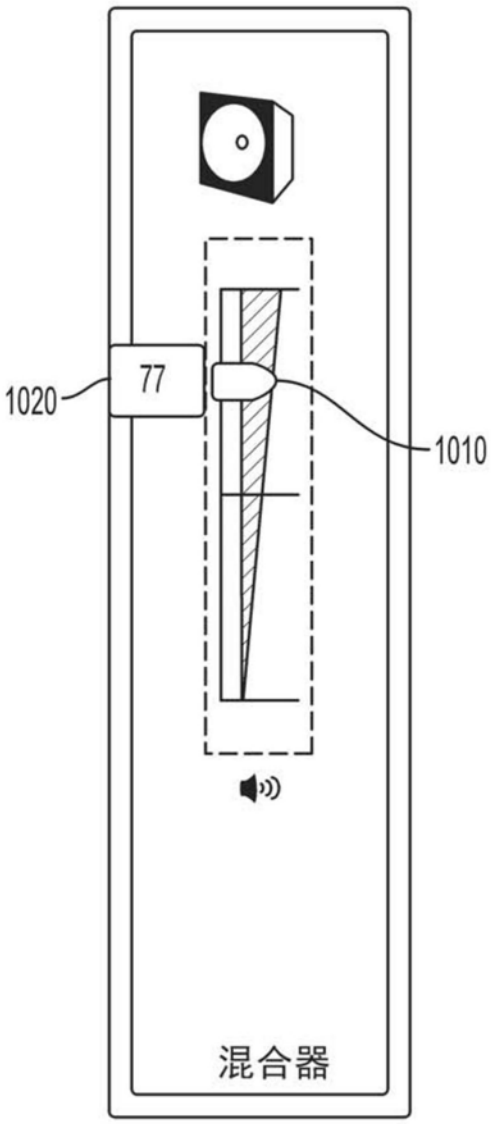


图10

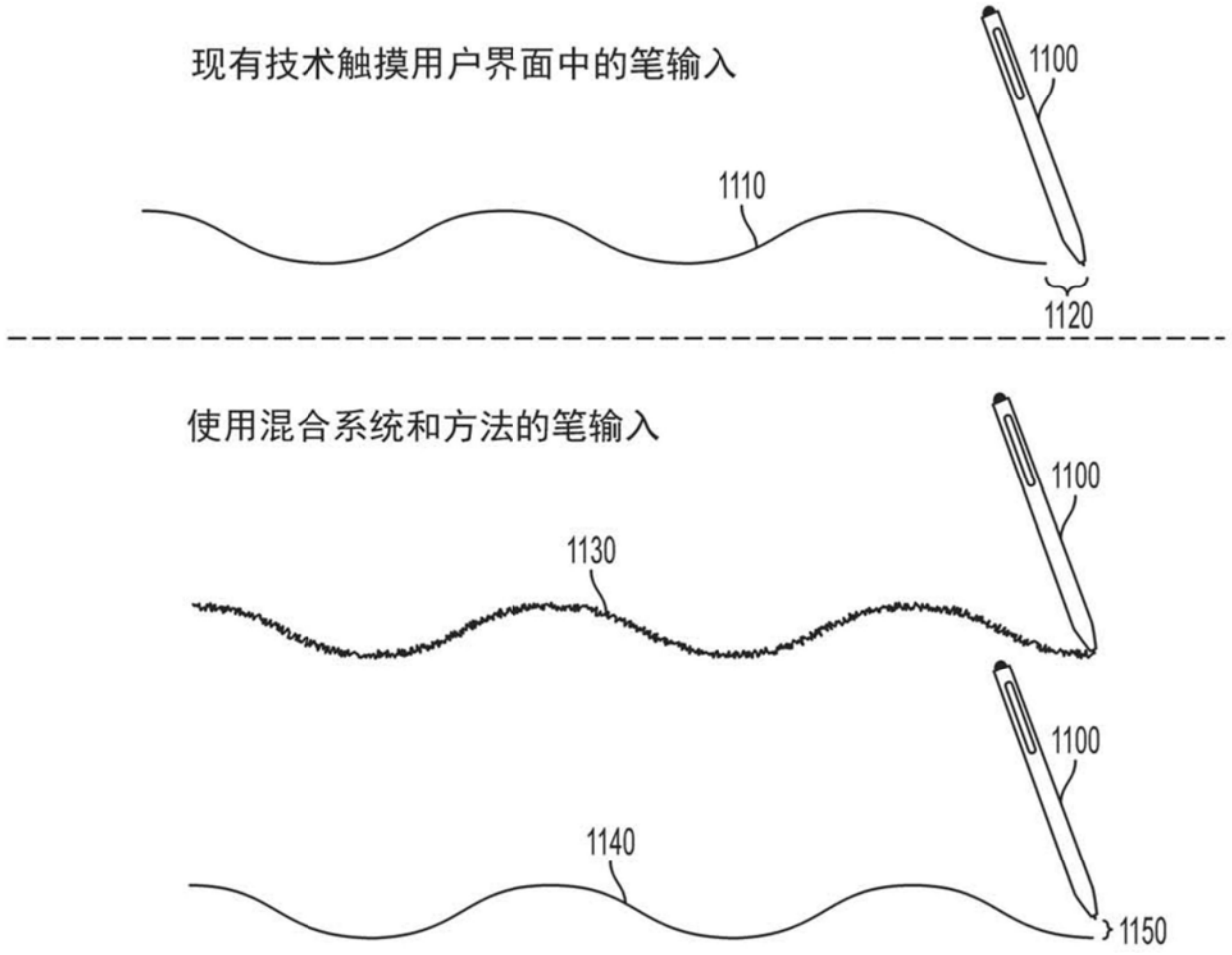

现有技术触摸用户界面中的笔输入
1100
1110
1120
使用混合系统和方法的笔输入
1100
1130
1100
1140
1150

图11

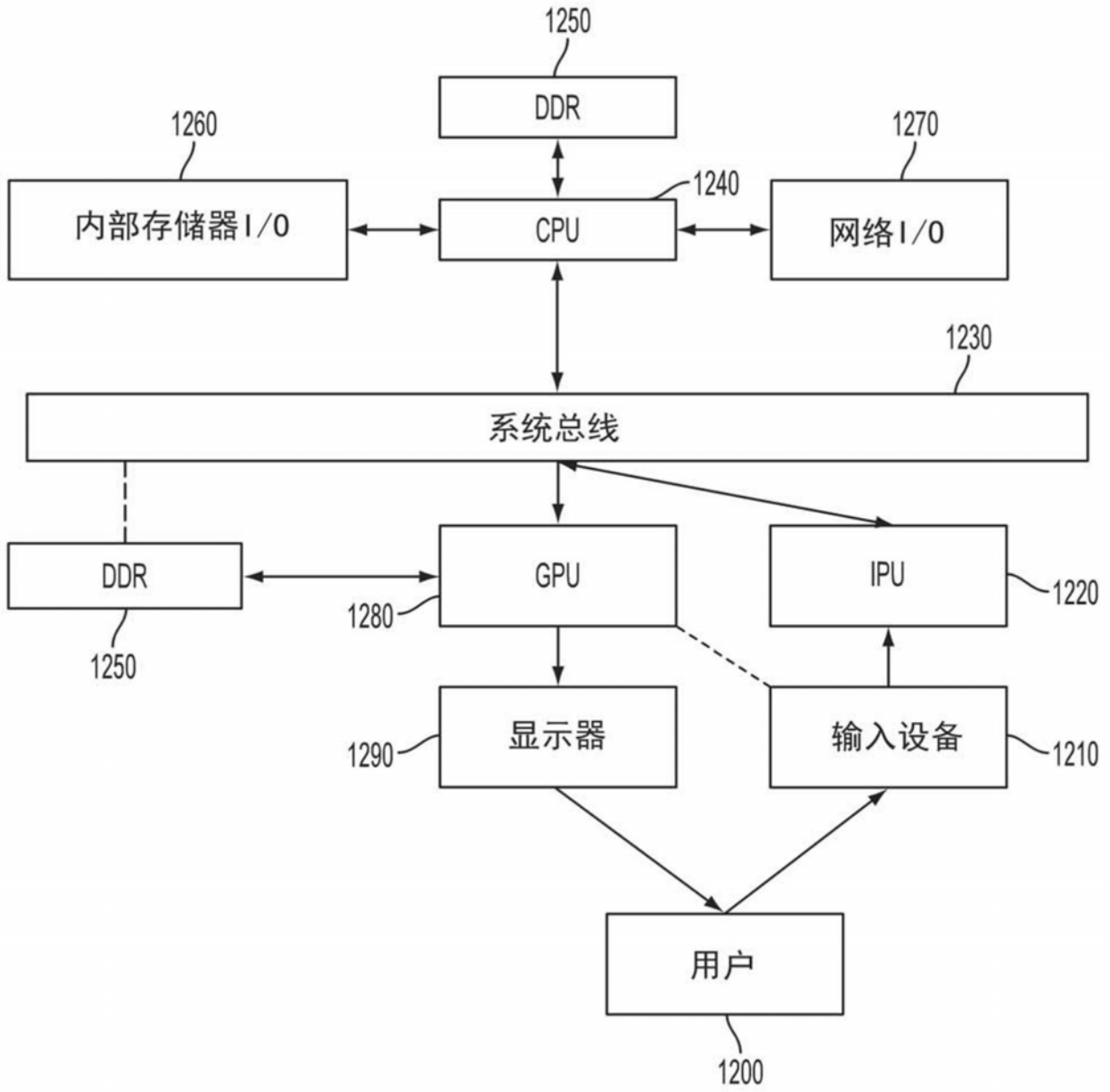

1250
DDR
1260
1270
内部存储器I/O
1240
CPU
网络I/O
1230
系统总线
DDR
GPU
IPU
1220
1280
1250
显示器
输入设备
1290
1210
用户
1200

图12

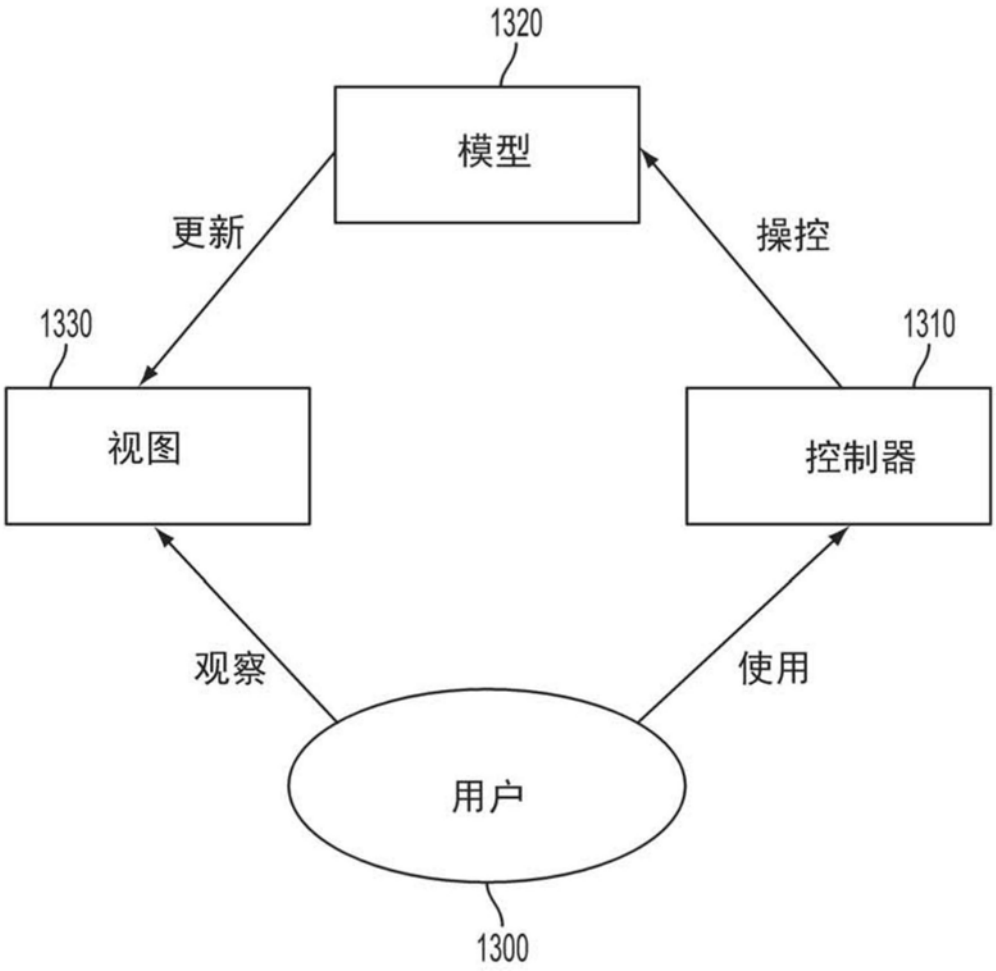


图13

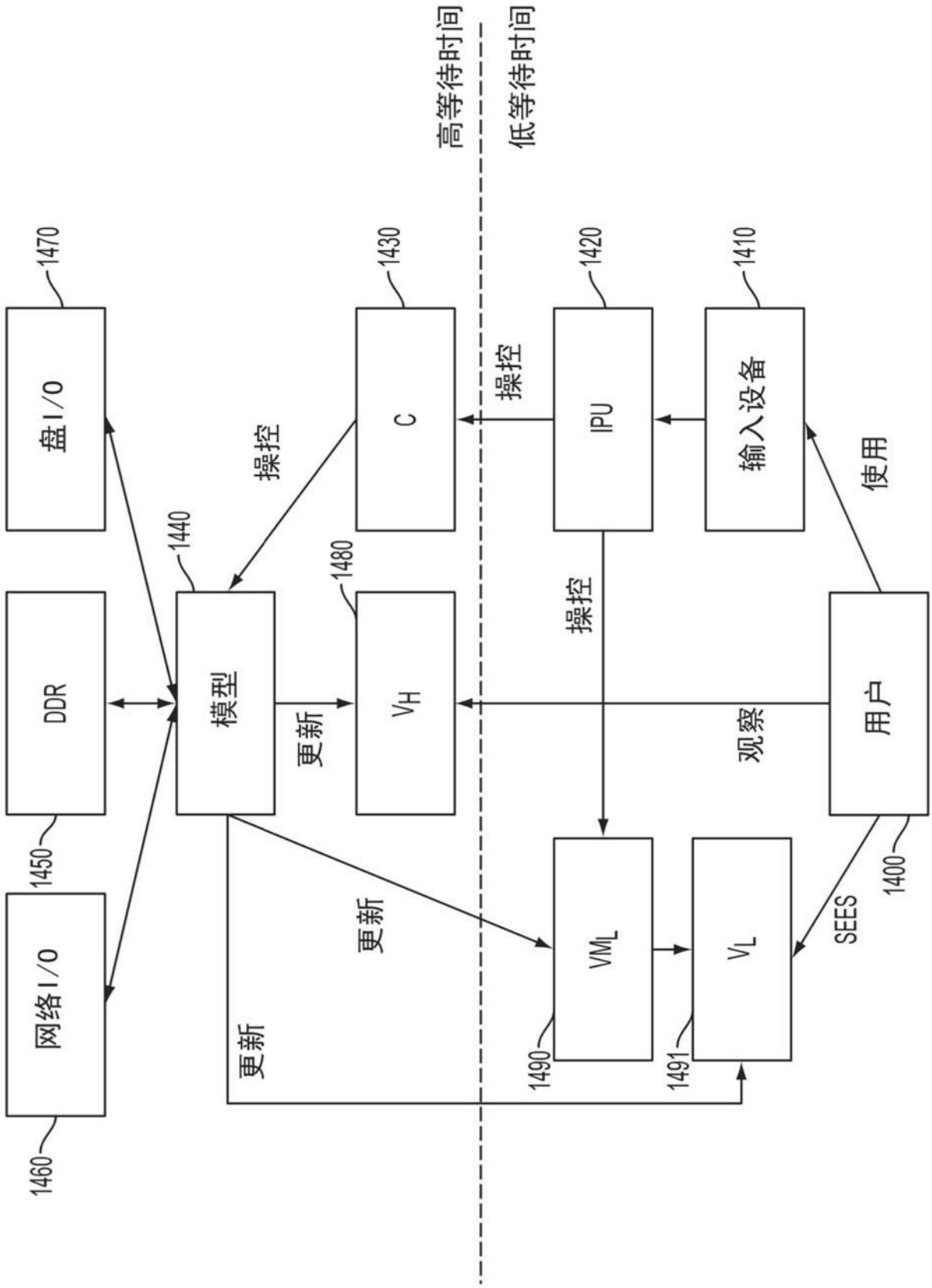

高等待时间
低等待时间
1470
盘I/O
1430
C
1420
IPU
1410
输入设备
操控
操控
使用
1440
模型
1480
V_H
操控
DDR
更新
观察
用户
1450
网络I/O
更新
VM_L
V_L
SEES
1400
更新
1490
1491
1460

图14

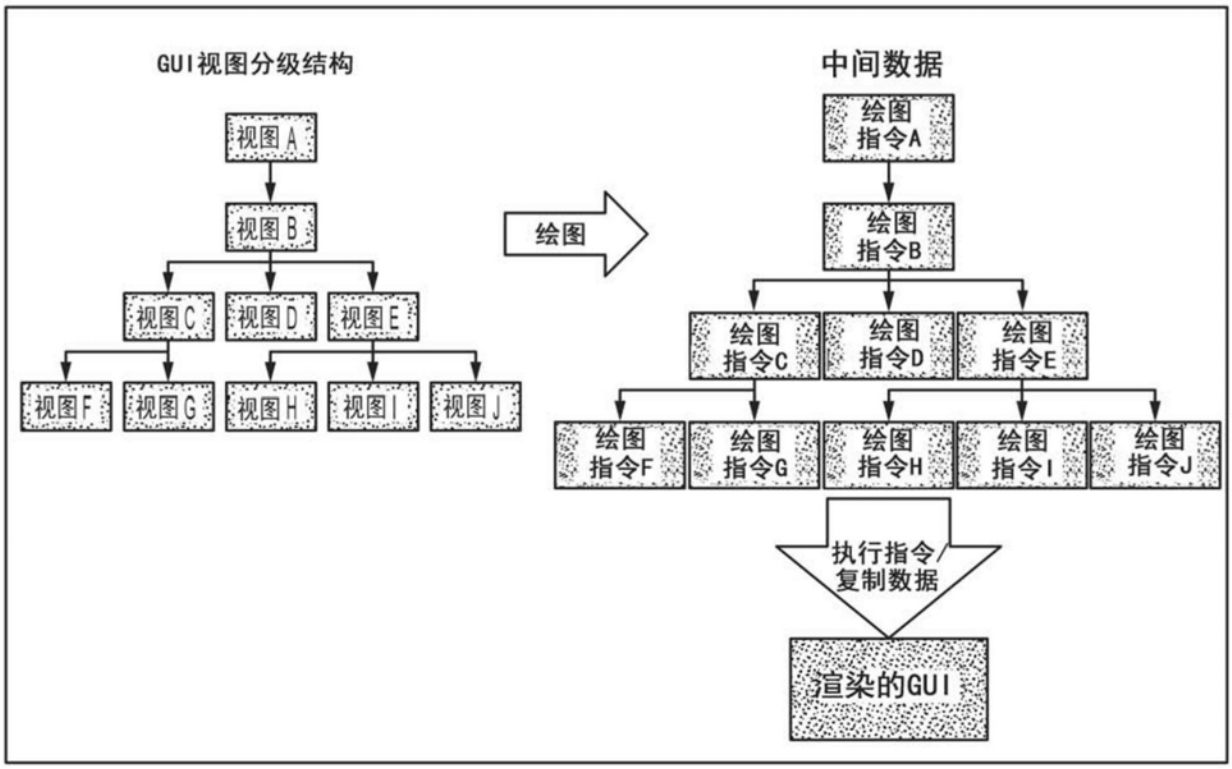

GUI视图分级结构
视图A
视图B
视图C
视图D
视图E
视图F
视图G
视图H
视图I
视图J
绘图
中间数据
绘图指令A
绘图指令B
绘图指令C
绘图指令D
绘图指令E
绘图指令F
绘图指令G
绘图指令H
绘图指令I
绘图指令J
执行指令/复制数据
渲染的GUI

图15现有技术

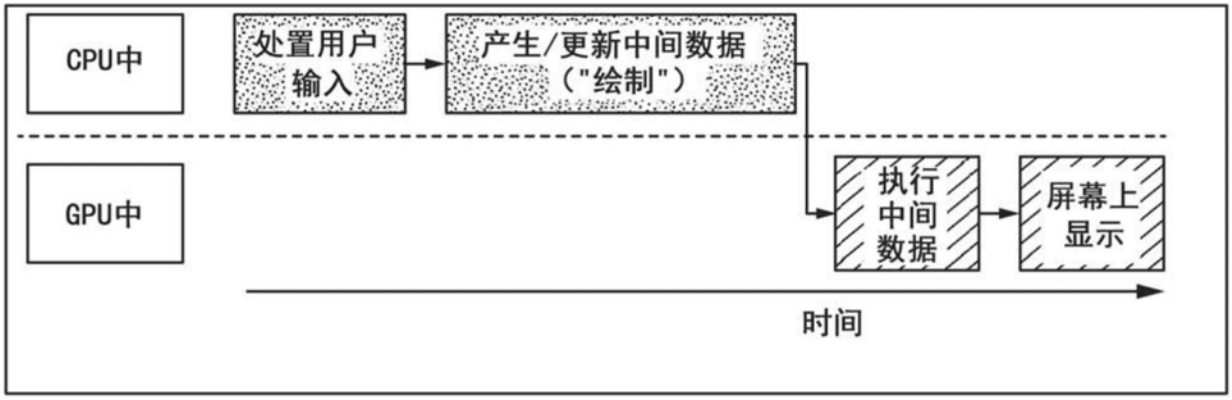

CPU中
处置用户输入
产生/更新中间数据（"绘制"）
GPU中
执行中间数据
屏幕上显示
时间

图16

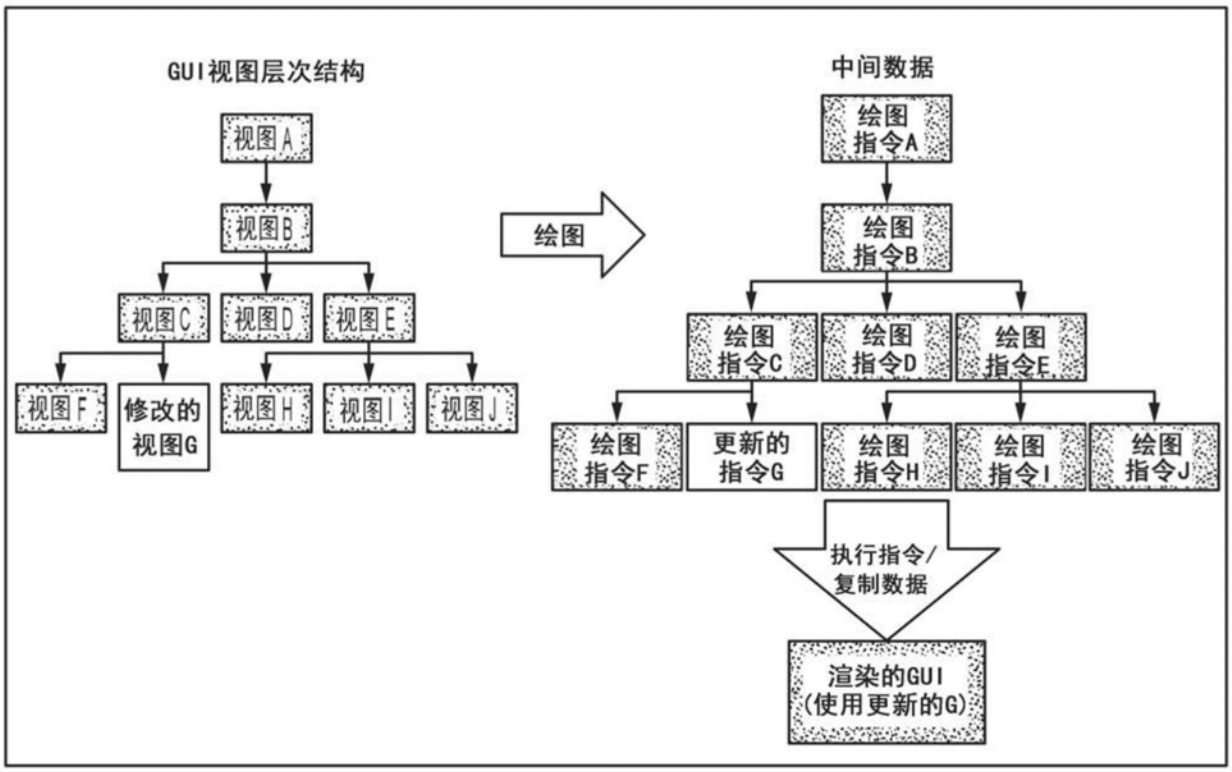

GUI视图层次结构
视图A
视图B
视图C
视图D
视图E
视图F
修改的视图G
视图H
视图I
视图J
绘图
中间数据
绘图指令A
绘图指令B
绘图指令C
绘图指令D
绘图指令E
绘图指令F
更新的指令G
绘图指令H
绘图指令I
绘图指令J
执行指令/复制数据
渲染的GUI(使用更新的G)

图17

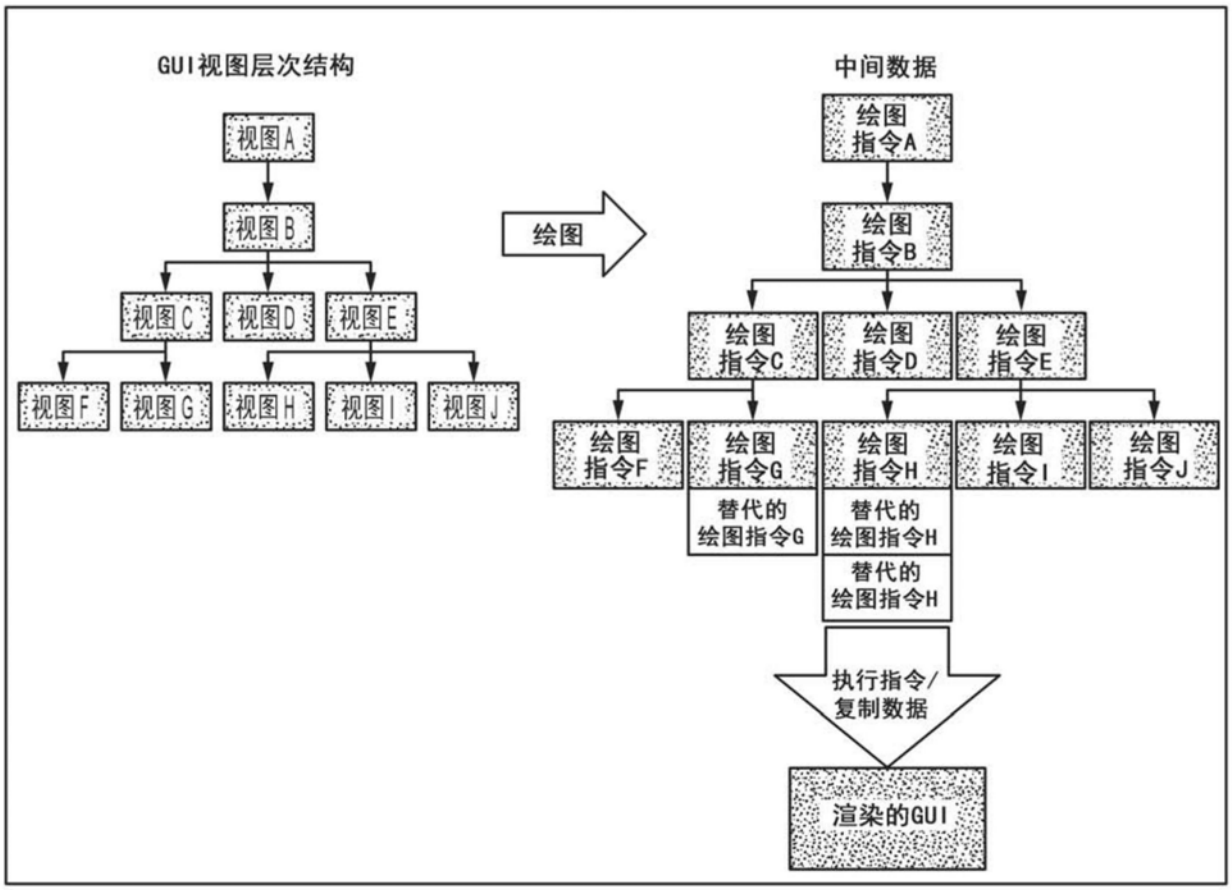

GUI视图层次结构
视图A
视图B
视图C
视图D
视图E
视图F
视图G
视图H
视图I
视图J
绘图
中间数据
绘图指令A
绘图指令B
绘图指令C
绘图指令D
绘图指令E
绘图指令F
绘图指令G
绘图指令H
绘图指令I
绘图指令J
替代的绘图指令G
替代的绘图指令H
替代的绘图指令H
执行指令/复制数据
渲染的GUI

图18

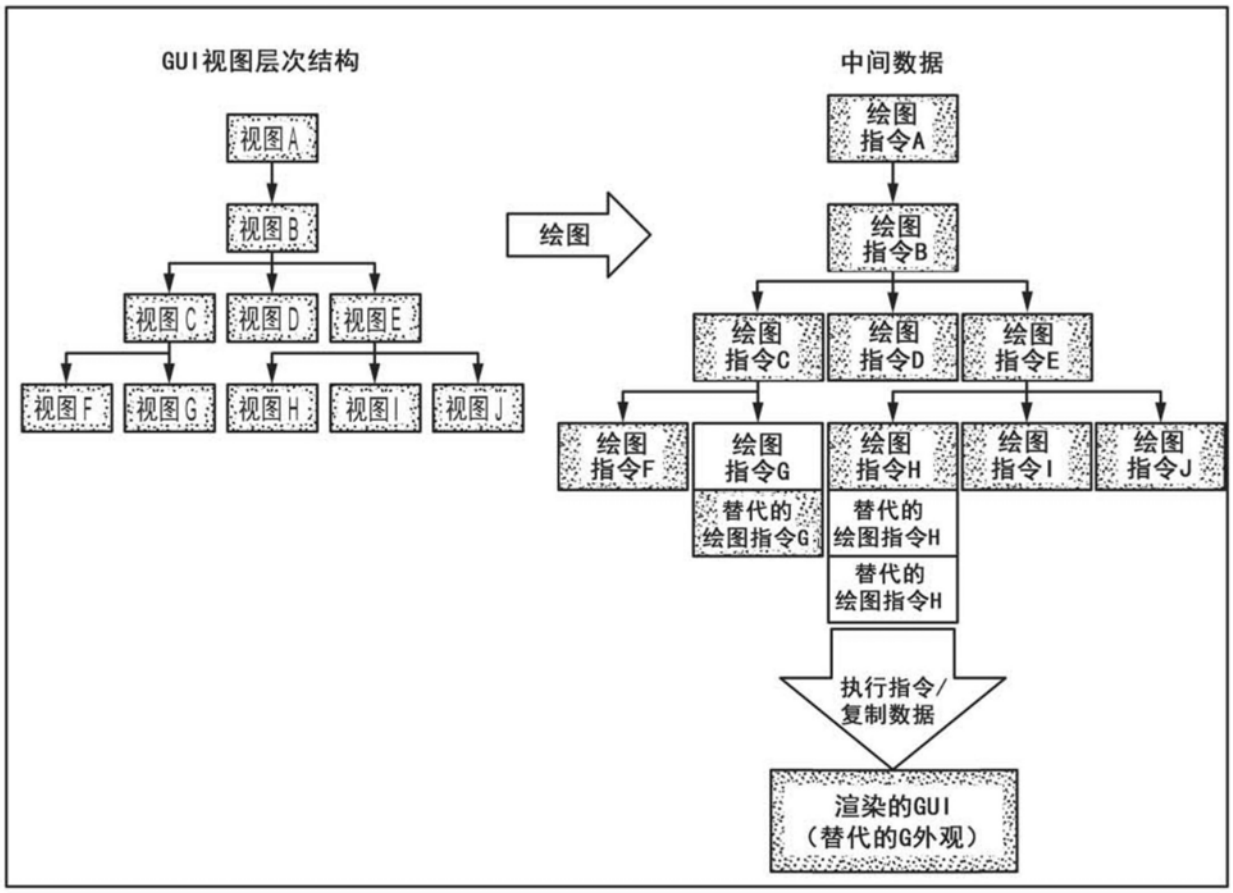

GUI视图层次结构
视图A
视图B
视图C
视图D
视图E
视图F
视图G
视图H
视图I
视图J
绘图
中间数据
绘图指令A
绘图指令B
绘图指令C
绘图指令D
绘图指令E
绘图指令F
绘图指令G
绘图指令H
绘图指令I
绘图指令J
替代的绘图指令G
替代的绘图指令H
替代的绘图指令H
执行指令/复制数据
渲染的GUI（替代的G外观）

图19

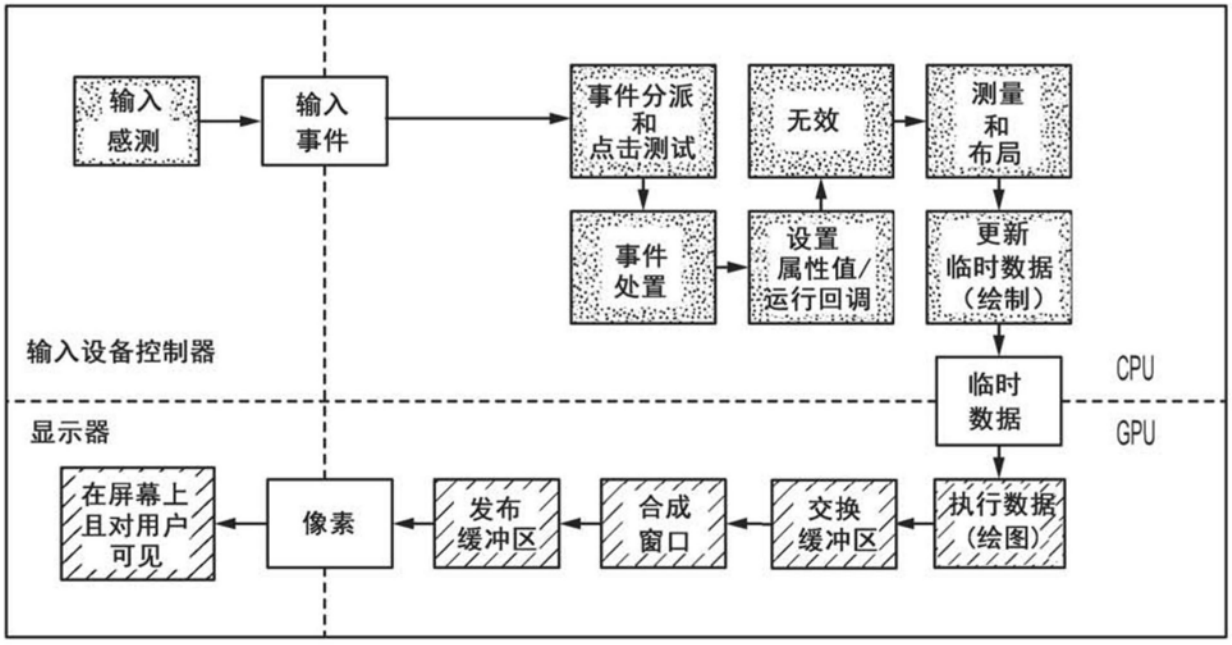

输入感测
输入事件
事件分派和点击测试
无效
测量和布局
事件处置
设置属性值/运行回调
更新临时数据（绘制）
输入设备控制器
CPU
临时数据
显示器
GPU
在屏幕上且对用户可见
像素
发布缓冲区
合成窗口
交换缓冲区
执行数据（绘图）

图20现有技术

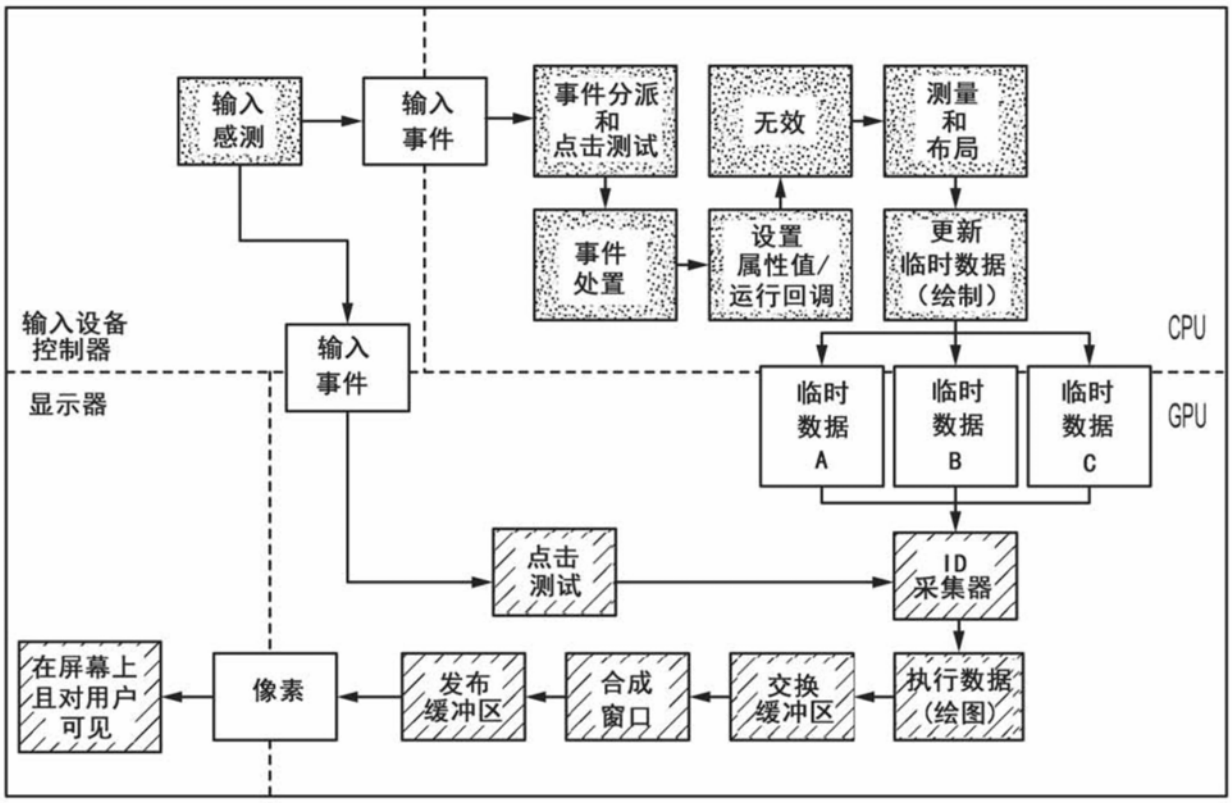

输入感测
输入事件
事件分派和点击测试
无效
测量和布局
事件处置
设置属性值/运行回调
更新临时数据（绘制）
输入设备控制器
输入事件
CPU
显示器
GPU
临时数据A
临时数据B
临时数据C
点击测试
ID采集器
在屏幕上且对用户可见
像素
发布缓冲区
合成窗口
交换缓冲区
执行数据（绘图）

图21

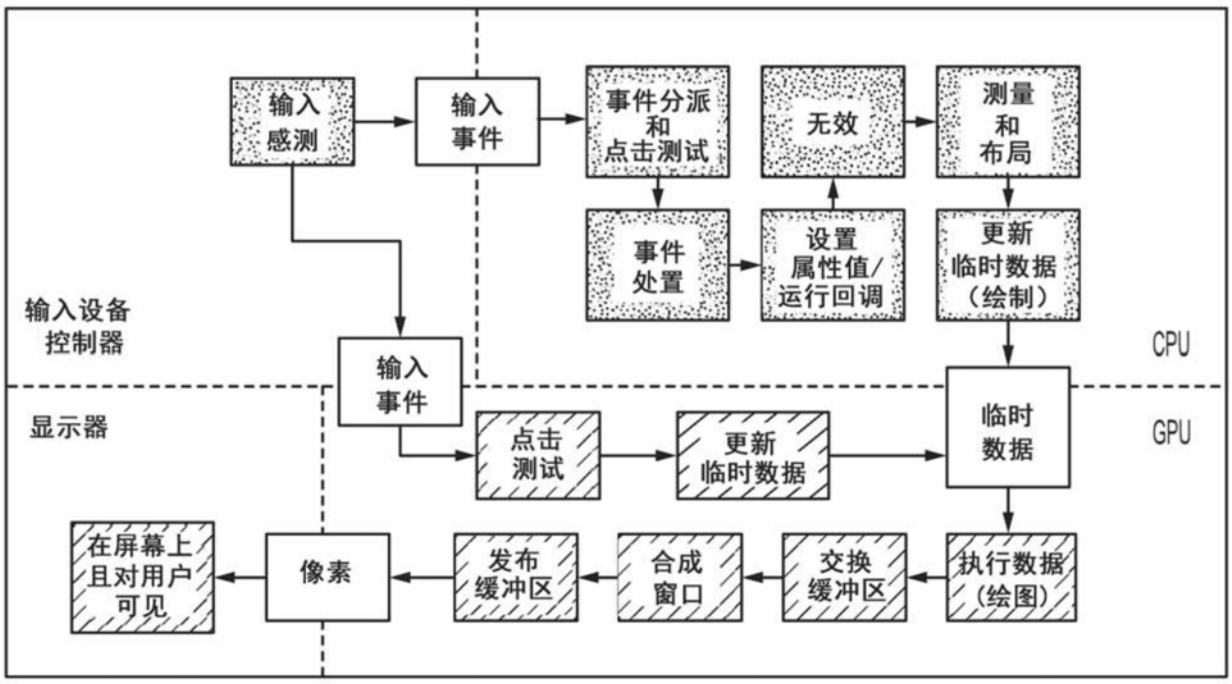

输入感测
输入事件
事件分派和点击测试
无效
测量和布局
事件处置
设置属性值/运行回调
更新临时数据（绘制）
输入设备控制器
输入事件
CPU
显示器
GPU
点击测试
更新临时数据
临时数据
在屏幕上且对用户可见
像素
发布缓冲区
合成窗口
交换缓冲区
执行数据（绘图）

图22

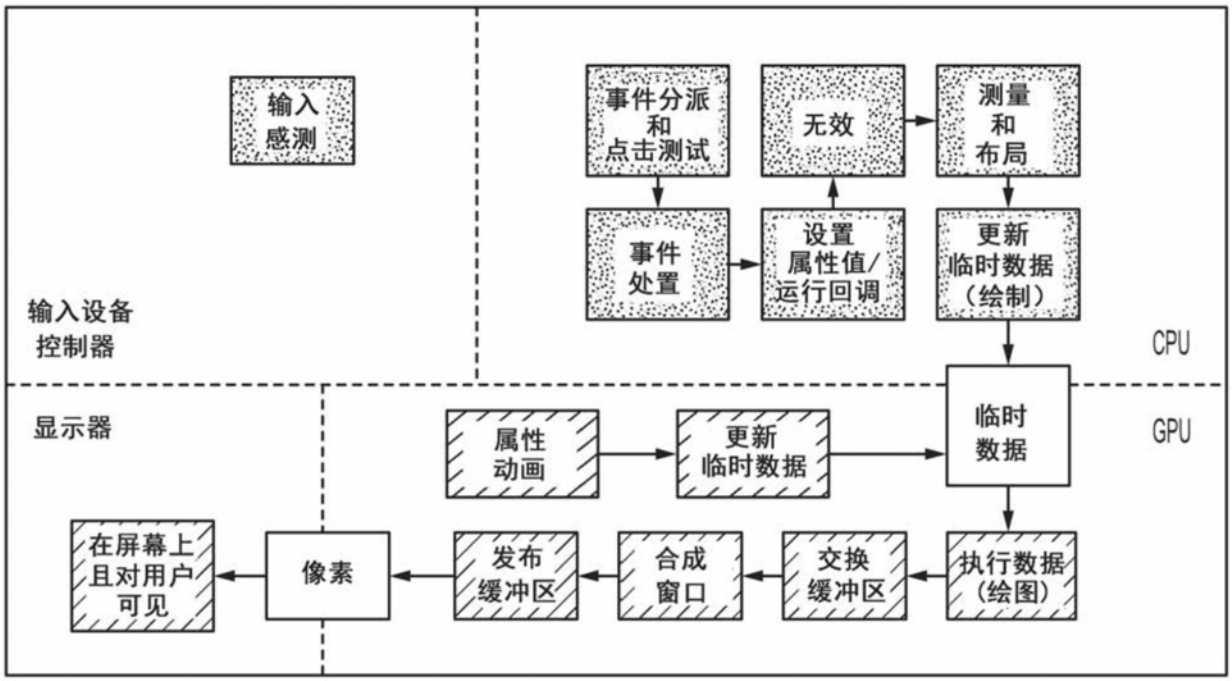

输入
感测
事件分派
和
点击测试
无效
测量
和
布局
事件
处置
设置
属性值/
运行回调
更新
临时数据
（绘制）
输入设备
控制器
CPU
显示器
属性
动画
更新
临时数据
临时
数据
GPU
在屏幕上
且对用户
可见
像素
发布
缓冲区
合成
窗口
交换
缓冲区
执行数据
（绘图）

图23