



(12) 发明专利申请

(10) 申请公布号 CN 113853241 A

(43) 申请公布日 2021. 12. 28

(21) 申请号 201980089044.7

(22) 申请日 2019.11.19

(30) 优先权数据

16/198,682 2018.11.21 US

(85) PCT国际申请进入国家阶段日

2021.07.14

(86) PCT国际申请的申请数据

PCT/US2019/062301 2019.11.19

(87) PCT国际申请的公布数据

W02020/106779 EN 2020.05.28

(71) 申请人 索尼互动娱乐有限责任公司

地址 美国加利福尼亚州

(72) 发明人 J. 贝尔特兰 K. 坎农 N. 格罗斯

D. 布特勒

(74) 专利代理机构 北京市柳沈律师事务所

11105

代理人 张晓明

(51) Int.Cl.

A63F 13/355 (2006.01)

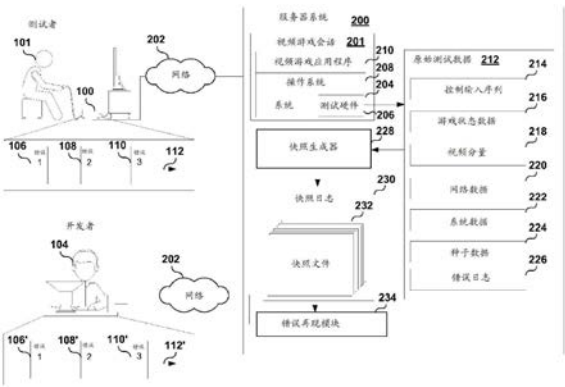
权利要求书3页 说明书18页 附图14页

(54) 发明名称

用于云游戏的测试即服务

(57) 摘要

描述了用于视频游戏的测试即服务(TaaS)的技术。在一个实施方案中,一种方法包括用于接收用于测试一个或多个错误的游戏应用程序的操作。所述方法还提供了由自动游戏测试模块执行游戏会话的多个自动会话,同时针对所述多个自动会话实现测试输入,所述测试输入包括控制输入、游戏状态、系统参数和网络参数。所述方法还包括用于进行以下项的操作:在所述执行所述多个会话期间检测错误的发生以生成快照文件,所述快照文件包括控制输入的一部分、游戏状态数据以及与错误的发生相关联的视频分量。



1. 一种方法,其包括:

在云测试系统的服务器上执行供玩家玩的视频游戏,所述执行所述视频游戏产生视频输出;

从所述玩家的客户端装置接收用于与所述视频游戏进行交互的控制输入序列,所述控制输入序列的一部分导致在与所述视频游戏的所述交互期间出现错误;

将控制输入序列的导致所述错误的所述部分与由所述视频游戏产生的所述视频输出和游戏状态数据一起记录在玩家生成的快照文件中;

使用机器学习模块处理所述玩家生成的快照文件,所述机器学习模块从所述控制输入序列、所述视频输出和所述游戏状态中提取特征以用于分类到错误检测模型中;

使用所述错误检测模型识别可能会通过所述视频游戏再现所述错误的一组测试控制输入。

2. 如权利要求1所述的方法,其还包括:

在分别输入用于再现所述错误的所述一组测试控制输入的同时执行多个自动视频游戏会话,所述执行所述多个自动视频游戏会话各自产生相应的游戏状态数据和相应的视频分量,其中所述相应的测试控制输入、游戏状态数据和视频分量记录在所述云测试系统的快照数据库中的相应的机器生成的快照文件中。

3. 如权利要求2所述的方法,其还包括:

使用所述机器学习模块,基于是否再现了错误来执行对所述相应的快照文件的分类,并基于所述分类来更新所述错误检测模型;

从所述错误检测模型中提取错误实现条件,所述错误实现条件能够用于识别导致所述错误的附加控制输入。

4. 如权利要求3所述的方法,其中所述错误实现条件包括与所述错误相关联的一个或多个规则,所述一个或多个规则能够用于识别导致所述错误的控制输入序列类别,所述控制输入序列类别包括所述控制输入序列和附加的控制输入序列。

5. 如权利要求3所述的方法,其中所述错误实现条件包括与所述错误相关联的游戏情境,所述游戏情境能够用于识别通常由与所述错误的发生相关联的所述玩家生成的快照文件和所述机器生成的快照文件共享的一个或多个游戏状态类别。

6. 如权利要求5所述的方法,其中所述一个或多个游戏状态类别指定级别、阶段、任务、使命、游戏中的动作、地图区域、角色、物品、能力、游戏中的对象、游戏中的对象之间的交互、人工智能(AI)角色、玩家控制的角色或结构中的一者或多者。

7. 如权利要求3所述的方法,其中所述错误实现条件包括与所述错误相关联的系统情境,所述系统情境能够用于识别通常由与所述错误的发生相关联的所述玩家生成的快照文件和所述机器生成的快照文件共享的一个或多个系统参数类别。

8. 如权利要求7所述的方法,其中所述系统参数类别指定中央处理单元(CPU)的时钟速率、图形处理单元(CPU)的时钟速率、通信链路的数据速率容量、所述通信链路的等待时间、与所述通信链路相关联的抖动或压缩速率中的一者或多者。

9. 如权利要求1所述的方法,其中多个自动视频游戏会话中的每一个在分布式游戏引擎上执行,所述分布式游戏引擎包括具有由相应的多个计算节点执行的操作的多个功能单元,其中由用于所述分布式游戏引擎的所述相应的多个计算节点执行的所述操作由分布式

游戏引擎管理器管理。

10. 如权利要求2所述的方法,其中所述自动视频游戏会话在相应的分布式游戏引擎上执行,每个分布式游戏引擎包括用于执行相应功能的多个游戏引擎节点,所述游戏节点中的每一个与相应的计算节点进行通信以处理用于所述执行所述相应功能的操作。

11. 一种计算机实现的方法,其包括:

从视频游戏的一个或多个玩家的玩游戏过程中生成玩家生成的快照文件,所述玩家生成的快照文件中的每一个包括与所述视频游戏的所述玩游戏过程的其中发生错误的部分相关联的控制输入序列、游戏状态数据和视频分量;

使用机器学习模块来处理所述玩家生成的快照文件以生成多个机器习得的控制输入,所述多个机器习得的控制输入是除了用于再现所述错误的来自所述玩家生成的快照文件的所述控制输入序列之外的;

在输入相应的机器习得的控制输入的同时执行多个自动视频游戏会话,所述执行所述多个自动视频游戏会话各自产生相应的游戏状态数据和相应的视频分量,其中所述相应的机器习得的控制输入、所述相应的游戏状态数据和所述相应的视频分量记录在相应的机器生成的快照文件中;以及

使用所述机器学习模块来处理所述机器生成的快照文件以识别错误实现条件,所述错误实现条件能够用于识别导致所述错误的控制输入序列类别,所述控制输入序列类别是除了所述玩家生成的快照文件的所述控制输入序列之外的。

12. 如权利要求11所述的计算机实现的方法,其中所述执行所述多个自动视频游戏会话还包括:

引入与执行所述自动视频游戏会话或所述输入所述机器习得的控制输入相关联的参数差异,所述参数差异也记录在所述机器生成的快照文件中,以供所述机器学习模块进行处理,其中所述错误实现条件还能够用于识别导致所述错误的特定参数差异。

13. 如权利要求11所述的计算机实现的方法,其中所述使用所述机器学习模块来处理所述玩家生成的快照文件还包括:

基于相应的控制输入序列、相应的游戏状态数据和相应的视频分量,从所述玩家生成的快照文件中提取特征;

使用错误检测模型,基于用于生成所述相应的控制输入序列的特性与所述错误的发生之间的相关性的所述特征,对所述玩家生成的快照文件进行分类;以及

基于错误检测模型构造所述机器习得的控制输入。

14. 如权利要求12所述的计算机实现的方法,其中所述引入与执行所述自动视频游戏会话相关联的参数差异包括以下项中的一者或多者:使中央处理单元(CPU)超频或降频、使图形处理单元(GPU)超频或降频、改变系统存储器的大小或速度、改变图形存储器的大小或速度、改变系统存储装置的大小或速度、改变帧缓冲区的大小或速度以及改变所述相应的视频分量的分辨率或帧速率。

15. 如权利要求11所述的计算机实现的方法,其中所述多个自动视频游戏会话在相应的分布式游戏引擎上执行,每个分布式游戏引擎包括用于执行相应功能的多个游戏引擎节点,所述游戏节点中的每一个与相应的计算节点进行通信以处理用于所述执行所述相应功能的操作。

16. 一种用于执行测试即服务 (TaaS) 的方法, 其包括:

在服务器处从客户端接收用于测试一个或多个错误的游戏应用程序;

由自动游戏测试模块执行所述游戏应用程序的第一多个自动会话, 同时针对所述第一多个自动会话实现相应的测试输入, 所述相应的测试输入包括控制输入、游戏状态数据、系统参数和网络参数;

在所述执行多个自动游戏会话期间, 由错误报告程序检测错误的发生; 以及

生成快照文件, 所述快照文件包括所述控制输入的一部分、所述游戏状态数据以及与所述错误的所述发生相关联的视频分量。

17. 如权利要求16所述的方法, 其还包括:

在错误再现模块处, 处理所述控制输入的所述部分、所述游戏状态数据以及与所述错误的所述发生相关联的所述视频分量以再现所述错误, 所述错误的所述再现被配置为被发送以供所述客户端显示。

18. 如权利要求16所述的方法, 其还包括:

使用机器学习模块来处理所述快照文件以生成多个机器习得的控制输入, 所述多个机器习得的控制输入是除了用于再现所述错误的控制输入部分之外的;

在输入所述机器习得的控制输入的同时使用所述自动游戏测试模块来执行第二多个自动会话, 以产生相应的游戏状态数据和相应的视频分量, 其中所述相应的机器习得的控制输入、所述相应的游戏状态数据和所述相应的视频分量记录在相应的机器生成的快照文件中; 以及

使用所述机器学习模块处理所述机器生成的快照文件, 以识别能够用于识别导致所述错误的附加控制输入的错误实现条件, 其中所述错误实现条件被配置为被递送给所述客户端。

19. 如权利要求16所述的方法, 其中用于所述游戏应用程序的所述第一多个自动会话的所述测试输入包括由混沌测试模块、或由负载测试模块、或由所述客户端、或由与测试所述游戏应用程序的先前版本时或测试不同的游戏应用程序时识别的所述错误的先前发生相关联的先前快照文件定义的控制输入序列。

20. 如权利要求16所述的方法, 其中所述执行第一多个自动游戏会话还包括向所述系统参数和网络参数引入差异以用于再现所述错误。

用于云游戏的测试即服务

技术领域

[0001] 本公开总体上涉及视频游戏开发,并且更具体地,涉及用于提供对视频游戏的云测试作为服务的方法和系统。

背景技术

[0002] 视频游戏测试是游戏开发过程的一部分,其保持对视频游戏的质量控制。游戏测试的功能之一是发现和记录会损害视频游戏使用的软件缺陷(例如,错误(bug))。过去,由于视频游戏的规模较小且复杂度较低,因此游戏测试所需的人力相对较少。如今,视频游戏测试是开发者和游戏发行商为确保其游戏应用程序平稳运行而采用的大规模且昂贵的工作。这是由于如今的视频游戏的大范围和复杂性以及消费者对几乎完美的游戏动作的需求所致。因此,游戏测试在确保发布的游戏平稳运行且没有错误、差错或其他缺陷上起到很大的作用。一些发行商可能会在给定的时间使数十名质量保证测试者致力于某个特定的游戏标题,这可能会非常昂贵。此外,当游戏测试者确实识别出错误时,由于与游戏测试者的游戏会话和其他因素相关联的差异,开发者并不总是能够再现所识别的错误。

[0003] 那些实施方案正是在这种背景下出现的。

发明内容

[0004] 本公开的实施方案涉及用于游戏测试回放、自动游戏测试以及测试即服务(TaaS)的方法和系统。

[0005] 在一个实施方案中,提供一种方法。所述方法包括用于在云测试系统的服务器上执行供玩家玩的视频游戏的操作,所述执行视频游戏产生视频输出。所述方法还包括用于从玩家的客户端装置接收用于与视频游戏进行交互的控制输入序列的操作,所述控制输入序列的一部分导致在与视频游戏的所述交互期间出现错误。此外,所述方法包括用于将控制输入序列的导致错误的所述部分与由视频游戏产生的视频输出和游戏状态数据一起记录在玩家生成的快照文件中的操作。此外,所述方法包括用于使用机器学习模块处理玩家生成的快照文件的操作,所述机器学习模块从控制输入序列、视频输出和游戏状态中提取特征以用于分类到错误检测模型中。此外,所述方法包括用于使用错误检测模型来识别可能通过视频游戏再现错误的一组测试控制输入的操作。

[0006] 在另一个实施方案中,提供一种方法。所述方法包括用于从视频游戏的一个或多个玩家的玩游戏过程中生成玩家生成的快照文件的操作,所述玩家生成的快照文件中的每一个包括与视频游戏的玩游戏过程的其中发生错误的部分相关联的控制输入序列、游戏状态数据和视频分量。所述方法还包括用于使用机器学习模块来处理玩家生成的快照文件以生成多个机器习得的控制输入的操作,所述多个机器习得的控制输入是除了用于再现错误的来自玩家生成的快照文件的控制输入序列之外的。此外,所述方法包括用于在输入相应的机器习得的控制输入的同时执行多个自动视频游戏会话的操作,所述执行多个自动视频游戏会话各自产生相应的游戏状态数据和相应的视频分量,其中所述相应的机器习得的控

制输入、所述相应的游戏状态数据和所述相应的视频分量记录在相应的机器生成的快照文件中。另外，所述方法包括用于使用机器学习模块来处理所述机器生成的快照文件以识别错误实现条件的操作，所述错误实现条件可用于识别导致错误的控制输入序列类别，所述控制输入序列类别是除了玩家生成的快照文件的控制输入序列之外的。

[0007] 在另一个实施方案中，提供一种方法。所述方法包括用于在服务器处接收用于测试一个或多个错误的游戏应用程序的操作。所述方法还包括用于由自动游戏测试模块执行游戏应用程序的自动会话同时针对所述自动会话实现测试输入的操作，所述测试输入包括控制输入、游戏状态数据、系统参数和网络参数。另外，所述方法提供了由错误报告程序在所述执行自动游戏会话期间检测错误的发生。此外，所述方法提供了生成快照文件，所述快照文件包括控制输入的一部分、游戏状态数据以及与错误的发生相关联的视频分量。

[0008] 从以下结合附图进行的详细描述中，本公开的其他方面将变得显而易见，所述附图以示例的方式示出了本公开的原理。

附图说明

[0009] 通过结合附图参考以下描述，可以最好地理解本公开，在附图中：

[0010] 图1示出了根据一个实施方案的错误报告和再现的概念图。

[0011] 图2示出了根据一个实施方案的用于如实地再现由游戏测试者发现的错误的改进的方法和系统的概念性图示。

[0012] 图3示出了根据一个实施方案的错误再现模块的概念性图示，所述错误再现模块用于通过处理捕获由游戏测试者识别的错误的快照文件来再现所述错误。

[0013] 图4示出了根据一个实施方案的包括多个快照文件的快照日志的概念图。

[0014] 图5示出了根据一个实施方案的基于云的测试服务器系统如何实现机器学习以生成在统计上与导致先前识别的错误或新识别的错误相关的新控制输入。

[0015] 图6示出了根据一个实施方案的差异实现模块如何创建与自动游戏测试模块处的自动游戏会话一起执行的差异包的概念图。

[0016] 图7示出了根据一个实施方案的正在执行多个自动游戏会话的自动游戏测试模块的概念性图示。

[0017] 图8示出了根据一个实施方案的利用分布式游戏引擎来执行多个自动游戏会话的自动游戏测试模块的架构的图。

[0018] 图9A和图9B概念性地示出了根据各种实施方案的机器学习模块、自动游戏测试模块和错误实现模块如何一起工作以识别可以实现先前或新识别的错误的条件集合。

[0019] 图10示出了根据一个实施方案的通过混沌 (chaos) 或霰弹枪 (shotgun) 测试发现新错误的方法。

[0020] 图11示出了根据一个实施方案的在游戏的新内部版本 (build) 或版本 (version) 中针对先前识别的错误进行自动测试的概念图。

[0021] 图12示出了根据一个实施方案的作为测试即服务 (TaaS) 的一部分如何将针对第一游戏已知的错误实现条件应用于第二游戏。

[0022] 图13是根据一个实施方案的可用于测试即服务 (TaaS) 的平台的概念性图示。

[0023] 图14示出了可以用于执行本公开的各种实施方案的各方面的示例装置 (例如TaaS

服务器)的部件。

具体实施方式

[0024] 本公开的实施方案涉及用于使游戏开发者能够再现由游戏测试者识别的错误的的方法和系统。本公开的实施方案还涉及使用机器学习算法来提供导致先前识别的错误或未识别的错误的附加的机器习得的控制输入的方法。此外,本公开的实施方案涉及用于使测试过程的各部分自动化以在变化的系统相关条件下测试游戏的方法和系统。此外,本公开的实施方案涉及允许将游戏测试的各方面作为服务来执行。然而,对于本领域的技术人员来说,显而易见的是,可以在没有这些具体细节中的一些或全部的情况下实践本公开。在其他实例中,没有详细描述众所周知的处理操作,以避免不必要地模糊本公开。

[0025] 游戏测试是一个昂贵且耗时的过程。实际上,所有进入商业阶段的游戏都将经过许多轮测试,以确保一定水平的功能和艺术品质。测试在开发视频游戏应用程序来发行的过程中起到多种作用。其中一些包括功能测试、合规性测试、兼容性测试、负载测试、beta测试、多玩家测试等。总体而言,游戏测试的目标是确保视频游戏在各种硬件平台上例如以预期的方式平稳运行。因此,游戏测试的目标之一是识别视频游戏中的妨碍游戏平稳运行的错误。错误可以通过多种方式展现。通常,错误是指游戏及其相关联代码中发生意外结果的部分,无论意外结果是否与游戏逻辑、游戏机制、差错、插图、系统故障或者不同于期望、预计或预期的其他游戏或系统响应有关。

[0026] 图1示出了根据一个实施方案的错误报告和再现的概念图。游戏测试者101被示为使用例如开发套件的测试装置100来玩视频游戏的片段112。游戏测试者101通过玩视频游戏的片段112而发现三个错误106、108和110。然后,游戏测试者101被示为提供错误报告102,所述错误报告被递送给开发者104。错误报告102可以通过缺陷跟踪系统提供,并且可以指定产生错误的情况(例如,级别或地图坐标)以及用于再现错误的特定步骤(例如,一组控制输入)。开发者104在尝试修复错误之前,会使用错误报告102来再现错误106、108和110。

[0027] 开发者104,可能是艺术家、程序员或游戏设计者等,成功地在三次尝试中的每一次再现了错误106'并且在三次尝试中的第二次再现了错误108'。开发者104无法再现错误110。无法再现某些错误(例如错误110)和能够间歇地再现错误(例如错误108)可能是由于与游戏测试者101的玩游戏过程相关联的许多因素造成的。例如,游戏测试者101和开发者104可能具有不同的种子数据或其他游戏状态数据,或者可能已经以时间上非常不相同的方式输入了控制输入序列。另外,可能存在与游戏测试者101的测试装置100相关联的系统差异(例如,等待时间、抖动、CPU时钟、GPU时钟、存储器使用情况等),这些因素会导致错误108和110的产生。对于开发者104来说,使用当前的方法,这些系统差异将难以跟踪并模仿错误108和110的再现。此外,可能的情况是,某些游戏测试者报告错误110,而其他游戏测试者没有报告所述错误,从而导致在错误110是否存在以及是否应对其进行处理方面的认识上的冲突。设想了用于提供对错误的如实再现的改进的方法和系统。

[0028] 图2示出了根据一个实施方案的用于如实地再现由游戏测试者101发现的错误的改进的方法和系统的概念性图示。类似于图1,图2的游戏测试者101被示为在与视频游戏的片段112交互期间已经产生了错误106、108、110。片段112可以是在服务器系统200上执行的

视频游戏会话201的一部分。也就是说,例如,视频游戏会话201可以由具有服务器系统200的测试硬件206的系统204托管。在一些实施方案中,测试硬件206可以是专有的。

[0029] 系统204被示为运行操作系统208,用于测试的视频游戏应用程序210正在所述操作系统上执行。游戏测试者101经由网络202与视频游戏会话201交互。例如,视频游戏应用程序210被预期为在服务器系统200处生成音频和视频分量,以用于递送和呈现给游戏测试者101。然后,服务器系统200经由网络202从游戏测试者101接收控制输入,以使游戏测试者101与视频游戏会话201进行交互。在其他实施方案中,视频游戏会话201可以在游戏测试者101本地的测试装置100上执行。然而,在任何情况下,由于游戏测试者101在视频游戏会话201期间与视频游戏应用程序210交互而产生了一组原始测试数据212。

[0030] 原始测试数据212被示为包括控制输入序列214,其可以是具有控制输入作为随时间变化的数据元素的任何合适的数据结构。例如,控制输入序列214被预期为记录如在游戏测试者101的某些输入装置上输入的各种控制输入中的每一个,包括按钮按下、击键、鼠标点击、光标移动、控制器移动、操纵杆移动、语音命令、身体移动以及由游戏测试者101做出的其他输入。可以根据视频游戏应用程序201记录控制输入的时刻以及它们在现实世界中在控制器处输入的时刻,来对控制输入序列214的数据元素中的每一个加时间戳。原始测试数据212还被示为包括游戏状态数据216,其可以具有用于捕获游戏环境的任何合适的数据结构,包括保存的数据、游戏值、地图数据、资产数据以及其他数据。游戏状态数据216还被预期为带有时间戳,使得可以将错误按时间引用到游戏状态数据216。

[0031] 当游戏测试者101与视频游戏应用程序201交互以发现错误106、108和110时,系统204会如视频游戏应用程序210所指示生成视频分量218。视频分量218可以包括呈视频帧序列形式的音频分量和视频分量两者。另外,当游戏测试者101与视频游戏应用程序210交互时,与视频游戏会话201相关联的各种网络和系统性质作为原始测试数据212的一部分分别实时地或接近实时地被捕获在网络数据220和系统数据222中。预期,网络数据220可以包括与服务器系统200和游戏测试者101之间的通信链路相关联的各种参数,包括例如等待时间、抖动、数据速率容量、分组丢失、服务质量(QoS)、吞吐量和其他网络性质。预期,系统数据222可以包括与执行视频游戏应用程序210的系统204或测试装置100相关联的各种参数,包括CPU时钟、GPU时钟、存储器使用率、帧速率、分辨率和其他系统性质。网络数据220和系统数据222两者被预期为带时间戳,使得可以将错误按时间引用到网络数据220和/或系统数据222内的特定时间。

[0032] 种子数据224和错误日志226也被示为包括在原始测试数据212中。种子数据224是指视频游戏应用程序210用来将一些变化或随机性引入到视频游戏中的初始数据集。在一些实施方案中,种子数据224可以利用随机数生成器。另外,错误日志226记录游戏测试者101输入的用于指示错误存在的标记或标志。例如,当游戏测试者101遇到错误106、108和110中的每一者时,它们可能在错误日志226中指示了这么多。错误日志226也被预期为带时间戳,使得错误106、108和110中的每一者发生的时间可以被引用到控制输入序列214、游戏状态数据216、视频分量218、网络数据220和系统数据222中的每一者中的特定窗口。

[0033] 快照生成器228被配置为生成存储在快照日志230中的快照文件232。每个快照文件232被预期为与特定错误相关联,并且包括控制输入序列214、游戏状态数据216、视频分量218、网络数据220、系统数据222、种子数据224的一部分以及错误日志226的与错误相关

联的部分。例如,错误106、108和110中的每一者被预期为包括在快照生成器228生成的快照文件232内。例如,当游戏测试者101指示存在错误106时,快照生成器228被预期为访问与错误日志226中的错误106的时间戳相对应的针对控制输入序列214、游戏状态数据216、视频分量218、网络数据220、系统数据222和种子数据224中的每一者的窗口,以生成错误106的快照文件232。所述窗口可能在错误106发生之前和之后大约1分钟到10分钟之间。快照生成器228被预期为还生成要存储在快照日志230中的针对错误108和110的快照文件232。除了游戏测试者101发现或产生的错误106、108和110的快照文件232之外,快照日志230可以是与其他游戏测试者或自动游戏测试代理发现或产生的各种附加错误相关联的快照文件232的数据库。

[0034] 服务器系统200还被预期为托管错误再现模块234,所述错误再现模块通过网络202与开发者104的装置对接以再现错误106'、108'和110'。错误再现模块234被预期为允许开发者104再现错误的方式有多种。三个非限制性实施方案包括回放模式、自动再现模式和手动再现模式。在回放模式下,快照文件232被回放给开发者104,使得错误106'、108'和110'如它们被显现给游戏测试者101的那样被回放给开发者104。因此,错误106、108和110的表示方式与它们展现给游戏测试者101的方式相同。

[0035] 在自动再现模式下,错误再现模块234被预期为加载与快照文件232相关联的游戏状态数据216,以用于执行视频游戏应用程序210。错误再现模块234可以访问系统204、测试硬件206、操作系统208和视频游戏应用程序210,以用于在自动再现模式下执行快照文件232。例如,系统204可以加载游戏状态数据216以在错误106、108、110中的每一者发生之前约5秒至约10分钟之间、或约10秒至5分钟之间、或约15秒至约60秒之间启动视频游戏会话201。一旦视频游戏会话201被启动,就基于控制输入序列214的每个控制输入的时间戳将控制输入序列214自动输入到视频游戏应用程序210。因此,控制输入序列214与游戏状态数据216同步,使得实时地为开发者104重新创建视频游戏会话201。在其他实施方案中,错误再现模块234可以配备有专用的回放系统硬件、操作系统、和视频游戏应用程序的开发版本。除了在执行视频游戏应用程序210期间复制控制输入序列214之外,还可以复制、镜像或模拟网络数据220、系统数据222和种子数据224,以在发生错误106、108和110的窗口期间重新创建操作条件。以这种方式,为开发者重新生成了错误106'、108'和110',但是所述错误可以在理论上与游戏测试者101所观察到的错误106、108和110相同。

[0036] 在手动再现模式中,由错误再现模块234加载快照文件232的游戏状态数据216,以用于在视频游戏会话201内介于错误106、108、110中的每一者发生之前约5秒至约10分钟之间、或约10秒至5分钟之间、或约15秒至约60秒之间的状态下执行视频游戏应用程序210。然而,代替自动地输入控制输入序列214,为开发者104提供了控制输入序列214的文本或视觉表示,使得开发者可以手动地重新创建错误106'、108'和110'。以这种方式,开发者104引起或触发错误106'、108'、110'的展现,并且可以尝试可能或不触发相同错误的控制输入的其他配置。因此,开发者104可以更好地理解导致错误的参数。在任何情况下,错误再现模块使开发者104能够再现如由游戏测试者101所识别的错误106'、108'和110',使得开发者104可以修复它们。

[0037] 图3示出了根据一个实施方案的错误再现模块234的概念性图示,所述错误再现模块用于通过处理捕获由游戏测试者所识别的错误106的快照文件232来再现所述错误106。

快照文件232a通过包含与发生错误106'时间上相关联的错误日志226a的一部分、控制输入序列214a的一部分、游戏状态数据216a的一部分以及视频分量218a的一部分来“捕获”错误106。尽管未示出,但是系统数据、网络数据和种子数据还被预期为包括在与错误106'相关联的快照文件232a内。在某些实施方案中,每个快照文件232a与错误的一个实例相关联。例如,如果错误106被快照文件232a捕获,则错误108和110可以被相应的快照文件捕获。然而,当错误彼此紧密相关联时,例如当第一错误导致第二错误时,或者当第一错误和第二错误在时间上彼此非常接近时,第一错误和第二错误两者可以由一个快照文件捕获。

[0038] 在所示的实施方案中,错误再现模块234被预期为执行错误回放模式302以简单地回放错误,执行自动再现模式304以自动地再现错误的相同实例,或执行手动再现模式306以用于指示开发者手动再现所述错误。在前述模式中的任一模式下,错误再现模块234生成错误再现GUI 300,以供开发者与之交互。错误再现GUI 300被示为包括多个用于显示的面板,包括源代码308面板、游戏状态310面板、视频分量312面板和控制输入314面板。视频分量312显示错误108涉及穿过田野的足球。如果错误再现模块234处于错误回放模式302,则如为游戏测试者所显示的那样,直接从视频分量218a获得在视频分量312中示出的视频帧。替代地,如果采用自动再现模式304或手动再现模式306,则视频分量312可以包括由于在输入控制输入序列214a的同时执行游戏应用程序而新生成的视频帧。然而,即使在自动再现模式304下或在手动再现模式306下,错误再现GUI 300也可以显示所生成的用于显示给游戏测试者的视频帧(例如,原始帧),使得开发者可以并排地观看原始帧和新生成的帧。

[0039] 源代码308可以包括视频游戏编程代码,所述视频游戏编程代码包含用于操作视频游戏的指令,包括用于游戏机制、游戏逻辑、资产管理、游戏物理、多玩家机制和元数据管理的代码。源代码308面板可以被预想为显示在错误108发生期间正在执行的代码。例如,源代码308可以包括用于使足球从5个单位的高度掉落的对游戏引擎的指令。源代码308面板被预期为在错误被再现时滚动到游戏代码的相关部分。

[0040] 游戏状态310被示为包括与足球相关联的各种游戏状态值。本文中,游戏状态310显示,“球反弹系数”被设置为“0”,而“球半径”被设置为“6”。游戏状态310面板使开发者能够查看在错误再现期间改变的各种游戏状态值和游戏引擎参数。

[0041] 另外,控制输入314面板使开发者能够查看可能已引起错误的控制输入序列214a。本文中,提供了游戏测试者使用的控制器的虚拟表示,以如控制输入序列214a被游戏测试者输入的那样呈现所述控制输入序列。例如,控制输入序列214a显示,游戏测试者输入了以下序列:“X-向下-向下-三角形-左_操纵杆_向下”。可以在虚拟控制器上以与控制输入序列214a相同的序列和时序虚拟地表示相同的序列。例如,虚拟控制器可以依次按下“X”按钮、按下两次向下按钮、按下三角形按钮、以及将左操纵杆向下倾斜。

[0042] 图4示出了根据一个实施方案的包括多个快照文件232的快照日志230的概念图。快照日志230可以包括由一个游戏测试者或若干游戏测试者生成的快照文件232。快照日志230中的每个快照文件232包括快照文件标识符401、错误标识符400、标签402、游戏相关数据404、系统数据406和网络数据408。错误标识符400可以与缺陷跟踪系统同步,使得相同令牌的错误被提供有相同的错误标识符400。标签402可以由游戏测试者或开发者注释,并且可以包括错误类型的描述、错误的严重性以及用于对快照日志230中的错误进行分类的描述符。游戏相关数据404可以在多维空间中表示为控制输入、视频输出、音频输出和游戏状

态数据的函数。游戏状态数据本身可以在多维空间中表示为各种游戏状态值、对象的坐标值、时间坐标、级别、资产、能力、种子数据等的函数。

[0043] 此外,系统数据406包括执行其中识别到相关联错误的游戏应用程序的系统的的时间相依状态数据。系统数据可以包括系统性能数据,例如CPU时钟和使用率、GPU时钟和使用率、存储器速度和使用率、帧速率、分辨率、视频压缩编解码器和比率、温度指示符以及其他系统性能指示符。网络数据408被预期为包括游戏测试者与托管视频游戏应用程序的服务器系统之间的通信信道的时间相依指示符。这些指示符可以指定等待时间、数据速率容量、抖动和其他通信性质。

[0044] 图5示出了根据一个实施方案的基于云的测试服务器系统500如何实现机器学习以生成在统计上与导致先前识别的错误或新识别的错误相关的新控制输入。服务器系统500包括快照数据库502、自动游戏测试模块504、机器学习模块506和错误实现模块508。多个游戏测试者501被示为与视频游戏交互以识别错误。当识别出错误时,与那些玩家生成的错误有关的玩家生成的快照文件514被记录在快照数据库502中。包括控制输入516的玩家生成的快照文件514经由应用程序接口522被输入到机器学习模块506中。在一些实施方案中,玩家生成的快照文件514可以用作训练数据集528或基准真相。特征提取器524分析玩家生成的快照文件514,确定与特定错误的生成有关的特征空间中的值。特征提取器524可以被定义为分析与控制输入序列、控制输入的时序、控制输入的先前历史等相关联的特征。另外,特征提取器524可以分析与游戏状态值、种子值、系统参数和网络参数相关联的特征。

[0045] 机器学习模块506使用机器学习算法526、训练数据集528、玩家生成的快照文件514来生成错误分类模型532。错误分类模型532可用于识别除了玩家生成的控制输入516之外的机器习得的控制输入530。机器习得的控制输入530在统计上预测为引起错误。可以实现任何合适的机器学习算法526以用于生成错误分类模型532,包括但不限于贝叶斯网络、线性回归、决策树、神经网络、k均值聚类等。错误分类模型532提供了统计上相关的组合、排列和时间相依的控制输入序列,这些可预测为引起玩家生成的错误或新识别的错误。训练数据集528和玩家生成的快照文件514可用于将标签分配给每组传入的控制输入516。标签可以包括“错误”或“不是错误”。

[0046] 在各种实施方案中,每组机器习得的控制输入530由自动游戏测试模块504实现。自动游戏测试模块504被配置为启动多个游戏会话以测试多组机器习得的控制输入514是否发生错误。自动游戏测试模块504包括GUI模块534、差异实现模块536、视频游戏应用程序538、游戏引擎540、操作系统542、游戏引擎分布模块544和错误报告程序546。GUI模块534使开发者能够与自动游戏测试模块504进行对接并且指定自动游戏会话要运行的条件。例如,开发者可以决定在运行自动游戏会话时经由差异实现模块536引入系统或网络参数中的差异(以下称为“差异”)。在一些实施方案中,机器学习模块506还可以提供一组机器习得的差异,其指示差异实现模块536关于如何在自动游戏会话期间引入差异。在其他实施方案中,开发者可以选择通过经由差异实现模块引入差异来对视频游戏应用程序进行负载测试。例如,预想开发者能够指定在执行自动视频游戏会话期间要模拟的帧速率、分辨率、CPU时钟和使用率、GPU时钟和使用率以及存储器使用率参数。另外,开发者可以指定网络差异,例如等待时间、抖动、数据速率容量以及在自动视频游戏会话期间要模拟的其他网络性质。

[0047] 依据要实现的若干机器习得的控制输入530、试图引入的差异以及运行之间参数

的变化或差别的粒度,可以并行地启动任何合适数量的视频游戏会话。在任何情况下,当由自动游戏测试模块504运行自动游戏会话时,视频游戏应用程序538可以被加载有特定游戏状态,所述特定游戏状态紧接在预期的错误发生之前,例如,约5秒至约10分钟、或约10秒至约5分钟、或约30秒至约1分钟之前。当执行视频游戏应用程序538时,以时间相依的方式输入机器习得的控制输入530。另外,可以在经由差异实现模块536实现差异的同时执行自动游戏会话。以这种方式,可以使用一组机器习得的控制输入514来实现多个自动游戏会话,其中每个会话测试不同的差异。例如,第一自动游戏会话可以实现低水平的抖动,而第二自动游戏会话可以实现较高水平的抖动。第三自动游戏会话可能使GPU超频,而第四自动游戏会话可能使类似的GPU降频,等等。

[0048] 此外,当执行自动游戏会话时,错误报告程序546可以在每个游戏会话之上分层以用于检测错误的发生。错误报告程序546可以被预定义为通过自动游戏会话产生的指示错误的游戏状态来自动检测错误。另外,错误报告程序546通过使用图像分析来识别指示所述错误的图像或图像的各部分,来检测与游戏机制或渲染相关联的错误。

[0049] 自动会话的结果(无论是否再现错误)被记录为机器生成的快照文件530。基于错误报告程序546的结果,将机器生成的快照文件518标记为“错误”或“不是错误”。然而,如果错误报告程序546的结果不确定或不完整,或者如果开发者不同意标签,则开发者也可以注释或改变标签。然后,将已标记的机器生成的快照文件518馈送到机器学习模块506中,用于训练或更新错误分类模型532。

[0050] 当错误分类模型532被更新时,可以使用自动游戏测试模块504来生成并测试一组新的机器习得的控制输入530。可以重复经由机器学习模块504进行学习、经由自动游戏测试模块504进行验证(例如,监督)以及完善错误分类模型532的过程,直到错误分类模型532将控制输入分类为预测导致错误或预测不导致错误的置信水平在一定程度上可接受为止。也就是说,例如,可以重复学习、验证和模型完善的过程,直到错误分类模型532能够以即99%的置信区间将新的控制输入分类为导致错误或不导致错误。另外,通过学习、验证和完善的每次迭代,可以预想,机器学习模块506可以生成多组机器习得的控制输入530,所述输入在统计地预测更可能引起所述错误。相反,机器学习模块506在识别预测不导致错误的那些组控制输入方面也得到了改善。

[0051] 尽管到目前为止已经将机器学习模块506描述为根据控制输入来学习,例如来自玩家生成的快照文件514的控制输入,但是机器学习模块506也被预想为基于系统和网络差异来学习。例如,类似于基于错误分类模型532生成机器习得的控制输入530,机器学习模块506还可以生成预测导致错误的机器习得的差异503。因此,机器学习模块506可以检测与错误的发生有关的系统和网络参数的模式,并且可以生成针对系统和网络参数的机器习得的差异503,以经由自动游戏测试模块504进行测试。可以经由差异实现模块536来实现自动游戏测试模块504,使得在自动游戏会话的执行期间模拟、模仿或再现由机器习得的差异503指定的系统和网络条件。可以将错误报告程序关于是否发生错误的结果与机器习得的控制输入530一起记录在机器生成的快照文件518中。例如,错误报告程序546可以用于用“错误”或“不是错误”或“不确定”的标签来标记每次运行。贴标人员稍后可以将标记为“不确定”的自动游戏会话解析为“错误”或“不是错误”。

[0052] 与机器习得的控制输入530一样,来自机器习得的差异503的结果然后可以反馈到

机器学习模块506中,所述机器学习模块用于更新错误分类模型532。学习、测试和更新错误分类模型532的周期以一种能力用于确定导致或近似导致特定错误或者共同导致或近似导致一组错误的一组系统和网络参数。

[0053] 在各种实施方案中,机器学习模块506以一种能力用于通过经验来学习导致特定错误或一组错误的控制输入、系统参数和网络参数的类型的一般性。一般性在错误分类模型532中表示,并且可以预期由错误实现模块508进行外推。错误实现模块508用于将错误分类模型532内的一般性形式化或表示为错误实现条件510,其包括规则548、类别550、游戏情境552和系统情境554。错误实现模块508还被示为包括具有错误实现控制输入556和错误实现系统参数520的错误实现库512。在某种意义上,错误实现模块508是用于触发或实现给定错误的平台。

[0054] 类似于图2的错误再现模块234,错误实现模块508可以回放或再现给定的错误,以供开发者查看。然而,另外,由于错误实现模块508访问机器学习模块506、快照数据库502、错误实现条件510和错误实现库512,因此错误实现模块508还可以向开发者展示导致相同的错误的多种其他方式以及关于导致错误的那些条件的某些限制或边界。错误实现模块508被进一步预期为能够识别并向用户呈现新发现的错误。例如,当机器学习模块506生成机器习得的控制输入530和机器习得的差异503时,它们可以发现先前未识别的错误,此时它们由自动游戏测试模块504来实现。

[0055] 错误分类模型532依据模型或机器学习算法526,以各种方式表示与错误的发生相关联的一般性。例如,一般性可以通过权重、图形连接性、聚类、分布、概率和其他关系来表示。根据这些,可以构建规则548、类别550、游戏情境552和系统情境554。规则548定义了导致特定错误的条件集合。例如,规则(例如“如果在角色A在地图上处于(x,y,z)处时输入了X-向下-向下-三角形,则将出现错误”)可被包括在规则548中。类别550定义了导致特定错误的条件的类别。例如,类别可能限制了发现导致错误的控制输入的范畴。

[0056] 游戏情境552和系统情境554还指定与导致错误的视频游戏或系统或网络中的情境有关的规则或类别。例如,可能存在游戏情境552,在该游戏情境中,当AI角色尝试跳跃时,该角色从世界中掉落。游戏情境552可以帮助识别不一定由玩家的输入产生的错误。另外,系统情境554描述了导致某一错误的系统和网络参数的状态。例如,系统情境554可以指定等待时间超过70毫秒、CPU超频和60FPS渲染的游戏会话将导致玩家从虚拟世界中掉落。当然,在规则548、类别550、游戏情境552、系统情境554中的每一者之间可能存在滑动,使得一个与另一个相关或依赖于另一个。

[0057] 还预期,错误实现模块508包括错误实现库512,所述错误实现库包括错误实现控制输入556和错误实现系统参数520,它们在被执行时将再现各种错误。当正在测试视频游戏应用程序538的新内部版本时,错误实现模块508可以用于针对存在的那些先前由测试者或机器识别的错误来测试新内部版本。以这种方式,开发者能够自动测试先前识别的错误,而游戏测试者501不必尝试从先前的内部版本中重新创建错误。

[0058] 图6示出了差异实现模块536如何创建与自动游戏测试模块504处的自动游戏会话一起执行的差异包606的概念图。输入差异600包括控制输入的序列和时序中的差异,所述控制输入被发送到用于自动游戏会话的自动游戏测试模块504并由其执行。例如,序列600a可以与由游戏测试者识别的玩家生成的控制输入序列相同。因此,序列600a可以用作预期

导致错误的控制项。序列600b与序列600a的不同之处在于，三角形输入在时间上移到了较早的某个时间。输入差异600可以帮助建立错误实现条件510。例如，如果序列600b返回与序列600a相同的错误，则可以将序列600a与序列600b之间的时序差别判定为导致所述错误的不确定起因。另一方面，如果序列600b不像序列600a那样返回错误，则在序列600a和600b之间保持所有其他都等同的情况下，输入三角形的时序可以确定是导致错误的起因。

[0059] 另外，种子数据差异602被示为改变用于初始化自动游戏会话的种子数据。种子数据602a和种子数据602b例如在它们的第一种子和第二种子方面不同。在种子数据中引入差异可以帮助错误实现模块508确定种子数据是否以及如何影响错误的起因。例如，如果种子数据602a和种子数据602b返回不同的错误结果，则可以推测出第一或第二种子或两者与错误的起因有因果关系。在差异实现模块536的第二次迭代中，可以引入种子数据变化以隔离与错误有因果关联的种子。

[0060] 此外，差异实现模块536被配置为将差异引入系统差异604中的系统和网络参数，以在自动测试会话期间进行模拟。在系统差异604中，第一列可以表示等待时间、第二列可以表示抖动、第三列可以表示CPU时钟，且第四列可以表示GPU时钟。系统差异604可用于模仿各种现实世界游戏条件，其中玩家在变化的等待时间性质、抖动性质、CPU和GPU配置以及其他硬件和网络变化的情况下玩视频游戏。系统差异604不仅有助于模拟现实世界游戏条件，而且还可以指出这些条件是否与错误的发生有因果关系。差异实现模块536将输入差异600、种子数据差异602和系统差异604进行组合以生成多个差异包。当由自动游戏测试模块504启动自动游戏会话时，操作系统542可以在自动游戏会话的执行期间实现由差异包606指定的条件集合。

[0061] 图7示出了根据一个实施方案的正在执行多个自动游戏会话700的自动游戏测试模块504的概念性图示。自动游戏会话700中的每一个在与操作游戏引擎440的操作系统542通信的系统708上执行。游戏引擎540又执行用于视频游戏会话700的视频游戏应用程序538。在运行视频游戏会话700时，与视频游戏应用程序538同步实现控制输入702、游戏状态704和系统参数706。例如，控制输入702可以在视频游戏应用程序538的执行期间经由游戏引擎540的输入插口来实现。控制输入702可以是机器习得的控制输入530或玩家生成的控制输入516，如由差异实现模块536修改的。另外，控制输入702可以包括来自多玩家游戏的一个以上虚拟玩家的控制输入。例如，玩家生成的快照文件514可以包括测试期间每个玩家的控制输入序列516。机器学习模块506可操作以生成机器习得的控制输入530，所述机器习得的控制输入针对仅一个玩家或一个以上玩家改变控制输入。因此，自动游戏测试模块504被预期为测试与其中某类玩家交互导致错误的多玩家游戏和情境相关联的错误。

[0062] 例如种子数据的游戏状态704也可以经由游戏引擎来实现，使得视频游戏应用程序被预先定义的种子数据初始化。系统参数706（例如，由差异实现模块536修改的机器学习差异503）可以在操作系统542级别或系统708级别实现。

[0063] 在自动游戏会话700的每次运行期间，在操作系统542级别或游戏引擎540级别执行错误报告程序546。错误报告程序546被配置为自动监测和检测预定义错误或先前未识别的错误的发生。自动游戏会话700的每次运行都产生对应的自动游戏会话报告710，所述报告包括与是否发生错误相关联的机器生成的快照文件。以这种方式，可以启动任意数量的自动游戏会话700以更好地理解导致错误的原因的条件和边界。了解条件及其边界使开发

者能够更有效地解决错误。

[0064] 游戏应用程序(例如视频游戏应用程序538)通常将需要服务于多个不同功能的游戏引擎。典型的游戏引擎具有许多子部件,以用于处理游戏环境的各个方面或特征并定义例如响应于玩家输入其应如何表现。这些子部件中的一些,或将被称为“游戏引擎节点”的那些部件,包括时序节点、物理节点、人工智能(AI)节点、游戏逻辑节点、游戏机制节点、渲染节点(其本身可能细分为多个节点)、地图节点、动画节点、资产管理节点、网络节点、通信节点、控制输入节点、聊天节点以及其他节点。渲染节点可以是指各种渲染功能(例如,像DirectX®或OpenGL®之类的图形API的典型功能),包括镜头转换、投影、裁剪、照明、着色、平滑化等。另外,游戏引擎可以提供有益于玩游戏过程的附加服务,例如社交网络交互节点、游戏帮助节点、游戏内资源呈现节点、玩游戏过程共享或广播节点等。

[0065] 在某些游戏引擎实施方案中,游戏引擎可以由单个计算节点(例如计算机、服务器、控制台或其他装置)提供服务。在其他实施方案中,计算节点可以是由服务器为游戏引擎部署的虚拟机,以及被部署为服务于那里运行的其他游戏引擎和游戏应用程序的其他虚拟机。因此,游戏引擎子部件中的每一个都由一个计算节点(例如,一个服务器、虚拟机、台式计算机或控制台)执行。游戏引擎与计算节点的这种一对一或多对一架构可能无法提供游戏引擎所需的性能,也无法提供计算节点的使用所需的效率,因为游戏引擎在其计算需求方面可能是弹性的,而不是固定的。也就是说,游戏引擎在任何给定时间可能具有与同一游戏内的任何其他给定时间不同的处理、图形、存储器和网络需求。

[0066] 例如,在大型多玩家在线角色扮演游戏(MMORPG)中,相关联的游戏引擎的计算需求可取决于游戏中发生的动作类型(例如,玩家是否彼此参与)、玩家数量(例如目前是否有10个玩家,但不久的将来会有100个玩家)以及其他因素。当向游戏引擎的一个计算节点提供资源(例如,在服务器内)时,相关联的硬件可能会具有狭窄的窗口,在所述窗口中,其像所需那样有效运行,同时具有所需的性能。例如,当玩家很少且在MMORPG中不交互时,相关联的计算节点可能未得到充分利用,而当玩家众多且相互参与时,相关联的计算节点可能表现不佳,从而导致服务质量降低、滞后、渲染质量问题、差错。

[0067] 这些考虑因素与自动游戏测试的背景同样重要,在自动游戏测试中,可以一次实例化数百个或数千个或更多游戏会话和相关联的游戏引擎。例如,游戏引擎对多个自动游戏会话进行实例化的计算需求将依据游戏环境和其中的对象的行为以时间相依的方式变化,例如在一个时间收集资源,而在另一时间与AI角色战斗。需求。此外,计算需求的类型可以根据时间或游戏状态而变化。例如,游戏引擎的处理、图形、存储器和网络需求可能会在如下场景之间有所变化,在所述场景中,一名球员正在罚球,而在错过罚球后,对方球队会抢篮板并将球推上球场。此外,自动游戏会话可以用于通过在运行期间同时实现多个玩家的控制输入来使多玩家玩游戏过程自动化。多玩家输入的存在还例如根据玩家的数量和他们的交互水平而在对应的游戏引擎的计算需求中引起进一步的弹性。因此,改进的游戏引擎架构被预期为执行针对自动游戏测试模块504的自动游戏会话。

[0068] 图8示出了根据一个实施方案的利用分布式游戏引擎800来执行多个自动游戏会话700的自动游戏测试模块504的架构的图。视频游戏应用程序538在具有多个游戏引擎节点800a和游戏引擎管理器800b的分布式游戏引擎800上执行。每个游戏引擎节点800a被预期为整体上服务于分布式游戏引擎800的子部件或子部件的一部分。在一些实施方案中,游

戏引擎节点800a可以基于其为分布式游戏引擎800提供的功能来预定义。例如,可能存在游戏引擎节点800a,其被定义或专用于执行与游戏逻辑、游戏机制、游戏时序、AI、物理、输入映射、脚本、网络、音频、图形、动画、资产管理以及游戏内服务中的每一者有关的功能。例如,对于自动游戏会话700a,游戏引擎节点1 (GEN 1) 可以专用于处理例如游戏逻辑子部件/功能,而不处理与图形或资产管理有关的任何功能。同时,游戏引擎节点2 (GEN 2) 可以被定义为专门处理AI,其中GEN 3被定义为用于物理,GEN 4用于脚本,GEN 5用于图形,GEN 6用于脚本,且GEN n用于例如实时游戏广播的服务。

[0069] 每个游戏引擎节点800a通过游戏引擎管理器800a与机架804的服务器802内的计算节点进行匹配,这取决于每个游戏引擎节点800a的计算需求。游戏引擎节点800a可以经由UDP或TCP或某一其他协议与计算节点通信。服务器802和机架804可以位于数据中心等处。游戏引擎管理器800b用作游戏引擎节点800a与分配的计算节点之间的总线。游戏引擎管理器800b执行与发送和接收数据、缓冲、路由、线程、排队、中继、拼接等有关的各种任务。以这种方式,游戏引擎管理器800b确保每个游戏引擎节点800a的操作由期望的计算节点执行,并且所得的返回值或其他结果被递送回适当的的游戏引擎节点800a,以实现到视频游戏应用程序538中。此外,游戏引擎管理器800a管理游戏引擎节点800a之间的通信,以确保整个视频游戏应用程序的正常运行。

[0070] 如图8中所示,游戏引擎节点1GEN 1被分配给计算节点806,所述计算节点806与位于机架1的服务器1上的硬件CPU相关联。例如,游戏引擎管理器可能因其适用于处理游戏逻辑的计算需求而采用了计算节点806。

[0071] 同时,自动游戏会话700b同样在具有多个游戏引擎节点GEN 1至n的分布式游戏引擎800上执行,每个游戏引擎节点被分配为处理分布式游戏引擎800的特定功能或子部件。游戏会话700n也是如此。例如,类似于自动游戏会话700a的GEN 1,自动游戏会话700b的GEN 1和自动游戏会话700n的GEN 1的操作都被分配给机架1的服务器1的计算节点806。因此,计算节点806执行由GEN 1、GEN 2和GEN 3中的每一者请求的过程。针对自动游戏会话700a、700b和700n中的每一者的GEN 2被分配给计算节点808,计算节点808是配备有CPU 2、GPU 1和16GB RAM的虚拟机VM 1。游戏引擎管理器800b可能已经基于GEN 2的预期需求识别了VM 1,所述GEN 2例如用于执行AI。在一些实施方案中,游戏引擎管理器800b可以自己请求部署VM 1,或者自动游戏测试模块504可以做出请求。在一些实施方案中,如果对于分配给计算节点的每个游戏引擎节点,请求由计算节点执行的操作基本上类似,则计算节点可以潜在地执行一次所请求的操作,并将结果返回给每个游戏引擎节点,而不是针对每个游戏引擎节点都执行一次所请求的操作。

[0072] 每个自动游戏会话700的GEN 3被示为与计算节点812相关联,所述计算节点812与GPU 1相关联,而GEN 4与计算814相关联,所述计算814与GPU 2相关联。GPU 1和GPU 2可以具有不同的配置。例如,GPU 1可能比GPU 2具有更多的核心、更高的时钟、更多的存储器或更高的存储器带宽。在这种情况下,如果GEN 3要求更大量的计算或更复杂的计算,则GEN 3的操作可能已分配给GPU 1。GEN 4的操作可能已分配给GPU 2,因为GEN 4可能需要较少的计算或不太复杂的计算。

[0073] GEN 5被示为分配给计算节点810,所述计算节点810是配备有CPU 3、GPU 2和8GB RAM的虚拟机VM 2。GEN 5可能已由游戏引擎管理器800b分配给计算节点810,以匹配GEN 5

的计算需求。自动游戏会话700的GEN 6被示为分配给容器化计算节点816,所述容器化计算节点816包括与机架3的服务器3上的主机操作系统对接的容器1至N。特别地,自动游戏会话700a的GEN 6被分配给容器1,自动游戏会话700b的GEN 6被分配给容器2,而自动游戏会话700n的GEN 6被分配给容器N。容器1至N中的每一个是软件和相依性的包含单元,例如,所述单元包括用于处理相关联的游戏引擎节点的操作的软件。每个容器都在容器运行时上运行,所述容器运行时与服务器的操作系统对接。从这个意义上说,操作系统是针对每个容器虚拟化的,而硬件是针对虚拟机虚拟化的。

[0074] 图9A和图9B概念性地示出了根据各种实施方案的机器学习模块506、自动游戏测试模块504和错误实现模块508如何一起工作以识别可以实现先前或新识别的错误的条件集合。错误实现条件是指与给定错误的发生有因果关系的条件的范围或范畴。条件是指控制输入、系统参数、网络参数和游戏状态,当在游戏应用程序的执行期间导致这些状态存在或共存时,它们单独或共同地导致给定的错误。因此,可以在多维空间中将错误实现条件概念化为导致或可能导致错误的条件的所有组合的集合。本文中,错误1实现条件900将是指在特定的系统和网络参数下并且在某些游戏状态下执行时导致或可能导致错误1的控制输入组合的集合。机器学习模块506、自动游戏测试模块504和错误实现模块508被预期为协同工作以识别、限制或近似错误1的实现条件,以帮助开发者了解错误的根本原因,从而获得更高的质量和更有效的修复。错误1实现条件900可以在规则、类别和情境方面在错误实现模块508中形式化,以传达给开发者。为了清楚起见,以下将关于用于填充错误1实现条件900的机器学习得的控制输入进行描述。然而,这些原理以类似的力度适用于使用机器学习得的系统和网络参数以及游戏状态数据来集结错误1实现条件的其余部分。

[0075] 当一个或多个游戏测试者识别到错误时,玩家生成的控制输入902将被记录在快照文件中,并由机器学习模块506处理。机器学习模块506使用错误分类模型来生成机器学习得的控制输入904,然后由自动游戏测试模块504实现所述机器学习得的控制输入以检测是否存在错误1。本文中,一部分机器学习得的控制输入904导致错误1。结果被反馈到机器学习模块504中,以更新错误分类模型并生成一组新的机器学习得的控制输入906。本文中,如可以预期的,与第一组机器学习得的控制输入904相比,第二组机器学习得的生成的控制输入906的更大部分会导致错误1。针对机器学习得的控制输入908重复学习和测试过程,依此类推,直到例如没有找到导致错误的控制输入的新集合为止。错误1实现条件900然后由错误实现模块508处理,以提取可归因于错误1实现条件900的任何规则、类别或情境。

[0076] 在图9B中,学习和测试过程在错误1实现条件900的分析期间发现了新的错误2。例如,机器学习得的控制输入904由自动游戏测试模块504进行测试,并被发现会触发错误1和错误2两者。在这种情况下,除了集结错误1实现条件900之外或与集结错误1实现条件900并行地,机器学习模块506、自动游戏测试模块504和错误实现模块508还可以对错误2实现条件901执行相同的操作。机器学习得的控制输入906至914由机器学习模块506生成,并由自动游戏测试模块904进行测试。错误2实现条件901的所得范围然后由错误实现模块508形式化为规则、类别和情境,这些将具有针对错误1和错误2两者的那些规则、类别和情境。

[0077] 图10示出了根据一个实施方案的通过混沌或霰弹枪测试发现新错误的方法。除了为已知错误或玩家识别的错误定义错误实现条件之外,本文提出的系统和方法还可以被预期为以自动方式检测未识别的错误。在一些实施方案中,提供混沌测试或霰弹枪测试以发

现未发现的错误。本文中,混沌测试模块1001生成多组混沌输入1002a至1002g。混沌输入1002a至1002g中的每一个可以包括一组随机地、偶估地或混沌地定义的输入。在其他实施方案中,混沌输入1002a至1002g可以是设计为加强游戏应用程序的已知或可知的输入序列。在任何情况下,在混沌输入1002a至1002g中,混沌输入1002c被示为导致如自动游戏测试模块504所检测到的错误3。自动游戏测试模块504将结果中继到机器学习模块506,所述机器学习模块506生成更具针对性的机器学习得的输入1004a和1004b。学习和测试过程继续进行,直到可以定义错误3实现条件1000的范围为止。在一些实施方案中,混沌测试模块1001可以继续生成混沌输入,以找到引起错误3或其他错误的其他方式。因此,本文中提出的系统方法进一步使得能够自动识别先前未知的错误。使用混沌测试模块1001、机器学习模块506、自动游戏测试模块504和错误实现模块508来识别先前未知的错误的过程可以作为服务(例如,测试即服务,TaaS)提供给开发者,如下文更详细地讨论。

[0078] 图11示出了在游戏的新内部版本或版本中针对先前识别的错误进行自动测试的概念图。假设错误实现条件1100的圆圈表示用于第一内部版本的控制输入、游戏状态、系统和数据参数空间中的错误实现条件1100。进一步假设错误实现模块已经定义了四个类别的控制输入1101a至1101d,这些输入与大部分错误实现条件1100有效地重叠并且触发第一内部版本的错误。如果控制输入1101a至1101d中的每一个再次导致第二内部版本的错误,则可能是所述错误未得到修复的情况。替代地,如果仅控制输入1101a至1101d的一部分导致了所述错误,则可能是所述错误被部分修复的情况。否则,如果当在第二内部版本中实现控制输入1101a至1101d时所述控制输入都未导致所述错误,则可能是所述错误已修复的情况。因此,当构造了第二内部版本时,可以以快速且有效的方式自动测试来自先前内部版本的先前错误。

[0079] 图12示出了根据一个实施方案的作为测试即服务(TaaS)的一部分如何将针对第一游戏已知的错误实现条件应用于第二游戏。在图12中,第一游戏的错误实现条件1100是已知的,并且可以使用控制输入1101a至1101d来再现。作为测试即服务过程的一部分,控制输入1101a至1101d可以应用于第二游戏标题,所述第二游戏标题可以由与第一游戏标题不同的开发者创建。在图12中,控制输入1101a至1101c被示为部分或完全地导致错误,而控制输入1101d被示为在游戏标题2的错误实现条件之外,且因此不导致所述错误。

[0080] 图13是根据一个实施方案的可用于测试即服务(TaaS)的平台的概念性图示。TaaS被预期为使游戏开发者能够测试视频游戏中的错误并验证错误修复,而不需要人类游戏测试者团队。本文中,开发者1300将游戏简单地拖放到TaaS服务器1303中,所述服务器包括第一内部版本的游戏标题1302。游戏标题1302被执行为在适当的游戏引擎540上运行的视频游戏应用程序538。游戏引擎540可以是以图8所示的方式实现的分布式游戏引擎。为游戏引擎540提供适当的操作系统540。如果打算在不同的游戏引擎、不同的操作系统或其各版本上执行游戏标题1302,则自动游戏测试模块504可以针对每种类型的游戏引擎540和操作系统542及其所需的组合对应地启动视频游戏应用程序538的自动视频游戏会话。自动游戏会话可以并行或以其他方式运行。

[0081] 可以由自动游戏测试模块504以许多方式来测试游戏标题1302。例如,TaaS可以经由混沌测试模块1001来提供视频游戏应用程序538的混沌测试。通过混沌测试识别出的任何错误都将记录到快照数据库502中。此外,如果开发者1300先前已经识别出错误并且希望

了解导致所述错误的情况,则他们可以在TaaS上执行时与视频游戏应用程序538交互以手动地导致所述错误。作为这种交互的结果,将生成快照文件,并且导致错误的控制输入将被记录到快照数据库502。机器学习模块506然后可以生成也导致错误的新的机器习得的控制输入530和机器习得的差异503。

[0082] 此外,可以针对错误实现模块508的错误实现条件510以及错误实现控制输入556和错误实现系统参数520来测试视频游戏应用程序。错误实现模块508可以为具有与游戏标题1302类似的特征的不同视频游戏或为具有共同的游戏引擎的不同游戏提供已知会导致该视频游戏的先前内部版本的错误的各种控制输入以及系统和网络参数。此外,TaaS服务器1303可以与一个或多个人类游戏测试者进行通信,以发现错误,这些错误的原始测试数据已被生成为快照数据库502处的快照文件。

[0083] 以上述方式识别的任何错误与导致错误的控制输入1302(以及系统和网络参数)一起记录在快照数据库中。这些将与开发者1300共享。此外,可以在回放模式302、自动再现模式304或手动再现模式306中的任一者下经由错误再现模块234为开发者1300再现这些错误。以这种方式,开发者可以在例如错误再现GUI 300之类的GUI中查看任何已发现的错误以及与所述错误相关联的源代码、游戏状态、控制输入以及视频分量。

[0084] 机器学习模块506访问快照数据库502,以用于生成预测导致先前识别的错误或新的错误的机器习得的控制输入530和机器习得的差异503。然后,在自动游戏测试模块504中测试这些机器习得的控制输入530和机器习得的差异503,以供错误报告程序546检测是否发生了先前识别的错误或先前未识别的错误。自动游戏测试模块504的结果再次被记录在快照数据库502中。本文中,再次,可以将机器习得的控制输入530和机器习得的差异503提供给开发者1300,使得可以将导致错误的附加控制输入以及系统和网络参数告知开发者。此外,开发者可能会被告知新的错误以及导致它们的控制输入以及系统和网络参数。

[0085] 可以重复学习和测试过程以完善错误分类模型532,使得错误实现模块508可以由规则548、类别550、游戏情境552和系统情境554来提取与错误的起因相关联的一般性。然后,可以将错误实现条件510提供给开发者1300,这有助于识别导致错误的条件。这使得开发者1300能够更好地理解错误的根本原因,使得可以应用更高质量的修复。在开发者1300尝试修复游戏标题1304的第二内部版本的错误之后,可以再次将游戏标题1304拖放到TaaS服务器1303中以进行验证和测试。为了查看错误是否已修复,可以在自动游戏测试模块504上测试已知导致先前内部版本(即第一内部版本)中的错误的某些控制输入1302。可以从错误实现模块508生成附加控制输入和系统参数,这些附加控制输入和系统参数虽然不一定被测试为导致先前内部版本的错误,但是预期具有使用规则548、类别550、游戏情境552和系统情境554。以这种方式,使开发者1300能够快速测试具有错误修复的新内部版本的操作。

[0086] 图14示出了可用于执行本公开的各种实施方案的各方面的示例装置1400的部件,所述示例装置例如服务器系统200、服务器系统500、服务器802中的一个或TaaS服务器1300。该框图示出了装置1400,所述装置1400可以结合或可以是适合于实践本公开的实施方案的个人计算机、视频游戏控制台、个人数字助理、服务器或其他数字装置。装置1400包括用于运行软件应用程序和可选地操作系统的中央处理单元(CPU) 1402。CPU 1402可以由一个或多个同构或异构处理核心构成。例如,CPU 1402是具有一个或多个处理核心的一个

或多个通用微处理器。还可以使用具有特别适于例如自动游戏测试、机器学习操作和错误再现过程等高度并行且计算密集的应用的微处理器架构的一个或多个CPU来实现其他实施方案。装置1400可以在玩游戏片段的玩家本地(例如,游戏控制台),或者远离玩家(例如,后端服务器处理器)。

[0087] 存储器1404存储由CPU 1402使用的应用程序和数据。存储装置1406为应用程序和数据提供非易失性存储装置和其他计算机可读介质并且可以包括固定磁盘驱动器、可移除磁盘驱动器、快闪存储器装置和CD-ROM、DVD-ROM、蓝光光碟、HD-DVD或其他光学存储装置,以及信号传输和存储介质。用户输入装置1408将用户输入从一个或多个用户传达到装置1400,所述装置的示例可以包括键盘、鼠标、操纵杆、触摸板、触摸屏、静态或视频记录器/相机、用于辨识手势的跟踪装置和/或传声器。网络接口1410允许装置1400经由电子通信网络与其他计算机系统通信,并且可以包括在局域网和例如互联网的广域网上的有线或无线通信。音频处理器1412适于从由CPU 1402、存储器1404和/或存储装置1406提供的指令和/或数据生成模拟或数字音频输出。装置1400的部件,包括CPU 1402、存储器1404、数据存储装置1406、用户输入装置1408、网络接口1410和音频处理器1412,经由一个或多个数据总线1422连接。

[0088] 图形子系统1420进一步与数据总线1422和装置1400的部件连接。图形子系统1420包括图形处理单元(GPU) 1416和图形存储器1418。图形存储器1418包括显示存储器(例如帧缓冲器),所述显示存储器用于存储输出图像的每个像素的像素数据。图形存储器1418可以集成在与GPU 1416相同的装置中、作为单独的装置与GPU 1416连接和/或实现在存储器1404内。像素数据可以直接从CPU 1402提供到图形存储器1418。替代地,CPU 1402可向GPU 1416提供定义期望输出图像的数据和/或指令,GPU 1416根据所述数据和/或指令生成一个或多个输出图像的像素数据。定义期望输出图像的数据和/或指令可以存储在存储器1404和/或图形存储器1418中。在实施方案中,GPU 1416包括3D呈现能力,所述3D呈现能力用于根据指令和数据生成输出图像的像素数据,所述指令和数据定义场景的几何形状、照明、着色、纹理化、动作和/或相机参数。GPU 1416还可以包括能够执行着色器程序的一个或多个可编程执行单元。

[0089] 图形子系统1420周期性地从图形存储器1418输出图像的像素数据,以显示在显示装置1414上。显示装置1414可以是能够响应于来自装置1400的信号来显示视觉信息的任何装置,包括CRT、LCD、等离子体和OLED显示器。装置1400可以向显示装置1414提供例如模拟或数字信号。

[0090] 应当注意,在广阔的地理区域上传递的访问服务(诸如提供对当前实施方案的游戏的访问)经常使用云计算。云计算是一种计算方式,其中动态可扩展和通常虚拟化的资源通过互联网作为服务提供。用户不需要成为支持他们的“云”中技术基础设施方面的专家。云计算可以分为不同的服务,例如基础设施即服务(IaaS)、平台即服务(PaaS)和软件即服务(SaaS)。云计算服务通常提供在线的从web浏览器访问的常用应用程序(例如视频游戏),而软件和数据则存储在云中的服务器上。基于计算机网络图中互联网的描绘方式,术语云被用作互联网的隐喻,并且是它所隐藏的复杂基础设施的抽象概念。

[0091] 互联网上进行的大多数视频游戏都是经由与游戏服务器的连接来操作的。通常,游戏使用专用的服务器应用程序,所述服务器应用程序从玩家收集数据并将其分发给其他

玩家。用户使用客户端装置访问远程服务,客户端装置至少包括CPU、显示器和I/O。客户端装置可以是PC、移动电话、上网本、PDA等。在一个实施方案中,在游戏服务器上执行的网络辨识客户端使用的装置类型并调整所采用的通信方法。在其他情况下,客户端装置使用标准的通信方法(例如html)来通过互联网访问游戏服务器上的应用程序。

[0092] 本公开的实施方案可以用包括手持式装置、微处理器系统、基于微处理器的或可编程的消费型电子产品、小型计算机、大型计算机等各种计算机系统配置来实践。本公开也可以在分布式计算环境中实践,其中由通过有线或无线网络链接的远程处理装置执行任务。

[0093] 应理解,可以针对特定平台和特定相关联的控制器装置开发给定的视频游戏或游戏应用程序。然而,当通过本文中呈现的游戏云系统使此类游戏可用时,用户可能正在使用不同的控制器装置来访问视频游戏。例如,可能已经为游戏控制台及其相关联的控制器开发了游戏,而用户可能正在使用键盘和鼠标从个人计算机访问基于云的游戏版本。在这种情况下,输入参数配置可以定义从用户的可用控制器装置(在这种情况下为键盘和鼠标)生成的输入到执行视频游戏可接受的输入的映射。

[0094] 在另一个示例中,用户可以经由平板计算装置、触摸屏智能手机或其他触摸屏驱动装置来访问云游戏系统。在这种情况下,客户端装置和控制器装置一起集成在同一装置中,其中输入通过检测到的触摸屏输入/手势来提供。对于这种装置,输入参数配置可以定义与视频游戏的游戏输入相对应的特定触摸屏输入。例如,在视频游戏的运行期间,按钮、方向盘或其他类型的输入元件可能被显示或覆盖,以指示用户可以触摸以生成游戏输入的触摸屏上的位置。例如在特定方向上的划动或特定触摸动作等手势也可以被检测为游戏输入。在一个实施方案中,可以(例如)在开始视频游戏的玩游戏之前向用户提供指示如何通过触摸屏提供输入以便玩游戏的指南,以便使用户适应在触摸屏上操作控件。

[0095] 在一些实施方案中,客户端装置用作控制器装置的连接点。也就是说,控制器装置经由无线或有线连接与客户端装置通信,以将输入从控制器装置传输到客户端装置。客户端装置可以依次处理这些输入,然后经由网络(例如,经由例如路由器之类的本地联网装置访问)将输入数据传输到云游戏服务器。然而,在其他实施方案中,控制器本身可以是联网装置,具有经由网络直接将输入传达到云游戏服务器的能力,而无需首先通过客户端装置传达这样的输入。例如,控制器可以连接到本地联网装置(例如上述路由器)以向云游戏服务器发送数据并从云游戏服务器接收数据。因此,尽管可能仍然需要客户端装置接收来自基于云的视频游戏的视频输出并将其渲染在本地显示器上,但是通过允许控制器通过网络直接将输入发送到云游戏服务器从而绕过客户端装置,可以减少输入等待时间。

[0096] 在一个实施方案中,联网控制器和客户端装置可以被配置为将某些类型的输入从控制器直接发送到云游戏服务器,以及经由客户端装置发送其他类型的输入。例如,其检测不依赖于控制器本身以外的任何附加硬件或处理的输入可以经由网络从控制器直接发送到云游戏服务器,从而绕过客户端装置。这样的输入可以包括按钮输入、操纵杆输入、嵌入式运动检测输入(例如,加速度计、磁力计、陀螺仪)等。然而,利用附加硬件或需要客户端装置进行处理的输入可以由客户端装置发送到云游戏服务器。这些可能包括从游戏环境中捕获的视频或音频,这些视频或音频可以在发送到云游戏服务器之前由客户端装置进行处理。另外,来自控制器的运动检测硬件的输入可由客户端装置结合所捕获的视频进行处理,

以检测控制器的位置和运动,随后客户端装置会将所述位置和运动传达到云游戏服务器。应理解,根据各种实施方案的控制器装置还可以从客户端装置或直接从云游戏服务器接收数据(例如,反馈数据)。

[0097] 应理解,可以使用本文公开的各种特征将本文定义的各种实施方案组合或组装成具体的实现方式。因此,提供的示例仅仅是一些可能的示例,而不是对通过组合各种要素以定义更多的实现方式而可能的各种实现方式的限制。在一些示例中,一些实现方式可以包括更少的要素,而不脱离所公开的或等同的实现方式的精神。

[0098] 本公开的实施方案可以用包括手持式装置、微处理器系统、基于微处理器的或可编程的消费型电子产品、小型计算机、大型计算机等各种计算机系统配置来实践。还可以在分布式计算环境中实践本公开的实施方案,在所述分布式计算环境中,通过基于有线或无线网络链接的远程处理装置执行任务。

[0099] 虽然以特定的顺序描述了方法操作,但是应当理解,可以在操作之间执行其他内务操作,或者可以调整操作,使得它们在稍微不同的时间发生,或者可以将操作分布在系统中,只要以期望的方式执行用于生成经修改的游戏状态的遥测和游戏状态数据的处理,所述系统便允许以与处理相关联的各种间隔发生处理操作。

[0100] 一个或多个实施方案也可以被制造为计算机可读介质上的计算机可读代码。计算机可读介质是可以存储数据的任何数据存储装置,该数据随后可以由计算机系统读取。计算机可读介质的示例包括硬盘驱动器、网络附接存储装置(NAS)、只读存储器、随机存取存储器、CD-ROM、CD-R、CD-RW、磁带以及其他光学和非光学数据存储装置。所述计算机可读介质可以包括分布在网络耦合的计算机系统上的计算机可读有形介质,使得以分布式方式存储和执行计算机可读代码。

[0101] 尽管为了清楚理解的目的而略微详细地描述了前述实施方案,但很显然,可以在所附权利要求的范围内实践某些变化和修改。因此,本实施方案被认为是说明性的而非限制性的,并且所述实施方案不限于本文给出的细节,而是可以在所附权利要求的范围和等效范围内进行修改。

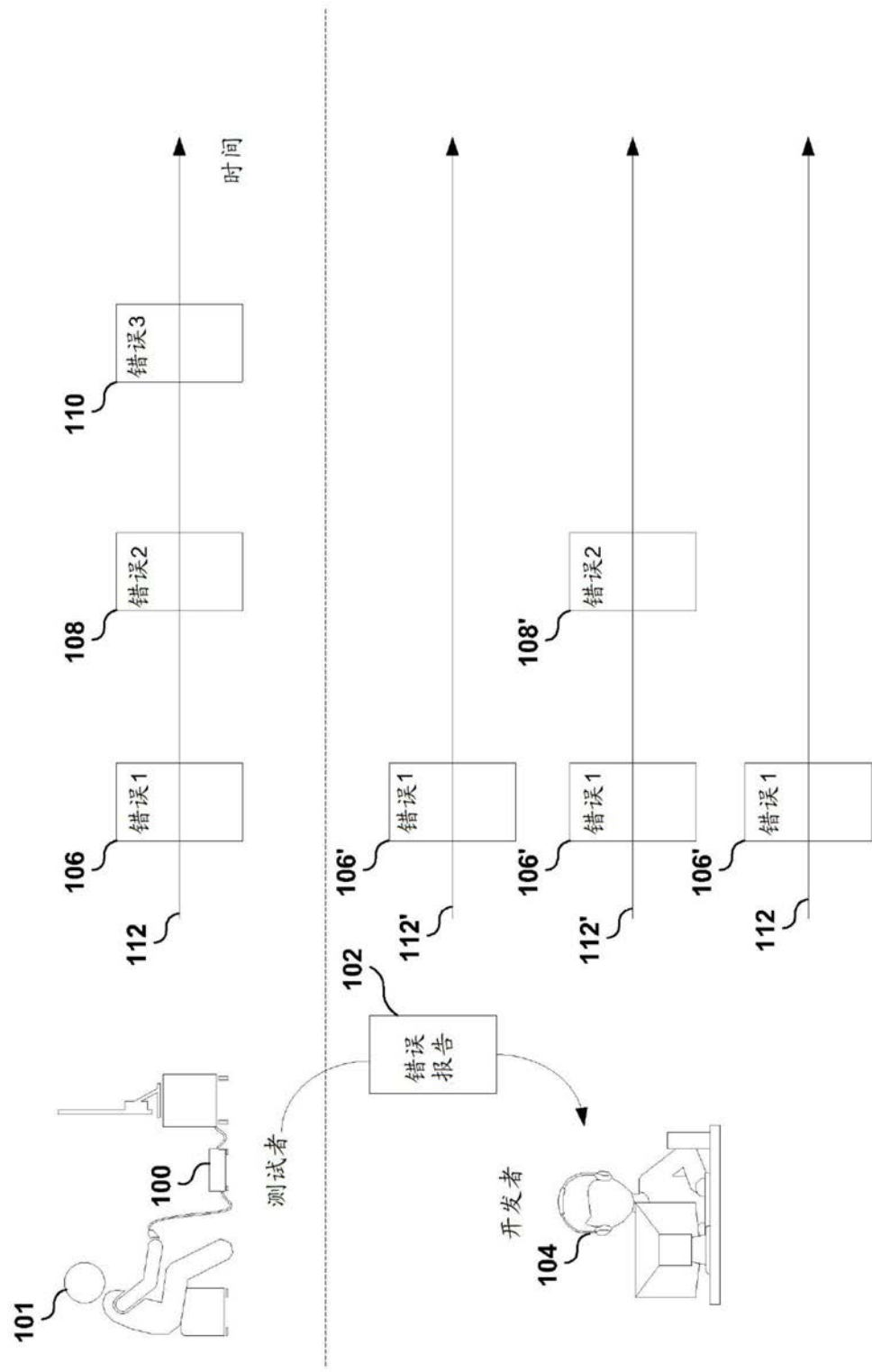


图1

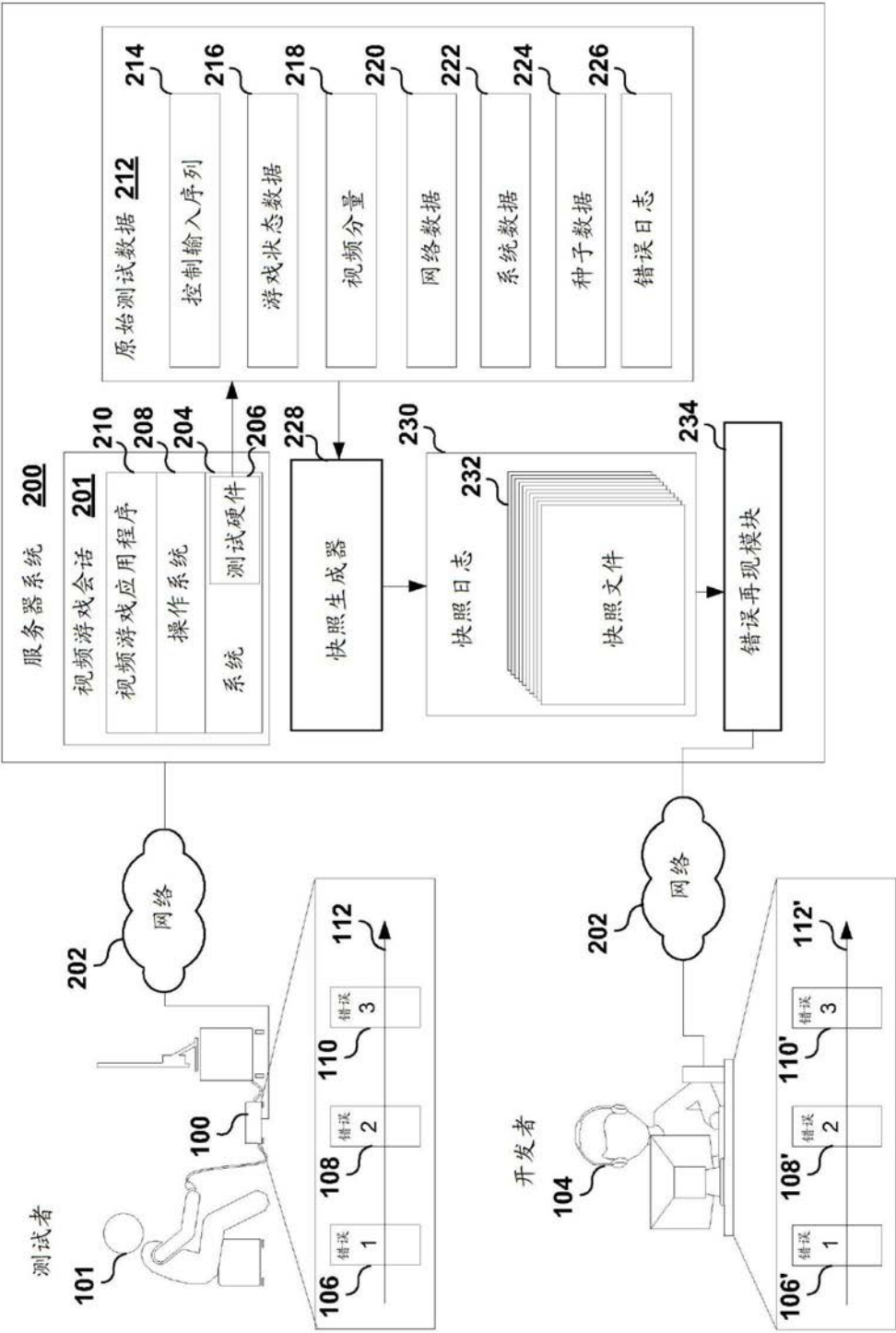


图2

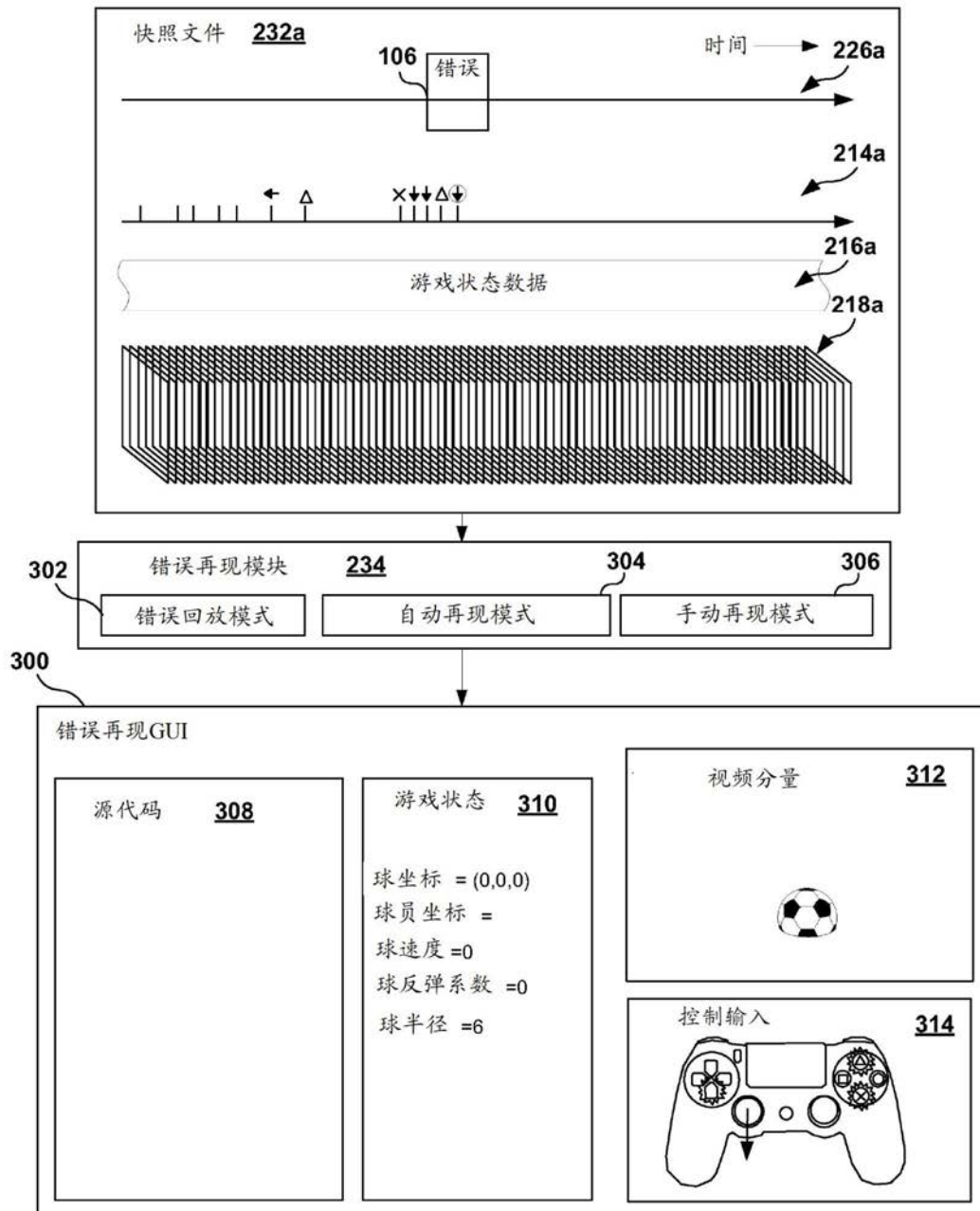


图3

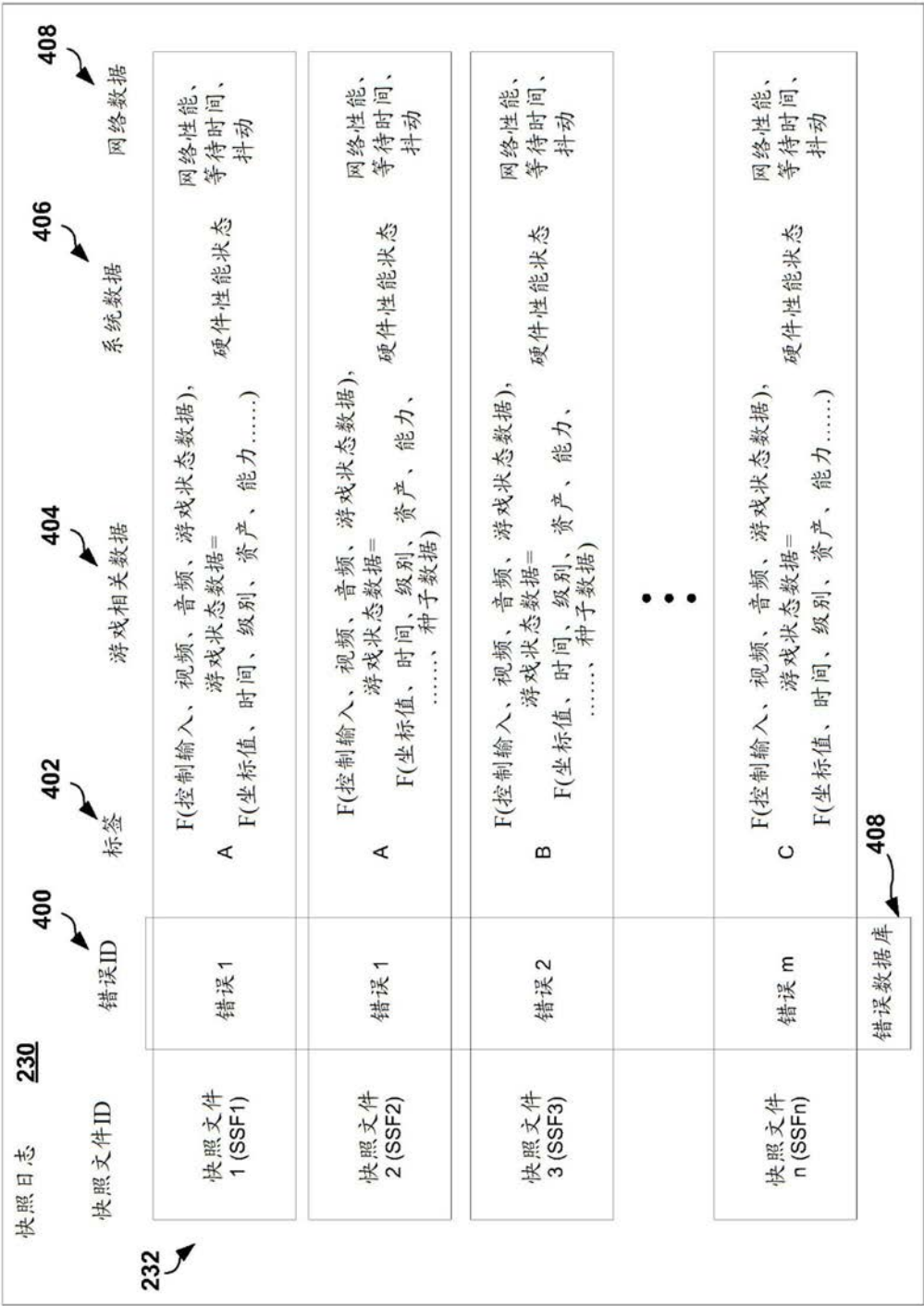


图4

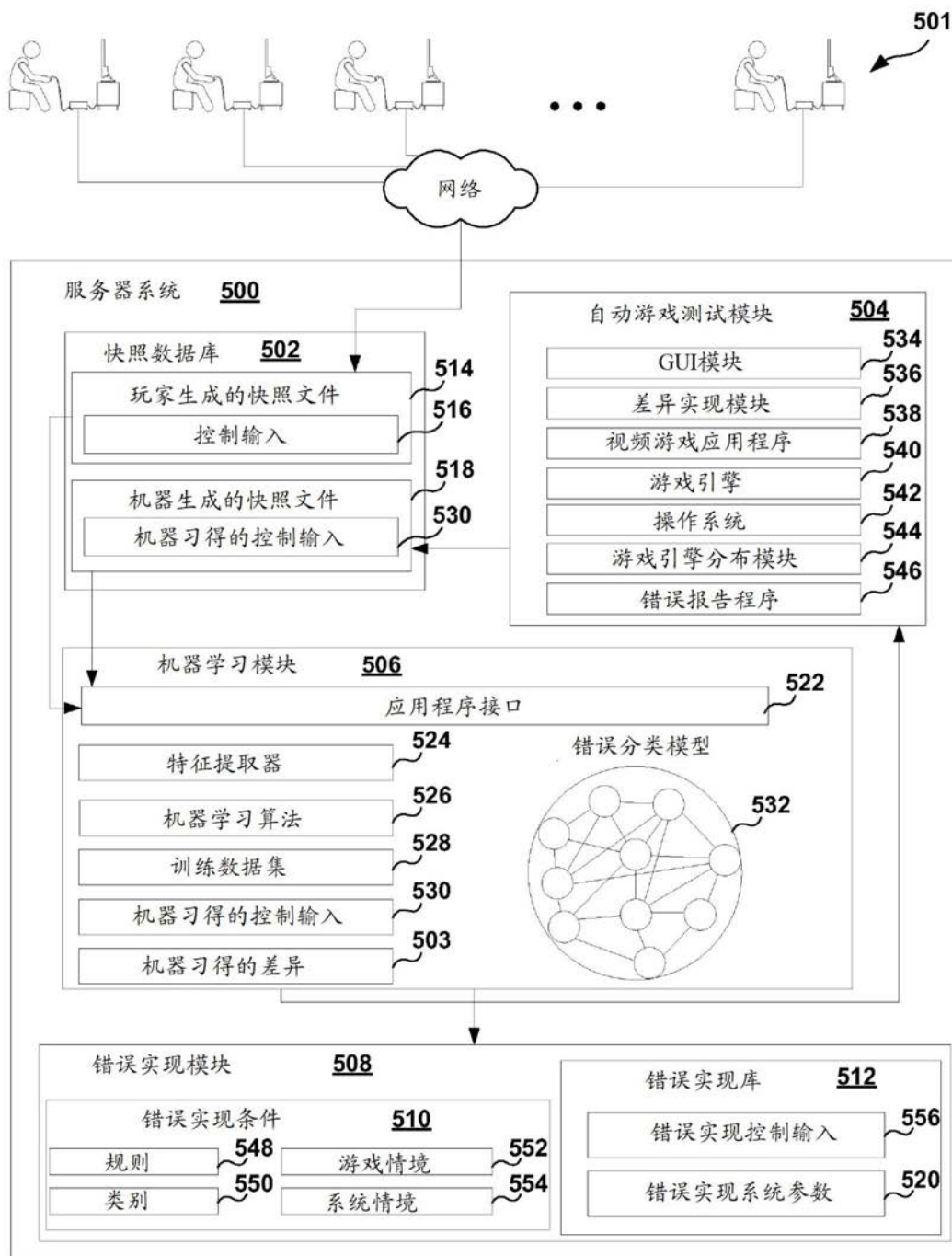


图5

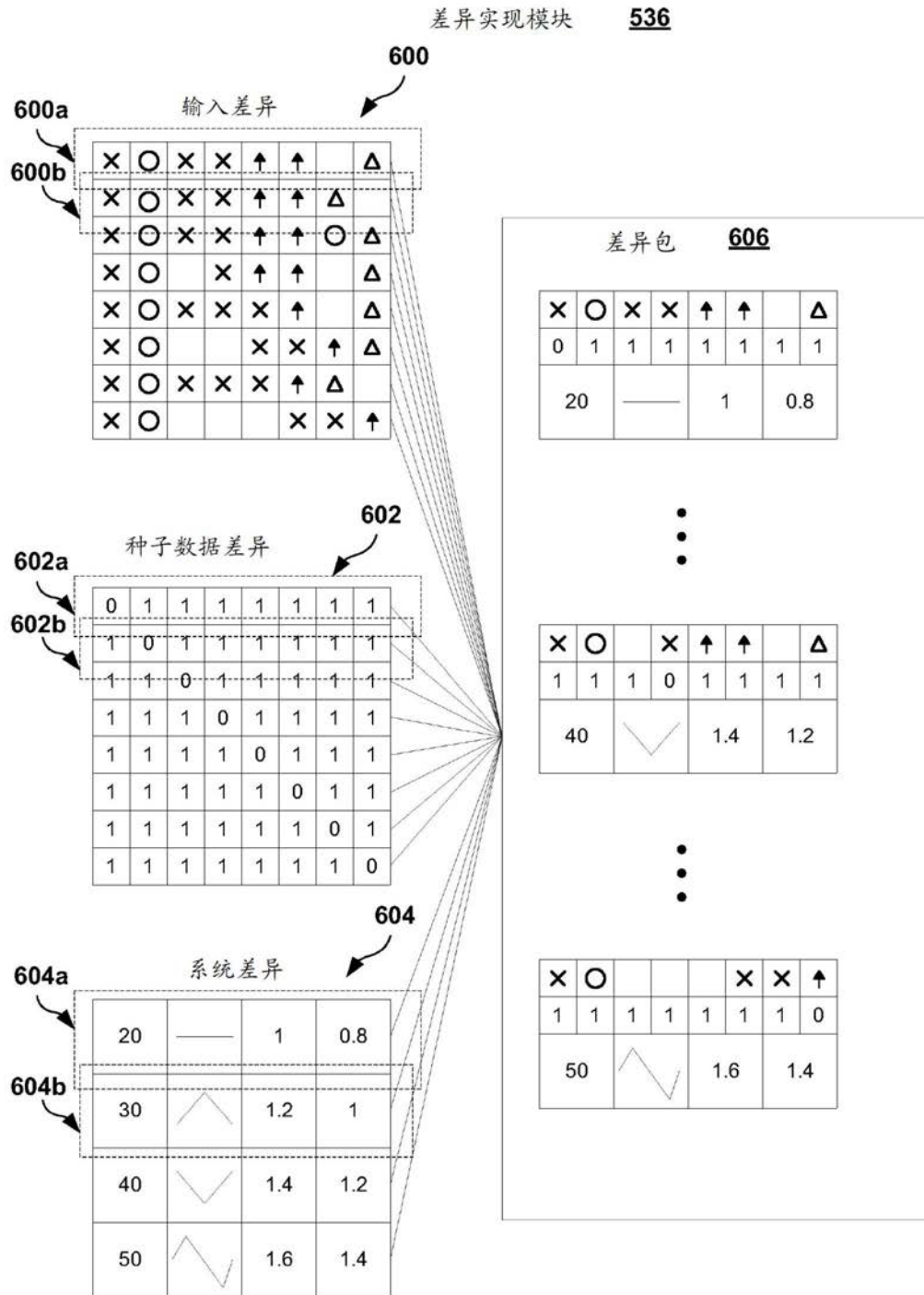


图6

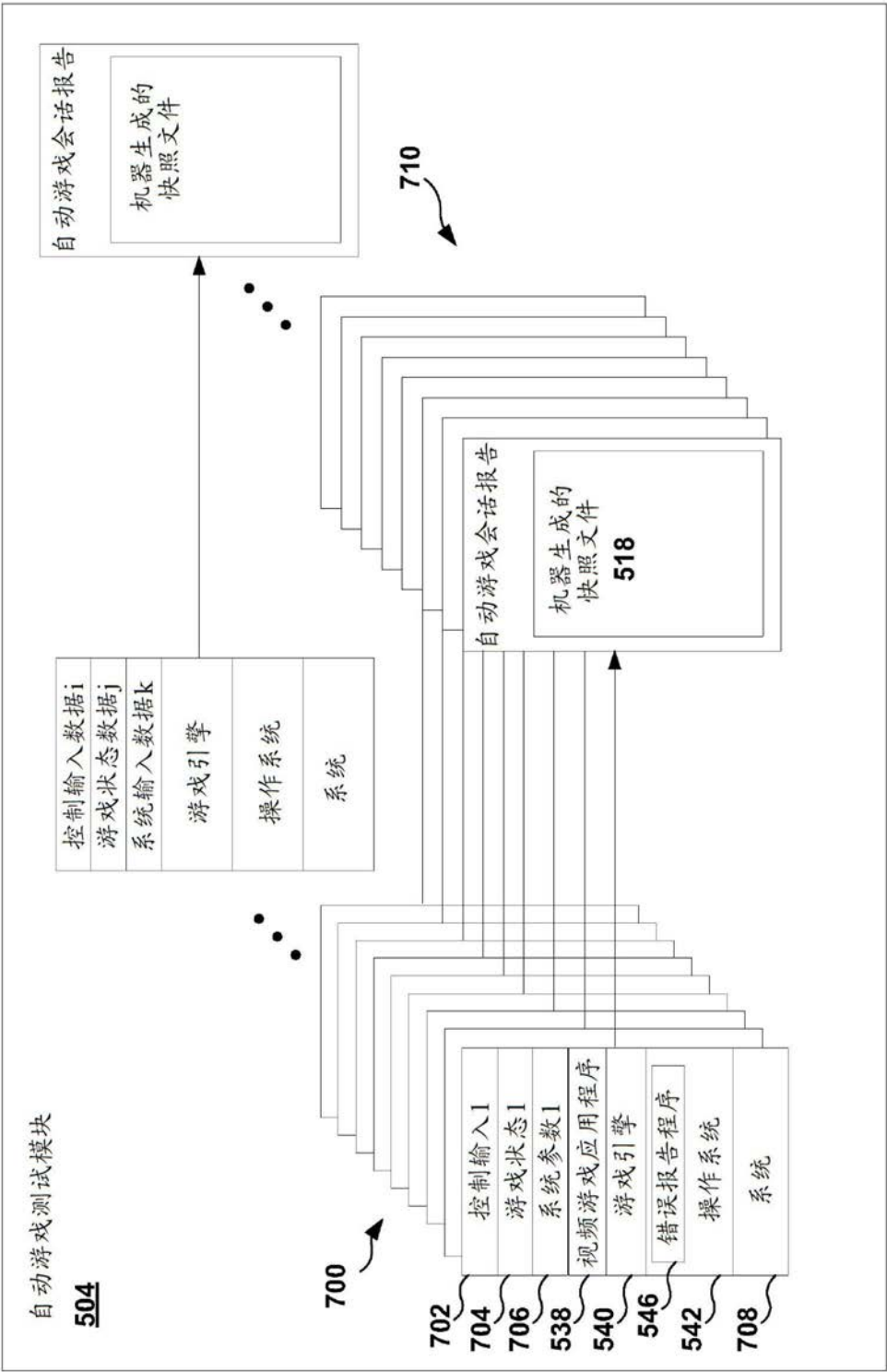


图7

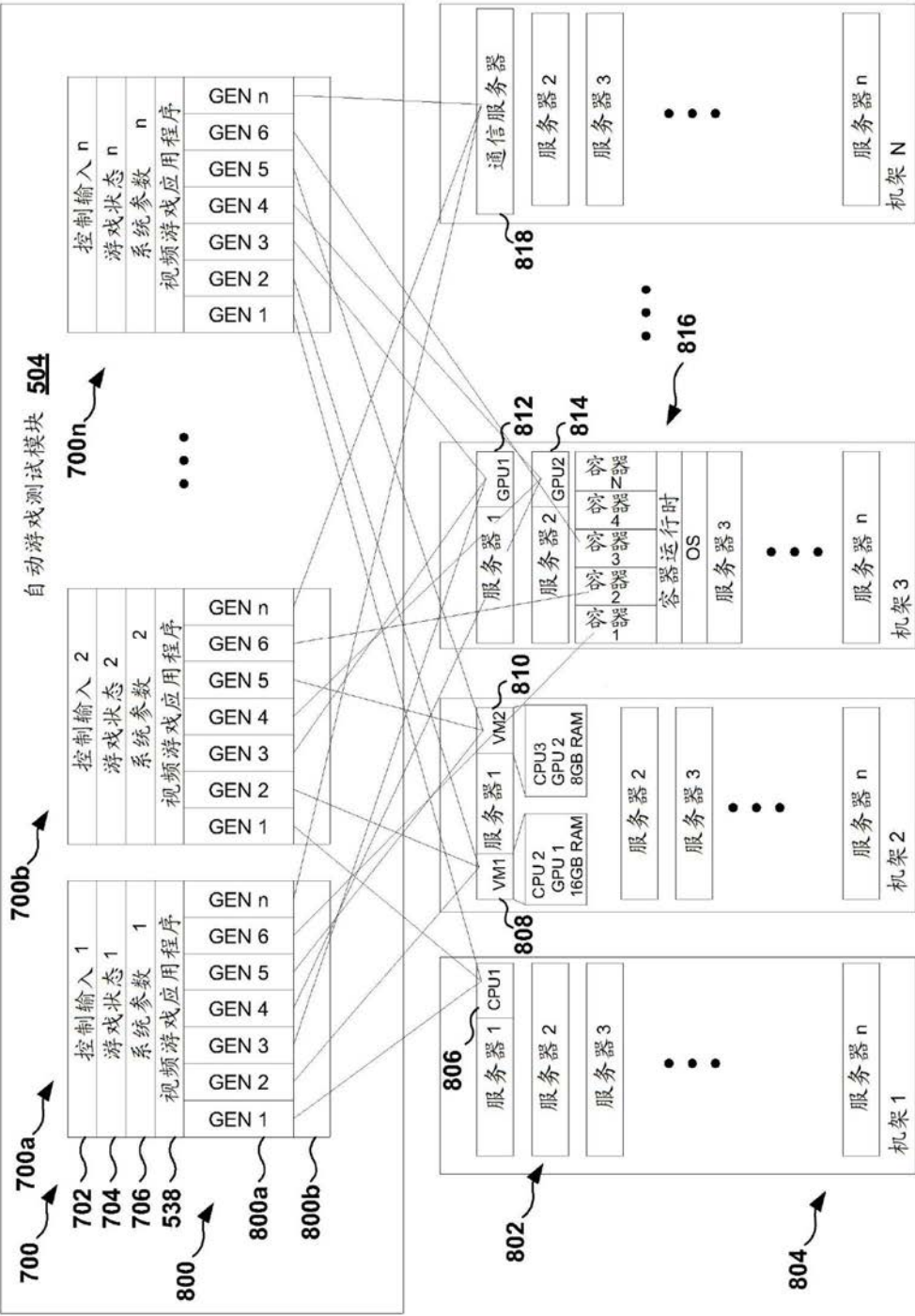


图8

在玩家生成的输入的帮助下发现用于
再现错误1的错误实现条件

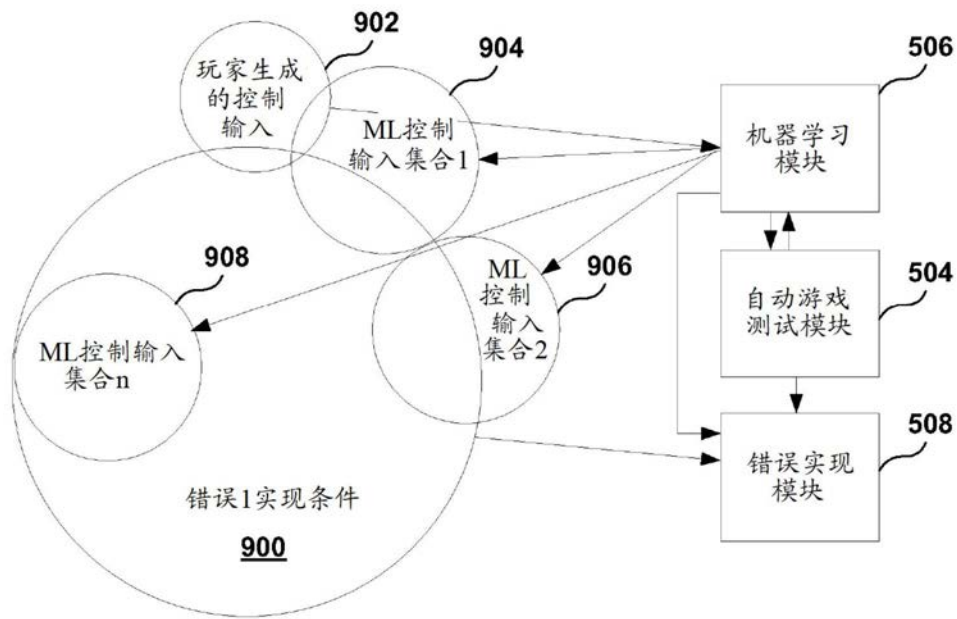


图9A

发现未被发现的错误2的错误实现条件

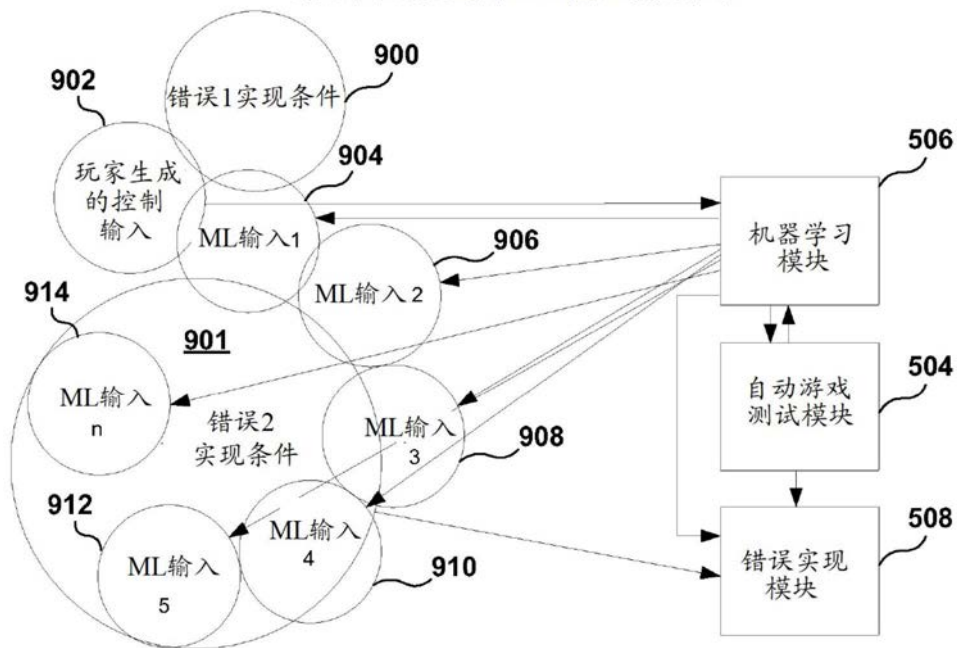


图9B

自动混沌测试

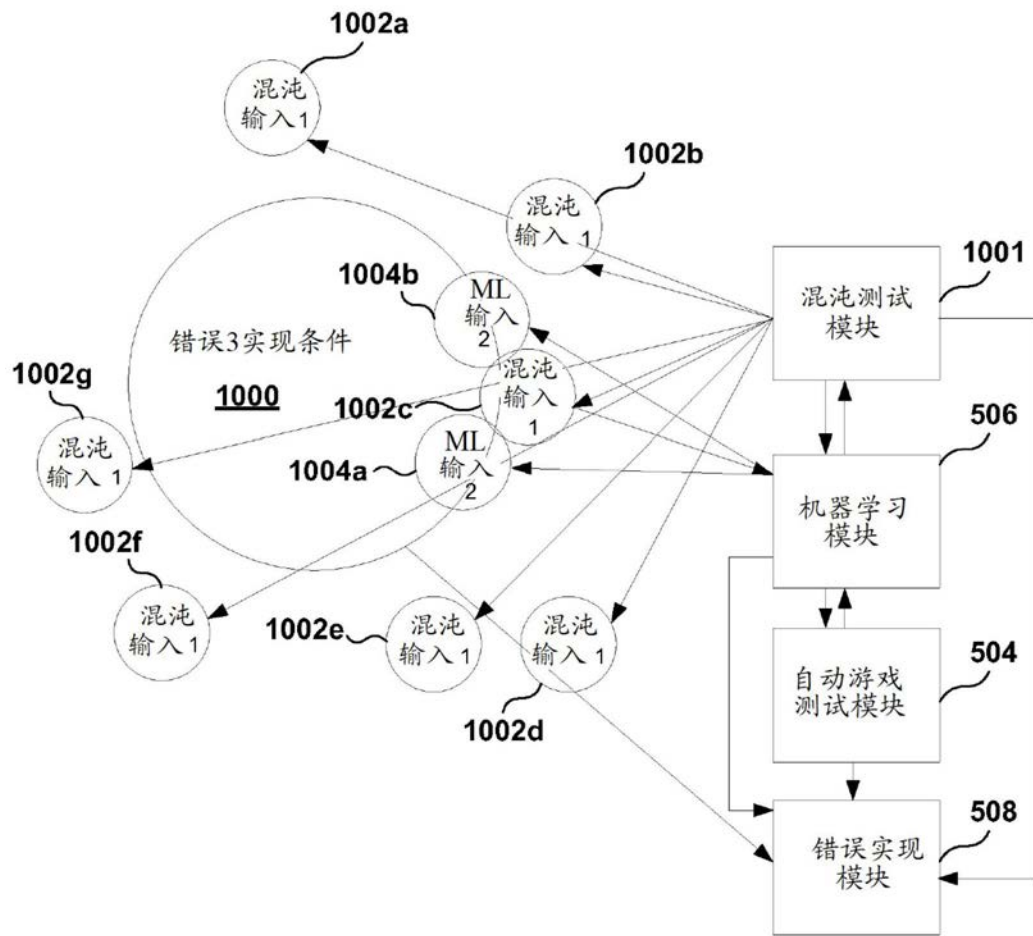


图10

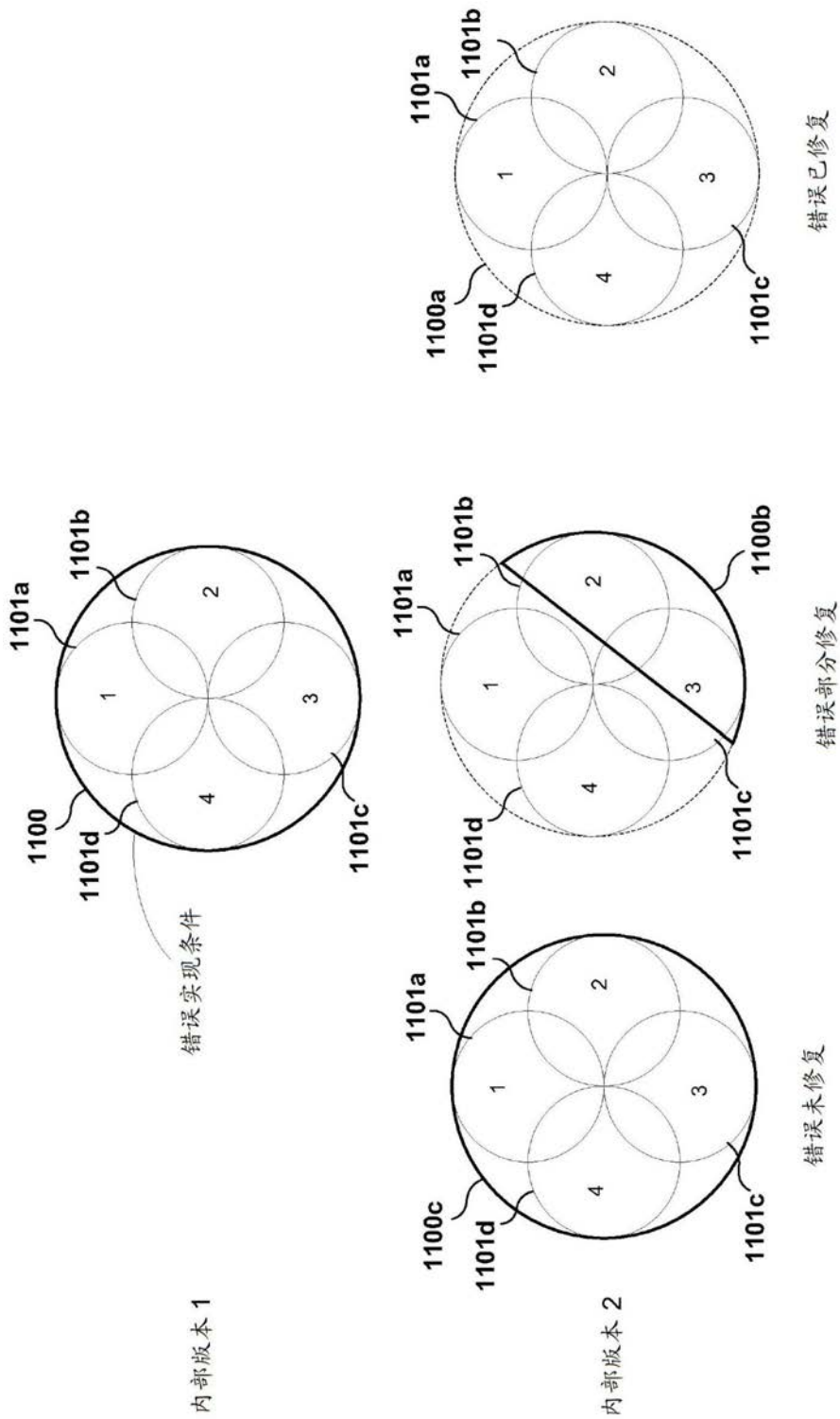
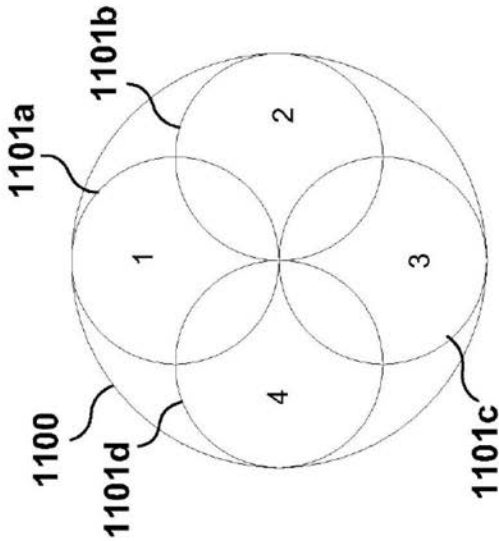


图11

测试即服务(TaaS)

游戏标题1的错误实现条件



游戏标题2的错误实现条件

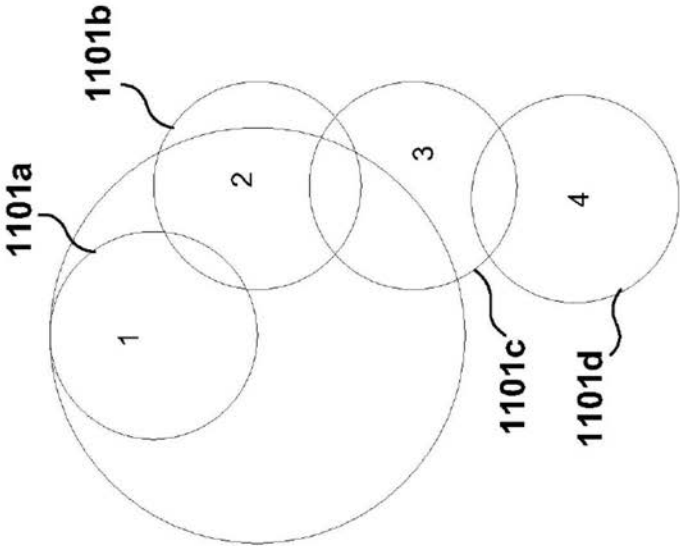


图12

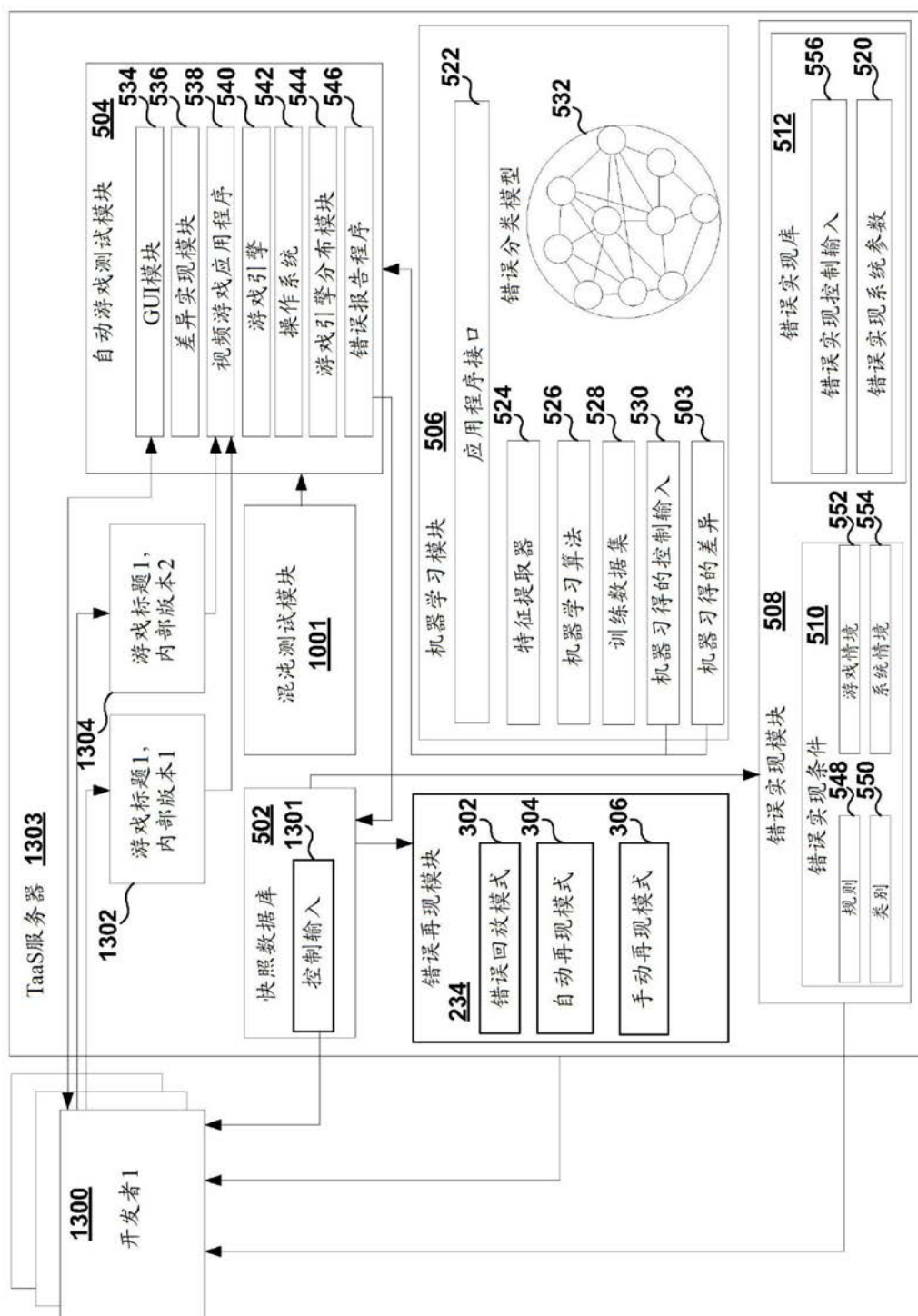


图13

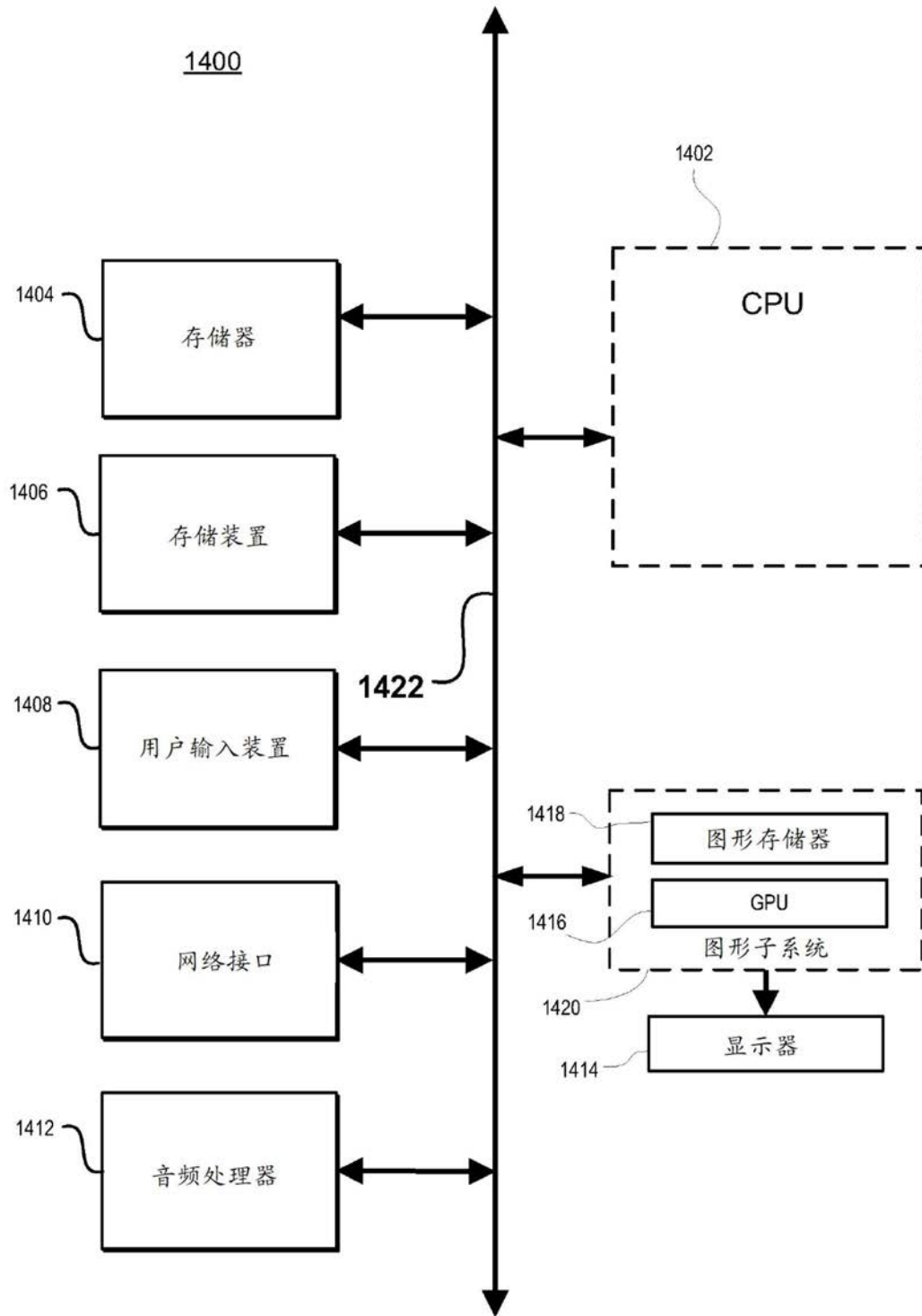


图14