



(12) 发明专利

(10) 授权公告号 CN 110781085 B

(45) 授权公告日 2022. 02. 11

(21) 申请号 201911015037.0

(22) 申请日 2019.10.23

(65) 同一申请的已公布的文献号

申请公布号 CN 110781085 A

(43) 申请公布日 2020.02.11

(73) 专利权人 腾讯科技(深圳)有限公司

地址 518057 广东省深圳市南山区高新区

科技中一路腾讯大厦35层

(72) 发明人 张帆

(74) 专利代理机构 广州三环专利商标代理有限公司

公司 44202

代理人 郝传鑫 熊永强

(51) Int. Cl.

G06F 11/36 (2006.01)

A63F 13/60 (2014.01)

(56) 对比文件

CN 109885480 A, 2019.06.14

CN 109885480 A, 2019.06.14

CN 103514023 A, 2014.01.15

CN 110096425 A, 2019.08.06

CN 105740154 A, 2016.07.06

CN 103336688 A, 2013.10.02

KR 101881804 B1, 2018.07.25

审查员 陈雅萍

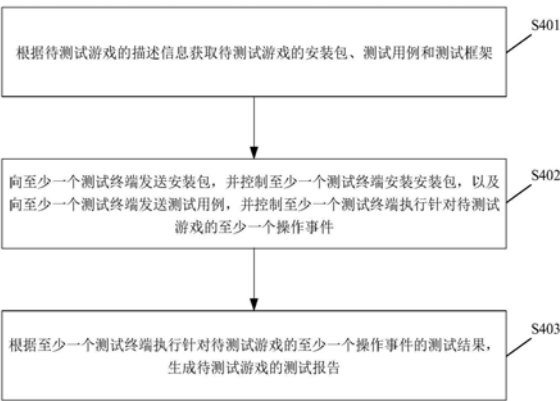
权利要求书3页 说明书22页 附图16页

(54) 发明名称

一种游戏自动化测试方法、装置、终端和计算机存储介质

(57) 摘要

本发明实施例公开了一种游戏自动测试方法、装置、终端和计算机存储介质。所述一种游戏自动测试方法包括：根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架；根据与所述待测试游戏匹配的测试脚本命令，向至少一个测试终端发送所述安装包，并控制所述至少一个测试终端安装所述安装包，以及向所述至少一个测试终端发送所述测试用例，并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件；根据所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件的测试结果，生成所述待测试游戏的测试报告。采用本发明实施例，可以对各类游戏实现自动下载、安装、测试和测试结果推送，提高游戏测试效率。



1. 一种游戏自动测试方法,其特征在于,包括:

根据预设的测试流程,录制针对待测试游戏的至少一个操作事件,并确定所述至少一个操作事件中的每一个操作事件对应的游戏控件、操作动作、游戏对象和游戏场景;

基于所述至少一个操作事件中的每一个操作事件对应的游戏控件、操作动作、游戏对象和游戏场景生成所述待测试游戏的测试用例;

根据所述待测试游戏的源代码,通过构建工具执行所述待测试游戏的构建任务,生成所述待测试游戏的安装包;

根据所述待测试游戏的描述信息查询所述待测试游戏的构建任务状态;

若所述待测试游戏的构建任务已完成,则根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架,所述测试用例包括针对所述待测试游戏的至少一个操作事件,所述测试框架包括与所述待测试游戏匹配的测试脚本命令;所述待测试游戏为具有多种操作系统的待测试游戏,与所述待测试游戏匹配的测试脚本命令是根据所述待测试游戏对应的操作系统确定的;

根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包,并控制所述至少一个测试终端安装所述安装包,以及向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端根据所述测试用例识别所述至少一个操作事件中的每一个操作事件对应的游戏控件、操作动作、游戏对象和游戏场景,并驱动所述至少一个测试终端执行所述至少一个操作事件;

根据所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件的测试结果,生成所述待测试游戏的测试报告包括:监控所述至少一个测试终端在执行针对所述待测试游戏的至少一个操作事件时的测试数据,根据所述测试数据得到相应的文字结果;根据所述至少一个测试终端在执行所述至少一个操作事件中的任意一个操作事件时出现游戏漏洞的游戏截图,得到图片结果,根据所述图片结果和所述文字结果生成所述待测试游戏的测试报告。

2. 根据权利要求1所述的方法,其特征在于,所述方法还包括:

确定所述测试报告对应的测试类别,并向所述测试类别对应的至少一个预设地址发送所述测试报告。

3. 根据权利要求1所述的方法,其特征在于,所述测试结果包括文字结果和图片结果;所述文字结果为所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件时产生的测试数据;所述图片结果为所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件时出现游戏漏洞的游戏截图,所述游戏漏洞包括游戏闪退、游戏卡顿、游戏闪屏和游戏崩溃中的任意一种。

4. 根据权利要求1所述的方法,其特征在于,所述方法还包括:

加载并解析预设的配置文件,确定所述待测试游戏的描述信息,所述描述信息包括所述待测试游戏的版本信息、名称信息、路径信息、操作系统信息和游戏引擎信息中的一种或多种。

5. 根据权利要求4所述的方法,其特征在于,所述根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包之前,包括:

根据所述描述信息确定与所述待测试游戏的操作系统信息匹配的所述至少一个测试

终端。

6. 根据权利要求1所述的方法,其特征在于,所述安装包包括与所述测试框架匹配的软件开发工具包SDK;所述向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件之前,包括:

根据与所述待测试游戏匹配的测试脚本命令控制所述至少一个测试终端启动所述待测试游戏,并运行所述SDK;

通过套接字socket与所述至少一个测试终端上运行的所述SDK建立测试连接,所述测试连接用于传递所述测试用例和所述测试结果。

7. 根据权利要求6所述的方法,其特征在于,所述SDK用于通过所述测试连接接收所述测试用例,并根据所述测试用例识别所述至少一个操作事件中的每一个操作事件对应的游戏控件、操作动作、游戏对象和游戏场景,并驱动所述至少一个测试终端执行所述至少一个操作事件。

8. 一种游戏自动测试装置,其特征在于,包括:

录制模块,用于根据预设的测试流程,录制针对待测试游戏的至少一个操作事件,并确定所述至少一个操作事件中的每一个操作事件对应的游戏控件、操作动作、游戏对象和游戏场景;

测试用例生成模块,用于基于所述至少一个操作事件中的每一个操作事件对应的游戏控件、操作动作、游戏对象和游戏场景生成所述待测试游戏的测试用例;

构建模块,用于根据所述待测试游戏的源代码,通过构建工具执行所述待测试游戏的构建任务,生成所述待测试游戏的安装包;

查询模块,用于根据所述待测试游戏的描述信息查询所述待测试游戏的构建任务状态;获取模块,用于若所述待测试游戏的构建任务已完成,则根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架,所述测试用例包括针对所述待测试游戏的至少一个操作事件,所述测试框架包括与所述待测试游戏匹配的测试脚本命令;所述待测试游戏为具有多种操作系统的待测试游戏,与所述待测试游戏匹配的测试脚本命令是根据所述待测试游戏对应的操作系统确定的;

测试模块,用于根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包,并控制所述至少一个测试终端安装所述安装包,以及向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端根据所述测试用例识别所述至少一个操作事件中的每一个操作事件对应的游戏控件、操作动作、游戏对象和游戏场景,并驱动所述至少一个测试终端执行所述至少一个操作事件;生成模块,用于根据所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件的测试结果,生成所述待测试游戏的测试报告包括:监控所述至少一个测试终端在执行针对所述对待测试游戏的至少一个操作事件时的测试数据,根据所述测试数据得到相应的文字结果;根据所述至少一个测试终端在执行所述至少一个操作事件中的任意一个操作事件时出现游戏漏洞的游戏截图,得到图片结果,根据所述图片结果和所述文字结果生成所述待测试游戏的测试报告。

9. 根据权利要求8所述的装置,其特征在于,所述装置还包括:

测试报告发送模块,用于确定所述测试报告的测试类别,并向所述测试类别对应的一个或多个目标地址发送所述测试报告。

10. 根据权利要求8所述的装置,其特征在于,所述装置还包括:

描述信息确定模块,用于加载并解析预设的配置文件,确定所述待测试游戏的描述信息,所述描述信息包括所述待测试游戏的版本信息、名称信息、路径信息、操作系统信息和游戏引擎信息中的一种或多种。

11. 根据权利要求10所述的装置,其特征在于,所述装置还包括:

测试终端确定模块,用于根据所述描述信息确定与所述待测试游戏的操作系统信息匹配的所述至少一个测试终端。

12. 一种测试服务端,包括:处理器和存储器;

所述处理器和存储器相连,其中,所述存储器用于存储程序代码,所述处理器用于调用所述程序代码,以执行如权利要求1-7任一项所述的方法。

13. 一种计算机存储介质,其特征在于,所述计算机存储介质存储有计算机程序,所述计算机程序包括程序指令,所述程序指令当被处理器执行时,执行如权利要求1-7任一项所述的方法。

一种游戏自动化测试方法、装置、终端和计算机存储介质

技术领域

[0001] 本发明涉及计算机技术领域,尤其涉及一种游戏自动测试方法、装置、终端和计算机存储介质。

背景技术

[0002] 随着智能手机行业的快速发展,手机游戏的市场占有率越来越高,游戏的内容和复杂程度也随之增加。现有的手机游戏大多采用Unity引擎进行开发,然而,随着用户对手游画面的精美程度越来越高的要求,依托于具有强大画面优势的虚幻引擎4(Unreal Engine 4,UE4)开发的手游产品也越来越多,与此同时,游戏的各类区域化版本更是层出不穷。为保证游戏的性能,游戏制作公司在游戏的开发过程中都需投入大量的资源来执行相关的游戏测试。例如,负载测试、耗电量测试、针对每日版本的冒烟测试、内存监测以及针对特定场景的崩溃测试等等,从而通过各类游戏测试不断检测和优化游戏性能。

[0003] 为保证游戏测试的高质量和高效率,市面上生产了很多游戏自动化测试工具,比较常见的例如有自动测试工具Airtest和自动测试工具GAutomator,这些测试工具可以为测试人员提供游戏的自动化测试框架,完成部分游戏测试内容。但是,上述自动化测试仅限于对游戏内容的自动测试,例如,在已手动安装好待测试游戏的测试终端上根据预设的测试项目以及流程执行相应的自动测试。该测试方法整体自动化程度较低,并且,由于在游戏测试之前涉及到手动安装等操作,而不同的测试人员手动安装的过程可能会存在操作差异,使得测试结果产生误差。因此,在手游市场不断扩大、游戏测试需求日益攀升的情况下,游戏测试人员更需要一种可以覆盖从游戏测试的前期准备工作到具体执行游戏测试在内的整个自动测试流程的游戏自动测试方案。

发明内容

[0004] 本发明实施例提供了一种游戏自动测试方法、装置、终端及计算机存储介质,可以实现各类游戏从下载、安装、测试到推送测试结果的自动化流程,提升游戏测试的自动化率,减轻测试负担,提高测试效率。

[0005] 一方面,本发明实施例提供了一种游戏自动测试方法,该游戏自动测试方法包括:

[0006] 根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架,所述测试用例包括针对所述待测试游戏的至少一个操作事件,所述测试框架包括与所述待测试游戏匹配的测试脚本命令;

[0007] 根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包,并控制所述至少一个测试终端安装所述安装包,以及向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件;

[0008] 根据所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件的测

试结果,生成所述待测试游戏的测试报告。

[0009] 其中,所述方法还包括:

[0010] 确定所述测试报告对应的测试类别,并向所述测试类别对应的至少一个预设地址发送所述测试报告。

[0011] 其中,所述测试结果包括文字结果和图片结果;所述文字结果为所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件时产生的测试数据;所述图片结果为所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件时出现游戏漏洞的游戏截图,所述游戏漏洞包括游戏闪退、游戏卡顿、游戏闪屏和游戏崩溃中的任意一种。

[0012] 其中,所述方法还包括:

[0013] 加载并解析预设的配置文件,确定所述待测试游戏的描述信息,所述描述信息包括所述待测试游戏的版本信息、名称信息、路径信息、操作系统信息和游戏引擎信息中的一种或多种。

[0014] 其中,所述根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包之前,包括:

[0015] 根据所述描述信息确定与所述待测试游戏的操作系统信息匹配的所述至少一个测试终端。

[0016] 其中,所述根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架之前,包括:

[0017] 根据所述待测试游戏的源代码,通过构建工具执行所述待测试游戏的构建任务,生成所述待测试游戏的安装包;

[0018] 根据所述待测试游戏的描述信息查询所述待测试游戏的构建任务状态;

[0019] 若所述待测试游戏的构建任务已完成,则执行所述根据所述待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架的步骤。

[0020] 其中,所述安装包包括与所述测试框架匹配的软件开发工具包SDK;所述向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件之前,包括:

[0021] 根据与所述待测试游戏匹配的测试脚本命令控制所述至少一个测试终端启动所述待测试游戏,并运行所述SDK;

[0022] 通过套接字socket与所述至少一个测试终端上运行的所述SDK建立测试连接,所述测试连接用于传递所述测试用例和所述测试结果。

[0023] 其中,所述方法还包括:

[0024] 根据预设的测试流程,录制针对所述待测试游戏的至少一个操作事件,并确定所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景;

[0025] 基于所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景生成所述测试用例。

[0026] 其中,所述SDK用于通过所述测试连接接收所述测试用例,并根据所述测试用例识别所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景,并驱动所述至少一个测试终端执行所述至少一个操作事件。

[0027] 另一方面,本发明实施例提供了一种游戏自动测试装置,该游戏自动测试装置包

括：

[0028] 获取模块，用于根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架，所述测试用例包括针对所述待测试游戏的至少一个操作事件，所述测试框架包括与所述待测试游戏匹配的测试脚本命令；

[0029] 测试模块，用于根据与所述待测试游戏匹配的测试脚本命令，向至少一个测试终端发送所述安装包，并控制所述至少一个测试终端安装所述安装包，以及向所述至少一个测试终端发送所述测试用例，并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件；

[0030] 生成模块，用于根据所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件的测试结果，生成所述待测试游戏的测试报告。

[0031] 其中，所述装置还包括：

[0032] 测试报告发送模块，用于确定所述测试报告的测试类别，并向所述测试类别对应的一个或多个目标地址发送所述测试报告。

[0033] 其中，所述测试结果包括文字结果和图片结果；所述文字结果为所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件时产生的测试数据；所述图片结果为所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件时出现游戏漏洞的游戏截图，所述游戏漏洞包括游戏闪退、游戏卡顿、游戏闪屏和游戏崩溃中的任意一种。

[0034] 其中，所述装置还包括：

[0035] 描述信息确定模块，用于加载并解析预设的配置文件，确定所述待测试游戏的描述信息，所述描述信息包括所述待测试游戏的版本信息、名称信息、路径信息、操作系统信息和游戏引擎信息中的一种或多种。

[0036] 其中，所述装置还包括：

[0037] 测试终端确定模块，用于根据所述描述信息确定与所述待测试游戏的操作系统信息匹配的所述至少一个测试终端。

[0038] 其中，所述根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架之前，所述装置还包括：

[0039] 构建模块，用于根据所述待测试游戏的源代码，通过构建工具执行所述待测试游戏的构建任务，生成所述待测试游戏的安装包；

[0040] 查询模块，用于根据所述待测试游戏的描述信息查询所述待测试游戏的构建任务状态；

[0041] 执行模块，用于若所述待测试游戏的构建任务已完成，则执行所述根据所述待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架的步骤。

[0042] 其中，所述安装包包括与所述测试框架匹配的软件开发工具包SDK；所述向所述至少一个测试终端发送所述测试用例，并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件之前，所述装置还包括：

[0043] 启动模块，用于根据与所述待测试游戏匹配的测试脚本命令控制所述至少一个测试终端启动所述待测试游戏，并运行所述SDK；

[0044] 连接模块，用于通过套接字socket与所述至少一个测试终端上运行的所述SDK建立测试连接，所述测试连接用于传递所述测试用例和所述测试结果。

[0045] 其中,所述装置还包括:

[0046] 录制模块,用于根据预设的测试流程,录制针对所述待测试游戏的至少一个操作事件,并确定所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景;

[0047] 测试用例生成模块,用于基于所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景生成所述测试用例。

[0048] 其中,所述SDK用于通过所述测试连接接收所述测试用例,并根据所述测试用例识别所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景,并驱动所述至少一个测试终端执行所述至少一个操作事件。

[0049] 再一方面,本发明实施例提供了一种终端设备,包括输入设备和输出设备,所述终端设备还包括:

[0050] 处理器,用于实现一条或一条以上指令;

[0051] 计算机存储介质,所述计算机存储介质存储有一条或一条以上指令,所述一条或一条以上指令用于由所述处理器加载并执行如下步骤:

[0052] 根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架,所述测试用例包括针对所述待测试游戏的至少一个操作事件,所述测试框架包括与所述待测试游戏匹配的测试脚本命令;

[0053] 根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包,并控制所述至少一个测试终端安装所述安装包,以及向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件;

[0054] 根据所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件的测试结果,生成所述待测试游戏的测试报告。

[0055] 再一方面,本发明实施例提供了一种计算机存储介质,所述计算机存储介质存储有一条或一条以上指令,所述一条或一条以上指令用于由所述处理器加载并执行如下步骤:

[0056] 根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架,所述测试用例包括针对所述待测试游戏的至少一个操作事件,所述测试框架包括与所述待测试游戏匹配的测试脚本命令;

[0057] 根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包,并控制所述至少一个测试终端安装所述安装包,以及向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件;

[0058] 根据所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件的测试结果,生成所述待测试游戏的测试报告。

[0059] 本发明实施例提供了一种游戏自动测试方法,可以根据本次测试任务中待测试游戏的描述信息(例如包括待测试游戏的名称信息、版本信息、路径信息、操作系统信息和游戏引擎信息等)获取待测试游戏的安装包,并获取与所述待测试游戏匹配的测试用例和测试框架,自动完成待测试游戏的测试环境的构建,即实现了在进行针对游戏内容的自动

测试前的准备工作的自动化;然后,根据 测试框架中与待测试游戏匹配的测试脚本命令控制后续整个游戏测试流程,其中包括向至少一个测试终端(例如可以是不同品牌、型号的手机、平板电脑等) 发送待测试游戏的安装包,并控制该至少一个测试终端进行安装;其次,还包括向至少一个测试终端发送测试用例,并控制该至少一个测试终端执行该测试用例,以完成对待测试游戏的自动测试,测试用例包括针对待测试游戏的至少 一个操作事件(例如登录游戏、选择游戏模式、开启战斗和退出游戏等等); 最终可以根据至少一个测试终端分别执行该至少一个操作事件的测试结果,生成该待测试游戏的测试报告。由此,对比现有的游戏自动测试方法仅仅实现了对待测试游戏内容的自动测试,仍需投入大量人力资源完成对待测试游戏的下载、安装等测试前的准备工作而言。在本发明实施例中可以完成包括待测试游戏的自动下载、自动安装、自动测试和测试报告自动生成在内的整个自动测试 流程,提升游戏测试的自动化率,减轻游戏开发人员的测试负担。与此同时, 在本发明实施例中还可以同时对不同测试终端执行同一待测试游戏的自动测试,得到同一待测试游戏运行在不同的测试终端上的测试结果,保证游戏测试的高 质量和高效率。

附图说明

[0060] 为了更清楚地说明本发明实施例的技术方案,下面将对实施例描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图是本发明的一些 实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

[0061] 图1是本发明实施例提供的一种游戏自动测试方法的系统架构图;

[0062] 图2是本发明实施例提供的一种不同操作系统的游戏自动测试方法的整体 架构示意图;

[0063] 图3是本发明实施例提供的一种游戏自动测试的整体流程示意图;

[0064] 图4A是本发明实施例提供的一种游戏自动测试方法的流程示意图;

[0065] 图4B是本发明实施例提供的一种游戏自动测试方法的整体流程示意图;

[0066] 图5A是本发明实施例提供的另一种游戏自动测试方法的流程示意图;

[0067] 图5B是本发明实施例提供的另一种游戏自动测试方法的整体流程示意图;

[0068] 图6是本发明实施例提供的一种游戏自动测试方法的定时任务界面示意图;

[0069] 图7是本发明实施例提供的一种待测试游戏的安装包下载流程示意图;

[0070] 图8是本发明实施例提供的一种下载失败消息的自动推送界面示意图;

[0071] 图9是本发明实施例提供的一种安装包下载过程示意图;

[0072] 图10是本发明实施例提供的一种录制流程示意图;

[0073] 图11是本发明实施例提供的一种自动测试流程示意图;

[0074] 图12是本发明实施例提供的另一种自动测试流程示意图;

[0075] 图13是本发明实施例提供的一种游戏内自动战斗的流程示意图;

[0076] 图14是本发明实施例提供的一种游戏测试的文字结果示意图;

[0077] 图15是本发明实施例提供的另一种游戏测试的文字结果示意图;

[0078] 图16是本发明实施例提供的一种崩溃测试的图片结果;

[0079] 图17是本发明实施例提供的一种测试报告的自动推送界面示意图;

- [0080] 图18是本发明实施例提供的另一种测试报告的自动推送界面示意图；
- [0081] 图19是本发明实施例提供的一种游戏自动测试装置的结构示意图；
- [0082] 图20是本发明实施例提供的一种测试服务端的结构示意图。

具体实施方式

[0083] 下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行清楚、完整地描述,显然,所描述的实施例仅仅是本发明一部分实施例,而不是全部的实施例。基于本发明中的实施例,本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例,都属于本发明保护的范围。

[0084] 请参见图1,图1是本发明实施例提供的一种游戏自动测试方法的系统架构图,本发明实施例的技术方案可以在图1举例所示的系统架构或类似的系统架构中具体实施。如图1所示,该系统架构可以包括测试服务端100a和多个测试终端,如图1所示,具体可以包括测试终端200a、200b和200c。其中,测试服务端100a可以根据本次测试任务中待测试游戏的描述信息,例如待测试游戏的名称信息、版本信息、路径信息、操作系统信息和游戏引擎信息等等,从服务器下载该待测试游戏的安装包,以及加载与该待测试游戏匹配的测试用例(例如可以包括针对待测试游戏的至少一个操作事件)和测试框架(例如可以为市面上较为常见的自动测试工具GAutomator和自动测试工具Airtest等);然后,测试服务端100a可以通过脚本命令将待测试游戏的安装包发送至测试终端200a、200b和200c,并控制测试终端200a、200b和200c安装该安装包;其次,测试服务端100a可以向测试终端200a、200b和200c发送该测试用例,并控制测试终端200a、200b和200c执行该测试用例(例如可以为执行登录游戏、选择游戏模式、进入战斗和退出游戏等操作事件),完成对待测试游戏的测试,获得测试结果;最终,测试服务端100a可以根据测试结果,生成测试报告,并将测试报告推送至相应的地址(例如为可以通过即时通讯软件将测试报告推送至相应的测试小组的通讯群)。其中,测试服务端100a可以通过接收到的测试指令触发整个游戏自动测试流程,该测试指令可以是预设的定时测试指令,也可以是开发人员通过测试服务端100a手动输入的测试指令。至此,测试服务端100a可以在测试指令的触发下完成包括待测试游戏的自动下载、自动安装、自动测试和测试结果自动推送在内的整个游戏自动测试流程。

[0085] 如上所述,其中,测试终端200a、200b和200c可以为具备上述功能的智能手机、智能可穿戴设备、平板电脑、膝上计算机以及台式电脑等。测试服务端100a可以是具备上述功能的终端设备,例如为智能手机、智能可穿戴设备、平板电脑、膝上计算机以及台式电脑等。测试服务端100a也可以是具有交互界面的服务器,所述服务器可以是一台服务器,也可以是由多台服务器组成的服务器集群,或者是一个云计算服务中心。测试服务端100a可以通过无线网络和有线网络与测试终端200a、200b和200c建立通信连接。

[0086] 结合上述图1实施例对应的系统架构图,在一种可能的实施方式中,本发明实施例还提供了一种在不同操作系统下(例如为Android系统和ios系统)的游戏自动测试方法的整体架构示意图。请参见图2,图2是本发明实施例提供的一种不同操作系统的游戏自动测试方法的整体架构示意图。除此之外,本发明实施例还针对图1实施例对应的方法流程提供了一种游戏自动测试的整体流程图。请参见图3,图3是本发明实施例提供的一种游戏自动测试的整体流程示意图。下面将结合如图2所示的整体架构以及如图3所示的整体流

程对本发明实施例提供的游戏自动测试方法进行进一步详细阐述。

[0087] 在一种可能的实施方式中,如图2所示,该架构可以应用于不同的操作系统,例如为如图2所示的Android系统和ios系统,所述个人计算机(personal computer,PC)和苹果电脑Mac可以对应于上述图1所示的测试服务端100a,所述安卓手机Android和苹果手机iphone可以对应于上述图1所示的测试终端200a、200b和200c中的任意一个测试终端。在Android系统的游戏测试过程中,所述PC和Android手机可以通过数据线连接,也可以通过无线网络连接。所述PC保持永不待机的状态,可以通过定时测试指令触发整个游戏自动测试流程,所述Android手机可以锁屏待机,但不可设置密码,以便根据PC的命令执行相应的操作,完成游戏的自动化测试。在ios系统的游戏测试过程中,所述Mac和iphone手机可以通过数据线连接,也可以通过无线网络连接。所述Mac保持永不待机的状态,可以通过定时测试指令触发整个游戏自动测试流程,所述iphone手机保持永不待机的状态,以便根据Mac的命令执行相应的操作,完成游戏的自动化测试。

[0088] 以Android系统为例,在测试开始前,所述PC可以通过构建工具(构建工具为实现将源代码生成可执行应用程序的过程自动化的程序)根据游戏开发人员编写的游戏源代码执行游戏的构建任务,生成游戏的安装包,安装包可以存放至服务器中。其中,所述构建工具可以为如图2所示的Soda构建工具,也可以为市面上较为常见的例如蓝盾等构建工具,本发明实施例对此不作具体限定。可选的,所述游戏源代码可以是多名游戏开发人员编写的多组代码,可以存放至该游戏的代码库中。所述构建任务可以在PC上(即本地机器)完成,也可以通过构建服务器完成。在一个游戏的开发过程中,往往需要多次修改代码或添加新的代码,不断构建不同代码对应的游戏的不同版本,并对每个版本进行测试,再根据测试结果修改代码或添加新的代码,循环往复,直至完善游戏性能。例如,一款游戏的开发小组中的多名开发人员可以在每日下班前修改或编写好新的游戏代码并存放至该游戏的代码库中,即更新该游戏的代码库。其次,可以通过设置定时构建指令,使得所述PC可以定时根据所述更新后的代码库执行该游戏新版本的构建任务。如图3所示,可以通过预设的定时触发指令触发测试任务,例如,可以通过蓝盾、Soda等工具部署定时任务,触发整个测试流程,也可以手动触发测试任务。如图3所示,在所述测试任务被触发后,所述PC可以首先从服务器下载本次测试任务的待测试游戏的安装包,并且加载与该待测试游戏匹配的测试用例和测试框架(例如可以为自动测试工具GAutomator和自动测试工具Airtest等等)。所述测试用例可以是根据预设的测试项目和流程编写或录制的包括针对待测试游戏的至少一个操作事件(例如可以为启动游戏、登录游戏、进入游戏和结束游戏等简单的操作事件)的测试用例。所述测试框架包括与待测试游戏匹配的测试脚本命令。该待测试游戏可以是当前构建的该款游戏的最新版本,也可以是任意一个指定的版本,本发明实施例对此不作具体限定。在下载完成后,所述PC可以通过安卓测试桥(Android Debug Bridge, ADB)命令将该安装包推送至Android手机,并命令所述Android手机完成安装。其次,所述PC可以通过测试工具根据测试用例对待测试游戏进行自动测试,获得测试结果。所述测试结果可以包括文字结果(比如包括待测试游戏的版本信息、测试开始时间、测试结束时间以及每个测试项目的执行时间、执行结果等等),还可以包括图片结果(比如待测试游戏在执行相关测试项目时出现卡顿、闪退、闪屏和崩溃等问题的游戏截图)。最终PC可以通过即时通讯软件等通信工具将所述测试结果推送至预设地址(例如为该游戏开发小组的通讯

群),使得开发人员可以第一时间掌握本次测试结果,以便及时根据所述测试结果进一步的修改或添加新的代码,完善该游戏的各项性能。如图2所示,所述PC和Mac用于运行上述构建工具、测试工具和通信工具,所述Android和iphone可以用于运行上述待测试游戏。可以理解的是,上述整个游戏自动测试均可以通过编写的Python脚本命令进行流程控制。

[0089] 如上所述,例如,游戏开发人员可以在当天下班前根据前一轮的测试结果编写好新的游戏代码,完成该游戏代码库的更新,并设置定时构建指令,比如在次日的上午六点执行构建任务,生成该游戏的新版本安装包。同时可以设置定时测试指令,比如在次日的上午八点执行该游戏的新版本测试任务,获得本次测试结果,并将该测试结果推送至开发人员的通讯群。如此,可以在开发人员上班时及时获取到该版本的测试结果,并根据所述测试结果进行新一轮的代码修改、版本构建以及测试。提升游戏测试的自动化率,减轻开发人员的测试负担,保证游戏开发的高质量和高效率。

[0090] 请参见图4A,图4A是本发明实施例提供的一种游戏自动测试方法的流程示意图。如图4A所示,该方法可应用于上述图1所述的系统架构中,其中的测试服务端可以为上述图1系统架构中的测试服务端100a,其中的测试终端可以为上述图1系统架构中的测试终端200a、200b和200c中的任意一个测试终端,可用于支持并执行图4A中所示的方法流程。下面将结合图4A从测试服务端侧进行描述,该游戏自动测试法可以包括以下步骤S401-S403:

[0091] 步骤S401,根据待测试游戏的描述信息获取待测试游戏的安装包、测试用例和测试框架。

[0092] 具体的,测试服务端可以根据本次测试任务中待测试游戏的描述信息,获取待测试游戏的安装包、测试用例和测试框架。所述测试用例规定了对待测试游戏执行自动测试的流程,可以包括根据流程串联的针对待测试游戏的至少一个操作事件(例如可以为针对该待测试游戏的登录游戏、选择游戏模式、进入战斗和返回游戏大厅等等操作事件),即本次测试任务中需要完成的针对待测试游戏的测试项目。所述测试框架包括与待测试游戏匹配的测试脚本命令,所述测试脚本命令可以用于控制待测试游戏的测试流程。

[0093] 例如,请参见图4B,图4B是本发明实施例提供的一种游戏测试方法的整体流程图。如图4B所示,该测试服务端可以是PC,可以根据待测试游戏的描述信息(例如包括待测试游戏的名称信息、版本信息、路径信息、操作系统信息和游戏引擎信息等等),从服务器下载待测试游戏的安装包。还可以根据待测试游戏的描述信息加载与之匹配的测试用例和测试框架。例如,针对王者荣耀等手游存在Android和ios等不同操作系统版本的情况,可以加载对应的自动测试工具GAutomator-Android版和自动测试工具GAutomator-ios版测试框架。又例如,针对不同的待测试游戏,其游戏引擎可能存在Unity引擎和虚幻引擎4(Unreal Engine 4,UE4)等不同引擎的情况,可以加载对应的自动测试工具GAutomator-Unity版和自动测试工具GAutomator-UE4版测试框架。从而实现根据本次测试任务针对待测试游戏的实际测试需求,自动获取相应的待测试游戏的安装包、测试用例和测试框架。进而自动完成对待测试游戏进行测试前的准备工作,构建完善的测试环境,减少测试人员手动操作执行这一步骤的工作负担以及手动操作带来的误差,提高游戏测试的自动化率。

[0094] 步骤S402,向至少一个测试终端发送安装包,并控制至少一个测试终端安装安装包,以及向至少一个测试终端发送测试用例,并控制至少一个测试终端执行针对待测试游

戏的至少一个操作事件。

[0095] 具体的,测试服务端可以根据与待测试游戏匹配的测试脚本命令,向与所述测试服务端连接的测试终端发送待测试游戏的安装包,并控制所述至少一个测试终端安装安装包,以及向至少一个测试终端发送测试用例,并控制至少一个测试终端执行针对待测试游戏的至少一个操作事件。

[0096] 例如,如图4B所示,所述至少一个测试终端可以为如图4B所示的多个手机(例如为手机1,……,手机2),所述多个手机可以是不同品牌、型号和操作系统的手机。如图4B所示,在与步骤S402对应的步骤S402-a,测试终端向多个手机发送待测试游戏的安装包和测试用例。例如,在Android系统的游戏自动测试中,所述测试服务端可以为如图4B所示的PC,所述测试终端可以为Android手机,所述PC可以通过ADB命令将安装包推送至Android手机,并控制其进行安装。所述ADB为Android开发工具之一,主要作用是提供电脑(测试服务端)与智能终端(测试终端)之间的交互服务,通过USB接线连接智能终端,电脑上的软件能够直接访问智能终端。可选的,在ios系统的游戏自动测试中,所述测试服务端可以为Mac,所述测试终端可以为iphone手机,所述Mac可以通过ideviceinstaller命令将安装包推送至iphone手机,并控制其进行安装。例如,针对Android操作系统的手机,测试服务端可以通过例如“Adb install待测试游戏”的安装命令向所述手机发送安装包,并控制所述手机安装所述安装包。又例如,针对ios系统的苹果手机,测试服务端可以使用苹果的开源工具 ideviceinstaller,例如通过“ideviceinstaller-I待测试游戏.ipa-o测试终端的特殊识别码(Unique Device Identifier,UDID)”的安装命令向苹果手机发送安装包,并控制所述手机安装所述安装包。例如,如图4B所示,在与步骤S402对应的步骤S402-b、步骤S402-c和步骤S402-d,每个手机可以通过测试服务端的命令接收安装包,并安装该安装包;以及接收测试服务端发送的测试用例,并根据所述测试用例执行这对待测试游戏的至少一个操作事件,从而获得测试结果。可以理解的是,上述命令均可以整合到与待测试游戏匹配的测试脚本命令中,控制安装包的发送和安装流程。从而可以实现待测试游戏在多个不同的测试终端上进行游戏测试,得到更加丰富、全面和更具参考价值的测试结果,提高游戏测试的效率和质量。

[0097] 步骤S403,根据至少一个测试终端执行针对待测试游戏的至少一个操作事件的测试结果,生成待测试游戏的测试报告。

[0098] 具体的,根据至少一个测试终端执行针对待测试游戏的至少一个操作事件的测试结果,生成待测试游戏的测试报告。例如,测试服务端可以监控所述至少一个测试终端在执行针对所述对待测试游戏的至少一个操作事件时的测试数据,根据所述测试数据得到相应的文字结果。例如,测试服务端可以根据所述至少一个测试终端在执行所述至少一个操作事件中的任意一个操作事件时出现游戏漏洞的游戏截图(例如在游戏闪退、游戏卡顿、游戏闪屏和游戏崩溃等等情况时的游戏截图)得到相应的图片结果。例如,如图4B所示,所述游戏截图可以是所述至少一个测试终端发送给测试服务端的游戏截图。如此,测试服务端可以根据所述文字结果和图片结果生成待测试游戏的测试报告(例如可以包括待测试游戏的描述信息、测试开始和结束时间、每一个操作事件和对应的操作结果等等)。从而实现对待测试游戏的测试结果的归纳整合,使得游戏开发人员能一目了然地根据所述测试报告掌握待测试游戏的各项性能,进而有针对性的对该待测试游戏进行改进、完

善,提升游戏测试和游戏开发的效率。

[0099] 请参见图5A,图5A是本发明实施例提供的另一种游戏自动测试方法的流程图。如图5A所示,该方法可应用于上述图1所述的系统架构中,其中的测试服务端可以为上述图1系统架构中的测试服务端100a,其中的测试终端可以为上述图1系统架构中的测试终端200a、200b和200c中的任意一个测试终端,可用于支持并执行图5A中所示的方法流程。下面将结合图5A从测试服务端侧进行描述,该游戏自动测试方法可以包括以下步骤S501-S506:

[0100] 步骤S501,加载并解析预设的配置文件,确定待测试游戏的描述信息。

[0101] 具体的,测试服务端加载并解析预设的配置文件,确定待测试游戏的描述信息。所述描述信息可以包括所述待测试游戏的版本信息(例如为同一游戏在开发过程中的不同新版本,或者同一游戏的不同区域化版本的版本信息)、名称信息(例如可以为“王者荣耀”、“和平精英”等简单游戏名称)、路径信息(例如可以为具体的存储路径)、操作系统信息(例如为Android系统、ios系统等主流操作系统和微软手机windows phone系统、塞班Symbian系统等操作系统)和游戏引擎信息(例如可以为Unity和UE4等)等。所述配置文件为游戏开发人员或游戏测试人员根据本次测试任务的实际需求预先编写好的配置文件,可以用于指定待测试游戏以及有待测试游戏匹配的测试用例和测试框架。例如,请参见图5B,图5B是本发明实施例提供的另一种游戏自动测试方法的整体流程图,如图5B所示,所述测试服务端可以是PC,所述配置文件可以是存储在测试服务端本地的配置文件,也可以是存放至服务器内的配置文件。测试服务端可以通过服务器获取配置文件并解析所述配置文件,确定待测试游戏的描述信息。

[0102] 可选的,测试服务端可以通过接收到的测试指令触发所述加载并解析预设的配置文件,确定待测试游戏的描述信息这一步骤,进而触发整个待测试游戏的自动测试流程。所述测试指令可以是预先设置的定时测试指令,例如通过蓝盾、Soda等工具布置的定时任务,也可以是开发人员通过测试服务端输入的测试指令。所述测试指令用于调用该游戏测试的入口脚本,触发整个游戏自动测试流程的启动。请参见图6,图6是本发明实施例提供的一种游戏自动测试方法的定时任务界面示意图。该定时任务界面可以包括如图6所示的游戏测试类别(例如为ios自动冒烟测试和Android自动冒烟测试),也可以包括测试执行次数(例如如图6所示的ios自动冒烟测试定时任务中的“已执行次数13”),还可以包括最新一次执行测试任务的时间(例如为图6中ios自动冒烟测试定时任务中的“最新执行时间:53秒前”),还可以包括手动触发测试任务按钮,以便在定时任务失败等情况下,游戏开发人员可以随时手动触发游戏测试。

[0103] 步骤S502,根据待测试游戏的描述信息获取待测试游戏的安装包、测试用例和测试框架。

[0104] 具体的,步骤S502可以参考上述图4A实施例中的步骤S401,此处不再赘述。

[0105] 可选的,测试服务端可以根据待测试游戏的源代码通过构建工具执行待测试游戏的构建任务,生成待测试游戏的安装包,所述安装包可以存放至服务器内。可选的,该构建任务可以在测试服务端(即本地机器)完成,也可以在服务器(例如为构建服务器)完成。所述构建服务器可以根据代码库中的最新状态构建最新版本的游戏,即每次开发人员对该游戏的代码库中的代码进行修改或添加新代码时,都会通知到构建服务器,该构建服务

器根据新的代码库重新 构建得到新的游戏版本。可选的,该构建任务可以通过定时构建指令触发,也 可以是开发人员通过测试服务端输入构建指令触发。其中,构建工具是一个把源代码生成可执行应用程序的过程自动化的程序(例如将Android app生成apk,市面上的构建工具例如有Soda、蓝盾等等)。通过构建工具可以帮助开发人员 创建一个可重复的、可靠的、可携带的且不需要手动干预的构建。例如,构建 任务可以简单包括如下几个步骤:编译源代码、拷贝class文件到目录和组装构件。每个构建任务都有输入和输出,其中源代码可以作为该构建任务的输入,将组装成可交付软件作为输出,此外,一个步骤的输出也可以作为下一步骤的 输入。如上所述,构建任务中每个步骤的顺序是由相互之间的依赖关系决定的,例如所需class文件没有被编译出来,则不能够进行组建构件,因此,编译任务 需最先进行,最终可以通过源代码生成可执行的应用程序。

[0106] 例如,所述待测试游戏还可以包括不同区域的区域化版本,同时也可以包 括Android系统版和ios系统版等等。上述游戏的不同版本均可以通过编写不同 版本对应的游戏代码并通过构建工具执行相应版本的构建任务得到。即通过本 发明实施例提供的方法,可以针对不同区域版本和不同系统版本的游戏进行自 动化测试,提升游戏的测试效率。

[0107] 可选的,Soda还可以提供一套开发应用编程接口OpenApi供客户端使用,可以用于查询构建任务状态,基于此,开发人员可以通过Python编写一套自动 下载安装包的脚本。请参见图7,图7是本发明实施例提供的一种待测试游戏的 安装包下载流程示意图,该下载流程可具体包括以下步骤s11-s14:

[0108] 步骤s11,解析配置文件。

[0109] 具体的,测试服务端解析配置文件,该配置文件为游戏自动测试任务开始 前的准备工作,用于确定待测试游戏的描述信息,所述描述信息可以包括待测 试游戏的版本信息、名称信息、路径信息、操作系统信息和游戏引擎信息等等。例如可以通过解析配置文件,从配置文件中读取待测试游戏的安装包名称、安 装包版本、安装包路径等等。

[0110] 可选的,该配置文件可以是指定下载当前服务器中待测游戏的最新版本,即所述待测试游戏可以为该待测游戏开发过程中构建的最新版本。例如,在一 款游戏的开发过程中,为保证开发进度和效率,最常见的测试方式是针对该游 戏的每日版本冒烟测试。所述每日版本可以是游戏开发人员每日针对该游戏中 的缺陷修改或添加新的代码,更新该游戏的代码库后,通过构建工具根据新的 代码库执行相应构建任务得到的该游戏的每日版本。则在每日版本的冒烟测试 中,所述待测试游戏可以为根据当前最新的代码库构建所得的最新游戏版本。可以理解的是,在实际情况中,因为游戏开发的难度和复杂程度不同,该 每日 版本也可以是每2-3天更新一次。

[0111] 可选的,该配置文件也可以是指定下载当前服务器中的任意游戏的任意版 本,本发明实施例对此不作具体限定,开发人员可以根据实际需求设置相应的 配置文件。

[0112] 步骤s12,查询构建任务状态。

[0113] 具体的,测试服务端根据解析配置文件得到的所述描述信息确定待测试游 戏的构建任务状态,即确定待测试游戏的构建任务是否已完成,是否已生成相 应的安装包可供本次测试任务下载使用。

[0114] 步骤s13,推送下载失败消息。

[0115] 具体的,若测试服务端查询到待测试游戏的构建任务还未完成,则测试服务端推送本次测试任务安装包下载失败的消息至一个或多个预设地址。

[0116] 可选的,若待测试游戏的构建任务还未完成,则游戏自动测试任务中断,在推送下载失败消息后,开发人员可以通过测试服务端输入测试指令,即手动触发游戏自动测试任务。

[0117] 例如,测试服务端可以通过即时通讯软件等将下载失败的消息推送至该游戏开发小组的通讯群中。请参见图8,图8是本发明实施例提供的一种下载失败消息的自动推送界面示意图。如图8所示,可以在游戏开发小组群里以测试机器人的方式推送安装包下载失败消息,以便开发人员及时掌握本次自动测试任务的状态和进度。该下载失败消息可以包括待测试游戏的版本号、任务开始、结束时间和下载失败原因等。除此之外,如图8所示,该下载失败消息还可以包括测试任务终止,请手动触发测试任务的提示信息。所述下载失败原因可以为待测试游戏正在构建,无法下载;或者待测试游戏还未开始构建,无法下载;又或者待测试游戏构建失败(例如可以为游戏代码错误或构建工具崩溃等),无法下载等。

[0118] 步骤s14,下载待测试游戏的安装包。

[0119] 具体的,若测试服务端查询到待测试游戏的构建已完成,即已生成待测试游戏的安装包,则测试服务端从服务器下载所述待测试游戏的安装包。

[0120] 可选的,测试服务端可以根据解析配置文件得到的安装包版本信息从服务器下载该版本信息对应的安装包,还可以根据安装包路径信息从服务器下载该路径对应的安装包,等等,此处不再赘述。

[0121] 例如,请参见图9,图9是本发明实施例提供的一种安装包下载过程示意图。如图9所示,在测试服务端下载待测试游戏的安装包过程中,在测试服务端的界面中可以实时监控下载过程。如图9所示,所述 iGame_N0178_0.12.0.10869_Google_GLOBAL.apk为所述待测试游戏的安装包名称,右侧的百分值可以表示待测试游戏安装包的下载进度,以便开发人员可以直观的掌握待测试游戏安装包的下载情况。

[0122] 可选的,若查询到待测试游戏的构建已完成,即已生成待测试游戏的安装包,测试服务端还可以加载与所述待测试游戏匹配的测试用例和测试框架,所述测试用例和测试框架可以存储于测试服务端本地,也可以存放至服务器云端,此处不再进行赘述。

[0123] 步骤S503,根据描述信息确定与待测试游戏的操作系统信息匹配的至少一个测试终端。

[0124] 具体的,根据描述信息确定与待测试游戏的操作系统信息匹配的至少一个测试终端。可选的,所述操作系统信息可以包括微软手机windows phone系统、塞班Symbian系统、安卓Android系统和苹果手机ios系统中的任何一种。

[0125] 例如,如图5B所示,该测试服务端为PC,与所述PC连接的有多部手机,例如其中可以包括微软手机系统的手机和安卓系统的手机等,而该待测试游戏的操作系统信息为安卓系统,则测试服务端可以确定与之连接的至少一个安卓系统的手机为所述与待测试游戏的操作系统信息匹配的至少一个测试终端。又例如,还可以根据所述操作系统信息中的具体版本号,例如Android 7.0、Android8.0和Android 9.0等等,确定更加具体的测试终端,得到更细致、更高质量的测试结果。

[0126] 步骤S504,向至少一个测试终端发送安装包,并控制至少一个测试终端安装安装包,以及向至少一个测试终端发送测试用例,并控制至少一个测试终端执行针对待测试游戏的至少一个操作事件。

[0127] 具体的,步骤S504可以参考上述图4A实施例中的步骤S402,此处不再赘述。

[0128] 例如,在Android系统的游戏自动测试中,所述Android手机可以根据所述PC的ADB命令进行待测试游戏的安装包的安装。可以理解的是,所述Android手机需要选用在安装时没有输入安全密码的手机,以便根据PC的命令执行相应的操作。可选的,在ios系统的游戏自动测试中,所述iphone手机可以通过Mac的ideviceinstaller命令进行待测试游戏的安装。上述命令(ADB命令和ideviceinstaller命令)均可以整合到该游戏自动测试的测试脚本中,以用于实现对应用程序的安装和卸载。卸载命令例如可以为“Adb uninstall待测试游戏”和“ideviceinstaller-U待测试游戏.ipa-o测试终端的UDID”。例如,如图5B所示,在与步骤S504对应的步骤S504-a,测试服务端可以向多个手机(例如图5B所示的手机1,……,手机2)发送安装包和测试用例。又如图5B所示,在与步骤S504对应的步骤S504-b、步骤S504-c和步骤S504-d,每个手机可以通过测试服务端的命令接收安装包,并安装该安装包;以及接收电脑发送的测试用例,并根据所述测试用例执行针对待测试游戏的至少一个操作事件,从而获得测试结果。

[0129] 可以理解的是,在进行自动测试之前,开发人员需根据本次测试要求编写具体的测试流程,即测试用例,比如在登录界面,需要执行那些操作(例如输入账号、密码以及点击登录控件等)才能真正登录进入游戏。可选的,开发人员可以通过自动测试工具(例如为自动测试工具GAutomator和自动测试工具Airtest等)根据要求的测试项目和流程编写具体的测试用例文件,该测试用例文件中可以包括相应的按测试流程顺序执行的一系列操作事件,该一系列操作事件可以包括其分别对应的用户界面(User Interface,UI)控件、操作动作(例如可以为针对控件的点击、长按和滑动等等)游戏对象(例如游戏虚拟人物等)以及游戏场景(例如为登录界面、游戏大厅界面和游戏战斗界面等等),所述测试用例文件用于模拟用户的实际操作。可选的,还可以通过自动测试工具的录制与播放的机制,把一系列的操作事件录制成一个或多个录制文件,并将所述一个或多个录制文件整合到游戏的测试用例文件中。在进行测试时,可以通过将所述测试用例加载到与待测试游戏匹配的测试脚本命令中,控制测试终端执行录制文件包含的至少一个操作事件,以执行对待测试游戏的自动化测试。

[0130] 可选的,请参见图10,图10是本发明实施例提供的一种录制流程示意图。如图10所示,该游戏可以是待测试游戏,即本次测试用的游戏版本,也可以是同一游戏的不同版本(例如为该待测试游戏的旧版本)。在进行如图10所示的录制流程时,该游戏可以运行于所述至少一个测试终端,也可以运行于其他可行的终端设备。以自动测试工具GAutomator为例,该自动测试工具GAutomator运行于所述测试服务端。由于GAutomator SDK主要是基于控件识别,即在录制用的游戏版本与测试用的游戏版本其控件标识(例如可以为控件的名称、序号和坐标等等)相同的情况下,可以采用与测试用的游戏版本不同的游戏版本进行一系列操作的录制,由此,同一份录制文件可以应用于不同游戏版本的自动测试,从而可以节约录制的资源投入。可选的,还可以通过改造SDK控件获取相关底层接口,例如,可以添加对于自定义控件的支持,丰富游戏内容,所述自定义控件为非游戏引擎自带的控

件;又例如,可以添加Index函数,以区分具有相同名称的控件,提升基于控件识别进行游戏测试的质量;还例如,可以改进控件遍历查找的方式,以实现从原本的局部查找到支持全局查找,保证控件识别的全面和准确性。如图10所示,在测试工具初始化后,当游戏在某一界面时(例如可以为用户登录界面和游戏大厅界面等),测试工具开始启动录制,开发人员可以在该界面上开始一系列操作(例如为在用户登录界面上点击用户账号框输入账号、点击用户密码框输入密码以及点击登录的控件等等,又例如为在游戏大厅界面上选择并点击相应的游戏模式,比如“经典模式”、“双人对战模式”等以及点击进入游戏等)。在录制过程中,测试工具可以记录每一个操作事件,具体记录每一个操作事件对应的控件、操作动作、游戏对象和游戏场景等等。如图10所示,在测试工具停止录制后,可以生成一份录制文件,该录制文件包含有上述一系列操作事件。例如,所述录制文件的格式可以为JSON(JsonScript Object Notation)文件,JSON是一种轻量级的数据交换格式,易于人们阅读和编写,同时也易于机器解析和生成。

[0131] 可选的,请参见图11,图11是本发明实施例提供的一种自动测试流程示意图。如上所述,测试服务端可以通过解析事先编写的配置文件,加载所述配置文件指定的与待测试游戏匹配的测试框架(例如为图11所示的自动测试工具GAutomator)。如图11所示,可以采用自动测试工具GAutomator完成对待测试游戏的自动测试,自动测试工具GAutomator具备以下优点:一是自动测试工具GAutomator支持UE4;二是自动测试工具GAutomator基于游戏控件识别,准确度高。如图11所示,在测试终端完成待测试游戏的安装后,通过测试工具自动拉起游戏,即在测试终端启动待测试游戏。并运行待测试游戏内的软件开发工具包(Software Development Kit,SDK),所述SDK为执行待测试游戏的构建任务时,集成在待测试游戏的安装包内的模块。如图10所示,在初始化GAutomator SDK后,等待测试工具与待测试游戏内的SDK通过套接字Socket(网络上的两个程序通过一个双向的通信连接实现数据的交换,这个连接的一端称为一个socket)建立测试连接。测试时,测试工具与待测试游戏内的SDK通过该Socket连接互相之间传递命令以及测试结果。其次,在Socket连接建立完成后,测试终端可以根据测试工具的测试脚本命令执行待测试游戏的测试。可选的,测试终端可以根据开发人员编写的测试用例文件中的测试流程在待测试游戏中执行相应的测试项目。可选的,测试终端也可以根据测试服务端播放的录制文件执行该录制文件中包含的一个或多个操作事件,以完成对游戏的自动化测试。如上所述,该录制文件可以是测试服务端在开始自动测试任务之前针对待测试游戏的测试流程录制的包含有相应操作事件的录制文件。例如,在测试过程中,SDK根据在测试终端识别到的当前游戏场景(例如为用户登录界面)和当前控件(例如为登录控件),驱动测试终端执行与其对应的录制文件中的操作事件(例如为在用户登录界面点击登录控件)。可选的,所述测试用例文件可以包括多种游戏测试情景,例如每日版本冒烟测试、每日版本内存监测(内存监测为检测内存的稳定度,同时还可以测试记忆的储存与检索资料的能力,还可以检测终端上正在使用的内存到底可不可信赖。)和特定场景崩溃测试(例如可以为重复多次的进入战斗和退出战斗,测试游戏的抗崩溃性能)等等。上述测试情景均可通过编写的测试用例文件实现。

[0132] 需要说明的是,测试工具(例如为自动测试工具GAutomator)可以为多种游戏提供自动测试框架,辅助游戏测试人员完成对待测试游戏的测试。但在对UE4这类游戏引擎的

游戏测试上,各项功能还不是很完善,然而,随着用户对 对游戏画面精美度的要求越来越高,依托于强大画面优势的UE4开发的游戏越来越多,对UE4版本的游戏测试需求也日益攀升。基于此,本发明还提供了一种针对自动测试工具GAutomator-UE4版本的改进方案,对自动测试工具 GAutomator-UE4版本进行了部分改进,使得自动测试工具GAutomator-UE4版本可以支持对ios系统的待测试游戏进行自动测试。其中包括:对SDK:增加 ios宏控制,解决ios编译问题。解决ios的网络连接问题。所述网络连接问题包括:解决底层socket创建连接时buffer过大,导致连接失败问题;解决 python->iproxy->sdk整个过程中网络异常中断,重连问题。对Python:解决ios 系统物理分辨率(硬件所支持的分辨率)和逻辑分辨率(软件所偶能达到的分辨率,一般小于物理分辨率)的转换,能正常的从屏幕坐标转换到游戏内坐标,主要用于滑动控件,需要精确获取滑动了多少距离。

[0133] 由此,本发明实施例可以实现对各类主流引擎和各类操作系统的待测试游戏进行自动测试,测试目标覆盖范围广,测试结果质量高,测试效率高。可以理解的是,本发明实施例中的测试框架可以是由现有测试工具提供的,也可以是通过该改进方案对测试工具进行改造后提供的测试框架。

[0134] 请参见图12,图12是本发明实施例提供的另一种自动测试流程示意图,如图12所示,所述测试用例可以包括多个录制文件,所述多个录制文件可以根据测试流程通过脚本命令串联起来。该多个录制文件可以分别对应针对待测试游戏的相关操作事件,即可以通过播放所述多个录制文件控制测试终端执行相应的操作事件,完成待测试游戏的测试。例如,如图12所示,在工具侧,即测试服务端侧可以通过播放“登录游戏”json,控制运行待测试游戏的测试终端执行登录游戏操作,并进入大厅。在测试服务端检查到进入大厅成功后,测试服务端再播放“设置游戏模式并进入战斗”json,控制测试终端设置游戏模式并进入战斗,同时,测试服务端可以检查是否进入战斗。在自动战斗结束后,测试服务端检查是否战斗结束,若结束,则播放“返回大厅”json,并控制测试终端执行返回大厅操作,最后结束测试。至此,完成了基于播放事先录制的多个操作事件对应的录制文件对待测试游戏进行自动测试的流程。如图12所示,进入战斗后的自动战斗可以由游戏内的逻辑驱动,请参见图13,图13是本发明实施例提供的一种游戏内自动战斗的流程示意图。如图13所示,游戏内部可以启动一个自动测试线程,通过解析测试工具传递的模式参数,每帧可以驱动游戏内的虚拟人物执行各种操作,例如走动、遇到物品拾取、遇到敌人开火等等。针对具体的测试场景还可以有多种测试模式,例如打点跑图(指定目标点)、自动往中心跑图等等。对此,本发明实施例还对SDK添加了CALL_GAME_METHOD 协议,并把对应的调用参数传递到游戏内部,进行具体游戏逻辑执行,同时,游戏侧(测试终端侧)可以具体执行测试工具侧(测试服务端侧)的函数调用命令,对于C++侧的函数接口,游戏侧需要指定暴露的接口给测试工具,对于 Lua侧的函数调用,测试工具侧可任意调用。

[0135] 步骤S505,根据至少一个测试终端执行针对待测试游戏的至少一个操作事件的测试结果,生成待测试游戏的测试报告。

[0136] 具体的,步骤S505可以参考上述图4A实施例中的步骤S403,此处不再赘述。

[0137] 可选的,测试服务端可以根据测试终端在执行针对待测试游戏的至少一个操作事件时产生的测试数据,生成该测试数据对应的文字结果。例如,请参见图14,图14是本发明实施例提供的一种游戏测试的文字结果示意图。如图14所示,该文字结果可以为测试报

告的文字内容部分,该文字结果可以包括测试类别,例如图14所示的Android自动冒烟测试。冒烟测试这一术语源自硬件行业,对一个硬件或硬件组件进行更改或修复后,直接给设备加电,如果没有冒烟,则该组件就通过了测试。在软件中,冒烟测试这一术语描述的是在讲代码更改嵌入到产品的源树中之前对这些更改进行验证的过程冒烟测试大多只测试最主要的流程,具体可以包括以下操作事件:启动游戏、登录游戏、进入游戏和结束游戏等等。如图14所示,该文字结果还可以包括待测试游戏的名称信息和版本信息(例如为如图14所示的 iGame_N0166_1.12.0.10867_Development_Google_GLOBAL.apk),还可以包括测试服务器(例如为图14所示的测试3服)、测试开始时间、结束时间、最终结果(例如为图14所示的“冒烟通过,主流程通畅”)以及如图14所示的各个测试节点的时间及其测试结果。显然,该文字结果表明了该游戏版本的 Android自动冒烟测试顺利通过,即开发人员针对该游戏修改或添加的代码没有对该游戏主流程造成影响。

[0138] 例如,请参见图15,图15是本发明实施例提供的另一种游戏测试的文字结果示意图。如图15所示,该文字结果可以为测试报告的文字内容部分,测试类别为针对ios系统的崩溃测试。所述崩溃测试为通过所述待测试游戏在特定的崩溃触发场景中(例如可以在测试终端接到来电或者短信的场景中、也可以在隧道或电梯等网络质量突然变差的场景中、还可以在测试终端内存不足的场景中、还可以在测试服务端通过测试工具对测试终端加载的场景中等等)进行测试,以验证待测试游戏是否会出现无响应等崩溃状况的测试。该文字结果也可以包括待测试游戏的名称信息和版本信息(例如图15所示的 iGame_N0132_1.12.0.10818_Development_AppStore_GLOBAL_enterprise_sign.ip a)。还可以包括测试服务器(例如为图15所示的测试3服),还可以包括游戏模式(例如为图15所示的生存模式),还可以包括游戏测试开始时间、结束时间、最终结果(例如为图15所示的“测试中出现错误!!!”)以及如图15所示的各个测试节点的时间及其测试结果。显然,该文字结果表明了该游戏版本未通过ios崩溃测试,即该游戏版本的抗崩溃性能仍有不足,需要开发人员修改相应的游戏代码从而进行新一轮新的测试。

[0139] 可选的,若待测试游戏的测试过程中出现了游戏漏洞(例如可以为游戏闪退、游戏崩溃、游戏闪退和游戏闪屏等问题),则所述出现游戏漏洞的测试终端会将出现问题的游戏截图发送至测试服务端。若待测试游戏的测试顺利结束,没有出现问题,则测试终端可以不需要向测试服务端发送游戏截图。例如,请参见图16,图16是本发明实施例提供了一种崩溃测试的图片结果。如图16所示,该图片结果为绝地求生PUBGM手游在加载过程中,无法继续加载,游戏无响应,游戏出现崩溃状况的游戏截图。该图片结果可以为测试报告的图片内容部分。

[0140] 步骤S506,确定测试报告对应的测试类别,并向测试类别对应的至少一个预设地址发送测试报告。

[0141] 具体的,确定测试报告对应的测试类别(例如可以包括每日版本冒烟测试、特定场景崩溃测试和每日版本内存测试等等测试类别),并向该测试类别对应的至少一个预设地址发送所述测试报告。例如,如图5B所示,在与步骤S506对应的步骤S506-b,该测试服务端(例如图5B所示的电脑)可以将测试报告发送至地址1、地址2、地址3和地址4等等(比如图5B所示的笔记本电脑、手机、一体式电脑等等)。又例如,具体可以通过即时通讯软件等通信工具,根据测试报告的测试类别,将测试报告发送至对应的游戏测试小组的通讯群

中。

[0142] 例如,请参见图17,图17是本发明实施例提供的一种测试报告的自动推送 界面示意图。如图17所示,所述测试报告(例如具体可以包括如图17所示的 文字内容和待测试游戏崩溃时的游戏画面截图)为针对IOS系统游戏的崩溃测试,则可以通过测试机器人将所述报告推送至如图17所示的游戏崩溃测试小组 通讯群中。可选的,如上所述,若待测试游戏的测试顺利结束,没有出现问题,则测试终端可以不需要向测试服务端发送游戏截图,即所述测试报告可以只包 括文字内容。例如,请参见图18,图18是本发明实施例提供的另一种测试报告 的自动推送界面示意图。如图18所示,两份测试报告分别是本日Android版本内存概况报告(例如包括如图18所示的在登录界面、大厅和训练场抓取的内存数据)和本日ios版本内存概况报告(例如包括如图18所示的在登录界面、大厅和训练场抓取的内存数据),则均可以通过测试机器人将所述报告推送至 如图18所示的游戏性能测试小组通讯群中。从而实现测试结果在第一时间的共享,以便负责各类测试的开发人员及时掌握测试结果,保证开发人员更加有针对性的及时根据所述测试报告对该待测试游戏进行新一轮的改善、优化,实现 测试任务的合理分配,提高测试效率,增强各小组之间的协同合作。可选的,还可以根据游戏区域版本的不同,将本次测试结果推送至相应的不同小组群中, 本发明实施例对此不作具体限定。

[0143] 本发明实施例提供了一种游戏自动测试方法,可以根据本次测试任务中待 测试游戏的描述信息(例如包括待测试游戏的名称信息、版本信息、路径信息、操作系统信息和游戏引擎信息等)获取待测试游戏的安装包,并获取与所述待 测试游戏匹配的测试用例和测试框架,自动完成待测试游戏的测试环境的构建,即实现了在进行针对游戏内容的自动测试前的准备工作的自动化;然后,根据 测试框架中与待测试游戏匹配的测试脚本命令控制后续整个游戏测试流程,其中包括向至少一个测试终端(例如可以是不同品牌、型号的手机、平板电脑等) 发送待测试游戏的安装包,并控制该至少一个测试终端进行安装;其次,还包括向至少一个测试终端发送测试用例,并控制该至少一个测试终端执行该测试用例,以完成对待测试游戏的自动测试,测试用例包括针对待测试游戏的至少 一个操作事件(例如登录游戏、选择游戏模式、开启战斗和退出游戏等等);最终可以根据至少一个测试终端分别执行该至少一个操作事件的测试结果,生 成该待测试游戏的测试报告。由此,对比现有的游戏自动测试方法仅仅实现了 对待测试游戏内容的自动测试,仍需投入大量人力资源完成对待测试游戏的下 载、安装等测试前的准备工作而言。在本发明实施例中可以完成包括待测试游 戏的自动下载、自动安装、自动测试和测试报告自动生成在内的整个自动测试 流程,提升游戏测试的自动化率,减轻游戏开发人员的测试负担。与此同时, 在本发明实施例中还可以同时对不同测试终端执行同一待测试游戏的自动测试,得到同一待测试游戏运行在不同的测试终端上的测试结果,保证游戏测试的高 质量和高效率。

[0144] 基于上述游戏自动测试方法实施例的描述,本发明实施例还公开了一种游 戏自动测试装置,所述游戏自动测试装置可以是运行于终端中的一个计算机程 序(包括程序代码)。请参见图19,图19是本发明实施例提供的一种游戏自动 测试装置的结构示意图,如图19所示,该游戏自动测试装置包括装置10,该装 置10可以执行图4A或图5A所示的方法,所述游戏自动测试装置可以包括: 获取模块107、测试模块109和生成模块112:

[0145] 获取模块107,用于根据待测试游戏的描述信息获取所述待测试游戏的安装 包、

测试用例和测试框架,所述测试用例包括针对所述待测试游戏的至少一个 操作事件,所述测试框架包括与所述待测试游戏匹配的测试脚本命令;

[0146] 测试模块109,用于根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包,并控制所述至少一个测试终端安装所述安装包,以及向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件;

[0147] 生成模块112,用于根据所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件的测试结果,生成所述待测试游戏的测试报告。

[0148] 其中,获取模块107、测试模块109和生成模块112的具体功能实现方式可以参见上述图4A所对应实施例中的步骤S401-步骤S403,获取模块107的具体功能实现方式还可以参见上述图5A所对应实施例中的步骤S502,测试模块109的具体功能实现方式还可以参见上述图5A所对应实施例中的步骤S504,生成模块112的具体功能实现方式还可以参见上述图5A所对应实施例中的步骤S505,此处不再进行赘述。

[0149] 请一并参见图19,所述装置还可以包括:录制模块101和测试用例生成模块102:

[0150] 录制模块101,用于根据预设的测试流程,录制针对所述待测试游戏的至少一个操作事件,并确定所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景;

[0151] 测试用例生成模块102,用于基于所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景生成所述测试用例。

[0152] 其中,录制模块101和测试用例生成模块102的具体功能实现方式可以参见上述图5A所对应实施例中的步骤S504,此处不再进行赘述。

[0153] 请一并参见图19,所述装置还可以包括:构建模块103、查询模块105和执行模块106:

[0154] 构建模块103,用于根据所述待测试游戏的源代码,通过构建工具执行所述待测试游戏的构建任务,生成所述待测试游戏的安装包;

[0155] 查询模块105,用于根据所述待测试游戏的描述信息查询所述待测试游戏的构建任务状态;

[0156] 执行模块106,用于若所述待测试游戏的构建任务已完成,则执行所述根据所述待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架的步骤。

[0157] 其中,构建模块103、查询模块105和执行模块106的具体功能实现方式可以参见上述图5A所对应实施例中的步骤S502,查询模块105的具体功能实现方式还可以参见上述图5A所对应实施例中的步骤s12,执行模块106的具体功能实现方式还可以参见上述图5A所对应实施例中的步骤s14,此处不再进行赘述。

[0158] 请一并参见图19,所述装置还可以包括:描述信息确定模块104:

[0159] 描述信息确定模块104,用于加载并解析预设的配置文件,确定所述待测试游戏的描述信息,所述描述信息包括所述待测试游戏的版本信息、名称信息、路径信息、操作系统信息和游戏引擎信息中的一种或多种。

[0160] 其中,描述信息确定模块104的具体功能实现方式可以参见上述图5A所对应实施例中的步骤S501,此处不再进行赘述。

[0161] 请一并参见图19,所述装置还可以包括:测试终端确定模块109:

[0162] 测试终端确定模块109,用于根据所述描述信息确定与所述待测试游戏的操作系统信息匹配的所述至少一个测试终端。

[0163] 其中,测试终端确定模块109的具体功能实现方式可以参见上述图5A所对应实施例中的步骤S503,此处不再进行赘述。

[0164] 请一并参见图19,所述装置还可以包括:启动模块110和连接模块111:

[0165] 启动模块110,用于根据与所述待测试游戏匹配的测试脚本命令控制所述至少一个测试终端启动所述待测试游戏,并运行所述SDK;

[0166] 连接模块111,用于通过套接字socket与所述至少一个测试终端上运行的所述SDK建立测试连接,所述测试连接用于传递所述测试用例和所述测试结果。

[0167] 其中,启动模块110和连接模块111的具体功能实现方式可以参见上述图5A所对应实施例中的步骤S504,此处不再进行赘述。

[0168] 请一并参见图19,所述装置还可以包括:测试报告发送模块113:

[0169] 测试报告发送模块113,用于确定所述测试报告的测试类别,并向所述测试类别对应的一个或多个目标地址发送所述测试报告。

[0170] 其中,测试报告发送模块113的具体功能实现方式可以参见上述图5A所对应实施例中的步骤S506,此处不再进行赘述。

[0171] 根据本发明提供的实施例,图19所示的游戏自动测试装置中的各个模块可以分别或全部合并为一个或若干个另外的模块来构成,或者其中的某个(些)模块还可以再拆分为功能上更小的多个单元来构成,这可以实现同样的操作,而不影响本发明的实施例的技术效果的实现。上述模块是基于逻辑功能划分的,在实际应用中,一个模块的功能也可以由多个模块来实现,或者多个模块的功能由一个模块实现。在本发明的其它实施例中,基于游戏自动测试装置也可以包括其它模块,在实际应用中,这些功能也可以由其它模块协助实现,并且可以由多个模块协作实现。

[0172] 根据本发明提供的实施例,可以通过在包括中央处理单元(Central Processing Unit,CPU)、随机存取存储介质(random access memory,RAM)、只读存储介质(Read Only Memory,ROM)等处理元件和存储元件的例如计算机的通用测试服务端上运行能够执行如图4A或图5A中所示的相应方法所涉及的各步骤的计算机程序(包括程序代码),来构造如图19中所示的游戏自动测试装置设备,以及来实现本发明实施例的游戏自动测试方法。所述计算机程序可以记载于例如计算机可读记录介质上,并通过计算机可读记录介质装载于上述测试服务端中,并在其中运行。

[0173] 基于上述方法实施例以及装置实施例的描述,本发明实施例还提供一种测试服务端。请参见图20,图20是本发明实施例提供的一种测试服务端的结构示意图。如图20所示,该测试服务端至少包括处理器1001、输入设备1002、输出设备1003以及计算机存储介质1004。其中,测试服务端内的处理器1001、输入设备1002、输出设备1003以及计算机存储介质1004可通过总线或其它方式连接。

[0174] 计算机存储介质1004可以存储在测试服务端的存储器中,所述计算机存储介质204用于存储计算机程序,所述计算机程序包括程序指令,所述处理器1001用于执行所述计算机存储介质1004存储的程序指令。处理器1001(或称CPU(Central Processing Unit,

中央处理器))是测试服务端的计算核心以及控制核心,其适于实现一条或多条指令,具体适于加载并执行一条或多条指令从而实现相应方法流程或相应功能;在一个实施例中,本发明实施例所述的处理器1001可以用于进行一系列的游戏自动测试处理,包括:根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架,所述测试用例包括针对所述待测试游戏的至少一个操作事件,所述测试框架包括与所述待测试游戏匹配的测试脚本命令;根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包,并控制所述至少一个测试终端安装所述安装包,以及向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件;根据所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件的测试结果,生成所述待测试游戏的测试报告,等等。

[0175] 本发明实施例还提供了一种计算机存储介质(Memory),所述计算机存储介质是终端中的记忆设备,用于存放程序和数据。可以理解的是,此处的计算机存储介质既可以包括终端中的内置存储介质,当然也可以包括终端所支持的扩展存储介质。计算机存储介质提供存储空间,该存储空间存储了终端的操作系统。并且,在该存储空间中还存放了适于被处理器1001加载并执行的一条或多条的指令,这些指令可以是一个或一个以上的计算机程序(包括程序代码)。需要说明的是,此处的计算机存储介质可以是高速RAM存储器,也可以是非稳定的存储器(non-volatile memory),例如至少一个磁盘存储器;可选的还可以是至少一个位于远离前述处理器的计算机存储介质。

[0176] 在一种实施方式中,可由处理器1001加载并执行计算机存储介质中存放的一条或多条指令,以实现上述有关游戏自动测试实施例中的方法的相应步骤;具体实现中,计算机存储介质中的一条或多条指令由处理器1001加载并具体执行:

[0177] 根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架,所述测试用例包括针对所述待测试游戏的至少一个操作事件,所述测试框架包括与所述待测试游戏匹配的测试脚本命令;

[0178] 根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包,并控制所述至少一个测试终端安装所述安装包,以及向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件;

[0179] 根据所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件的测试结果,生成所述待测试游戏的测试报告。

[0180] 在一种实施方式中,所述一条或多条指令还可由处理器1001加载并具体执行:

[0181] 确定所述测试报告对应的测试类别,并向所述测试类别对应的至少一个预设地址发送所述测试报告。

[0182] 在一种实施方式中,所述测试结果包括文字结果和图片结果;所述文字结果为所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件时产生的测试数据;所述图片结果为所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件时出现游戏漏洞的游戏截图,所述游戏漏洞包括游戏闪退、游戏卡顿、游戏闪屏和游戏崩溃中的任意一种。

[0183] 在一种实施方式中,所述一条或多条指令还可由处理器1001加载并具体执行:

[0184] 加载并解析预设的配置文件,确定所述待测试游戏的描述信息,所述描述信息包括所述待测试游戏的版本信息、名称信息、路径信息、操作系统信息和游戏引擎信息中的一种或多种。

[0185] 在一种实施方式中,在所述根据与所述待测试游戏匹配的测试脚本命令,向至少一个测试终端发送所述安装包之前,所述一条或多条指令还可由处理器 1001 加载并具体执行:

[0186] 根据所述描述信息确定与所述待测试游戏的操作系统信息匹配的所述至少一个测试终端。

[0187] 在一种实施方式中,在所述根据待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架之前,所述一条或多条指令还可由处理器 1001 加载并具体执行:

[0188] 根据所述待测试游戏的源代码,通过构建工具执行所述待测试游戏的构建任务,生成所述待测试游戏的安装包;

[0189] 根据所述待测试游戏的描述信息查询所述待测试游戏的构建任务状态;

[0190] 若所述待测试游戏的构建任务已完成,则执行所述根据所述待测试游戏的描述信息获取所述待测试游戏的安装包、测试用例和测试框架的步骤。

[0191] 在一种实施方式中,所述安装包包括与所述测试框架匹配的软件开发工具包 SDK;在所述向所述至少一个测试终端发送所述测试用例,并控制所述至少一个测试终端执行针对所述待测试游戏的至少一个操作事件之前,所述一条或多条指令还可由处理器 1001 加载并具体执行:

[0192] 根据与所述待测试游戏匹配的测试脚本命令控制所述至少一个测试终端启动所述待测试游戏,并运行所述 SDK;

[0193] 通过套接字 socket 与所述至少一个测试终端上运行的所述 SDK 建立测试连接,所述测试连接用于传递所述测试用例和所述测试结果。

[0194] 在一种实施方式中,所述一条或多条指令还可由处理器 1001 加载并具体执行:

[0195] 根据预设的测试流程,录制针对所述待测试游戏的至少一个操作事件,并确定所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景;

[0196] 基于所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景生成所述测试用例。

[0197] 在一种实施方式中,所述 SDK 用于通过所述测试连接接收所述测试用例,并根据所述测试用例识别所述至少一个操作事件中的每一个操作事件对应的游戏控件、游戏对象和游戏场景,并驱动所述至少一个测试终端执行所述至少一个操作事件。

[0198] 本发明实施例提供了一种游戏自动测试方法,可以根据本次测试任务中待测试游戏的描述信息(例如包括待测试游戏的名称信息、版本信息、路径信息、操作系统信息和游戏引擎信息等)获取待测试游戏的安装包,并获取与所述待测试游戏匹配的测试用例和测试框架,自动完成待测试游戏的测试环境的构建,即实现了在进行针对游戏内容的自动测试前的准备工作的自动化;然后,根据测试框架中与待测试游戏匹配的测试脚本命令控制后续整个游戏测试流程,其中包括向至少一个测试终端(例如可以是不同品牌、型号的手机、平板电脑等)发送待测试游戏的安装包,并控制该至少一个测试终端进行安装;其

次,还包括向至少一个测试终端发送测试用例,并控制该至少一个测试终端执行该测试用例,以完成对待测试游戏的自动测试,测试用例包括针对待测试游戏的至少一个操作事件(例如登录游戏、选择游戏模式、开启战斗和退出游戏等等);最终可以根据至少一个测试终端分别执行该至少一个操作事件的测试结果,生成该待测试游戏的测试报告。由此,对比现有的游戏自动测试方法仅仅实现了对待测试游戏内容的自动测试,仍需投入大量人力资源完成对待测试游戏的下载、安装等测试前的准备工作而言。在本发明实施例中可以完成包括待测试游戏的自动下载、自动安装、自动测试和测试报告自动生成以及推送在内的整个自动测试流程,提升游戏测试的自动化率,减轻游戏开发人员的测试负担。与此同时,在本发明实施例中还可以同时对不同测试终端执行同一待测试游戏的自动测试,得到同一待测试游戏运行在不同的测试终端上的测试结果,保证游戏测试的高质量和高效率。

[0199] 以上所揭露的仅为本发明较佳实施例而已,当然不能以此来限定本发明之权利范围,因此依本发明权利要求所作的等同变化,仍属本发明所涵盖的范围。

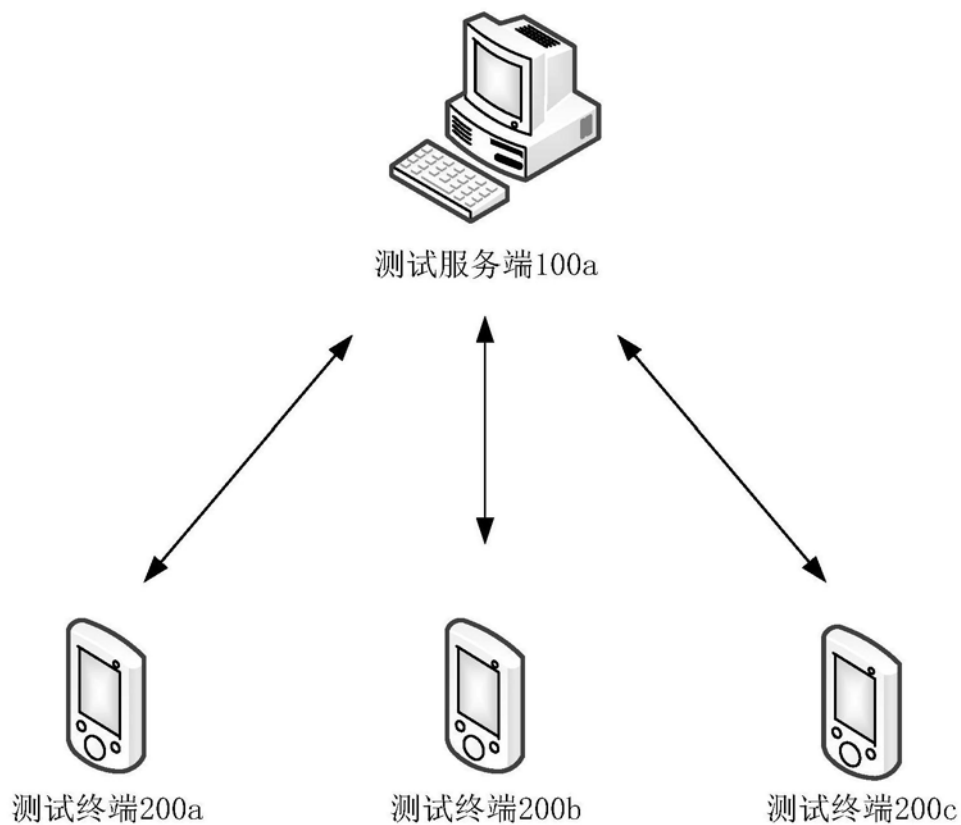


图1

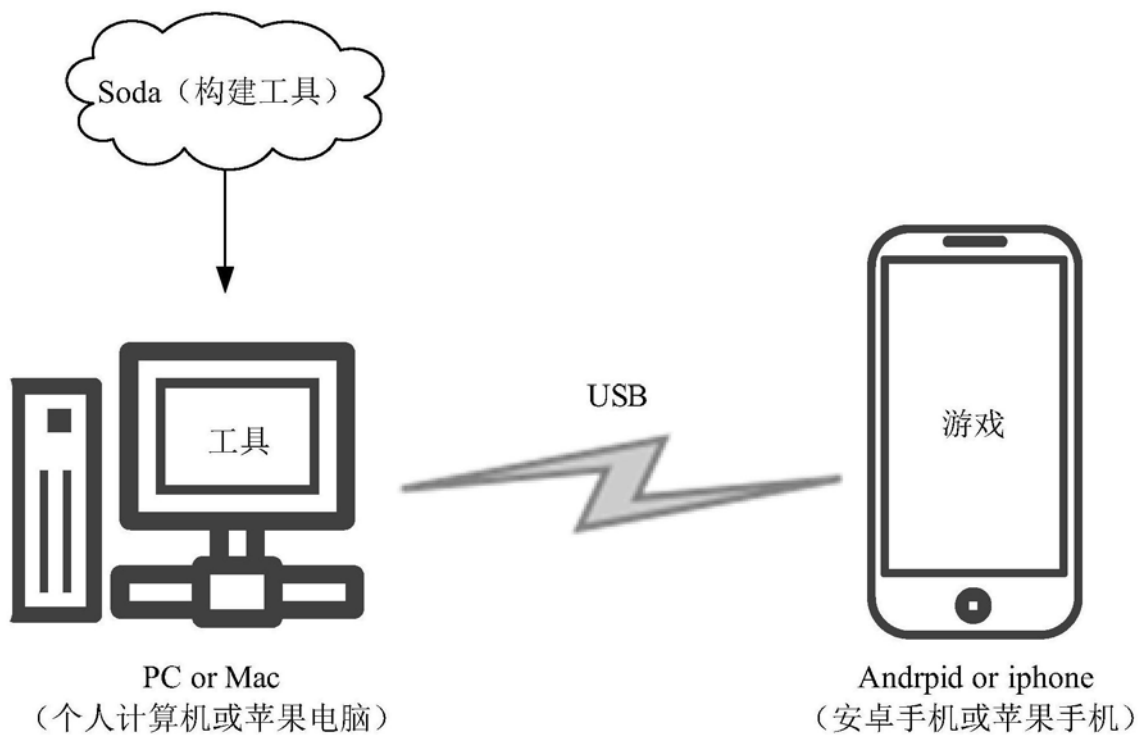


图2

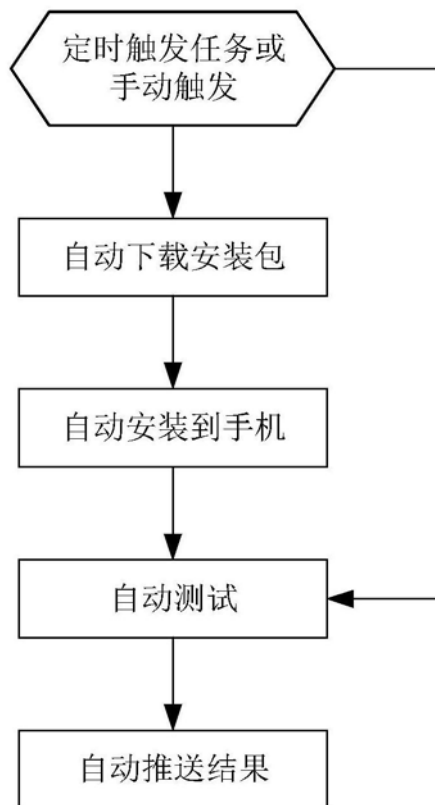


图3

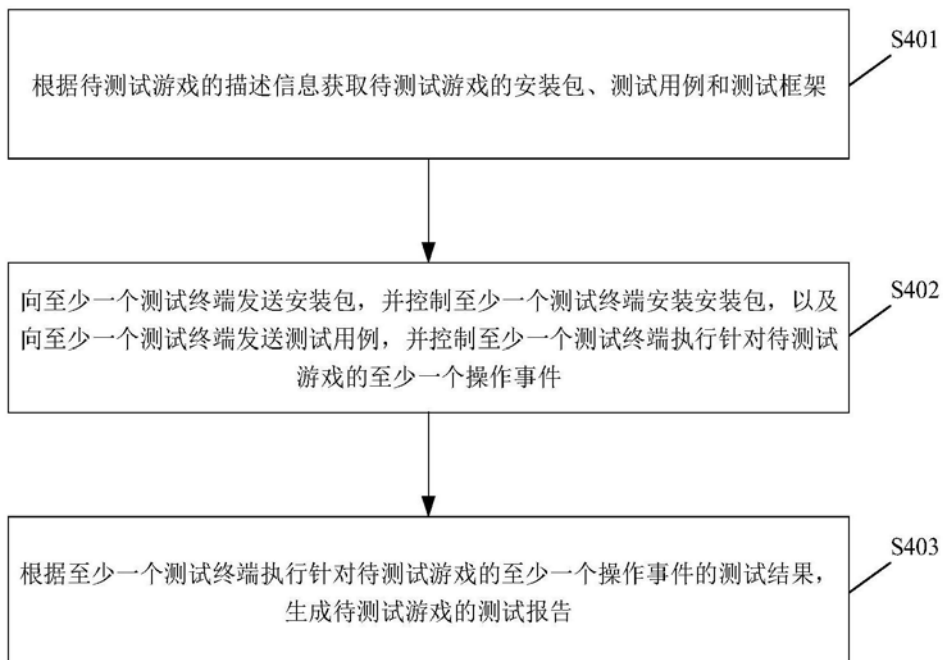


图4A

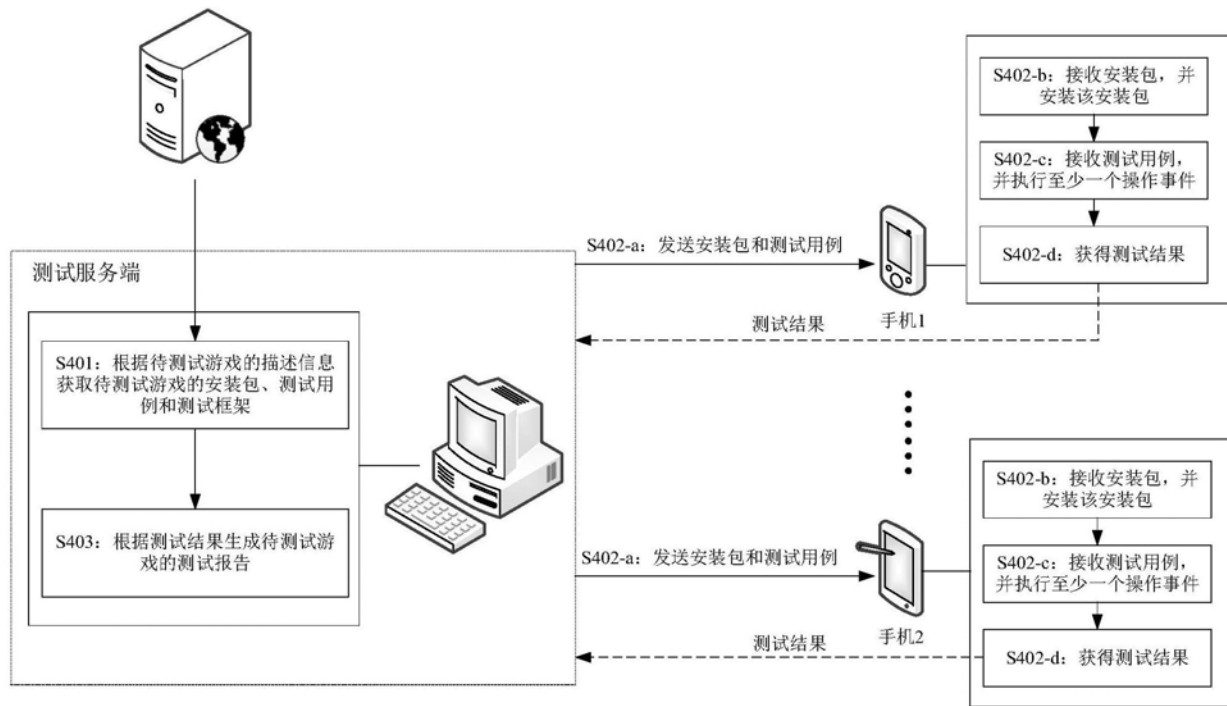


图4B

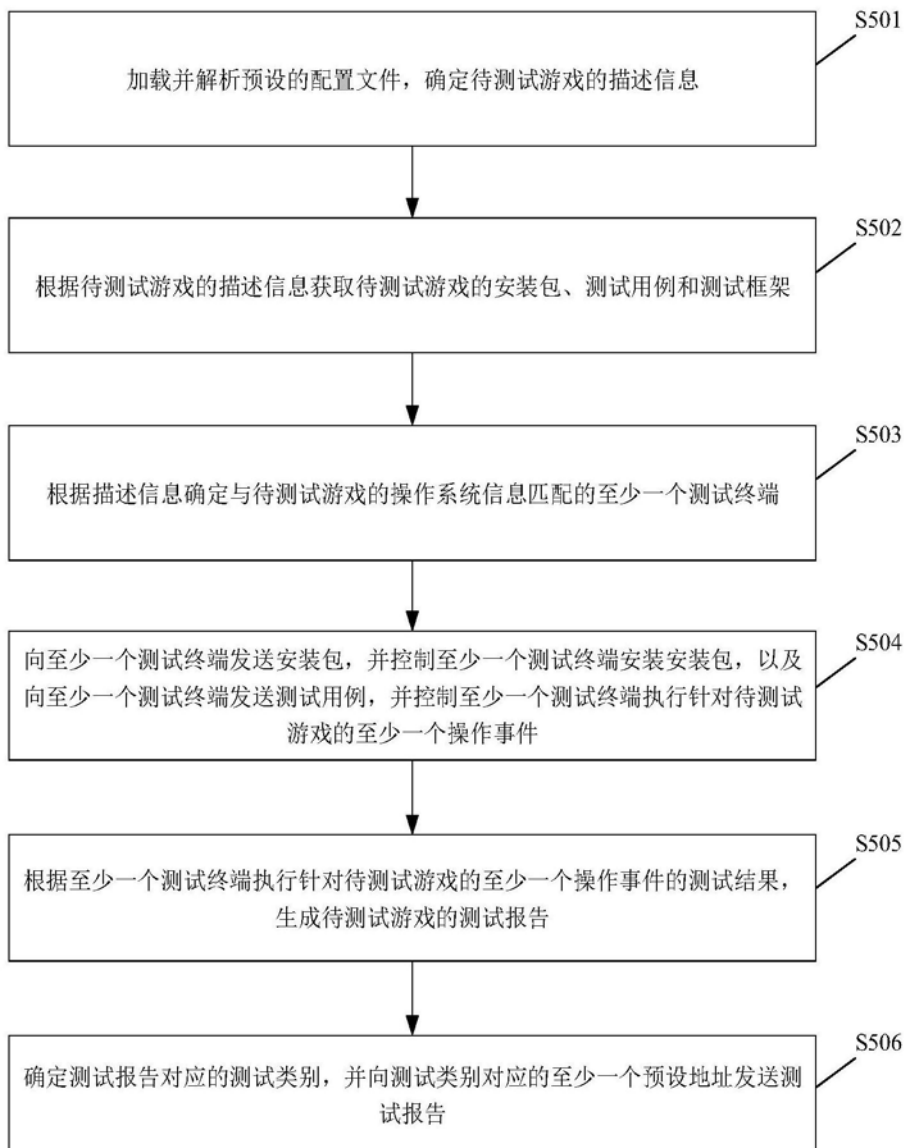


图5A

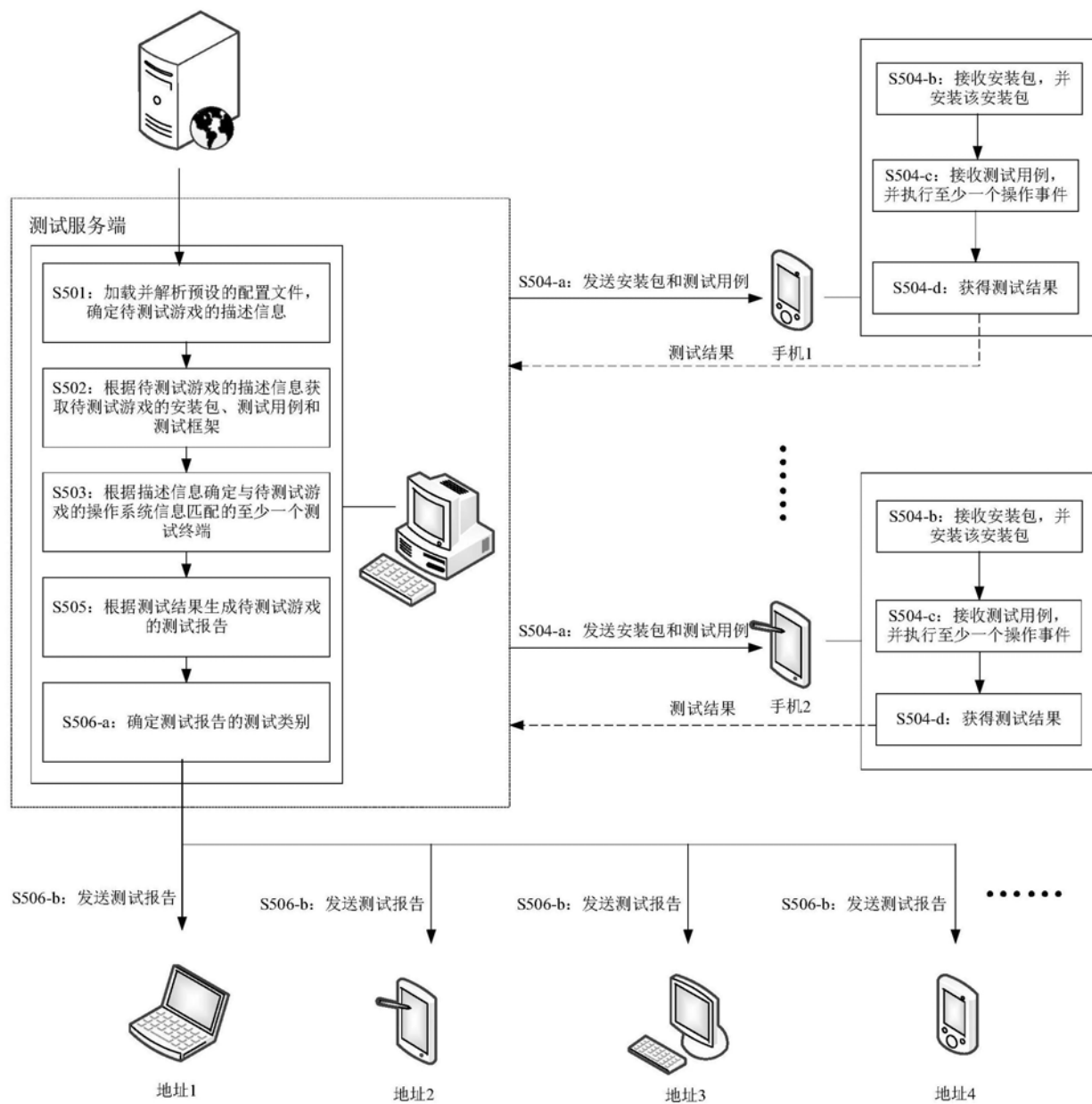


图5B

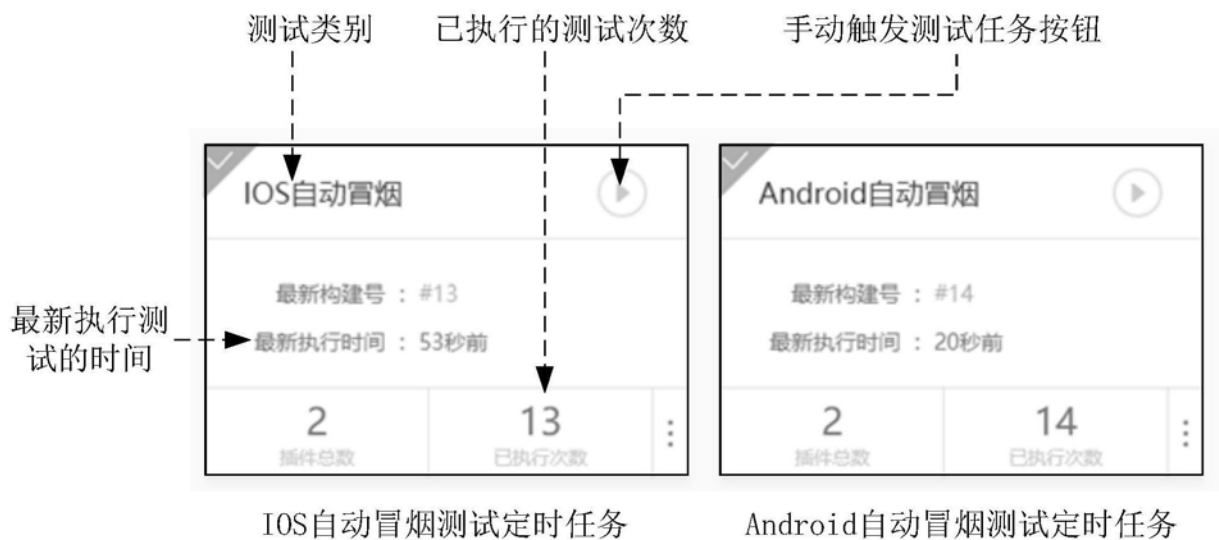


图6

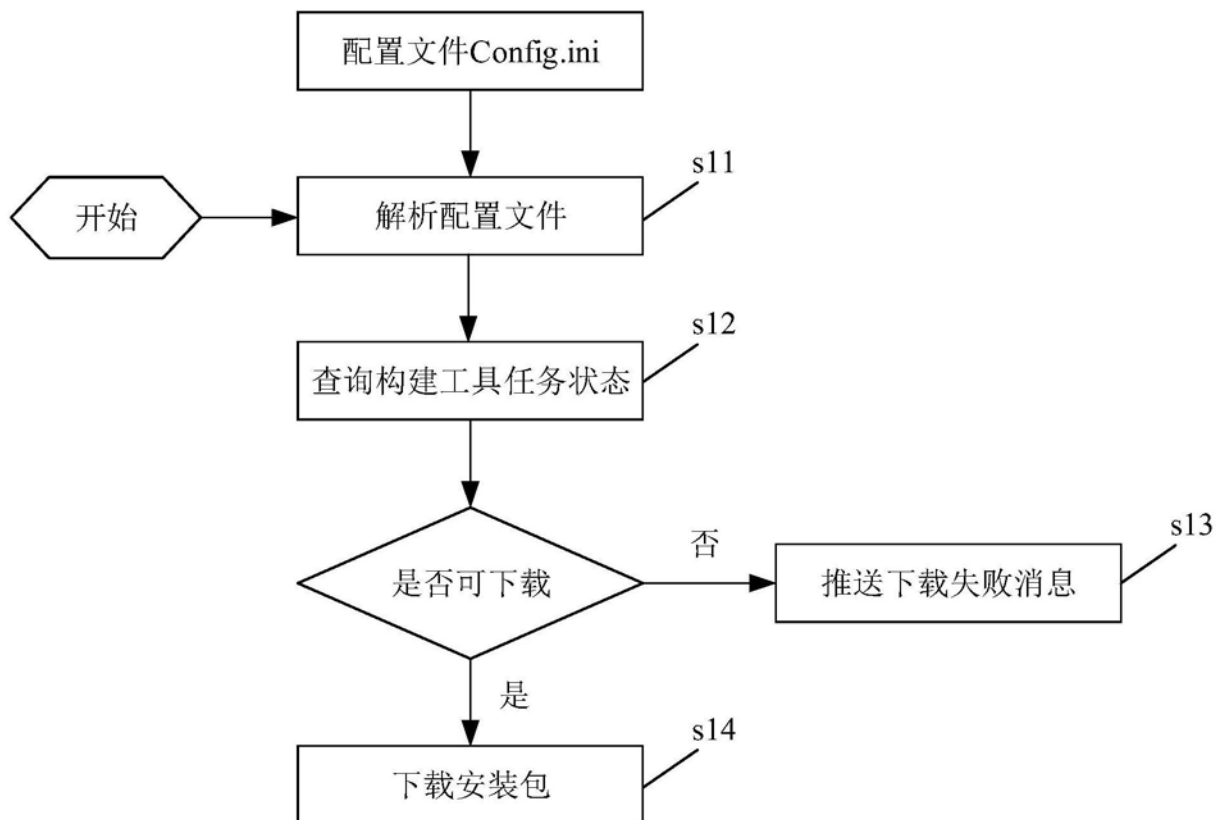


图7

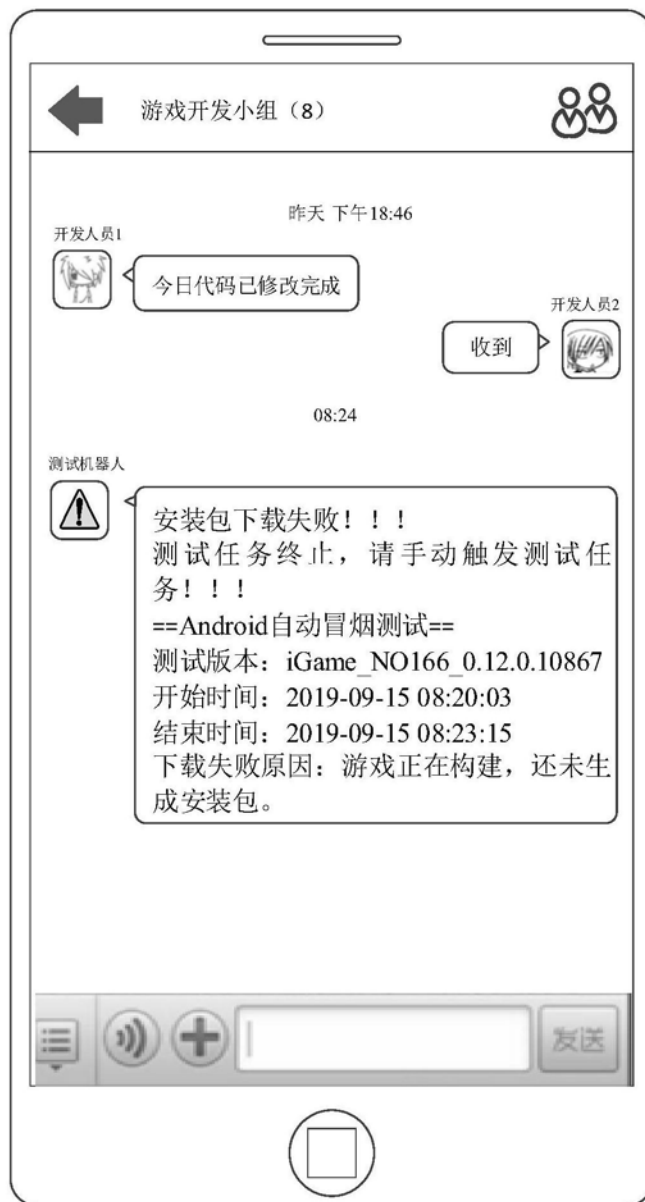


图8



图9

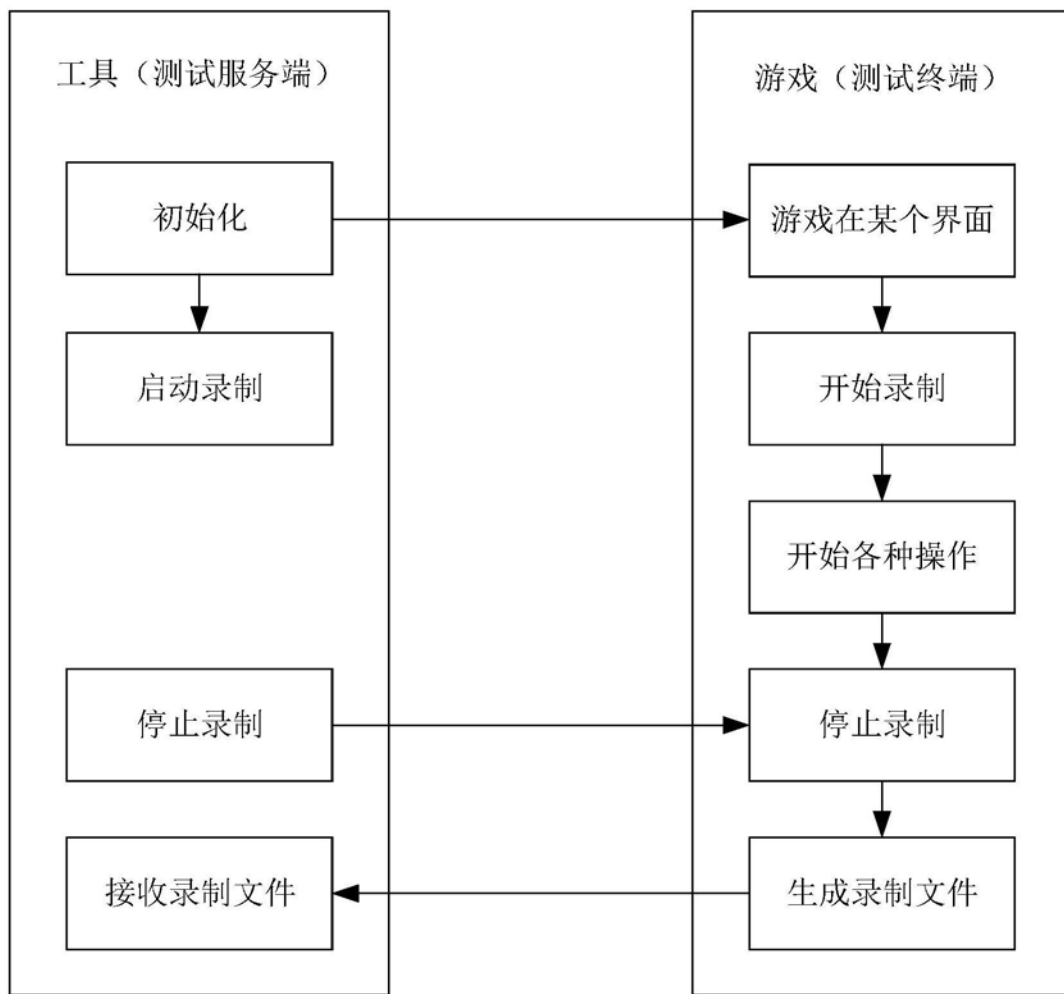


图10

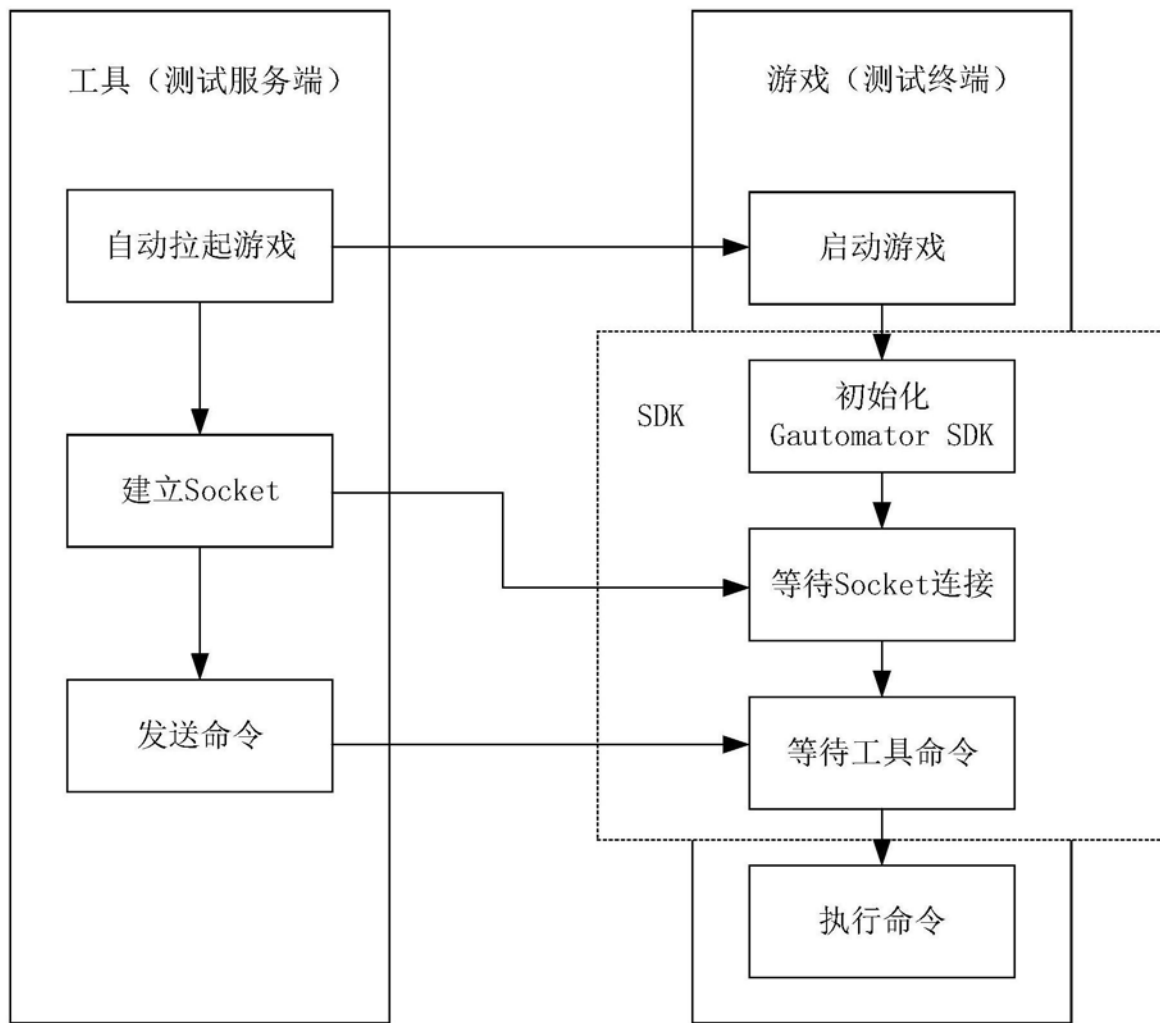


图11

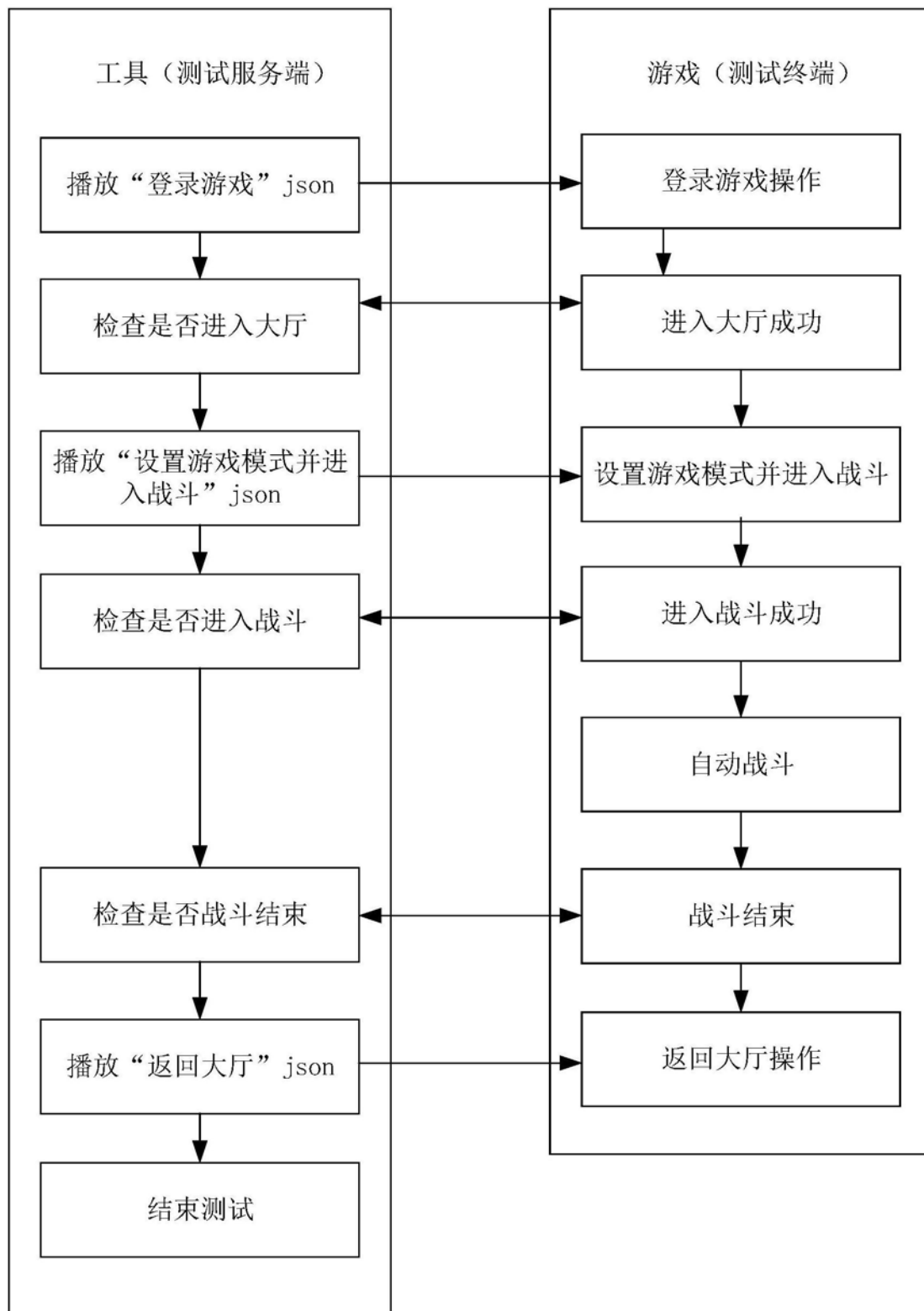


图12

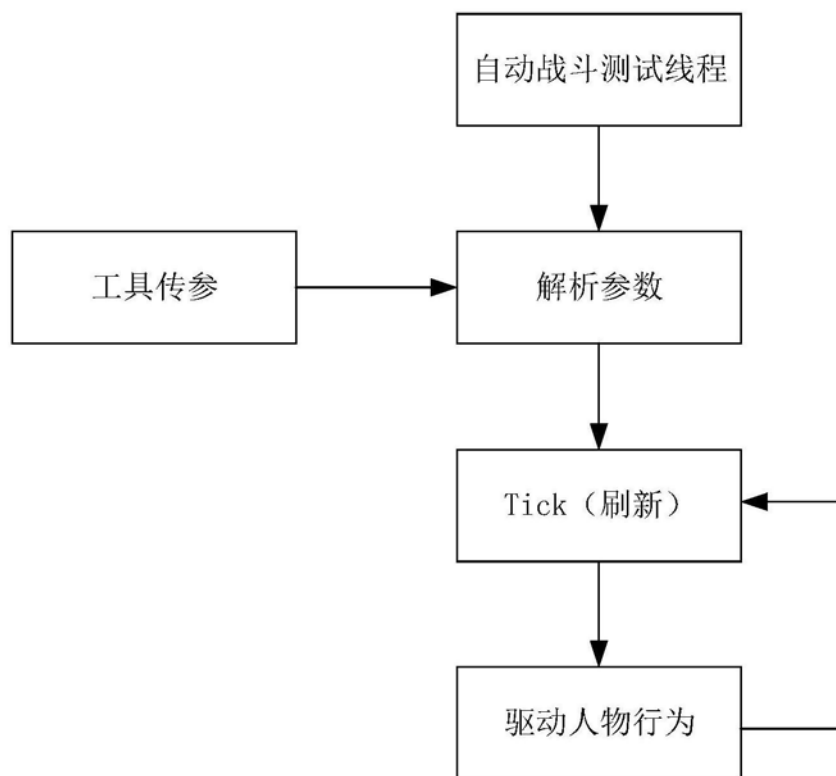


图13



图14

崩溃测试机器人 BOT 3-22 19:20:09

==IOS崩溃测试==

测试版本: iGame_NO132_0.12.0.10818_Development_AppStore_GLOBAL_enterprise_sign.ipa

测试服务器: 测试3服

游戏模式: 生存模式

开始时间: 2019-03-22 19:16:13

结束时间: 2019-03-22 19:20:09

测试结果: 测试过程中出现错误!!!

19:16:13 启动游戏 ✓

19:17:14 登陆游戏 ✓

19:17:14 准备第1次测试战斗 ✓

19:20:09 战斗非正常结束(Crash or DS断开) ×

← IOS崩溃测试
的文字结果

图15



← 崩溃测试的
图片结果

图16

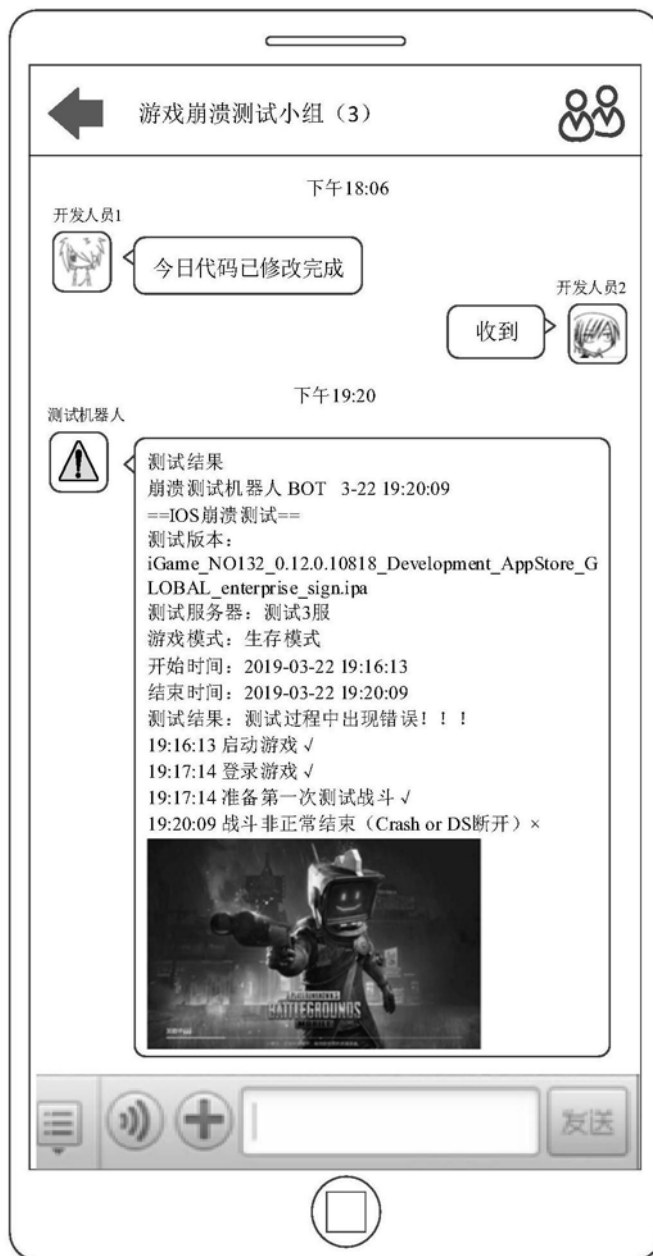


图17

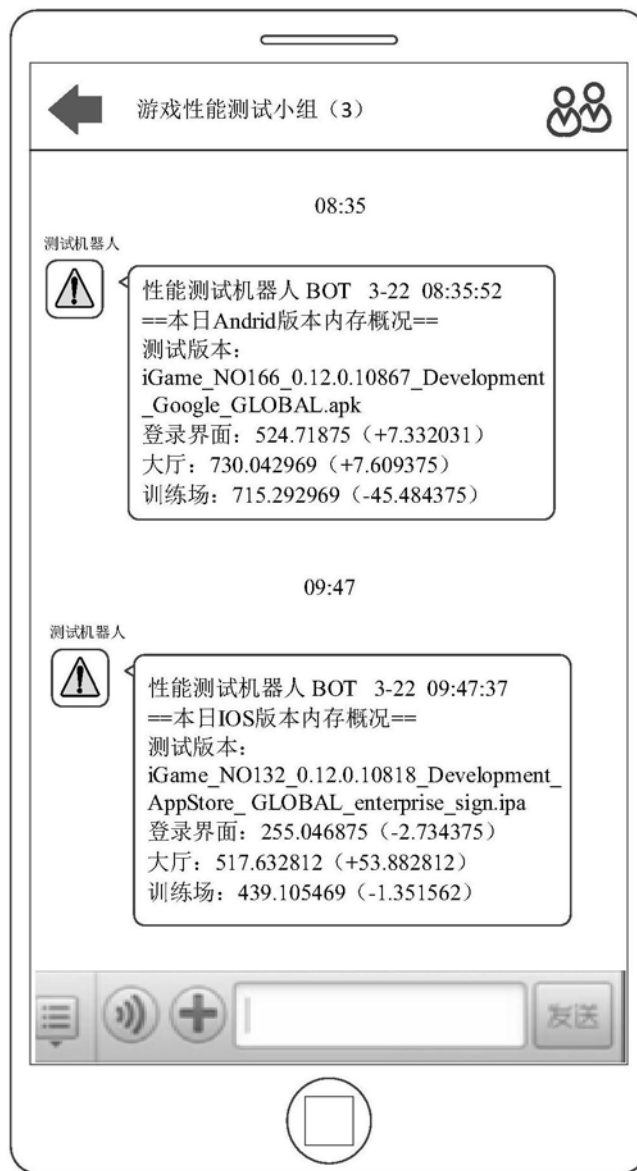


图18

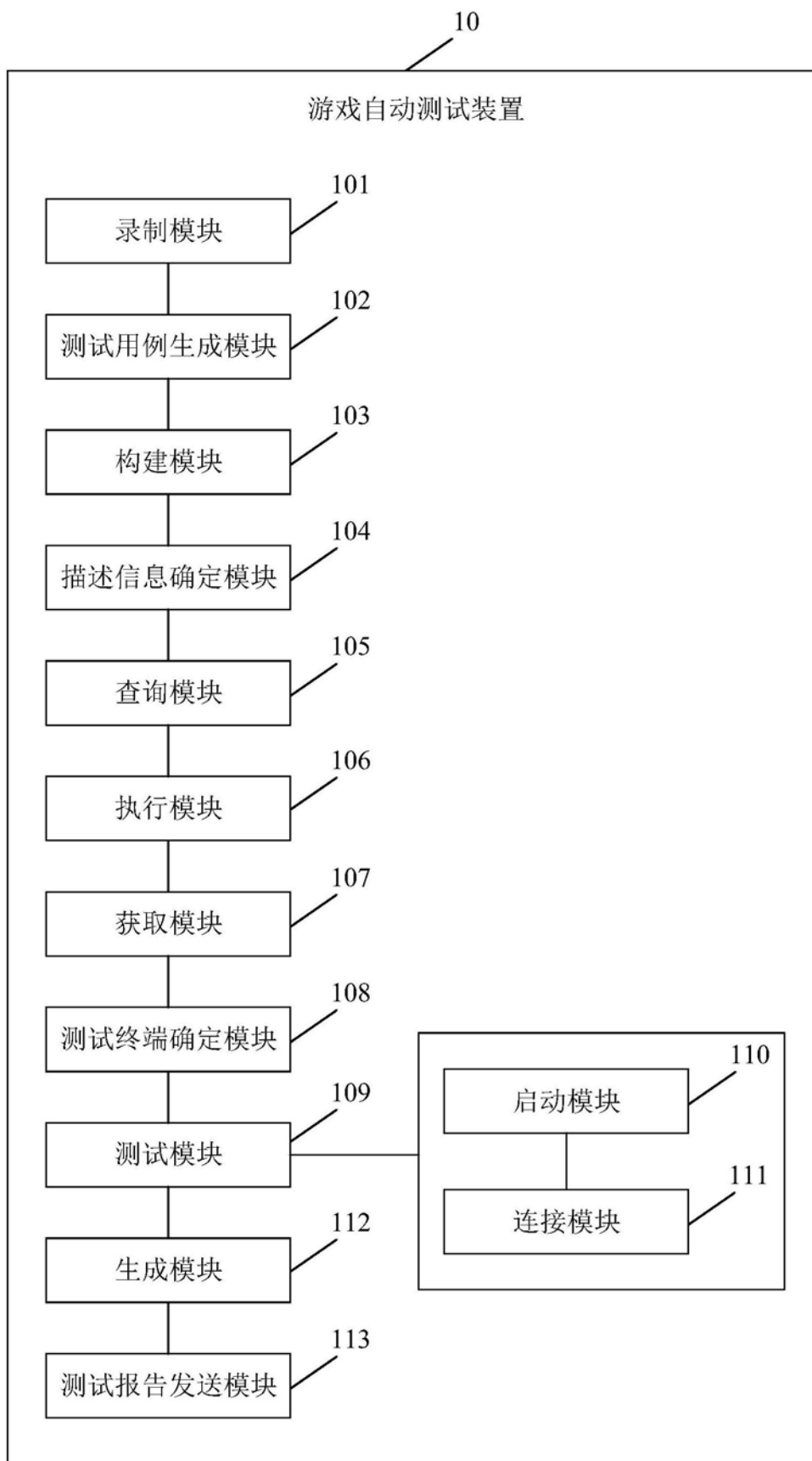


图19

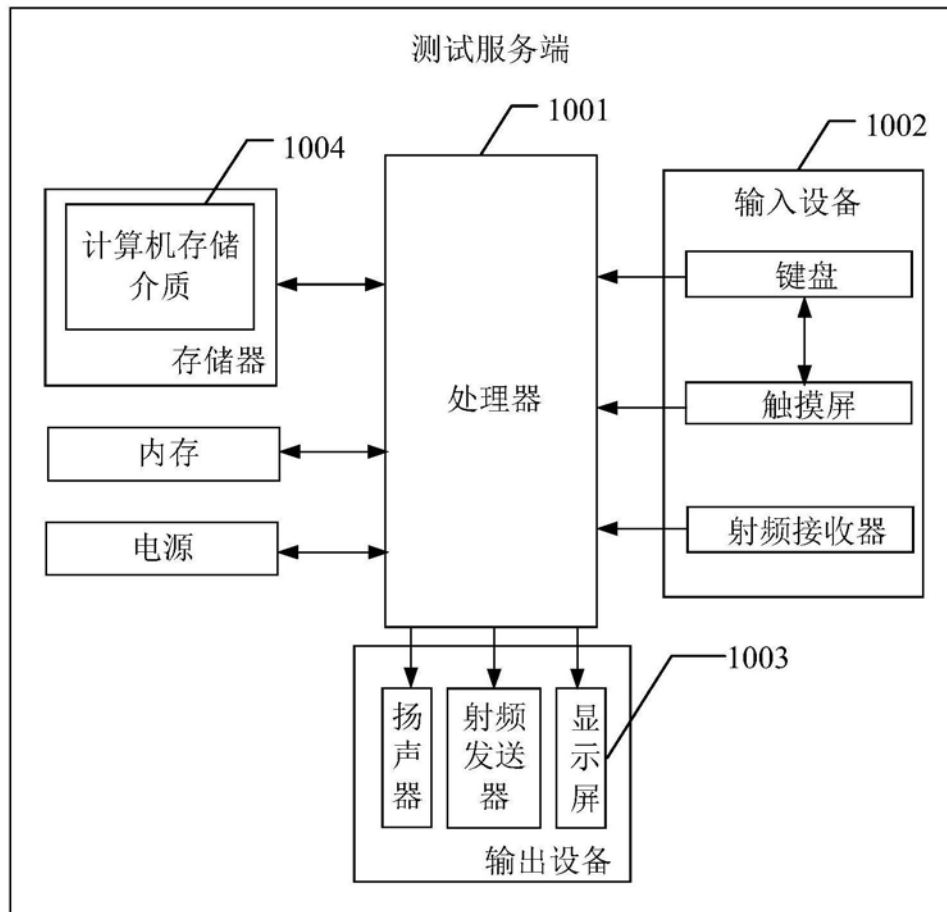


图20