



(12) 发明专利申请

(10) 申请公布号 CN 118860847 A

(43) 申请公布日 2024. 10. 29

(21) 申请号 202310487974.6

(22) 申请日 2023.04.28

(71) 申请人 深圳市腾讯网络信息技术有限公司

地址 518000 广东省深圳市宝安区新安街
道新安六路御景湾花园1栋101、201A、
301、401之二楼201A-K49

(72) 发明人 陈琼雯

(74) 专利代理机构 深圳市智圈知识产权代理事

务所(普通合伙) 44351

专利代理师 韩绍君

(51) Int. Cl.

G06F 11/36 (2006.01)

G06F 3/0483 (2013.01)

G06F 3/04847 (2022.01)

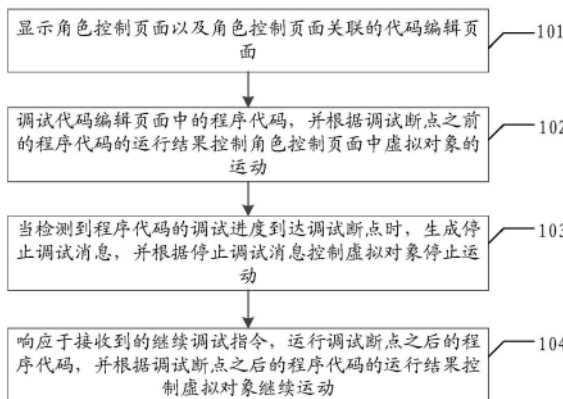
权利要求书3页 说明书29页 附图13页

(54) 发明名称

程序调试方法、装置、设备和计算机可读存
储介质

(57) 摘要

本申请公开了一种程序调试方法、装置、设备和计算机可读存储介质。通过显示角色控制页面以及角色控制页面关联的代码编辑页面,代码编辑页面显示实时编辑的程序代码以及位于程序代码中的调试断点;调试代码编辑页面中的程序代码,并根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动;当检测到程序代码的调试进度到达调试断点时,生成停止调试消息,并根据停止调试消息控制虚拟对象停止运动;响应于接收到的继续调试指令,运行调试断点之后的程序代码,并根据调试断点之后的程序代码的运行结果控制虚拟对象继续运动。以此,可实时调试程序代码,并展示程序代码的控制逻辑效果,提高用户编程进阶学习的效率和体验。



1. 一种程序调试方法,其特征在于,包括:

显示角色控制页面以及所述角色控制页面关联的代码编辑页面,所述代码编辑页面显示实时编辑的程序代码以及位于所述程序代码中的调试断点;

调试所述代码编辑页面中的程序代码,并根据所述调试断点之前的程序代码的运行结果控制所述角色控制页面中虚拟对象的运动;

当检测到所述程序代码的调试进度到达所述调试断点时,生成停止调试消息,并根据所述停止调试消息控制所述虚拟对象停止运动;

响应于接收到的继续调试指令,运行所述调试断点之后的程序代码,并根据所述调试断点之后的程序代码的运行结果控制所述虚拟对象继续运动。

2. 根据权利要求1所述的方法,其特征在于,所述显示角色控制页面以及所述角色控制页面关联的代码编辑页面,包括:

显示角色控制页面,所述角色控制页面至少包含虚拟对象和代码编辑组件;

响应于针对所述代码编辑组件的触发操作,显示所述代码编辑组件对应的代码编辑页面,其中,所述代码编辑页面与所述角色控制页面之间并排分布。

3. 根据权利要求1所述的方法,其特征在于,所述显示所述代码编辑组件对应的代码编辑页面之后,还包括:

响应于在所述代码编辑页面中的输入操作,在所述代码编辑页面中展示实时编辑的程序代码;

获取所述代码编辑页面内被选中的程序代码,并根据调试断点标记所述被选中的程序代码。

4. 根据权利要求3所述的方法,其特征在于,所述代码编辑页面包含动作逻辑库和代码存储文件,所述响应于在所述代码编辑页面中的输入操作,在所述代码编辑页面中展示实时编辑的程序代码,包括:

响应于针对所述动作逻辑库的触发操作,显示所述动作逻辑库对应的动作逻辑列表;

获取所述动作逻辑列表内被选中的目标函数,并以所述目标函数作为程序代码嵌入至所述代码存储文件内;

在所述代码编辑页面展示所述代码存储文件对应的文件信息窗口,所述文件信息窗口展示实时编辑的程序代码。

5. 根据权利要求1至4任一项所述的方法,其特征在于,所述代码编辑页面还包含调试检测组件,所述调试所述代码编辑页面中的程序代码,包括:

响应于针对所述调试检测组件的触发操作,运行所述代码编辑页面中的程序代码;

检测所述程序代码在运行时的调试进度。

6. 根据权利要求1或5所述的方法,其特征在于,所述根据所述调试断点之前的程序代码的运行结果控制所述角色控制页面中虚拟对象的运动,包括:

运行所述调试断点之前的程序代码,并从所述调试断点之前的程序代码中选取所述虚拟对象的动作控制代码;

将所述动作控制代码传输至所述角色控制页面对应的服务器,以使得所述服务器在执行所述动作控制代码后返回对应的动作指令;

基于所述动作指令,控制所述角色控制页面中的虚拟对象进行运动。

7. 根据权利要求6所述的方法, 其特征在于, 所述基于所述动作控制指令, 控制所述角色控制页面中的虚拟对象进行运动, 包括:

若识别到所述动作控制指令为以目标速度的行走指令, 则根据所述目标速度控制所述角色控制页面中的虚拟对象向前行走;

在所述虚拟对象的后方区域显示行走轨迹。

8. 根据权利要求1所述的方法, 其特征在于, 所述代码编辑页面还包括表示代码调试过程的可控状态的调试控制组件, 所述生成停止调试消息, 包括:

暂停运行所述调试断点之后的程序代码, 并将调试状态切换为暂停调试状态;

基于所述暂停调试状态, 将所述调试控制组件的状态图标显示为继续调试状态图标, 并生成所述暂停调试状态对应的停止调试消息。

9. 根据权利要求1或8任一项所述的方法, 其特征在于, 所述生成停止调试消息之后, 还包括:

将所述代码编辑页面中所述调试断点对应的程序代码的行区域进行彩色标记;

获取位于所述调试断点之前的程序代码的运算结果和运行堆栈信息;

在所述代码编辑界面中生成调试信息窗口, 所述调试信息窗口用于显示所述运算结果和运行堆栈信息。

10. 根据权利要求9所述的方法, 其特征在于, 所述根据所述停止调试消息控制所述虚拟对象停止运动, 包括:

将所述停止调试消息发送至所述角色控制页面对应的服务器, 使得所述服务器在解析所述停止调试消息后返回停止运动指令;

基于所述停止运动指令, 控制所述角色控制页面中的所述虚拟对象停止运动。

11. 根据权利要求1所述的方法, 其特征在于, 所述响应于接收的继续调试指令, 运行所述调试断点之后的程序代码, 包括:

响应于接收到的继续运行指令, 运行所述调试断点之后的程序代码, 并生成继续执行消息;

将所述继续执行消息发送至所述角色控制页面对应的服务器, 使得所述服务器恢复所述角色控制页面对应的运行逻辑、物理系统和动画系统。

12. 一种程序调试装置, 其特征在于, 包括:

显示单元, 用于显示角色控制页面以及所述角色控制页面关联的代码编辑页面, 所述代码编辑页面显示实时编辑的程序代码以及位于所述程序代码中的调试断点;

第一调试单元, 用于调试所述代码编辑页面中的程序代码, 并根据所述调试断点之前的程序代码的运行结果控制所述角色控制页面中虚拟对象的运动;

生成单元, 用于当检测到所述程序代码的调试进度到达所述调试断点时, 生成停止调试消息, 并根据所述停止调试消息控制所述虚拟对象停止运动;

第二调试单元, 用于响应于接收到的继续调试指令, 运行所述调试断点之后的程序代码, 并根据所述调试断点之后的程序代码的运行结果控制所述虚拟对象继续运动。

13. 一种计算机设备, 其特征在于, 包括处理器和存储器, 所述存储器存储有计算机程序, 所述处理器用于运行所述存储器内的计算机程序实现权利要求1至11任一项所述的程序调试方法中的步骤。

14.一种计算机可读存储介质,其特征在于,所述计算机可读存储介质存储有多条指令,所述指令适于处理器进行加载,以执行权利要求1至11任一项所述的程序调试方法中的步骤。

15.一种计算机程序产品,其特征在于,包括计算机指令,所述计算机指被执行时实现权利要求1至11任一项所述的程序调试方法中的步骤。

程序调试方法、装置、设备和计算机可读存储介质

技术领域

[0001] 本申请涉及计算机技术领域,具体涉及一种程序调试方法、装置、设备和计算机可读存储介质。

背景技术

[0002] 编程语言作为用户与计算机之间的沟通语言,用户可使用编程语言来编写计算机程序指令,以使得计算机按照计算机程序指令执行对应的程序逻辑,从而完成相应的任务。因此,这不仅要求用户需要了解编程语言,还要求用户能够掌握编程技能。

[0003] 为了满足用户学习进阶代码编程,相关技术一般在用户编写完整程序后,运行该编写的程序,并对程序的运行结果进行显示,以使用户基于显示的运行结果来确定所编写的程序代码的漏洞。

[0004] 在对相关技术的研究和实践过程中,本申请的发明人发现相关技术虽然可通过程序的运行结果来确定是否存在漏洞,但在程序代码存在漏洞时,需要用户逐句检查程序代码,花费较多的时间成本,延长了编程学习周期,降低了用户在编程进阶学习的效率,影响了用户的体验。

发明内容

[0005] 本申请实施例提供一种程序调试方法、装置、设备和计算机可读存储介质,可实时调试程序代码,无需人工逐句检查程序代码,缩短了编程学习周期,提高用户的编程进阶学习的效率,提高用户的体验。

[0006] 本申请实施例提供一种程序调试方法,包括:

[0007] 显示角色控制页面以及所述角色控制页面关联的代码编辑页面,所述代码编辑页面显示实时编辑的程序代码以及位于所述程序代码中的调试断点;

[0008] 调试所述代码编辑页面中的程序代码,并根据所述调试断点之前的程序代码的运行结果控制所述角色控制页面中虚拟对象的运动;

[0009] 当检测到所述程序代码的调试进度到达所述调试断点时,生成停止调试消息,并根据所述停止调试消息控制所述虚拟对象停止运动;

[0010] 响应于接收到的继续调试指令,运行所述调试断点之后的程序代码,并根据所述调试断点之后的程序代码的运行结果控制所述虚拟对象继续运动。

[0011] 相应的,本申请实施例提供一种程序调试装置,包括:

[0012] 显示单元,用于显示角色控制页面以及所述角色控制页面关联的代码编辑页面,所述代码编辑页面显示实时编辑的程序代码以及位于所述程序代码中的调试断点;

[0013] 第一调试单元,用于调试所述代码编辑页面中的程序代码,并根据所述调试断点之前的程序代码的运行结果控制所述角色控制页面中虚拟对象的运动;

[0014] 生成单元,用于当检测到所述程序代码的调试进度到达所述调试断点时,生成停止调试消息,并根据所述停止调试消息控制所述虚拟对象停止运动;

[0015] 第二调试单元,用于响应于接收到的继续调试指令,运行所述调试断点之后的程序代码,并根据所述调试断点之后的程序代码的运行结果控制所述虚拟对象继续运动。

[0016] 在一些实施方式中,所述显示单元,还用于:

[0017] 显示角色控制页面,所述角色控制页面至少包含虚拟对象和代码编辑组件;

[0018] 响应于针对所述代码编辑组件的触发操作,显示所述代码编辑组件对应的代码编辑页面,其中,所述代码编辑页面与所述角色控制页面之间并排分布。

[0019] 在一些实施方式中,所述显示单元,还用于:

[0020] 响应于在所述代码编辑页面中的输入操作,在所述代码编辑页面中展示实时编辑的程序代码;

[0021] 获取所述代码编辑页面内被选中的程序代码,并根据调试断点标记所述被选中的程序代码。

[0022] 在一些实施方式中,所述代码编辑页面包含动作逻辑库和代码存储文件,所述显示单元,还用于:

[0023] 响应于针对所述动作逻辑库的触发操作,显示所述动作逻辑库对应的动作逻辑列表;

[0024] 获取所述动作逻辑列表内被选中的目标函数,并以所述目标函数作为程序代码嵌入至所述代码存储文件内;

[0025] 在所述代码编辑页面展示所述代码存储文件对应的文件信息窗口,所述文件信息窗口展示实时编辑的程序代码。

[0026] 在一些实施方式中,所述代码编辑页面还包含调试检测组件,所述第一调试单元,还用于:

[0027] 响应于针对所述调试检测组件的触发操作,运行所述代码编辑页面中的程序代码;

[0028] 检测所述程序代码在运行时的调试进度。

[0029] 在一些实施方式中,所述第一调试单元,还用于:

[0030] 运行所述调试断点之前的程序代码,并从所述调试断点之前的程序代码中选取所述虚拟对象的动作控制代码;

[0031] 将所述动作控制代码传输至所述角色控制页面对应的服务器,以使得所述服务器在执行所述动作控制代码后返回对应的动作指令;

[0032] 基于所述动作指令,控制所述角色控制页面中的虚拟对象进行运动。

[0033] 在一些实施方式中,所述第一调试单元,还用于:

[0034] 若识别到所述动作控制指令为以目标速度的行走指令,则根据所述目标速度控制所述角色控制页面中的虚拟对象向前行走;

[0035] 在所述虚拟对象的后方区域显示行走轨迹。

[0036] 在一些实施方式中,所述代码编辑页面还包括表示代码调试过程的可控状态的调试控制组件,所述生成单元,还用于:

[0037] 暂停运行所述调试断点之后的程序代码,并将调试状态切换为暂停调试状态;

[0038] 基于所述暂停调试状态,将所述调试控制组件的状态图标显示为继续调试状态图标,并生成所述暂停调试状态对应的停止调试消息。

- [0039] 在一些实施方式中,所述程序调试装置还包括提示单元,用于:
- [0040] 将所述代码编辑页面中所述调试断点对应的程序代码的行区域进行彩色标记;
- [0041] 获取位于所述调试断点之前的程序代码的运算结果和运行堆栈信息;
- [0042] 在所述代码编辑界面中生成调试信息窗口,所述调试信息窗口用于显示所述运算结果和运行堆栈信息。
- [0043] 在一些实施方式中,所述生成单元,还用于:
- [0044] 将所述停止调试消息发送至所述角色控制页面对应的服务器,使得所述服务器在解析所述停止调试消息后返回停止运动指令;
- [0045] 基于所述停止运动指令,控制所述角色控制页面中的所述虚拟对象停止运动。
- [0046] 在一些实施方式中,所述第二调试单元,还用于:
- [0047] 响应于接收到的继续运行指令,运行所述调试断点之后的程序代码,并生成继续执行消息;
- [0048] 将所述继续执行消息发送至所述角色控制页面对应的服务器,使得所述服务器恢复所述角色控制页面对应的运行逻辑、物理系统和动画系统。
- [0049] 此外,本申请实施例还提供一种计算机设备,包括处理器和存储器,所述存储器存储有计算机程序,所述处理器用于运行所述存储器内的计算机程序实现本申请实施例提供的任一种程序调试方法中的步骤。
- [0050] 此外,本申请实施例还提供一种计算机可读存储介质,所述计算机可读存储介质存储有多条指令,所述指令适于处理器进行加载,以执行本申请实施例所提供的任一种程序调试方法中的步骤。
- [0051] 此外,本申请实施例还提供一种计算机程序产品,包括计算机指令,所述计算机指令被执行时实现本申请实施例所提供的任一种程序调试方法中的步骤。
- [0052] 本申请实施例可以显示角色控制页面以及角色控制页面关联的代码编辑页面,代码编辑页面显示实时编辑的程序代码以及位于程序代码中的调试断点;调试代码编辑页面中的程序代码,并根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动;当检测到程序代码的调试进度到达调试断点时,生成停止调试消息,并根据停止调试消息控制虚拟对象停止运动;响应于接收到的继续调试指令,运行调试断点之后的程序代码,并根据调试断点之后的程序代码的运行结果控制虚拟对象继续运动。由此可得,本方案可先显示角色控制页面和关联的代码编辑页面,并通过代码编辑页面显示用户实时编辑的程序代码和设定在程序代码之间的调试断点,然后,对程序代码进行调试,以及根据调试过程的运行结果控制角色控制页面中的虚拟对象进行运动,同时,检测针对程序代码的调试进度,最后,在调试进度达到调试断点时生成停止调试消息,以根据停止调试消息控制角色控制页面中的虚拟对象停止运动,并在检测到针对调试断点之后的程序代码的继续调试指令时,控制虚拟对象继续运动;以此,可根据设定的调试断点来实时调试程序代码,无需逐句检查程序代码,缩短了编程学习周期,提高用户的编程进阶学习的效率,此外,将用户所编辑的程序代码的控制逻辑展示角色控制页面中,提供良好的编程学习方式,提高用户的体验。

附图说明

[0053] 为了更清楚地说明本申请实施例中的技术方案,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,下面描述中的附图仅仅是本申请的一些实施例,对于本领域技术人员来讲,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。

[0054] 图1是本申请实施例提供的程序调试系统的第一场景示意图;

[0055] 图2是本申请实施例提供的程序调试系统的第二场景示意图;

[0056] 图3是本申请实施例提供的程序调试方法的步骤流程示意图;

[0057] 图4是本申请实施例提供的角色控制页面的示意图;

[0058] 图5是本申请实施例提供的代码编辑页面的初始示意图;

[0059] 图6是本申请实施例提供的代码编辑页面与角色控制页面并排分布的场景示意图;

[0060] 图7是本申请实施例提供的包含代码附加组件的代码编辑页面示意图;

[0061] 图8是本申请实施例提供的包含调试检测组件的代码编辑页面示意图;

[0062] 图9是本申请实施例提供的虚拟对象向左摇晃的角色控制页面示意图;

[0063] 图10为本申请实施例提供的虚拟对象向右摇晃的角色控制页面示意图;

[0064] 图11是本申请实施例提供的虚拟对象向前行走的角色控制页面示意图;

[0065] 图12是本申请实施例提供的包含停止调试时的调试信息的代码编辑页面示意图;

[0066] 图13是本申请实施例提供的虚拟对象停止运动的角色控制页面示意图;

[0067] 图14是本申请实施例提供的程序调试方法的另一步骤流程示意图;

[0068] 图15是本申请实施例提供的虚拟对象继续运动的角色控制页面示意图;

[0069] 图16是本申请实施例提供的程序调试方法的模块框架示意图;

[0070] 图17是本申请实施例提供的调试检测组件的程序的安装场景示意图;

[0071] 图18是本申请实施例提供的调试检测组件的程序的安装界面示意图;

[0072] 图19是本申请实施例提供的程序代码的运行流程示意图;

[0073] 图20是本申请实施例提供的调试消息的传输架构示意图;

[0074] 图21是本申请实施例提供的程序调试方法的时序流程图;

[0075] 图22是本申请实施例提供的程序调试装置的结构示意图;

[0076] 图23是本申请实施例提供的计算机设备的结构示意图。

具体实施方式

[0077] 下面详细描述本申请的实施方式,实施方式的示例在附图中示出,其中自始至终相同或类似的标号表示相同或类似的元件或具有相同或类似功能的元件。下面通过参考附图描述的实施方式是示例性地,仅用于解释本申请,而不能理解为对本申请的限制。

[0078] 在说明书、权利要求书和上述附图所描述的一些流程中,包含了按照特定顺序出现的多个步骤,但应该清楚了解,这些步骤可以不按照其在本文中出现的顺序来执行或并行执行,步骤序号仅仅是用于区分各个不同的步骤,序号本身不代表任何的执行顺序。此外,本文中的“第一”和“第二”等描述,是用于区别类似的对象,而不必用于描述特定的顺序或先后次序。

[0079] 下面将结合本申请实施例中的附图,对本申请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申请一部分实施例,而不是全部的实施例。基于本申请中的实施例,本领域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,都属于本申请保护的范围。

[0080] 本申请实施例提供一种程序调试方法、装置、设备和计算机可读存储介质。具体地,本申请实施例将从程序调试装置的维度进行描述,该程序调试装置具体可以集成在计算机设备中,该计算机设备可以是服务器,也可以是用户终端等设备。其中,该服务器可以是独立的物理服务器,也可以是多个物理服务器构成的服务器集群或者分布式系统,还可以是提供云服务、云数据库、云计算、云函数、云存储、网络服务、云通信、中间件服务、域名服务、安全服务、CDN、以及大数据和人工智能平台等基础云计算服务的云服务器。其中,用户终端可以是智能手机、平板电脑、笔记本电脑、台式计算机、智能音箱、智能手表、智能家电、车载终端、智能语音交互设备、飞行器等,但并不局限于此。

[0081] 可以理解的是,在本申请的具体实施方式中,涉及到用户信息、用户使用记录、用户状况等相关的数据,当本申请以上实施例运用到具体产品或技术中时,需要获得用户许可或者同意,且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准。

[0082] 需要说明的是,本申请实施例提供的程序调试方法可应用于任意的编程代码调试场景,如编程进阶学习、编写程序代码等场景,这些场景不限于通过云服务、云控制、游戏控制等一种或多种组合方式实现,具体通过如下实施例进行说明:

[0083] 例如,参见图1,为本申请实施例提供的程序调试系统的第一场景示意图,该场景中的系统可以包括服务器和终端,终端与服务器之间通信连接。其中,终端上安装有编程应用(客户端),该编程应用可用于编辑程序代码以及通过页面展示程序代码的运行效果,服务器上可包括页面展示的运行效果对应的程序代码的执行环境。

[0084] 具体的,终端上的客户端可以显示角色控制页面以及角色控制页面关联的代码编辑页面,代码编辑页面显示实时编辑的程序代码以及位于程序代码中的调试断点;调试代码编辑页面中的程序代码,并根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动;当检测到程序代码的调试进度到达调试断点时,生成停止调试消息,并根据停止调试消息控制虚拟对象停止运动;响应于接收到的继续调试指令,调试调试断点之后的程序代码,并根据调试断点之后的程序代码的运行结果控制虚拟对象继续运动。

[0085] 其中,终端上的客户端在对程序代码进行调试时,终端上的客户端会将角色控制页面中虚拟对象相关的部分程序代码发送至用于角色运行控制的服务器。而后,服务器执行接收到到的该部分程序代码,以生成针对虚拟对的动作指令,从而,将动作控制指令返回至终端上的客户端。最后,终端上的客户端基于该动作指令,对角色控制页面中的虚拟对象进行控制,使得虚拟对象按照动作指令进行运动。

[0086] 需要说明的是,还可通过终端上的浏览器来取代客户端,以通过浏览器(World Wide,web)版的编程来实现以上程序调试过程。

[0087] 例如,以用户的编程进阶学习场景为例,用户可在终端上安装用于编程学习的客户端或浏览器,在编程学习过程中,可对编写的程序代码进行实时调试,而代码调试过程中的代码运行情况会被展示出来,如通过代码编辑页面展示代码运行情况,和/或将代码运行

情况通过动画视图展示出来。具体的,以客户端来实现编程学习为例,假设角色控制页面为游戏动画视图,用户可通过客户端显示游戏动画视图以及关联的代码编辑页面,用户可在该代码编辑页面中编写代码,使得代码编辑页面中实时展示用户编辑的程序代码,以及用户在程序代码中设定的调试断点,该调试断点用于中断整篇程序代码的调试过程;响应于用户触发的代码调试指令,对程序代码进行调试,并检测程序代码在调试过程中的调试消息,当检测到调试进度到达调试断点时,会针对调试断点之后的程序代码停止调试消息,并基于该停止调试消息生成调试提示信息,该提示信息用于提示用户关于程序代码的调试情况,以表示程序代码当前的调试进度停止在调试断点处;用户触发继续调试组件(按钮)时,继续对调试断点之后的程序代码进行调试。需要说明的是,在以上调试过程中,可根据程序代码在调试过程中实时的运行结果来控制游戏动画视图中的虚拟对象进行运动,并根据停止调试消息来控制虚拟对象停止运动,以及根据继续执行的运行结果来控制虚拟对象继续运动。

[0088] 示例性的,用户可通过客户端打开游戏动画页面以及该游戏动画视图关联的代码编辑页面,用户在代码编辑页面中编写程序代码,并在需要单独调试或实时调试的代码程序中设定调试断点,此时,代码编辑页面中会实时显示用户编辑的程序代码和调试断点;进而,用户点击代码编辑页面中的调试检测组件时,触发开始调试指令,对整篇程序代码进行调试,以及检测调试过程中的调试消息,在调试过程中,会根据程序代码调试时的运行结果来控制游戏动画视图中的虚拟对象(如机器狗、机器人等游戏主角)运动,如控制虚拟对象作出摇晃、行走、打招呼、换装备等动作;当程序代码的调试进度到达调试断点时,会生成停止调试消息,并基于该停止调试消息控制游戏动画视图中的虚拟对象停止运动,即停止作出当前的动作,并在用户触发继续执行指令时,控制游戏动画视图中的虚拟对象继续运动,如控制虚拟对象继续前行。以上仅为示例,不作为实施本申请实施方式的限定。

[0089] 又如,参见图2,为本申请实施例提供的程序调试系统的第二场景示意图,该场景中的系统可以包括终端,终端上安装有角色控制页面的程序以及编程环境,通过该角色控制页面的程序可以拉起编程环境,以使用户基于编程环境来编辑程序代码,从而,通过编辑的程序代码来的运行逻辑来控制角色控制页面中的虚拟对象运动。

[0090] 具体的,终端用于显示角色控制页面以及角色控制页面关联的代码编辑页面,代码编辑页面显示实时编辑的程序代码以及位于程序代码中的调试断点;调试代码编辑页面中的程序代码,并根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动;当检测到程序代码的调试进度到达调试断点时,生成停止调试消息,并根据停止调试消息控制虚拟对象停止运动;响应于接收到的继续调试指令,调试调试断点之后的程序代码,并根据调试断点之后的程序代码的运行结果控制虚拟对象继续运动。

[0091] 示例性的,以游戏画面作为角色控制页面,用户可打开客户端来运行游戏进程,使得终端界面显示游戏画面,并触发该游戏画面上的相关组件,以拉起编程环境,并运行编程环境的进程,使得终端界面上展示该编程环境关联的代码编辑页面。此后,用户在代码编辑页面中编写程序代码,并在需要单独调试或实时调试的代码程序中设定调试断点,此时,代码编辑页面中会实时显示用户编辑的程序代码和调试断点;进而,用户点击代码编辑页面中的调试检测组件时,运行调试检测模块的程序,通过全双工通信协议(WebSocket)建立编程进程与游戏进程之间的连接,同时,触发开始调试指令,以开始对整篇程序代码进行调

试,并检测调试过程中的调试消息。具体的,在调试过程中,这针对游戏画面中虚拟对象的动作的部分程序代码,编程环境会通过WebSocket服务将该部分程序代码发送给游戏进程,以使得游戏进程中的控制模块运行该部分程序代码,以对游戏画面中的虚拟对象(如机器狗、机器人等游戏主角)的动作进行控制,如控制虚拟对象作出摇晃、行走、打招呼、换装备等动作。其中,在对程序代码进行调试时,当检测到调试进度到达调试断点时,调试检测模块会获取代码执行模块生成的停止调试消息,并通过WebSocket服务将停止调试消息发给游戏进程的控制模块,而游戏进程的控制模块会基于停止调试消息控制游戏画面中的虚拟对象停止运动,即停止作出当前的动作;此外,在用户触发继续执行指令时,编程进程生成继续调试消息,游戏进程中的控制模块基于继续调试消息恢复游戏画面中虚拟对象的运动,并基于调试断点之后的程序代码,控制游戏动画中的虚拟对象继续运动,如控制虚拟对象继续前行。以上仅为示例,不作为实施本申请实施方式的限定。

[0092] 基于以上,用户可在编程进阶学习过程中对编辑的程序代码进行实时调试,无需人工逐句检查程序代码的漏洞,节省了编程学习的周期,并通过相应的动画视图来展示程序代码的控制逻辑和运行效果,以通过良好的编程学习方式来进行编程进阶学习,提高了用户在编程进阶学习中的体验。

[0093] 为了便于理解,以下将分别对程序调试方法的各步骤进行详细说明。需说明的是,以下实施例的顺序不作为对实施例优选顺序的限定。

[0094] 在本申请实施例中,将从程序调试装置的维度进行描述,以该程序调试装置具体可以集成在计算机设备如终端或服务器中。参见图3,图3为本申请实施例提供的一种程序调试方法的步骤流程示意图,本申请实施例以程序调试装置具体集成在终端上为例,终端上的处理器执行程序调试方法对应的程序指令时,具体流程如下:

[0095] 101、显示角色控制页面以及角色控制页面关联的代码编辑页面。

[0096] 在本申请实施例中,为了提供良好的编程进阶学习途径,可将代码编辑页面与动态视图相结合,使得用户在编程进阶学习中,通过动态视图来展示用户编辑的程序代码在调试过程中的运行效果,以使用户根据每一句程序代码的运算情况的预期视图效果来审视动态视图的展示效果,满足用户在编辑完的程序代码后的快速调试需求,节省了学习周期,使得用户基于良好的编程进阶学习途径来提高学习效率。

[0097] 需要说明的是,为了满足用户在编程进阶学习中的程序代码实时调试需求,用户在编辑程序代码时可在某一句程序代码之前设定调试断点,如在相邻的两句程序代码之间设定调试断点,该调试断点可以理解为调试中断点,用于中断程序代码的调试过程,以实现对调试断点前后的两部分程序代码进行分别调试,以使得用户能够实时审视每部分代码在调试时的运行情况;基于此,用户可根据实际调试需求在任意的程序代码之间设定调试断点,以满足对程序代码的实时调试需求,无需用户人工逐句检查程序代码,节省了学习周期,提高编程学习效率。

[0098] 具体的,用户可通过终端设备上安装的客户端或浏览器等途径来启动编程学习页面,该编程学习页面不限于包括角色控制页面和代码编辑页面,其中,代码编辑页面为用户提供代码编辑环境,当用户在代码编辑页面编写程序代码以及设定调试断点时,代码编辑页面显示实时编辑的程序代码以及位于程序代码中的调试断点;而角色控制页面用于展示每一句程序代码被调试时的运行效果。以此,能够为用户提供良好编程学习途径,使得用户

能够结合动态视图来了解程序代码的情况,提高用户在编程进阶学习的体验。

[0099] 其中,该角色控制页面可以是一种可控制的动态视图,具体的,该角色控制页面可包含受控制的虚拟对象,页面中的虚拟对象可以基于相关的控制指令来运动,从而,实现以动态视图方式来展示对虚拟对象的控制效果。例如,以游戏页面作为角色控制页面为例,该虚拟对象可以是游戏页面中的游戏主角,用户可在编辑程序代码后,调试所编辑的程序代码,以根据调试时的代码运行结果来实时控制游戏页面中的游戏主角进行运动,该虚拟对象可以是虚拟武侠角色、虚拟机器狗、虚拟机器人等;以上仅为示例,不作为本申请实施场景的具体限定。

[0100] 其中,该代码编辑页面可以是编程环境对应的编辑器的显示页面,该页面中至少包括代码编辑区,用户可在该代码编辑页面中的代码编辑区内编写的程序代码,以及在程序代码之间设定调试断点,从而,代码编辑页面显示有实时编辑的程序代码以及位于程序代码中的调试断点,以供用户查看。

[0101] 其中,该程序代码不限于包括动作控制代码和非动作控制代码,具体的,动作控制代码可以理解为用于控制虚拟对象的动作的代码,动作控制代码的运行结果用于控制虚拟对象作出相应的动作,不同的动作控制代码控制虚拟对象作出不同的动作样式;而非动作控制代码可以理解而为与虚拟对象作出的动作样式无关的代码,非动作控制代码不限用作调试或运行时的判断条件,用于辅助动作控制代码的运行。通过运行程序代码,可以基于运行结果来控制角色控制页面中的虚拟对象作出相应的动作。

[0102] 在本申请实施例中,该代码编辑页面与角色控制页面之间是具有关联关系的,具体的,代码编辑页面用于编辑和运行程序代码,以控制角色控制页面中的虚拟对象运动,而角色控制页面则基于虚拟对象的动态视图来反映/表示代码编辑页面中的程序代码的控制逻辑。

[0103] 在本申请实施例中,编程环境与角色控制过程可通过不同的进程来运行,即代码编辑页面与角色控制页面之间属于不同的进程,代码编辑页面与角色控制页面可并排显示在同一视觉界面中。

[0104] 在一些实施方式中,可先显示其中一个页面,并通过线程启动方式来打开另一页面。例如,以通过线程启动代码编辑页面为例,步骤101可以包括:

[0105] (101.1) 显示角色控制页面,角色控制页面至少包含虚拟对象和代码编辑组件;

[0106] (101.2) 响应于针对代码编辑组件的触发操作,显示代码编辑组件对应的代码编辑页面,其中,代码编辑页面与角色控制页面之间并排分布。

[0107] 其中,该虚拟对象可以是任意样式的虚拟物体,如机器狗、机器人、等任意形式的虚拟动画物体,该虚拟对象可根据程序代码的运行情况来变化,如在角色控制页面中作出行走、跳舞、摇晃、停留、蹲坐等相应动作。

[0108] 其中,该代码编辑组件可以是触发程序代码编辑的控件,该代码编辑组件显示在角色控制页面上,用于触发显示代码编辑页面。示例性的,以用于编程进阶学习的客户端为例,用户在终端上打开该客户端后,显示界面中会显示出角色控制页面,该角色控制页面中包含代码编辑组件,当用户需要学习编辑代码时,可点击该代码编辑组件,此时,显示界面上会显示该代码编程组件对应的代码编辑页面。

[0109] 具体的,用户在终端上打开客户端后,显示界面会显示一个角色控制页面,该角色

控制页面中包含一个或多个虚拟对象,此外,该角色控制页面中还至少包括代码编辑组件,当用户点击该代码编辑组件后,会响应于针对该代码编辑组件的触发操作,通过运行线程拉起一个代码编辑环境,并通过进程来运行该代码编辑环境,同时,在显示界面中显示出对应的代码编辑页面。需要说明的是,代码编辑页面和角色控制页面分别对应不同的程序,可通过不同的进程来运行,但该代码编辑页面与角色控制页面之间是具有关联性,通过在该代码编辑页面中编写程序代码并调试、运行,可对角色控制页面中的虚拟对象进行控制,基于此,为了提高用户在编程学习或开发时的便利性,在显示代码编辑页面时,可以将代码编辑页面与角色控制页面相互并排分布,使得代码编辑页面和角色控制页面同时显示在同一显示界面中,以利于用户同时观察代码编辑页面中的程序代码的相关情况和角色控制页面中虚拟对象的动态效果,为用户提供一个良好的编程学习或使用的途径,提高编程学习或使用效率。

[0110] 示例性的,参见图4所示,为角色控制页面的示例图,该角色控制页面中包含的虚拟对象为机器狗,角色控制页面的左侧包含一些控件图标,如包括“返回”组件、“虚拟场景”组件、“代码编辑”组件、“运行”组件等,当用户触发该“代码编辑”组件,终端会通过线程管理来拉起代码编辑器,并通过进程来运行该代码编辑器,同时显示对应的代码编辑页面,如图5所示,该代码编辑页面可包含代码编辑区、功能组件栏、文件目录区等,其中,用户可在代码编辑区内编写一些程序代码。参见图6所示,代码编辑页面和角色控制页面属于并排分布显示在同一显示界面中,且代码编辑页面与角色控制页面之间无重叠覆盖,或至少不覆盖页面重要内容,如代码编辑页面不遮挡角色控制页面中的虚拟对象,或者角色控制页面不遮挡代码编辑页面中的代码编辑区和功能组件栏。需要说明的是,对于多屏幕技术组合得到的一个显示界面,当代码编辑页面和角色控制页面分别显示在不同硬件设备的显示屏幕中时,仍属于分布显示在同一显示界面中。以上仅为示例,不作为实施本申请实施例场景的具体限定。

[0111] 在本申请实施例中,当打开角色控制页面以及关联的代码编辑页面之后,用户可在代码编辑页面中编写程序代码,以使得代码编辑页面中实时显示有用户编辑的程序代码,并当用户在程序代码中设定调试断点时,还会在用户的设定位置显示该调试断点,以表示程序代码在后续调试时会运行至该调试断点处停止调试,以便于用户对相关部分的程序代码进行实时单独调试。

[0112] 在一些实施方式中,代码编辑页面可实时显示用户输入的程序代码和设定的调试断点,以供用户在编辑过程和后续的调试过程中查看,其中,用户可先编辑程序代码,再在程序代码中的特定位置设定调试断点。例如,在步骤(101.2)中的“显示代码编辑组件对应的代码编辑页面”之后,还包括:

[0113] (101.2.1) 响应于在代码编辑页面中的输入操作,在代码编辑页面中展示实时编辑的程序代码;

[0114] (101.2.2) 获取代码编辑页面内被选中的程序代码,并根据调试断点标记被选中的程序代码。

[0115] 其中,在代码编辑页面中的输入操作,不限于包括编写、调用、引用程序代码等操作,此处不作限定。

[0116] 具体的,用户可以在代码编辑页面中选定代码的输入位置,并针对代码编辑页面

中的输入位置进行代码输入操作,此时,终端会响应于用户的输入操作,并基于检测到的输入指令来确定代码内容,并在代码编辑页面中实时显示用户编写的程序代码;此外,为了实现对程序代码的进行实时或单独调试,用户可以在程序代码篇幅中的任意两句程序代码之间的添加调试断点,具体的,用户可以选中代码编辑页面中需要设定调试断点的目标程序代码,并通过调试断点来对选中的目标程序代码进行标记,以使得需要实时、单独或分段调试的程序代码之前标记有“调试断点”,具体可以是在代码编辑页面中的被选中的程序代码之前显示一个“调试断点”的标记图标。需要说明的是,可以在编辑代码过程中设定调试断点,还可以在编辑完所有的程序代码后,根据需求在任意两句程序代码之间设定调试断点,此处不做限定。以此,以便后续在代码调试时基于编辑的程序代码和设定的调试断点进行实时、单独或分段调试,无需人工逐句检查代码的漏洞,提高编程学习效率。

[0117] 示例性的,如图5所示,代码编辑页面中包含代码编辑区,用户可以点击代码存储文件,该代码存储文件用于存储用户编写的程序代码,此时,代码编辑区会展示该文件对应的代码编辑窗口,用户可在代码编辑窗口中选定代码的输入位置,如选定在第1、2、3……或11行,并在针对选定的行位置处进行代码输入,此时,该代码编辑窗口中显示有用户编写的程序代码;此外,当用户在编写的程序代码之间设定调试断点时,代码编辑窗口中相应的行位置的前面会显示有代表调试断点的标记,如图5所示,对第10行和第11行的程序代码设定调试断点,该调试断点可以显示为“黑圆点”,此外,该调试断点还可以表示为“加粗显示的圆点”、“红色圆点”、三角形、圆形、多边形、箭头等任意形式的图标。以上仅为示例,不作为实施本申请的具体限定。

[0118] 在一些实施方式中,可以通过代码调用的方式引入封装好的代码库,节省代码编辑时间,提高程序代码编辑效率。例如,代码编辑页面包含动作逻辑库和代码存储文件,步骤(101.2.1),可以包括:

[0119] (101.2.1.1) 响应于针对动作逻辑库的触发操作,显示动作逻辑库对应的动作逻辑列表;

[0120] (101.2.1.2) 获取动作逻辑列表内被选中的目标函数,并以目标函数作为程序代码嵌入至代码存储文件内;

[0121] (101.2.1.3) 在代码编辑页面中展示代码存储文件对应的文件信息窗口,文件信息窗口展示实时编辑的程序代码。

[0122] 其中,该动作逻辑库可以包含控制虚拟对象的动作或运动的类或函数,可以理解的是,类或函数属于封装好的动作控制代码,动作逻辑库中包含多个封装好的动作控制代码,可在编辑程序代码时,引入不同的动作控制代码来控制虚拟对象作出不同的动作或运动。

[0123] 其中,该代码存储文件可以是用于存储所编辑的程序代码的文件,其属于目标程序的入口文件,通过在该代码存储文件中编写程序代码,以用于控制虚拟对象的运动。

[0124] 具体的,代码编辑页面中显示有针对虚拟对象封装好的动作逻辑库以及代码存储文件,该动作逻辑库和代码存储文件可以由用户自己添加,还可有代码编辑页面对应的编程环境默认打开。需要说明的是,每个文件都具有对应的一个窗口,代码编辑区可以显示一个或多个窗口,用户在编写程序代码时,可以打开代码存储文件,使得代码编辑区上叠加显示文件信息窗口,用户可以在代码的代码存储文件对应的文件信息窗口内编写程序代码。

在用户编写程序代码的过程中,还可打开封装好虚拟对象的动作控制代码的动作逻辑库,该动作逻辑库包含的函数可通过列表或窗口等形式展示,用户可从动作逻辑列表内选定需要引用的目标函数,此时,并通过调用方式将目标函数作为一部分程序代码添加至代码存储文件内;此后,用户编写的和引入的程序代码都会在代码编辑页面的代码存储文件对应的文件信息窗口内展示。以此,可通过代码引入方式来提高代码编写效率,节省用户编程时间,提高用户的体验。

[0125] 示例性的,结合图5所示,该代码编辑页面的左侧区域包含文件目录区,该文件目录区内显示有“动作逻辑库”和“代码存储文件”,该“动作逻辑库”和“代码存储文件”可以由编程环境被启动时默认打开,还可由用户添加、创建。用户可以点击“代码存储文件”,并在代码边界页面中的代码编辑区编写程序代码,此外,用户还可以打开动作逻辑库,并从动作逻辑库中选定需要引用的目标函数,并通过“from***import***”的函数调用语句在编程时引入类或目标函数,如通过“from动作逻辑库import Robot_Max”语句来引入针对虚拟对象的相关函数或类等,如引入“Robot_Max.action(“****”)”、“Robot_Max.move(“****”,10)”等目标函数或类作为一部分程序代码,以节省用户的编程时间,提高用户的编程效率。

[0126] 在一些实施方式中,可将从动作逻辑库中引入的程序代码与用户自己编辑的程序代码之间进行类别区分,以便对不同类别的程序代码进行管理,为了实现以上,可在不同的文件中保存用户人工编写的程序代码,进而,通过调用方式将用户编写的程序代码引入到代码存储文件中。例如,代码编辑页面中还包括代码附加组件,步骤(101.2.1.3)之前,还可包括:响应于针对代码附加组件的触发操作,创建代码附加文件,并展示代码附加文件对应的附加代码窗口;获取附加代码窗口中输入的附加程序代码;通过代码调用函数将附加程序代码添加至代码存储文件。

[0127] 其中,该代码附加文件用于存储人工编辑的附加程序代码,该附加程序代码不限于包括条件函数代码、非动作控制代码,该附加程序代码还可以是动作逻辑库中不包括的其他动作控制代码。

[0128] 示例性的,参见图5所示,在代码编辑页面的文件目录区中包含“代码附加组件”,用户可点击该“代码附加组件”,以创建“代码附加文件”,当用户打开该代码附加文件后,代码编辑区会展示该代码附加文件对应的附加代码窗口,用户可以在该的附加代码窗口中编写的附加程序代码。进而,客户端可获取附加代码窗口中的附加程序代码,并通过代码调用函数将附加代码窗口中的附加程序代码引入至代码存储文件内,如利用“from***import***”的代码调用函数,在代码存储文件对应的程序代码窗口中编写“from附加代码文件import***”,以便将附加程序代码引入至代码存储文件中。以此,可实现多文件代码编辑,将从动作逻辑库中引入的程序代码与用户自己编辑的程序代码之间进行类别区分,以利于对不同类别的程序代码进行管理。

[0129] 在一些实施方式中,还可支持多人协同开发,具体的,可将多个用户编辑的程序代码合并到代码存储文件内,提高开发效率和质量。例如,在步骤(101.2.1.3)之前,还可包括:获取不同目标对象编辑的目标附加文件;读取每个目标附加文件对应的目标附加代码;通过代码调用函数将目标附加代码合并至代码存储文件内。

[0130] 示例性的,参见图7所示,当其他目标对象编辑好程序代码后,可以目标附加文件的方式进行保存,文件的命名后缀与代码存储文件的命名后缀相同,如保持相同的“.py”后

缀(图中未示出),并将目标附加文件在多个参与协同开发的用户之间进行共享,或者至少存储至同一终端。进而,可在代码编辑页面中打开不同目标对象编辑的目标附加文件,如通过点击“代码附加组件”将不同目标对象的目标附加文件添加至文件目录区中,图7中仅示出一个“目标附加文件”,还可以包括多个;进一步的,读取目标附加文件中的目标附加代码,并通过代码调用函数语句“from目标附加文件import***”,将每个目标附加文件中的目标附加代码嵌入至代码存储文件中,以此类推,实现将多个用户编辑的程序代码合并至代码存储文件内,实现多人协同开发,提高开发效率和质量。

[0131] 通过以上方式,可在显示界面中同时显示角色控制页面和关联的代码编辑页面,并在代码编辑页面中显示用户实时编辑的程序代码以及用户在程序代码之间设定的调试断点,以便用户在后续的代码调试过程中可以实时查看程序代码的运行以及角色控制页面内虚拟对象的控制动态,为用户提供了良好的编程学习途径,提高用户的编程学习体验。

[0132] 102、调试代码编辑页面中的程序代码,并根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动。

[0133] 在本申请实施例中,为了确保编辑的程序代码无误,可对程序代码进行调试,以便对编辑的程序代码进行检验。其中,在调试程序代码时,可根据程序代码在调试时的运行结果来对角色控制页面中虚拟对象的运动进行控制,如基于运行结果控制虚拟对象作出相应的动作,以使用户结合角色控制页面中虚拟对象的动态视图来检验程序代码在调试时的运行情况,提高用户在编程进阶学习或开发过程中的体验。

[0134] 需要说明的是,为了满足用户对程序代码的实时、单独或分段调试,在根据调试需求在程序代码中的特定位置设定调试断点之后,还需要检测程序代码的调试情况,如调试进度、调试信息等,以便基于程序代码的调试情况来生成调试消息。

[0135] 在一些实施方式中,代码编辑页面还包含调试检测组件,用户可以触发该调试检测组件,以使得编程环境的相关器件对程序代码进行调试,并检测调试过程中的调试进度。例如,步骤102中的“调试代码编辑页面中的程序代码”,可以包括:

[0136] (102.a.1) 响应于针对调试检测组件的触发操作,运行代码编辑页面中的程序代码;

[0137] (102.a.2) 检测程序代码在运行时的调试进度。

[0138] 具体的,代码编辑页面中可包含调试检测组件,当用户点击该调试检测组件时,即可开始代码调试流程,在调试开始时,启动一个调试会话,并运行程序代码。其中,在运行程序代码时可包括两步,第一步,对每一句程序代码进行解释,得到每一句程序代码对应的待执行代码,该待执行代码为“0”和“1”形式的计算机可读的语言,即机器码,第二步,依序运行每一句待执行代码,以便基于每一运行结果来对角色控制页面中的虚拟对象进行控制。此外,在程序代码的调试过程中,为了使得用户的能够获悉程序代码的调试情况,可以检测程序代码在运行时的调试进度,以便后续在调试进度生成相关的调试消息,从而基于调试会话的机制将调试消息传输至编程环境中相应的调试控制组件,以请求角色控制页面对应的系统作出响应。

[0139] 示例性的,参见图8所示,代码编辑页面的功能组件栏中包含“调试检测组件”,用户点击该代码编辑组件,以触发针对代码存储文件中的程序代码的开始调试指令,从而,编程环境会响应于该开始调试指令,对程序代码进行调试。具体的,以“Python”编程语言的程

序代码为例,以“VSCode”编辑器作为本示例中的编程环境,在进入调试流程时,“VSCode”编辑器启动一个调试会话(session),并请求“Python”解释器对程序代码进行逐句解释,以获取每一句程序代码对应的待执行程序逻辑,并通过代码解释器逐句运行每一个待执行的程序逻辑,以便基于程序逻辑的运行结果来控制角色控制页面中的虚拟对象的运动;需要说明的是,在以上调试过程中,可以实施检测程序代码的调试进度,以便获悉程序代码的调试情况,如在调试程序代码过程中,可在遇到漏洞时可基于调试进度确定漏洞的位置,或者基于调试进度检测是否达到调试断点,以便基于以上情况,以事件消息的形式生成对应的调试消息,并基于调试会话机制将调试消息传输至“VSCode”编辑器中的调试控制器,以使得调试控制器请求角色控制页面对应的控制系统作出响应,具体响应情况根据实际的调试消息而定。

[0140] 本申请实施例在对程序代码进行调试时,需要根据程序代码的运行结果来实时控制角色控制页面中的虚拟对象进行运动。由于程序代码中设定有调试断点,这使得程序代码的调试过程会出现中断,具体的,在开始调试后,会从第一句程序代码运行至调试断点的前一句程序代码,并在调试断点的前一句程序代码运行完毕时,停止调试。因此,在调试过程的初始阶段,一般根据调试断点之前的程序代码的运行结果来控制虚拟对象进行运动。

[0141] 在一些实施方式中,角色控制页面可以理解为客户端的动态画面,其具有对应的服务器,可由服务器运行相应的程序逻辑,并根据运行结果来控制角色控制页面中的虚拟对象运动。例如,步骤102中的“根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动”,可以包括:

[0142] (102.b.1) 运行调试断点之前的程序代码,并从调试断点之前的程序代码中选取虚拟对象的动作控制代码;

[0143] (102.b.2) 将动作控制代码传输至角色控制页面对应的服务器,以使得服务器在执行动作控制代码后返回对应的动作指令;

[0144] (102.b.3) 基于动作指令,控制角色控制页面中的虚拟对象进行运动。

[0145] 为了便于理解,可将程序代码分为用户编辑的非动作控制代码以及从库中引入的封装好的动作控制代码,动作控制代码的运行结果主要用于控制角色控制页面中的虚拟对象进行运动;而非动作控制代码可包括用户编写的程序代码,其不限用作调试或运行时的判断条件,用于辅助动作控制代码的运行。需要说明的是,针对于控制虚拟对象的动作控制代码,一般通过角色控制页面关联的后台服务器来运行,而对于非动作控制代码,则在终端上运行,如通过在终端的编程环境中的代码解释器来运行。

[0146] 因此,在程序代码的初始调试阶段,首先,针对调试断点之前的程序代码,需要是选择出用于控制虚拟对象的动作控制代码,该选择过程可根据代码调用函数中的参数信息来确定。然后,将动作控制代码发送至角色控制页面关联的服务器,以通过服务器来运行接收到的动作控制代码,并将得到的运行结果返回给终端,此时,该运行结果可以是动作指令,用于对角色控制页面中的虚拟对象进行动作或运动的控制。最后,基于该运行结果控制虚拟对象进行运动,如控制虚拟对象在原地作出各种动作,或者控制虚拟对象行走。至此,在程序代码的初始调试阶段,实现根据调试断点之前的程序代码的运行结果来控制角色控制页面中的虚拟对象的运动,以使用户结合角色控制页面的动态视图来检验调试断点之前的程序代码的调试情况和效果,提高用户在编程学习或程序开发时的体验。

[0147] 在一些实施方式中,程序代码可包含动作代码组合,该动作代码组合由多个动作代码与代码条件函数组成,基于此,需要从动作代码组合中选取当前针对虚拟对象的动作控制代码。具体的,步骤(102.b.1)可以包括:确定调试断点之前的程序代码中针对虚拟对象的动作代码组合,动作代码组合包含代码条件函数和至少两个动作代码;运行代码条件函数,得到函数条件结果;根据函数条件结果,从至少两个动作代码中选取虚拟对象的动作控制代码。

[0148] 需要说明的是,程序代码中可以包括直接确定的动作控制代码以及无法直接确定的动作代码组合,该动作代码组合由动作代码与代码条件函数组成,例如,该动作代码组合中可包含至少两个动作代码和用于选择其中一个动作代码的代码条件函数。为了从动作代码组合中选取当前针对虚拟对象的动作控制代码,需要运行动作代码组合中的代码条件函数,以获取函数条件结果,进而,根据该函数条件结果来作出判定,以选取出针对虚拟对象的动作控制代码。

[0149] 示例性的,假设动作代码组合为“if add(1,2)>2:Robot_Max.move(“Wheel”,10); else:Robot_Max.move(“Wheel”,6)”,则代码条件函数为“if add(1,2)>2”,动作代码分别为“Robot_Max.move(“Wheel”,10)”和“Robot_Max.move(“Wheel”,6)”。为了从动作代码组合中选出针对虚拟对象的动作控制代码,运行代码条件函数为“if add(1,2)>2”,当得到函数条件结果大于2时,选择“Robot_Max.move(“Wheel”,10)”作为动作控制代码。以上仅为示例,不作为实施本申请的具体限定。

[0150] 在一些实施方式中,步骤(102.b.3)可以包括:若识别到动作控制指令为摇晃动作指令,则根据摇晃动作指令,控制角色控制页面中的虚拟对象作出摇晃动作。

[0151] 示例性的,结合图9和图10所示,角色控制页面中的虚拟对象为机器狗,当服务器在接收到动作控制代码后,执行该动作控制代码,以得运行结果,即动作指令,将动作指令返回至终端,以对角色控制页面中机器狗作出动作控制。例如,当前的动作指令为表示摇晃动作的指令,则根据该指令控制机器狗作出左右摇晃的动作,如图9所示,表示机器狗作出向左摇晃时的动作,以及如图10所示,表示机器狗作出向右摇晃时的动作。以上仅为示例,不作为实施本申请的具体限定。

[0152] 在一些实施方式中,步骤(102.b.3)可以包括:若识别到动作控制指令为以目标速度的行走指令,则根据目标速度控制角色控制页面中的虚拟对象向前行走;在虚拟对象的后方区域显示行走轨迹。

[0153] 示例性的,参见图11所示,角色控制页面中的虚拟对象为机器狗,角色控制页面关联的服务器返回至终端的动作指令为以每秒10米的速度向前行走,则控制机器狗以10米/秒的速度前行,具体动态视图表示机器狗的向前行走的动作,同时,在机器狗的后方显示“脚印”图标,以表示和机器狗的行走轨迹。以上仅为示例,不作为实施本申请的具体限定。

[0154] 在一些实施方式中,终端上可同时运行编程环境和角色控制页面的控制程序,通过控制程序来执行针对虚拟对象的动作的程序代码,以控制角色控制页面中的虚拟对象进行运动。例如,步骤102中的“根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动”,可以包括:运行调试断点之前的程序代码,并从调试断点之前的程序代码中选取虚拟对象的动作控制代码;将动作控制代码传输至角色控制页面关联的控制程序;通过角色控制页面关联的控制程序运行动作控制代码,并根据动作控制代码的运行结

果控制角色控制页面中的虚拟对象进行运动。

[0155] 其中,在将动作控制代码传输至角色控制页面关联的控制程序时,首先,针对动作控制代码,生成请求协议消息,该控制请求协议中携带有动作逻辑接口和动作控制参数,进而,将请求协议消息发送至角色控制页面关联的控制程序。基于此,控制程序可解析请求协议消息,得到动作逻辑接口和动作控制参数,并从动作逻辑库中调用该动作逻辑接口对应的角色控制脚本,并根据角色控制脚本和动作控制参数,生成控制逻辑,以执行该控制逻辑,控制虚拟对象进行运动。

[0156] 示例性的,以Unity游戏的游戏画面作为角色控制页面,以Vscode编辑器作为编程环境,编程环境与Unity游戏之间可通过Websocket通信连接。在调试程序代码的过程中,Vscode编辑器的代码执行模块针对调试断点之前的程序代码,可将每一条针对虚拟对象的动作控制代码生成请求协议消息,并发送给Unity游戏程序;其中,请求协议消息可以为Json格式的消息,具体可以是字段形式。

[0157] 进一步的,以控制虚拟对象向前运动的代码为例,该请求协议消息可以为{"id":3,"type":"request","command":move,"args":[wheel,10]},其中,"id"为编程环境中代码执行时与Unity游戏运行时之间所传输协议消息的顺序标识;"type"表示消息种类;"command"表示控制逻辑命令(如"move",移动),command字段的值决定并执行业务逻辑(即游戏逻辑),可用于控制动画系统和物理系统,以实现对虚拟对象进行控制;"args"表示控制参数(如"wheel,10")。进一步的,在游戏引擎(Unity)侧,游戏程序解析接收到的协议消息,按照"move"的接口去封装好的动作逻辑库中调用"move"的角色控制脚本,并基于"args"指示的"wheel,10"的控制参数,将虚拟对象的运动方式设置为"轮式",固定移动速度为10米/秒,进而,调用虚拟对象的接口(如动画系统、物理系统的接口),播放轮式前进的动画,使得游戏画面实时显示虚拟对象按照10米/秒向前移动的动画视图。以此,实现在终端侧控制角色控制页面中虚拟对象的运动。

[0158] 通过以上方式,可对程序代码进行调试,并根据程序代码在调试时的运行结果来控制角色控制页面中的虚拟对象进行运动,以使用户结合角色控制页面中虚拟对象的动态视图来检验程序代码在调试时的运行情况,提高用户在编程进阶学习或开发过程中的体验。

[0159] 103、当检测到程序代码的调试进度到达调试断点时,生成停止调试消息,并根据停止调试消息控制虚拟对象停止运动。

[0160] 在本申请实施例中,程序代码中设置的调试断点可用于触发中断代码调试过程,为了在调试断点处对程序代码的调试过程进行中断,需要实时检测程序代码的调试进度,并在检测到程序代码的调试进度达到调试断点时,生成停止调试消息,以便提示用户程序代码的调试情况,同时,控制角色控制页面中的虚拟对象处于停止运动状态,使得角色控制页面的动画运作与程序代码的运行状态同步,有利于用户检验调试断点之前的程序代码的调试情况,提高用户的编程学习的体验。

[0161] 需要说明的是,在调试进度达到调试断点之前,若调试过程中检测出调试断点之前的程序代码存在错误,则确定存在错误的目标代码,并暂停运行目标代码之后的程序代码,进而,生成停止调试消息,并执行与调试进度达到调试断点的相同处理方式。

[0162] 在一些实施方式中,代码编辑页面还包括用于切换调试状态的调试控制组件,该

调试控制组件用于在代码调试过程中切换调试状态,如暂停调试或继续调试。具体的,步骤103中的“生成停止调试消息”,可以包括:

[0163] (103.1) 暂停运行调试断点之后的程序代码,并将调试状态切换为暂停调试状态;

[0164] (103.2) 基于暂停调试状态,将调试控制组件的状态图标显示为继续调试状态图标,并生成暂停调试状态对应的停止调试消息。

[0165] 具体的,为了满足用户针对程序代码的实时或分段调试需求,在检测到代码调试进度达到程序代码中设定的调试断点时,暂停运行位于调试断点之后的程序代码,并将当前的代码调试状态切换至暂停调试状态。进而,为了使得用户后续能够触发继续调试指令,将代码编辑页面中的调试控制组件的状态图标进行更换显示,具体更换显示为继续调试状态图标,表示后续可点击该继续调试状态图标对应的调试控制组件,以触发继续调试指令;同时,在该暂停调试状态下,生成停止调试消息,以便后续基于停止调试消息来请求服务器暂停对角色控制页面中的虚拟对象进行控制。

[0166] 示例性的,参见图12所示,代码编辑页面的右上方包含一个控件区,该控件区中包含有“调试控制组件”,通过主动触发该“调试控制组件”,可变更代码的调试状态,如将“暂停调试状态”切换为“继续调试状态”,或者将“继续调试状态”切换为“暂停调试状态”。在本实施方式中,当针对程序代码的调试进度达到调试断点时,暂停运行调试断点之后的程序代码,并将调试状态切换为暂停调试状态,在该暂停调试状态下,可将“调试控制组件”的状态图标显示为“三角形”形状图像,以表示后续可点击该状态图标,以将当前的暂停调试状态切换至继续调试状态。

[0167] 在一些实施方式中,当调试进度到达调试断点时,为了使得用户获悉对调试断点之前的程序代码的调试情况,需要将调试断点之前的调试信息进行展示。具体的,在步骤103中的“生成停止调试消息”之后,还可以包括:将代码编辑页面中调试断点对应的程序代码的行区域进行彩色标记;获取位于调试断点之前的程序代码的运算结果和运行堆栈信息;在代码编辑界面中生成调试信息窗口,调试信息窗口用于显示运算结果和运行堆栈信息。

[0168] 其中,该行区域可以是调试断点在程序代码中的行位置区域,可以理解的是,调试断点所在的行位置区域还可包括至少一句程序代码。

[0169] 其中,该运算结果可以是位于调试断点之前的程序代码在调试过程中的运算值,具体根据实际情况而定。

[0170] 其中,该运行堆栈信息可以是多个程序代码之间在调试过程中的运算顺序堆栈,用于表示调试断点之前的程序代码之间的运行顺序、运行关系等信息。

[0171] 示例性的,当调试进度到达调试断点时,会暂停对调试断点之后的程序代码的调试,当前的调试状态处于暂停调试状态,此时,为了便于用户能够从代码编辑页面侧获悉代码调试情况,可加强对代码编辑页面的信息提示。例如,结合图12所示,在暂停调试状态下,在代码编辑页面中,将当前到大的调试断点所在行区域进行色彩显示(图中未示出具体颜色),如以绿色、蓝色、红色等颜色进行显示,以便于用户快速查看程序代码当前的调试进度;此外,获取调试断点之前的程序代码的调试信息,该调试信息不限于包括运算结果和运行堆栈信息,并在代码编辑页面的左侧(原文件目录区)生成一个调试信息窗口,以通过该调试信息窗口来展示调试断点之前的程序代码的运算结果和运行堆栈信息。以此,以利于

用户快速了解调试断点之前的程序代码的调试情况,且满足用户对程序代码的实时调试需求,无需人工逐句检查程序代码,提高了编程进阶学习或开发的效率和体验。

[0172] 在一些实施方式中,当程序代码的暂停/停止调试时,需要通知角色控制页面对应的服务器暂停对相关系统的运作,以避免由于系统混乱而导致角色控制页面显示有误,从而影响了用户体验。具体的,步骤103中的“根据停止调试消息控制虚拟对象停止运动”。可以包括:将停止调试消息发送至角色控制页面对应的服务器,使得服务器在解析停止调试消息后返回停止运动指令;基于停止运动指令,控制角色控制页面中的虚拟对象停止运动。

[0173] 需要说明的是,在对角色控制页面中的虚拟对象进行控制时,其对应的后台系统涉及多个维度的控制因素,如包含物理系统、动画系统和运行逻辑等控制因素。因此,在调试过程中,当暂停调试该调试断点之后的程序代码时,需要将生成的停止调试消息发送至角色控制页面关联的服务器,以使得服务器在基于停止调试消息作出响应,具体可以是暂停运行物理系统、动画系统和运行逻辑等,并生成停止运动指令返回至终端,以控制角色控制页面中的虚拟对象停止运动。

[0174] 示例性的,以游戏画面作为角色控制页面为例,假设虚拟对象为机器狗,当暂停运行调试断点之后的程序代码时,将生成的停止调试消息发送至游戏服务器,以使得游戏服务器基于接收到的停止调试消息作出响应,具体为暂停运行用于控制机器狗的动作逻辑、物理系统和动画系统,并返回针对游戏画面中机器狗的停止运动指令;基于停止运动指令控制游戏画面中的机器狗停止前行,如图13所示,机器狗在“斑马线”前停止前行,并保持站立的动作状态。

[0175] 通过以上方式,可在检测到程序代码的调试进度达到调试断点时,中断对程序代码的调试,并生成停止调试消息,以控制角色控制页面中的虚拟对象停止运动,使得角色控制页面的动画运作与程序代码的运行状态同步,有利于用户准确地检验调试断点之前的程序代码的调试情况,提高用户的编程学习的体验。

[0176] 104、响应于接收到的继续调试指令,运行调试断点之后的程序代码,并根据调试断点之后的程序代码的运行结果控制虚拟对象继续运动。

[0177] 在本申请实施例中,用户在查看调试断点之前的程序代码的调试情况后,可选择对调试断点之后的程序代码进行继续调试。当用户确定继续对调试断点之后的程序代码进行调试时,可触发代码响应的组件来生成继续调试指令,以使得编程环境对继续调试指令进行响应,从而运行调试断点之后的程序代码,从而,根据运行结果来控制角色控制页面中的虚拟对象继续运动。以此,避免全篇幅调试代码,可实现对编辑的程序代码进行实时、单独或分段调试,有利于用户的快速获悉各句程序代码的调试情况,提高用户在编程进阶学习或开发时的效率和体验。

[0178] 在一些实施方式中,当继续运行调试断点之后的程序代码时,需要通知角色控制页面关联的服务器中的相关控制系统,以使得服务器能继续基于程序代码调试时的运行结果来控制角色控制页面中的虚拟对象运动。具体的,步骤104中的“响应于接收到的继续调试指令,运行调试断点之后的程序代码”,可以包括:响应于接收到的继续运行指令,运行调试断点之后的程序代码,并生成继续执行消息;将继续执行消息发送至角色控制页面对应的服务器,使得服务器恢复角色控制页面对应的运行逻辑、物理系统和动画系统。

[0179] 示例性的,如图12所示,代码编辑页面的右上方区域中包含“继续调试状态图标”

的调试控制组件,用户可点击该状态图标下的调试控制组件,以生成继续调试指令,从而触发针对调试断点之后的程序代码的继续调试流程,继续运行调试断点之后的程序代码,同时,为了通知角色控制页面关联的服务器恢复对虚拟对象的控制,需要生成继续调试消息,并将该继续调试消息发送至服务器,以使得服务器恢复之前暂停运行的运行逻辑、物理系统和动画系统。此后,恢复后服务器可基于基于程序代码的运行结果来生成动作指令,以用于控制角色控制页面中的虚拟对象继续运动。

[0180] 由上可知,本申请实施例可以显示角色控制页面以及角色控制页面关联的代码编辑页面,代码编辑页面显示实时编辑的程序代码以及位于程序代码中的调试断点;调试代码编辑页面中的程序代码,并根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动;当检测到程序代码的调试进度到达调试断点时,生成停止调试消息,并根据停止调试消息控制虚拟对象停止运动;响应于接收到的继续调试指令,运行调试断点之后的程序代码,并根据调试断点之后的程序代码的运行结果控制虚拟对象继续运动。由此可得,本方案可先显示角色控制页面和关联的代码编辑页面,并通过代码编辑页面显示用户实时编辑的程序代码和设定在程序代码之间的调试断点,然后,对程序代码进行调试,以及根据调试过程的运行结果控制角色控制页面中的虚拟对象进行运动,同时,检测针对程序代码的调试进度,最后,在调试进度达到调试断点时生成停止调试消息,以根据停止调试消息控制角色控制页面中的虚拟对象停止运动,并在检测到针对调试断点之后的程序代码的继续调试指令时,控制虚拟对象继续运动;以此,可根据设定的调试断点来实时调试程序代码,无需逐句检查程序代码,缩短了编程学习周期,提高用户的编程进阶学习的效率,此外,将用户所编辑的程序代码的控制逻辑展示角色控制页面中,提供良好的编程学习方式,提高用户的体验。

[0181] 根据上面实施例所描述的方法,以下将举例作进一步详细说明。

[0182] 本申请实施例以程序调试为例,对本申请实施例提供的程序调试方法作进一步叙述。

[0183] 图14是本申请实施例提供的程序调试方法的另一步骤流程示意图,图15是本申请实施例提供的虚拟对象继续运动的页面示意图,图16是本申请实施例提供的程序调试方法的模块框架示意图,图17是本申请实施例提供的调试检测组件的程序的安装场景示意图,图18是本申请实施例提供的调试检测组件的程序的安装界面示意图,图19是本申请实施例提供的程序代码的运行流程示意图,图20是本申请实施例提供的调试消息的传输架构示意图,图21是本申请实施例提供的程序调试方法的时序流程图。为了便于理解,本申请实施例结合图4-21进行描述。

[0184] 在本申请实施例中,将从程序调试装置的维度进行描述,该程序调试装置具体可以集成在计算机设备如终端和/或服务器中。例如,该计算机设备上的处理器执行程序调试方法对应的程序时,该程序调试方法的具体流程如下:

[0185] 201、显示角色控制页面。

[0186] 在本申请实施例中,为了提供一种良好的编程进阶学习或开发的途径,可将编程环境与具有产品表现的动态视图相结合,以便通过动态视图来展示用户编辑的程序代码在调试过程中的运行效果,以此,后续可根据动态视图的表现情况来检验编辑的程序代码,满足用户的学习周期,使得用户基于良好的编程进阶学习途径来提高编程学习效率。

[0187] 其中,该角色控制页面可以理解为可控制的动态视图,其作为辅助用户编程学习或开发的专门用途,该角色控制页面中至少包括受控制的虚拟对象以及代码编辑组件,该代码编辑组件用于在角色控制页面上启动编程环境,如代码编辑器。为了方便理解角色控制页面,举一个示例,以游戏画面作为角色控制页面为例,该虚拟对象为游戏画面中的游戏主角,用户可在编辑好程序代码后,可调试所编辑的程序代码,以根据调试时的代码运行结果来实时控制游戏画面中的游戏主角进行运动,该虚拟对象可以是虚拟武侠角色、虚拟机器狗、虚拟机器人等;以上仅为示例,不作为本申请实施场景的具体限定。

[0188] 示例性的,以编程进阶学习为例,用户在终端上打开一个编程学习的客户端后,终端的显示界面会展示出一个角色控制页面。结合图4所示,该角色控制可以视为一个游戏画面,该角色控制页面中包含有虚拟对象,该虚拟对象作为游戏主角,如以机器狗作为游戏主角,该机器狗可作为用户编程学习过程中的控制对象;此外,角色控制页面的右下方叠加显示有一个任务视图窗口,该任务视图窗口中可展示有任务信息,用户后续在编程学习时,可以根据任务视图窗口中的任务指示信息来编写程序代码,以便后续在对程序代码进行调试时,可控制机器狗作出相应的运动,直至完成任务指示信息的要求,如此,可结合娱乐方式来提高编程学习或开发的趣味性,为用户提供一个良好学习或开发途径,提高用户的体验。

[0189] 此外,如图4所示,角色控制页面的左侧还包括一系列的控件,如“返回”、“虚拟场景”、“代码编辑”、“运行”等组件,其中,“虚拟场景”组件不限于切换角色控制页面的显示背景、对比度等显示属性信息,用户可以根据个人需求自定义设定对应的显示属性信息。其中,该“运行”组件用于触发直接运行用户编辑的程序代码,以实现相应的程序代码的逻辑,其与代码调试存在区别。其中,该“代码编辑”组件被触发时,会拉起一个编程环境的页面,以供用户编辑程序代码,因此,为了实现以上编程进阶学习或开发途径,用户可以触发“代码编辑”组件,以拉起编程环境对应的页面,以便在页面中编写程序代码。

[0190] 202、响应于针对代码编辑组件的触发操作,显示代码编辑组件对应的代码编辑页面。

[0191] 其中,该代码编辑页面可以是编程环境对应的编辑器的显示页面,该页面中至少包括代码编辑区,用户可在该代码编辑页面中的代码编辑区内编写的程序代码。

[0192] 具体的,当用户点击该代码编辑组件后,会响应于针对该代码编辑组件的触发操作,通过运行线程拉起一个代码编辑环境,并通过进程来运行该代码编辑环境,同时,在显示界面中显示出对应的代码编辑页面。

[0193] 需要说明的是,该代码编辑页面与角色控制页面之间是存在关联关系的,具体的,代码编辑页面用于编辑和运行程序代码,以控制角色控制页面中的虚拟对象运动,而角色控制页面则基于虚拟对象的动态视图来反映/表示代码编辑页面中的程序代码的控制逻辑。但是,代码编辑页面和角色控制页面分别对应不同的程序,即通过不同的进程来分别运行代码编辑页面和角色控制页面。

[0194] 在本申请实施例中,在显示代码编辑页面时,可以将代码编辑页面与角色控制页面相互并排分布,使得代码编辑页面和角色控制页面同时显示在同一显示界面中,以利于用户同时观察代码编辑页面中的程序代码的相关情况和角色控制页面中虚拟对象的动态效果,提高编程学习或使用效率,提高用户在编程学习或开发时的便利性。示例性的,结合图6所示,代码编辑页面和角色控制页面并排显示。

[0195] 可以理解的是,代码编辑页面用于展示用户实时编辑情况。

[0196] 203、响应于在代码编辑页面中的输入操作,在代码编辑页面中展示实时编辑的程序代码,并获取代码编辑页面内被选中的程序代码,并根据调试断点标记被选中的程序代码。

[0197] 在本申请实施例中,当打开角色控制页面以及关联的代码编辑页面之后,用户可在代码编辑页面中编写程序代码。具体的,可响应于用户在代码编辑页面中的编写、调用、引用程序代码等输入操作,确定用户输入的代码内容,并在代码编辑页面中实时显示用户编写的程序代码。此外,当用户在程序代码中设定调试断点时,还会在用户的设定位置显示该调试断点,以表示程序代码在后续调试时会运行至该调试断点处停止调试,以便于用户对相关部分的程序代码进行实时或单独调试。

[0198] 其中,该调试断点可以理解为代码调试过程中断位置点,用于阻断程序代码的调试过程,以实现调试断点前后的两部分程序代码进行分别调试,以使得用户能够实时审视每部分代码在调试时的运行情况。然而,用户可根据对程序代码的实际调试需求,在任意两句程序代码之间设定调试断点,具体示例场景如下:

[0199] 示例1,用户在编辑完所有的程序代码后,可根据实际调试需求,在任意两句程序代码之间设定调试断点,以使得后续在对程序代码调试时,使得调试进度达到调试断点时,都会暂停调试当前的调试断点之后的程序代码。需要说明的是,调试断点的数量不受限制,可针对每一句程序代码都设定一个调试断点,具体根据实际需求设定。

[0200] 示例2,用户在编辑完一句目标程序代码后,可下一句程序代码之前的位置设定一个调试断点,以此,用户无需编辑完所有的程序代码即可开始对目标程序代码进行调试,并由于调试断点的中断调试作用,在调试完该目标程序代码之后即可停止调试,如此,满足用户对程序代码的实时调试需求。

[0201] 为了便于理解用户的代码编辑过程,可结合图5所示,代码编辑页面中包含代码编辑区,用户可以点击代码存储文件,代码编辑区会展示该代码存储文件对应的代码编辑窗口,用户可选定代码编辑窗口中的任意行作为代码的输入位置,并在选定的行位置出进行代码输入,此时,当前的代码编辑窗口会显示用户实时编辑的程序代码;此外,用户还可根据实际调试需求,在相应的行位置前面设定调试断点,如,在第10行的程序代码之前设定调试断点,该调试断点可以显示为“黑圆点”。以上仅为示例,不作为实施本申请实施例的具体限制。

[0202] 在本申请实施例中,用户在编辑程序代码时,可以通过代码调用的方式来引入代码,以作为用户编辑的部分程序代码。具体示例如下:

[0203] 示例a,动作逻辑库包含已经封装好的各种样式动作的动作控制代码,每个动作控制代码可以对应一个函数或类。其中,可通过动作逻辑库中引入相应的动作控制代码,以作为针对角色控制页面中的虚拟对象的动作的程序代码。例如,结合图5所示,代码编辑页面的左侧区域包含文件目录区,该文件目录区内默认显示有“动作逻辑库”和“代码存储文件”,通过“from动作逻辑库import Robot_Max”语句来引入针对虚拟对象的相关函数或类等对应的动作控制代码,如引入“Robot_Max.action(“****”)”、“Robot_Max.move(“****”,10)”等作为一部分程序代码,以节省用户的编程时间,提高用户的编程效率。

[0204] 示例b,结合图5所示,基于多文件编辑的方式,用户可在文件目录区中创建代码附

加文件,并在代码附加文件中编写个人需求的部分附加程序代码;进而,在代码存储文件的代码编辑窗口中的第2行编写“form目标附加文件import***”的代码调用函数语句,以通过“from附加代码文件import***”的代码调用函数语句来将附加程序代码引入至代码存储文件中。以此,可将从动作逻辑库中引入的程序代码与用户自己编辑的程序代码之间进行类别区分,以利于对不同类别的程序代码进行管理。

[0205] 示例c,结合图5所示,通过多用户协同编辑开发的方式,将每个用户编写的目标附加文件导入代码编辑页面中的文件目录区内;进而,在代码存储文件的代码编辑窗口中的第3行编写“form目标附加文件import***”的代码调用函数语句,以将每个目标附加文件中的目标附加代码嵌入至代码存储文件中。以此,实现将多个用户编辑的程序代码合并至代码存储文件内,实现多人协同开发,提高开发效率和质量。

[0206] 204、响应于接收到的开始调试指令,调试代码编辑页面中的程序代码,并检测程序代码在运行时的调试进度。

[0207] 在本申请实施例中,为了确保编辑的程序代码无误,可对程序代码进行调试,以便对编辑的程序代码进行检验。具体的,代码编辑页面中可包含调试检测组件,当用户点击该调试检测组件时,会生成一个开始调试指令,而编程环境基于该开始调试指令,触发代码调试流程,具体如下:

[0208] 在调试开始时,启动一个调试会话,并运行程序代码。其中,在运行程序代码时可包括两步,具体如下:

[0209] 第一步,对每一句程序代码进行解释,得到每一句程序代码对应的待执行代码,该待执行代码为“0”和“1”形式的计算机可读的语言,即机器码。

[0210] 第二步,依序运行每一句待执行代码,以便后续基于每一运行结果来对角色控制页面中的虚拟对象进行控制。

[0211] 此外,在程序代码的调试过程中,为了满足对程序代码的实时、单独或分段调试,可在调试过程中实时检测程序代码在运行时的调试进度,以便后续在调试进度达到调试断点时生成停止调试消息,并基于调试会话的机制反馈调试情况,如此,可使得用户在实时、单独或分段调试中能够分别获取每阶段的程序代码的调试情况,避免用户人工逐句检查程序代码的现象,具有可靠性。

[0212] 205、根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动。

[0213] 本申请实施例在调试程序代码时,可根据程序代码在调试时的运行结果来对角色控制页面中虚拟对象的运动进行控制,如基于运行结果控制虚拟对象作出相应的动作,以便用户结合角色控制页面中虚拟对象的动态视图来检验程序代码在调试时的运行情况,提高用户在编程进阶学习或开发过程中的体验。

[0214] 在本申请实施例中,角色控制页面可以理解为客户端的动态画面,其具有对应的服务器,首先,可基于TCP的全双工通信协议(WebSocket)来建立客户端与服务器之间的长连接。然后,在程序代码的初始调试阶段,针对调试断点之前的程序代码,需要是选择出用于控制虚拟对象的动作控制代码,并以Json格式生成动作控制代码对应的Json格式消息。进而,以Json格式消息方式向服务器发送程序代码逻辑,并由服务器运行该程序代码逻辑,以得到运行结果(即动作指令),并将运行结果返回至客户端。最后,基于服务器返回的运行

结果,控制角色控制页面中的虚拟对象运动。

[0215] 示例性的,参见图9和10所示,角色控制页面中的虚拟对象为机器狗,当服务器返回的运动结果为摇晃动作的指令,则根据该指令控制机器狗作出左右摇晃的动作,如图9所示,表示机器狗作出向左摇晃时的动作,以及如图10所示,表示机器狗作出向右摇晃时的动作。又如,参见图11所示,当服务器返回的运动结果为以每秒10米的速度向前行走的动作指令,则控制机器狗以10米/秒的速度前行,具体动态视图表示机器狗的向前行走的动作,同时,在机器狗的后方显示“脚印”图标,以表示和机器狗的行走轨迹。

[0216] 206、当检测到程序代码的调试进度到达调试断点时,暂停运行调试断点之后的程序代码,并生成停止调试消息。

[0217] 在本申请实施例中,在检测到程序代码的调试进度达到调试断点时,生成停止调试消息,以便提示用户程序代码的调试情况,使得用户能够检验调试断点之前的程序代码的调试情况,提高用户的编程学习的体验。

[0218] 具体的,为了满足用户针对程序代码的实时或分段调试需求,在检测到代码调试进度达到程序代码中设定的调试断点时,暂停运行位于调试断点之后的程序代码,并将当前的代码调试状态切换至暂停调试状态。进而,将代码编辑页面中的调试控制组件的状态图标进行更换显示为继续调试状态图标,表示后续可点击该继续调试状态图标来触发继续调试指令;同时,在该暂停调试状态下,生成停止调试消息,以便后续基于停止调试消息来请求服务器暂停对角色控制页面中的虚拟对象进行控制。

[0219] 需要说明是,当调试进度到达调试断点时,为了使得用户获悉对调试断点之前的程序代码的调试情况,需要将调试断点之前的调试信息进行展示。示例性的,结合图12所示,在暂停调试状态下,在代码编辑页面中,将当前到大的调试断点所在行区域进行色彩显示(图中未示出具体颜色),如以绿色、蓝色、红色等颜色进行显示,以便于用户快速查看程序代码当前的调试进度;此外,获取调试断点之前的程序代码的调试信息,该调试信息不限于包括运算结果和运行堆栈信息,并在代码编辑页面的左侧(原文件目录区)生成一个调试信息窗口,以通过该调试信息窗口来展示调试断点之前的程序代码的运算结果和运行堆栈信息。

[0220] 207、根据停止调试消息控制虚拟对象停止运动。

[0221] 在本申请实施例中,在检测到程序代码的调试进度达到调试断点时,可将生成的停止调试消息发送至角色控制页面对应的服务器,以使得服务器在基于停止调试消息作出响应,具体可以是暂停运行物理系统、动画系统和运行逻辑等,并生成停止运动指令。进而,将停止运动指令返回至终端,以控制角色控制页面中的虚拟对象停止运动,如此,使得角色控制页面的动画运作与程序代码的运行状态同步,有利于用户准确地检验调试断点之前的程序代码的调试情况,提高用户的体验。

[0222] 示例性的,结合图13所示,以游戏画面作为角色控制页面,则虚拟对象为机器狗,将生成的停止调试消息发送至游戏服务器,以使得游戏服务器基于接收到的停止调试消息,暂停运行用于控制机器狗的动作逻辑、物理系统和动画系统,并返回针对游戏画面中机器狗的停止运动指令;基于停止运动指令控制游戏画面中的机器狗停止前行,例如,机器狗在“斑马线”前停止前行,并保持站立的动作状态。

[0223] 208、响应于接收到的继续调试指令,运行调试断点之后的程序代码。

[0224] 具体的,当用户确定继续对调试断点之后的程序代码进行调试时,可触发代码响应的组件来生成继续调试指令,使得编程环境对应的客户端响应于该继续调试指令,生成继续调试消息,并将该继续调试消息发送至服务器,以使得服务器恢复之前暂停运行的运行逻辑、物理系统和动画系统。同时,继续运行调试断点之后的程序代码,以便后续基于调试断点之后的程序代码的运行结果控制虚拟对象继续运动。

[0225] 209、根据调试断点之后的程序代码的运行结果控制虚拟对象继续运动。

[0226] 在本申请实施例中,在恢复对调试断点之后的程序代码进行调试时,可根据一句或多句程序代码的运行结果来控制虚拟对象继续运动,直至全篇幅的程序代码调试完毕时停止调试。

[0227] 示例性的,参见图15所示,以游戏画面作为角色控制页面,虚拟对象为机器狗,在游戏服务器恢复运行逻辑、物理系统和动画系统后,可将调试断点之后的动作控制代码传输至游戏服务器,以通过游戏服务器来执行调试断点之后的动作控制代码,并将执行得到的运行结果返回给客户端,以便控制游戏画面中的机器狗继续运动,如控制机器狗继续强行,结合图13和14所示,机器狗已经通过了“斑马线”,继续前行。

[0228] 需要说明的是,本申请实施例在调试进度到达每一个调试断点处时都会停止调试,并生成停止调试消息,以控制角色控制页面关联的进程暂停运行逻辑、物理系统和动画系统;并基于继续执行指令,恢复暂停的运行逻辑、物理系统和动画系统,以按照当前调试断点之后的程序代码的运行结果控制虚拟对象继续运动。以此类推,直至全篇幅的程序代码调试完毕,则停止调试流程。

[0229] 通过执行以上步骤201-209,可避免全篇幅地调试代码,可实现对编辑的程序代码进行实时、单独或分段调试,有利于用户的快速获悉各句程序代码的调试情况,提高用户在编程进阶学习或开发时的效率和体验。

[0230] 为了便于对本申请实施例的理解,将以具体的应用场景实例对本申请实施例进行描述。具体的,通过执行以上步骤201-209,以及结合图4-图21,对该应用场景实例进行描述。

[0231] 具体的,该程序调试方法的场景实例具体描述如下:

[0232] 一、为了更好理解该程序调试方法的场景实例,先对本场景示例所涉及的术语进行解释,具体分别如下:

[0233] VSCode:本场景实例主要是基于VSCode的编程环境来实现的,该VSCode为轻量级集成开发环境(IDE),可在Windows、macOS和Linux等操作环境上运行。可用于各种编程语言和框架的开发,如JavaScript、Python、Java、C++、TypeScript、React、Angular等语言。

[0234] VSCode扩展:VSCode编程环境(编辑器)支持开发扩展,以为VSCode编辑器添加其他功能,例如,扩展其他编程语言、主体、代码格式化等。

[0235] Unity:可跨平台的游戏引擎,用于开发各种类型的游戏,例如,不限用于开发电脑(PC)端、移动终端、主机游戏机、虚拟现实和增强现实等类型的游戏应用程序。为了便于描述,将基于Unity开发的游戏简述为“Unity游戏”

[0236] Python解释器:属于可用于执行Python代码的程序,Python解释器可以将Python代码翻译成计算机可读的语言,并执行该代码以产生运行结果。此外,该解释器也可以支持调试功能,可以让程序停止在某个指定位置,以便用户可以逐步执行代码,并查看变量的值

和程序执行流程,以获悉程序代码的调试情况。

[0237] 基于以上术语,对该程序调试方法的场景的方案流程进行介绍,该方案具体为:以Python作为编程语言为例,首先,用户在VSCode编辑器中编写Python代码(程序代码),并添加调试断点;然后,在调试程序代码时,通过VSCode编辑器中的Python调试器来启动Python解释器,从而,通过Python解释器来运行、调试Python代码;最后,基于Python代码的运行结果来控制Unity游戏的游戏逻辑(即前述的“运行逻辑”),以执行针对游戏主角动作的Python代码的逻辑,从而控制游戏画面中游戏主角的动作。

[0238] 此外,在Python代码的调试过程中,通过开发的断点调试功能,检测Python解释器发送给Python调试器的停止调试消息,并在检测到停止调试消息时,将调试消息通过WebSocket的通信方式转发给Unity游戏,以请求Unity游戏暂停运行游戏逻辑、物理系统和动画系统等,以此,实现了Python调试器与Unity游戏之间的交互。

[0239] 二、本场景实例在框架架构上可包括:(1)编程模块、(2)代码管理模块、(3)游戏控制模块。结合图16,关于这三个模块的介绍如下:

[0240] (1)编程环境模块包括:代码编辑器,用于支持用户编写程序代码。Python解释器,用于解释、执行用户编辑的程序代码,该Python解释器中可包含用于支持调试Python代码功能的Python调试器。VSCode内置的Python扩展,用于支持Python代码提示、运行、调试、环境配置。Coding扩展,属于新开发的VSCode拓展,用于检测在调试Python代码时的调试消息,并转发检测到的调试信息至Unity游戏;需要说明的是,该Coding扩展为本申请实施例中与“调试检测组件”关联的程序。

[0241] (2)代码管理模块,用于管理用户编辑的Python代码,该Python代码包括引用的codingtge编程库和用户编写的代码。其中,该codingtge编程库属于已经封装开发好的Python编程库,用于控制Unity游戏角色的动作。其中,用户自己在编写的代码时,可以引入不同的编程库,以获取所需的程序逻辑。

[0242] (3)游戏控制模块,用于控制Unity游戏。

[0243] 以上三个模块之间存在交互,结合图16和图21进行描述,交互流程为:

[0244] 首先,用户通过代码编辑器来编写Python编程语言的程序代码(Python代码),并在编辑Python代码时,可以引入已将封装好codingtge编程库,以实现在编辑的程序代码中添加控制游戏角色的动作控制代码。

[0245] 然后,用户通过Coding扩展向Python扩展请求启动调试,使得Python扩展启动Python解释器,以通过Python解释器运行编辑的Python代码。一方面,Python扩展接收Python解释器发送的停止调试消息,以实现调试过程的停止控制,此时,Coding扩展会检测到Python扩展接收到的停止调试消息;另一方面,用户可通过Coding扩展请求Python扩展启动继续调试流程,Python扩展在Python扩展请求Python解释器继续调试后,接收Python解释器反馈的继续调试消息,此时,Coding扩展可检测到Python扩展从Python解释器处接收到的继续调试消息。需要说明的是,Coding扩展还建立与Unity游戏之间的websocket通信连接,以使得与游戏控制模块中的调试控制模块之间能够同步停止调试消息或继续调试消息。

[0246] 最后,游戏控制模块中调试控制模块会基于停止调试消息或继续调试消息,对程序控制模块进行暂停或恢复的控制,而程序控制模块则基于暂停状态控制游戏的物理系

统、动画系统、角色控制(即游戏逻辑)暂停运行,或者基于恢复状态控制物理系统、动画系统、角色控制(即游戏逻辑)恢复运行。以此,实现对Unity游戏中的游戏主角的动作控制。

[0247] 三、关于程序调试的场景示例的具体过程,包括:Coding扩展的安装阶段和使用阶段。具体的,对着两个阶段进行描述,具体如下:

[0248] (A) Coding扩展的安装阶段:结合图17所示,开发者可在VSCode编辑器编程关于Coding扩展的程序代码,并打包封装为VSIX文件;进一步的,将VSIX文件发布至软件平台审核。

[0249] 用户可以基于终端上的VSCode编辑器提供的扩展接口,下载安装Coding扩展。具体的,结合图18所示,用户可以点击VSCode编辑器中左侧的“扩展管理”按钮,并在搜索框中输入“Coding扩展”,以在搜索结果列表中选择“Coding扩展”进行安装。在安装完成后,VSCode编辑器左侧的功能栏中会显示有一个“调试检测组件”,具体参见图5所示。

[0250] (B)关于Coding扩展的使用阶段,可包括代码运行和代码调试这两部分。需要说明的是,Unity游戏中开发有WebSocket服务器,而Coding扩展作为WebSocket客户端,启动WebSocket服务来实现Vscod编程器与Unity游戏之间的代码或消息传输,而在调试阶段,Coding扩展在启动时就通过WebSocket连接到Unity游戏。

[0251] 首先,用户打开编程环境对应的客户端,显示界面中会展示出游戏画面(如图4所示),用户点击游戏画面左侧的“代码编辑”,拉起VSCode编辑器,同时并排分布显示VSCode编辑器对应的代码编辑页面,关于点编辑页面与游戏画面的排布关系可参见图6。进而,用户在代码编辑页面中编辑Python代码,关于Python代码的编辑过程参见前述,此处不做赘述。

[0252] 然后,当用户编写好自己的程序后,可进入Coding扩展的使用阶段,以开始调试Python代码。

[0253] 其中,在代码运行方面,可结合图16、19和21所示,该过程可简括为:Coding扩展请求Python扩展启动调试流程,而Python扩展请求Python解释器运行Python代码,该运行过为:codingtge编程库会和Unity游戏建立WebSocket连接;针对程序代码中涉及codingtge库的接口,都会转换成一条请求协议消息,具体的,将调用的codingtge的api和参数信息,转换为JSON格式的指令消息,如“reuset:SetRateInFoot[2]”、“reuset:Wait[0.5]”,并通过WebSocket客户端连接发送给Unity游戏;Unity游戏接收到消息后,会解析JSON格式的消息,并将其转换为游戏中主角的相应动作,发送给游戏中的程序控制模块,最终操作游戏对象的动作;此外,Unity游戏处理完消息后会发送一个response的消息,用于通知codingtge编程库指令已经完成,如定时/实时通过发送“sync_data”协议,同步机器狗和场景相关的数据,给Python运行时保存,并文字方式展示在代码编辑页面内中,以便用户实施查看。

[0254] 其中,在代码调试方面,可结合图16、20和21所示,该过程可以检测简括为:第一步,Coding扩展检测到Python扩展发来的调试消息,将调试消息通过WebSocket连接发送给Unity游戏。

[0255] 需要说明的是,Coding扩展在代码调试阶段,仅处理两个类型的事件消息,如“stopped事件消息”(停止调试消息)和“continued事件消息”(即继续调试信息)。关于这两个的事件消息的介绍如下:

[0256] 当Coding扩展接收到stopped事件消息时,检测检查停止调试的原因(例如,是调试断点还是异常),并基于WebSocket连接,向Unity游戏服务器发送一个包含停止调试消息的JSON对象,以通知Unity开发的游戏暂停游戏逻辑。

[0257] 当Coding扩展接收到continued事件消息时,向Unity游戏服务器发送一个包含继续调试消息的JSON对象,以通知Unity开发的游戏恢复游戏逻辑。

[0258] 第二步,Unity开发的游戏收到消息后,根据不同类型的消息,通过程序控制模块决定游戏的暂停、恢复。具体的,游戏收到调试事件的消息后,会主动调用程序控制模块,去控制所有和游戏主角相关的系统,该系统不限于包括用于实现物理碰撞效果的物理系统、播放角色动作的动画系统以及游戏逻辑。以此,使得游戏暂停时,Unity游戏中的游戏主角不受其他系统的影响,完全静止,不可操作;当游戏恢复时,Unity游戏中的游戏主角可以从当前状态中恢复过来,继续按原定逻辑运行。

[0259] 通过以上应用场景实例,基于VSCode编辑器的基础功能和插件,开发一个Python-Unity游戏拓展(即coding扩展),以此,使得VSCode、Python和Unity游戏之间关联,以为用户提供一下良好的编程进阶学习的途径。具体可实现如下效果:提供更加丰富、灵活的编辑学习工具,得编程的过程更加愉悦和自由,同时,提高用户的编程效率和质量,提高用户的编程学习体验。

[0260] 可实现如下效果:通过改进的命名实体识别算法将热点与目标事件的关联判别、目标事件情感倾向判别、目标事件主体抽取的3个任务相结合,简化了热点关联负面事件的识别任务,提升了模型准确率及运行效率。

[0261] 由以上可知,本申请实施例可先显示角色控制页面和关联的代码编辑页面,并通过代码编辑页面显示用户实时编辑的程序代码和设定在程序代码之间的调试断点,然后,对程序代码进行调试,以及根据调试过程的运行结果控制角色控制页面中的虚拟对象进行运动,同时,检测针对程序代码的调试进度,最后,在调试进度达到调试断点时生成停止调试消息,以根据停止调试消息控制角色控制页面中的虚拟对象停止运动,并在检测到针对调试断点之后的程序代码的继续调试指令时,控制虚拟对象继续运动;以此,可根据设定的调试断点来实时调试程序代码,无需逐句检查程序代码,缩短了编程学习周期,提高用户的编程进阶学习的效率,此外,将用户所编辑的程序代码的控制逻辑展示角色控制页面中,提供良好的编程学习方式,提高用户的体验。

[0262] 为了更好地实施以上方法,本申请实施例还提供一种程序调试装置。例如,如图22所示,该程序调试装置可以包括显示单元401、第一调试单元402、生成单元403和第二调试单元404。

[0263] 显示单元401,用于显示角色控制页面以及角色控制页面关联的代码编辑页面,代码编辑页面显示实时编辑的程序代码以及位于程序代码中的调试断点;

[0264] 第一调试单元402,用于调试代码编辑页面中的程序代码,并根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动;

[0265] 生成单元403,用于当检测到程序代码的调试进度到达调试断点时,生成停止调试消息,并根据停止调试消息控制虚拟对象停止运动;

[0266] 第二调试单元404,用于响应于接收到的继续调试指令,运行调试断点之后的程序代码,并根据调试断点之后的程序代码的运行结果控制虚拟对象继续运动。

[0267] 在一些实施方式中,显示单元401,还用于:显示角色控制页面,角色控制页面至少包含虚拟对象和代码编辑组件;响应于针对代码编辑组件的触发操作,显示代码编辑组件对应的代码编辑页面,其中,代码编辑页面与角色控制页面之间并排分布。

[0268] 在一些实施方式中,显示单元401,还用于:响应于在代码编辑页面中的输入操作,在代码编辑页面中展示实时编辑的程序代码;获取代码编辑页面内被选中的程序代码,并根据调试断点标记被选中的程序代码。

[0269] 在一些实施方式中,代码编辑页面包含动作逻辑库和代码存储文件,显示单元401,还用于:响应于针对动作逻辑库的触发操作,显示动作逻辑库对应的动作逻辑列表;获取动作逻辑列表内被选中的目标函数,并以目标函数作为程序代码嵌入至代码存储文件内;在代码编辑页面展示代码存储文件对应的文件信息窗口,文件信息窗口展示实时编辑的程序代码。

[0270] 在一些实施方式中,代码编辑页面还包含调试检测组件,第一调试单元402,还用于:响应于针对调试检测组件的触发操作,运行代码编辑页面中的程序代码;检测程序代码在运行时的调试进度。

[0271] 在一些实施方式中,第一调试单元402,还用于:运行调试断点之前的程序代码,并从调试断点之前的程序代码中选取虚拟对象的动作控制代码;将动作控制代码传输至角色控制页面对应的服务器,以使得服务器在执行动作控制代码后返回对应的动作指令;基于动作指令,控制角色控制页面中的虚拟对象进行运动。

[0272] 在一些实施方式中,第一调试单元402,还用于:若识别到动作控制指令为以目标速度的行走指令,则根据目标速度控制角色控制页面中的虚拟对象向前行走;在虚拟对象的后方区域显示行走轨迹。

[0273] 在一些实施方式中,代码编辑页面还包括表示代码调试过程的可控状态的调试控制组件,生成单元403,还用于:暂停运行调试断点之后的程序代码,并将调试状态切换为暂停调试状态;基于暂停调试状态,将调试控制组件的状态图标显示为继续调试状态图标,并生成暂停调试状态对应的停止调试消息。

[0274] 在一些实施方式中,程序调试装置还包括提示单元,用于:将代码编辑页面中调试断点对应的程序代码的行区域进行彩色标记;获取位于调试断点之前的程序代码的运算结果和运行堆栈信息;在代码编辑界面中生成调试信息窗口,调试信息窗口用于显示运算结果和运行堆栈信息。

[0275] 在一些实施方式中,生成单元403,还用于:将停止调试消息发送至角色控制页面对应的服务器,使得服务器在解析停止调试消息后返回停止运动指令;基于停止运动指令,控制角色控制页面中的虚拟对象停止运动。

[0276] 在一些实施方式中,第二调试单元404,还用于:响应于接收到的继续运行指令,运行调试断点之后的程序代码,并生成继续执行消息;将继续执行消息发送至角色控制页面对应的服务器,使得服务器恢复角色控制页面对应的运行逻辑、物理系统和动画系统。

[0277] 由以上可知,本申请实施例可先显示角色控制页面和关联的代码编辑页面,并通过代码编辑页面显示用户实时编辑的程序代码和设定在程序代码之间的调试断点,然后,对程序代码进行调试,以及根据调试过程的运行结果控制角色控制页面中的虚拟对象进行运动,同时,检测针对程序代码的调试进度,最后,在调试进度达到调试断点时生成停止调

试消息,以根据停止调试消息控制角色控制页面中的虚拟对象停止运动,并在检测到针对调试断点之后的程序代码的继续调试指令时,控制虚拟对象继续运动;以此,可根据设定的调试断点来实时调试程序代码,无需逐句检查程序代码,缩短了编程学习周期,提高用户的编程进阶学习的效率,此外,将用户所编辑的程序代码的控制逻辑展示角色控制页面中,提供良好的编程学习方式,提高用户的体验。

[0278] 本申请实施例还提供一种计算机设备,如图23所示,其示出了本申请实施例所涉及的计算机设备的结构示意图,具体来讲:

[0279] 该计算机设备可以包括一个或者一个以上处理核心的处理器501、一个或一个以上计算机可读存储介质的存储器502、电源503和输入单元504等部件。本领域技术人员可以理解,图23中示出的计算机设备结构并不构成对计算机设备的限定,可以包括比图示更多或更少的部件,或者组合某些部件,或者不同的部件布置。其中:

[0280] 处理器501是该计算机设备的控制中心,利用各种接口和线路连接整个计算机设备的各个部分,通过运行或执行存储在存储器502内的软件程序和/或模块,以及调用存储在存储器502内的数据,执行计算机设备的各种功能和处理数据。可选的,处理器501可包括一个或多个处理核心;优选的,处理器501可集成应用处理器和调制解调处理器,其中,应用处理器主要处理操作系统、用户界面和应用程序等,调制解调处理器主要处理无线通信。可以理解的是,上述调制解调处理器也可以不集成到处理器501中。

[0281] 存储器502可用于存储软件程序以及模块,处理器501通过运行存储在存储器502的软件程序以及模块,从而执行各种功能应用以及程序调试过程。存储器502可主要包括存储程序区和存储数据区,其中,存储程序区可存储操作系统、至少一个功能所需的应用程序(比如声音播放功能、图像播放功能等)等;存储数据区可存储根据计算机设备的使用所创建的数据等。此外,存储器502可以包括高速随机存取存储器,还可以包括非易失性存储器,例如至少一个磁盘存储器件、闪存器件、或其他易失性固态存储器件。相应地,存储器502还可以包括存储器控制器,以提供处理器501对存储器502的访问。

[0282] 计算机设备还包括给各个部件供电的电源503,优选的,电源503可以通过电源管理系统与处理器501逻辑相连,从而通过电源管理系统实现管理充电、放电、以及功耗管理等功能。电源503还可以包括一个或一个以上的直流或交流电源、再充电系统、电源故障检测电路、电源转换器或者逆变器、电源状态指示器等任意组件。

[0283] 该计算机设备还可包括输入单元504,该输入单元504可用于接收输入的数字或字符信息,以及产生与用户设置以及功能控制有关的键盘、鼠标、操作杆、光学或者轨迹球信号输入。

[0284] 尽管未示出,计算机设备还可以包括显示单元等,在此不再赘述。具体在本申请实施例中,计算机设备中的处理器501会按照如下的指令,将一个或一个以上的应用程序的进程对应的可执行文件加载到存储器502中,并由处理器501来运行存储在存储器502中的应用程序,从而实现各种功能,如下:

[0285] 显示角色控制页面以及角色控制页面关联的代码编辑页面,代码编辑页面显示实时编辑的程序代码以及位于程序代码中的调试断点;调试代码编辑页面中的程序代码,并根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动;当检测到程序代码的调试进度到达调试断点时,生成停止调试消息,并根据停止调试消息控制虚拟

对象停止运动;响应于接收到的继续调试指令,运行调试断点之后的程序代码,并根据调试断点之后的程序代码的运行结果控制虚拟对象继续运动。

[0286] 以上各个操作的具体实施可参见前面的实施例,在此不作赘述。

[0287] 由此可得,本方案可先显示角色控制页面和关联的代码编辑页面,并通过代码编辑页面显示用户实时编辑的程序代码和设定在程序代码之间的调试断点,然后,对程序代码进行调试,以及根据调试过程的运行结果控制角色控制页面中的虚拟对象进行运动,同时,检测针对程序代码的调试进度,最后,在调试进度达到调试断点时生成停止调试消息,以根据停止调试消息控制角色控制页面中的虚拟对象停止运动,并在检测到针对调试断点之后的程序代码的继续调试指令时,控制虚拟对象继续运动;以此,可根据设定的调试断点来实时调试程序代码,无需逐句检查程序代码,缩短了编程学习周期,提高用户的编程进阶学习的效率,此外,将用户所编辑的程序代码的控制逻辑展示角色控制页面中,提供良好的编程学习方式,提高用户的体验。

[0288] 本领域普通技术人员可以理解,上述实施例的各种方法中的全部或部分步骤可以通过指令来完成,或通过指令控制相关的硬件来完成,该指令可以存储于一计算机可读存储介质中,并由处理器进行加载和执行。

[0289] 为此,本申请实施例提供一种计算机可读存储介质,其中存储有多条指令,该指令能够被处理器进行加载,以执行本申请实施例所提供的任一种程序调试方法中的步骤。例如,该指令可以执行如下步骤:

[0290] 显示角色控制页面以及角色控制页面关联的代码编辑页面,代码编辑页面显示实时编辑的程序代码以及位于程序代码中的调试断点;调试代码编辑页面中的程序代码,并根据调试断点之前的程序代码的运行结果控制角色控制页面中虚拟对象的运动;当检测到程序代码的调试进度到达调试断点时,生成停止调试消息,并根据停止调试消息控制虚拟对象停止运动;响应于接收到的继续调试指令,运行调试断点之后的程序代码,并根据调试断点之后的程序代码的运行结果控制虚拟对象继续运动。

[0291] 以上各个操作的具体实施可参见前面的实施例,在此不再赘述。

[0292] 其中,该计算机可读存储介质可以包括:只读存储器(ROM,Read Only Memory)、随机存取记忆体(RAM,Random Access Memory)、磁盘或光盘等。

[0293] 由于该计算机可读存储介质中所存储的指令,可以执行本申请实施例所提供的任一种程序调试方法中的步骤,因此,可以实现本申请实施例所提供的任一种程序调试方法所能实现的有益效果,详见前面的实施例,在此不再赘述。

[0294] 根据本申请的一个方面,提供了一种计算机程序产品或计算机程序,该计算机程序产品或计算机程序包括计算机指令,该计算机指令存储在计算机可读存储介质中。计算机设备的处理器从计算机可读存储介质读取该计算机指令,处理器执行该计算机指令,使得该计算机设备执行上述实施例提供的各种可选实现方式中提供的方法。

[0295] 以上对本申请实施例所提供的一种程序调试方法、装置、设备和计算机可读存储介质进行了详细介绍,本文中应用了具体个例对本申请的原理及实施方式进行了阐述,以上实施例的说明只是用于帮助理解本申请的方法及其核心思想;同时,对于本领域的技术人员,依据本申请的思想,在具体实施方式及应用范围上均会有改变之处,综上,本说明书内容不应理解为对本申请的限制。

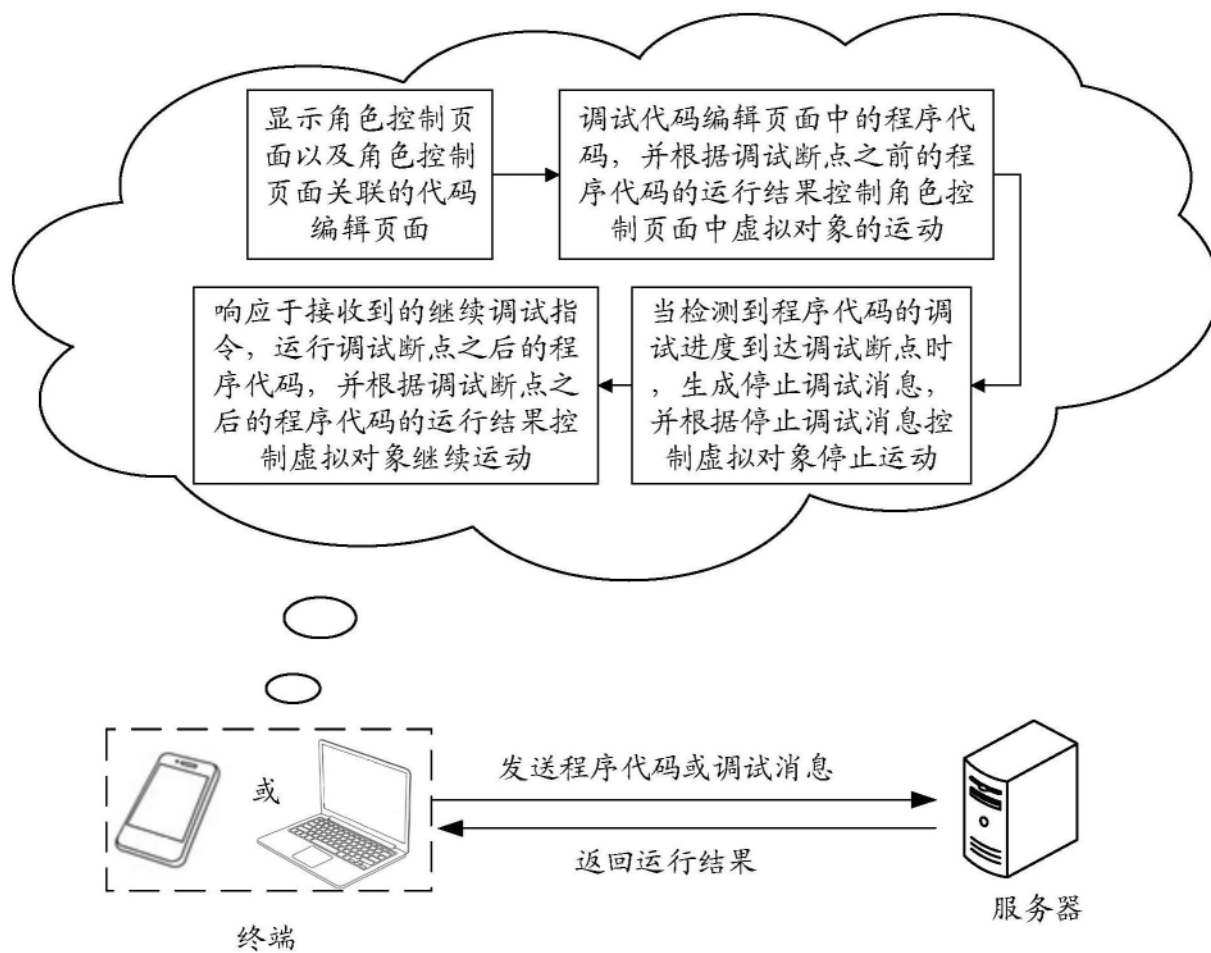


图1

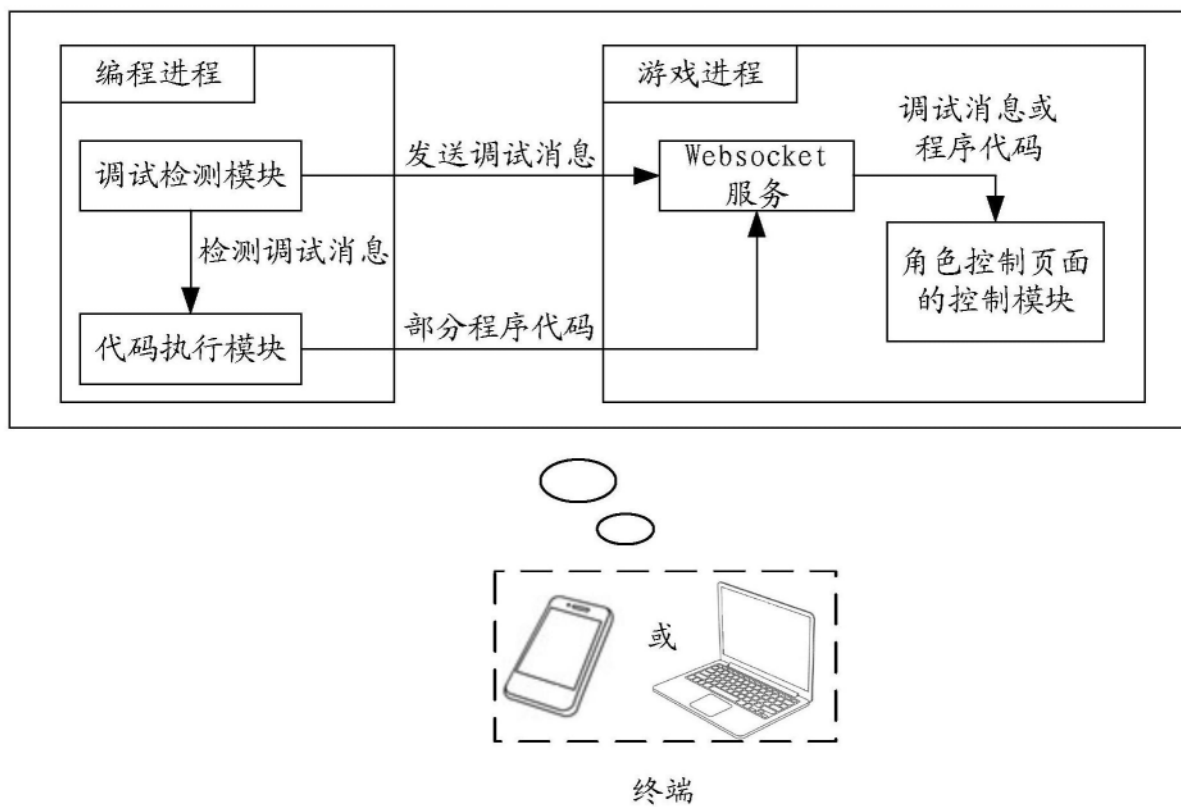


图2

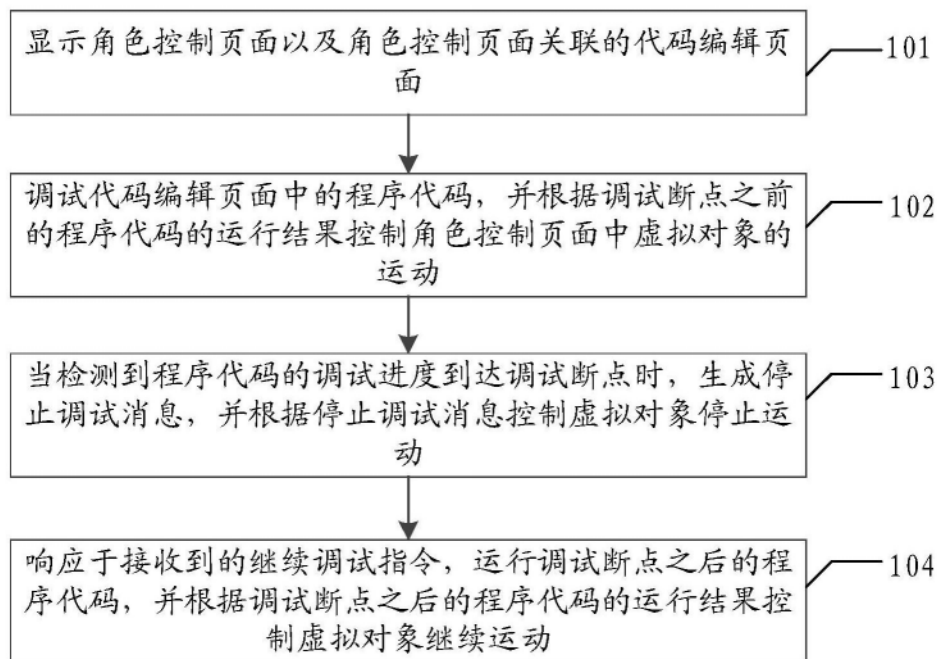


图3

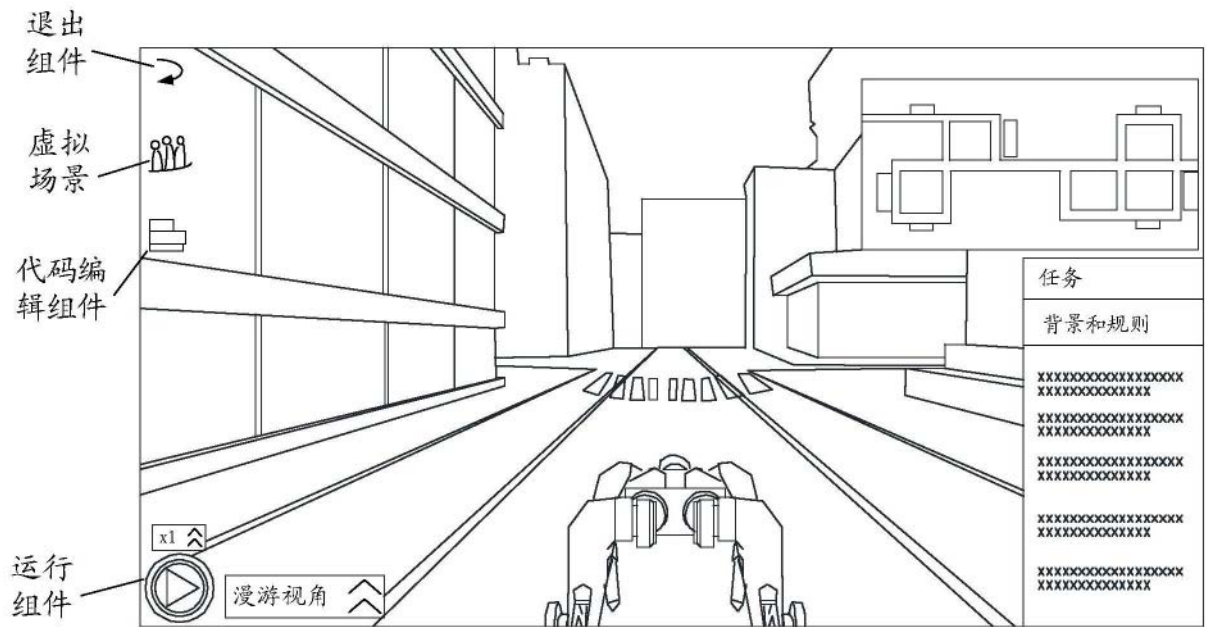


图4

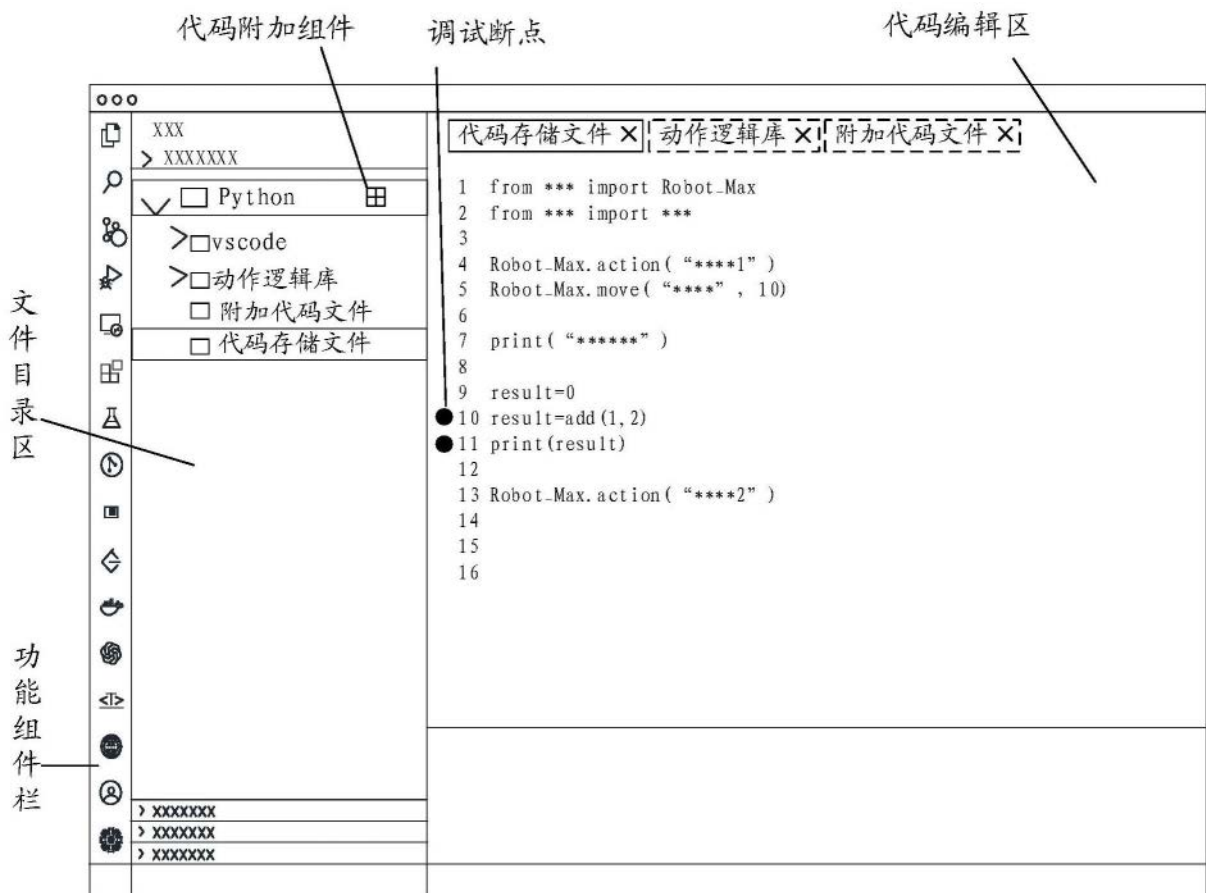


图5

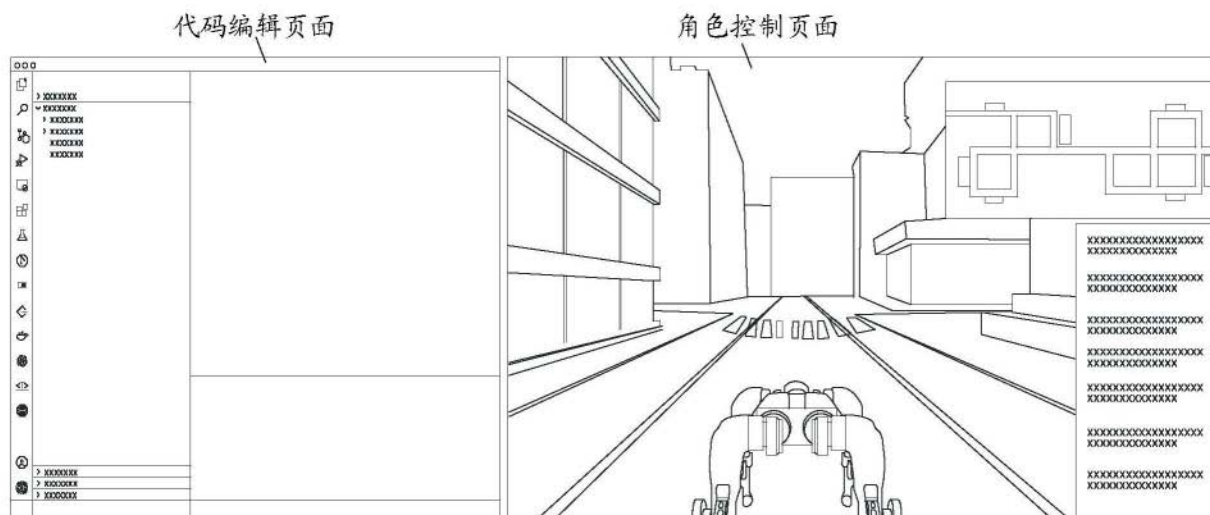


图6

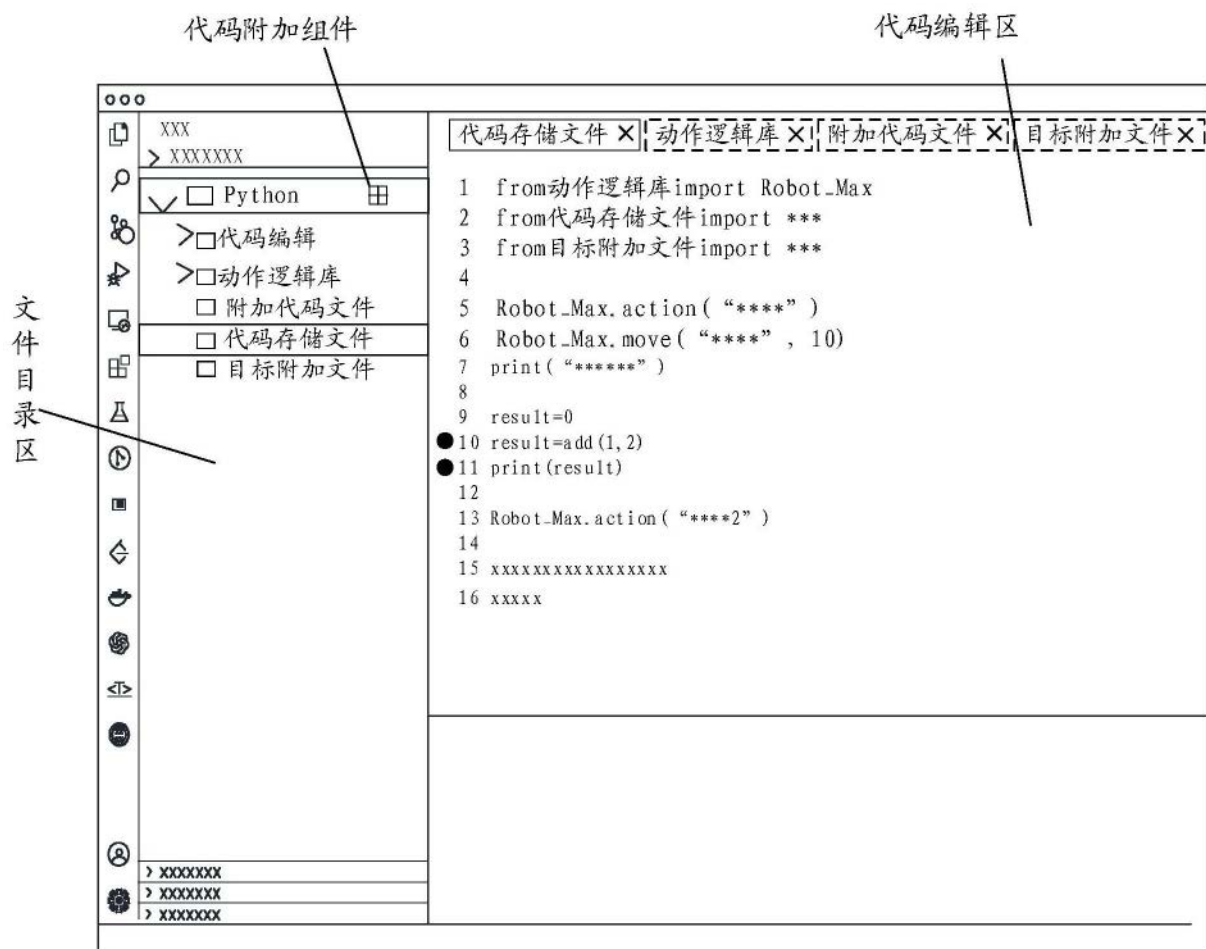


图7

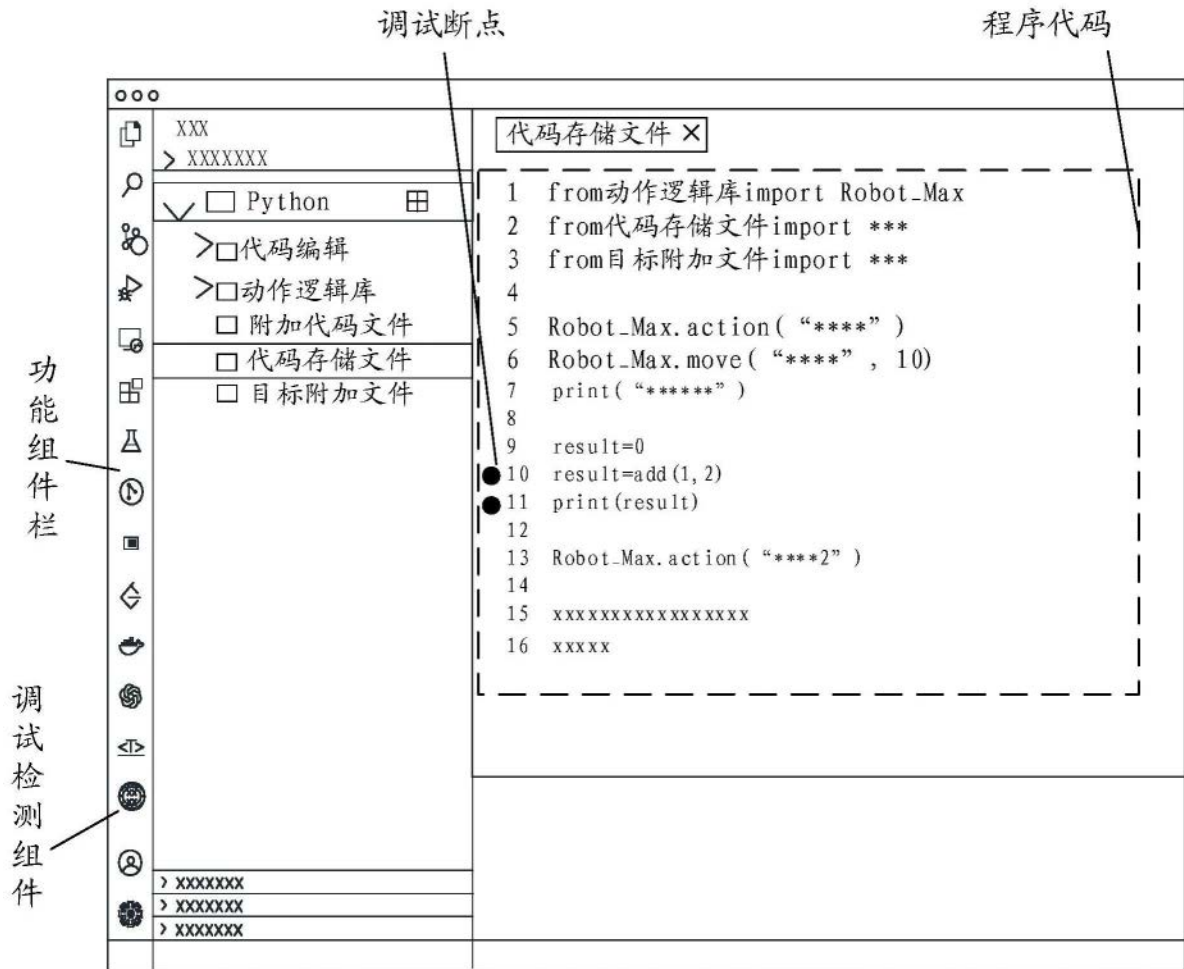


图8

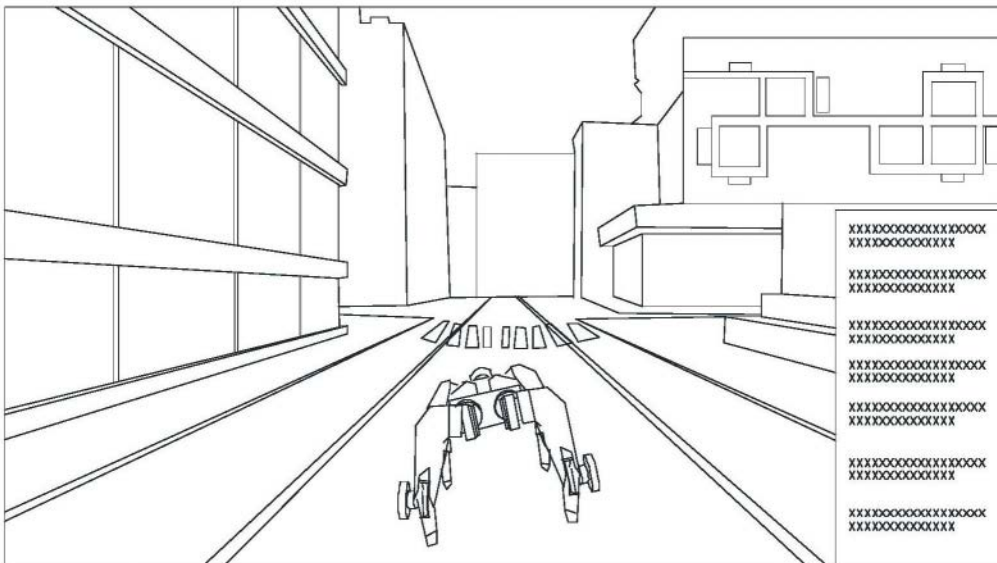


图9

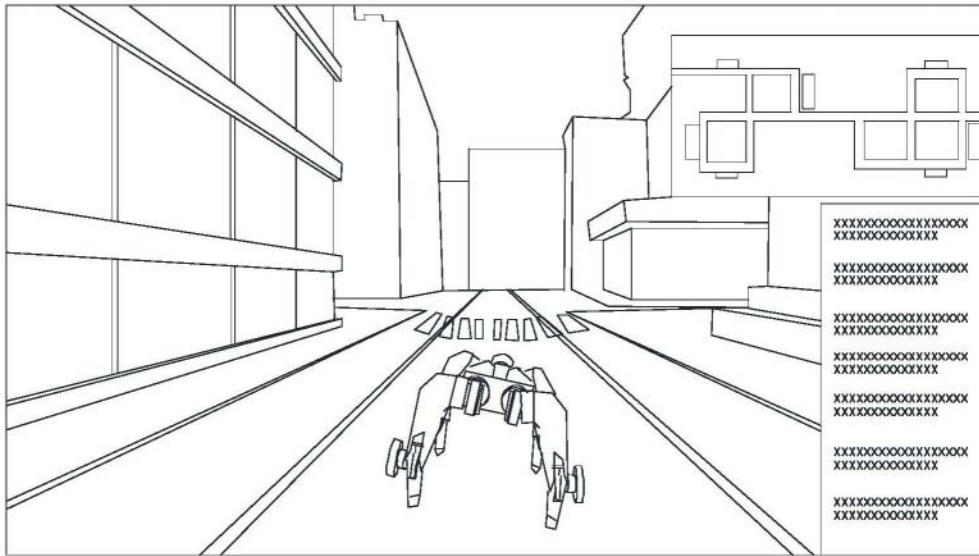


图10

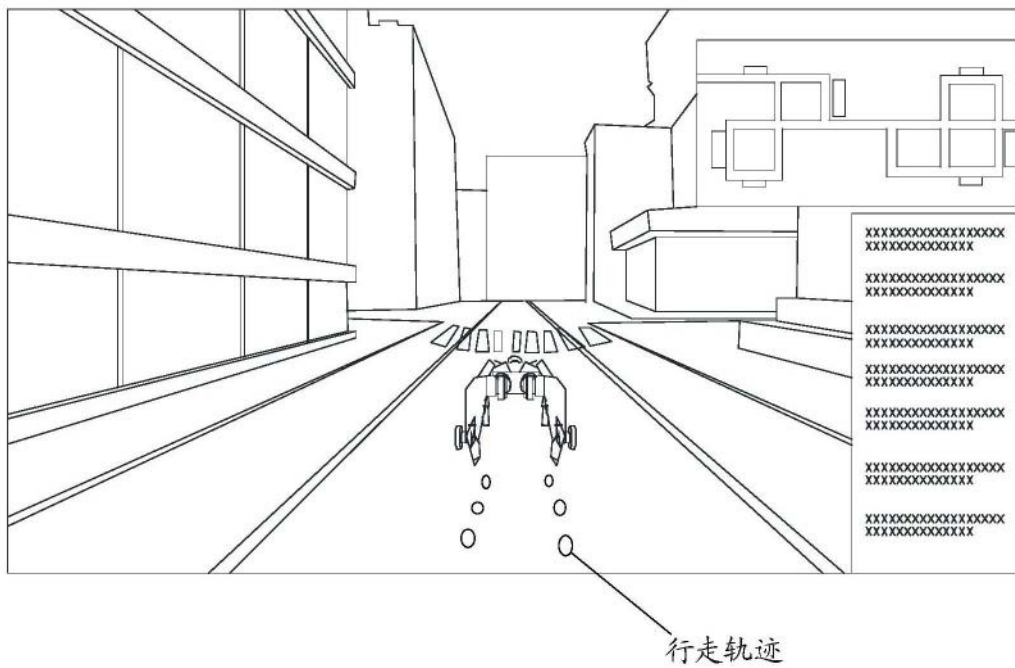


图11

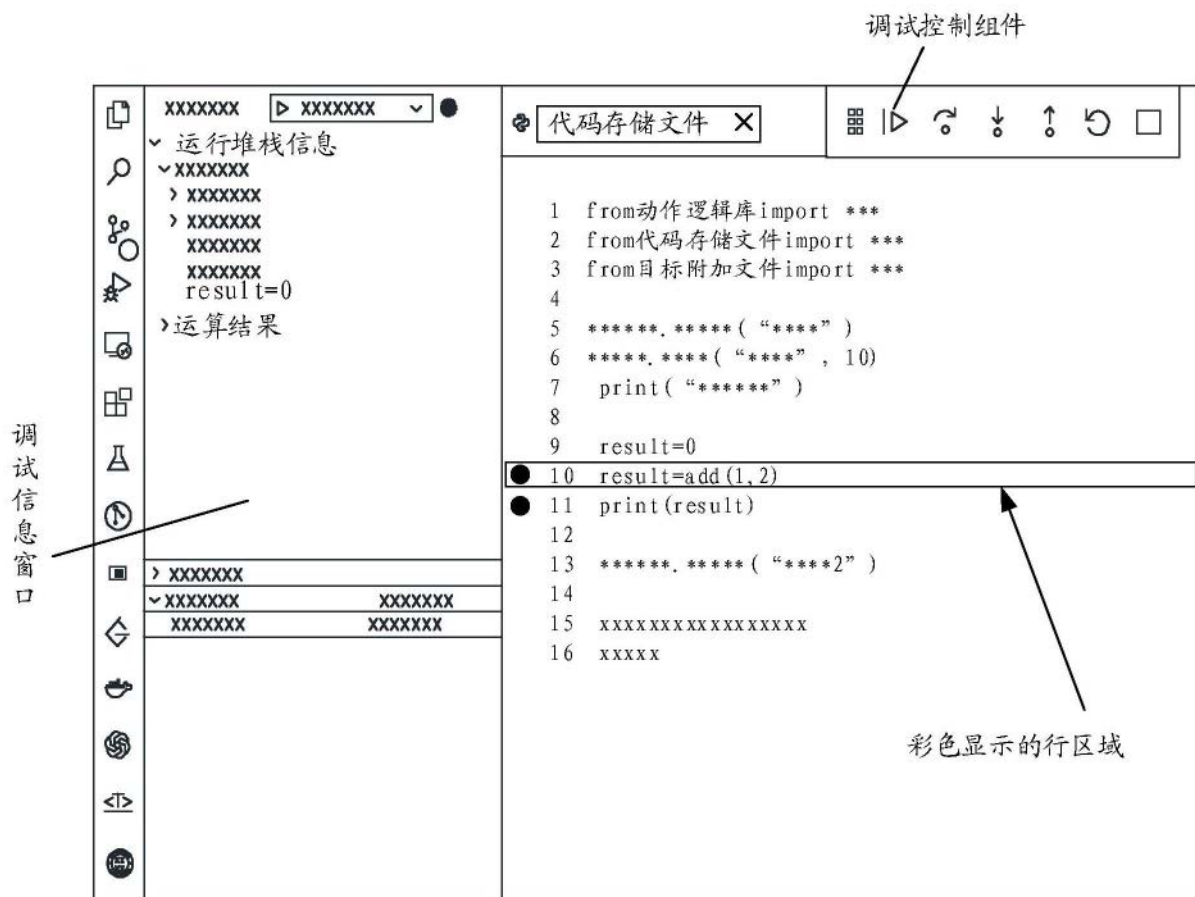


图12

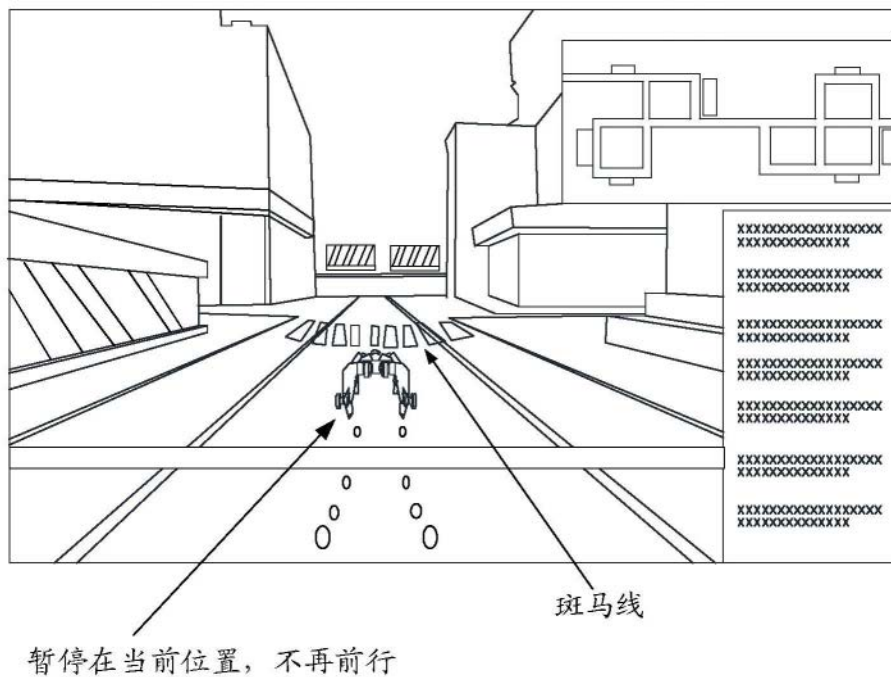


图13

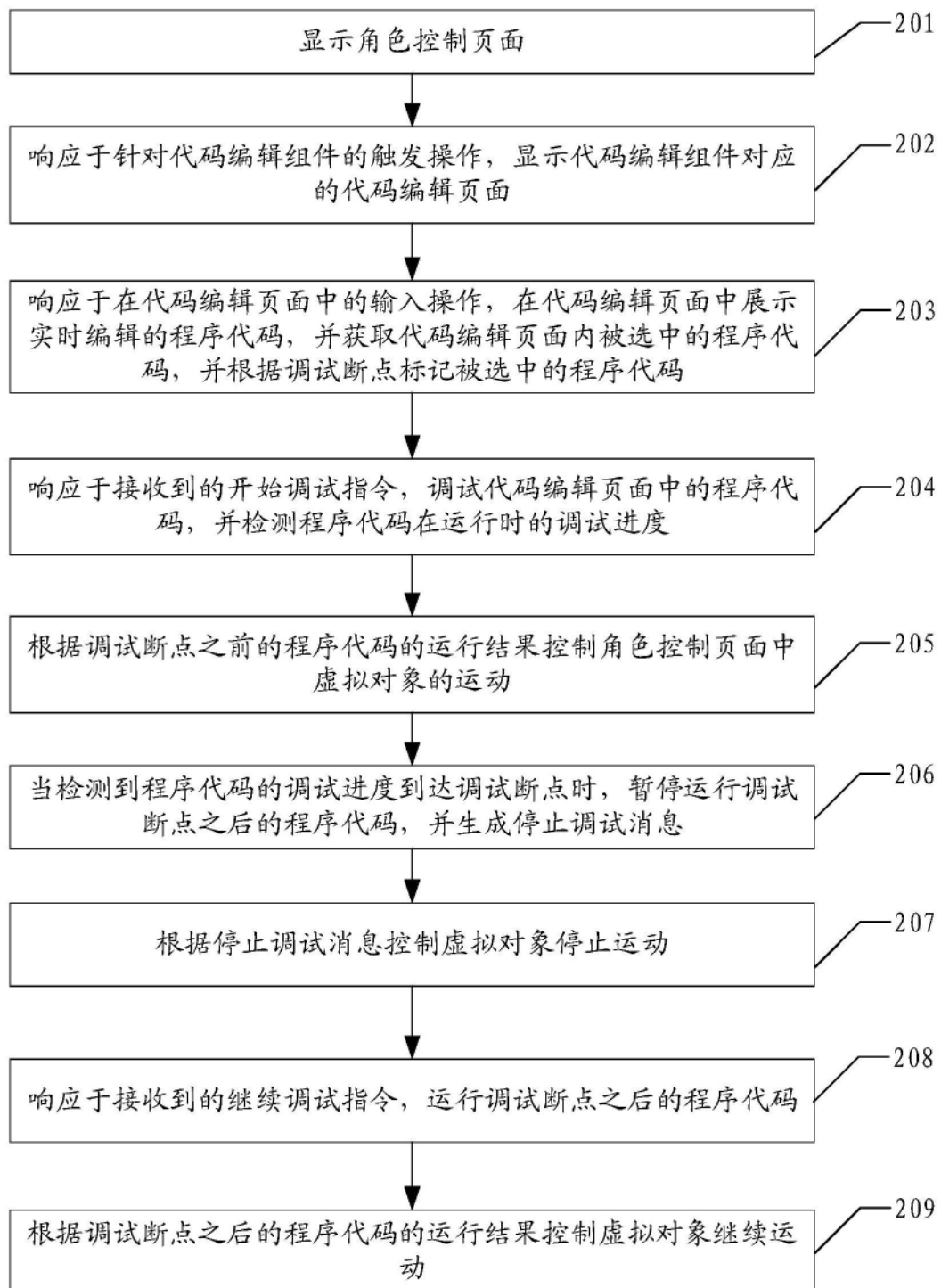


图14

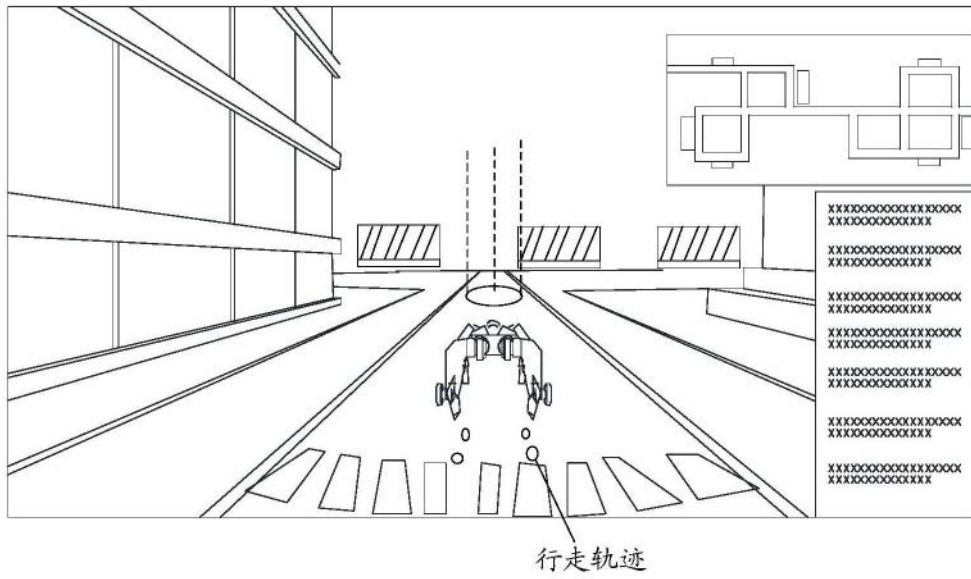


图15

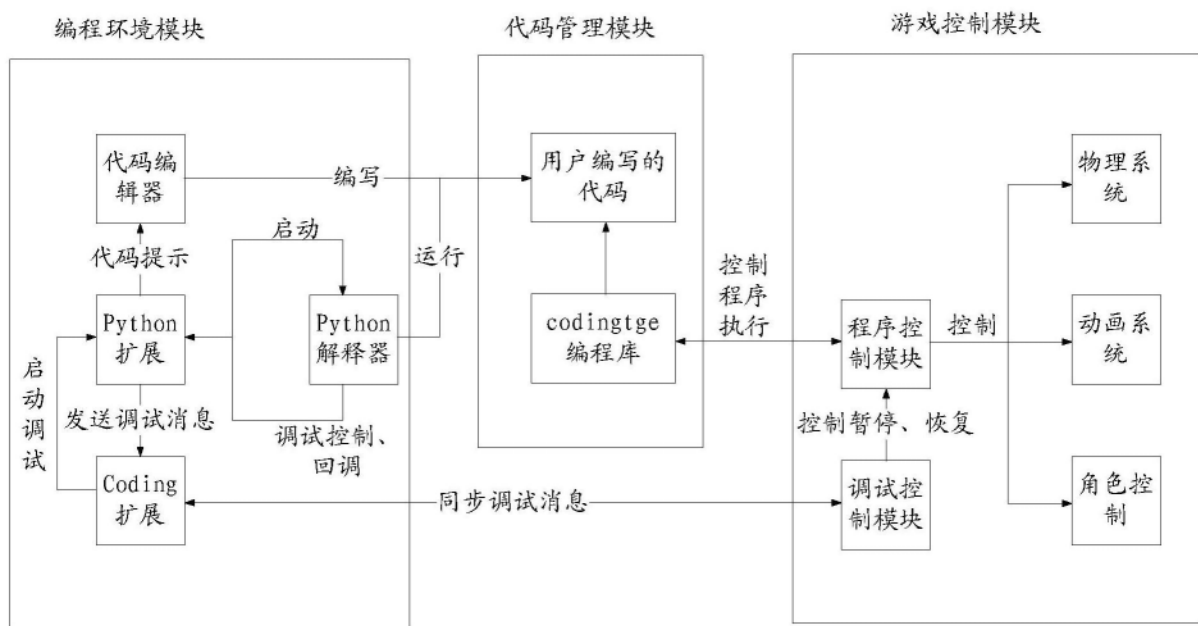


图16

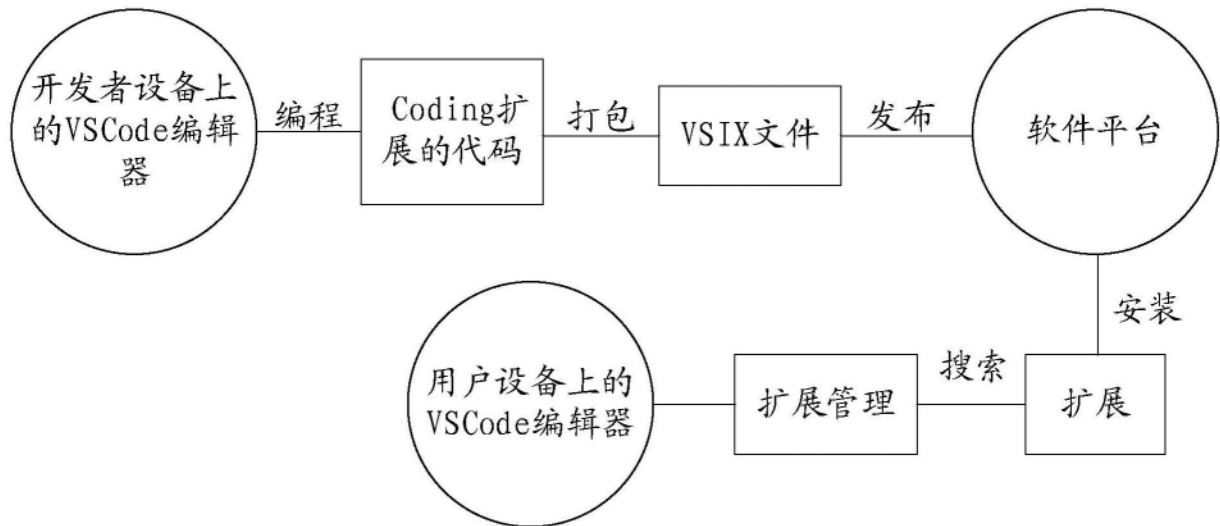


图17

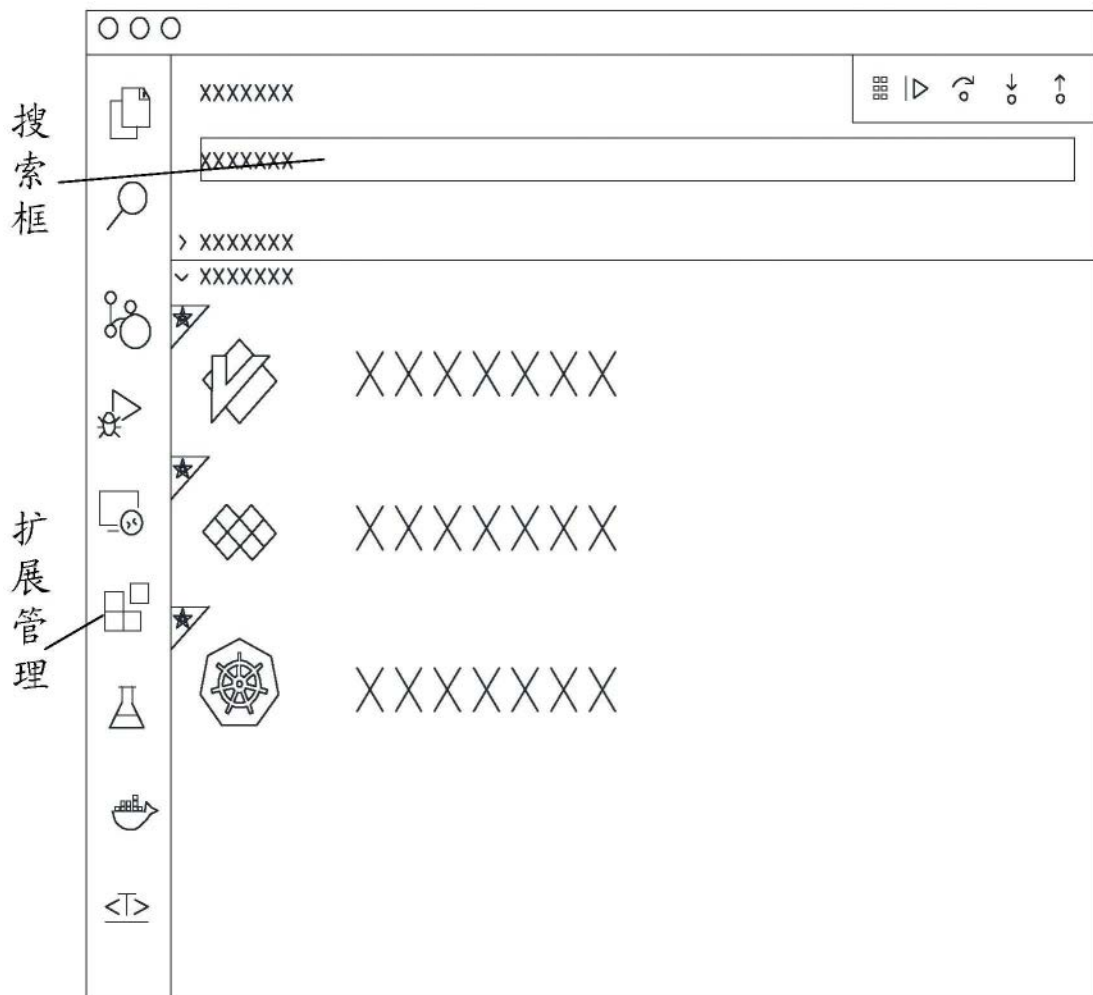


图18

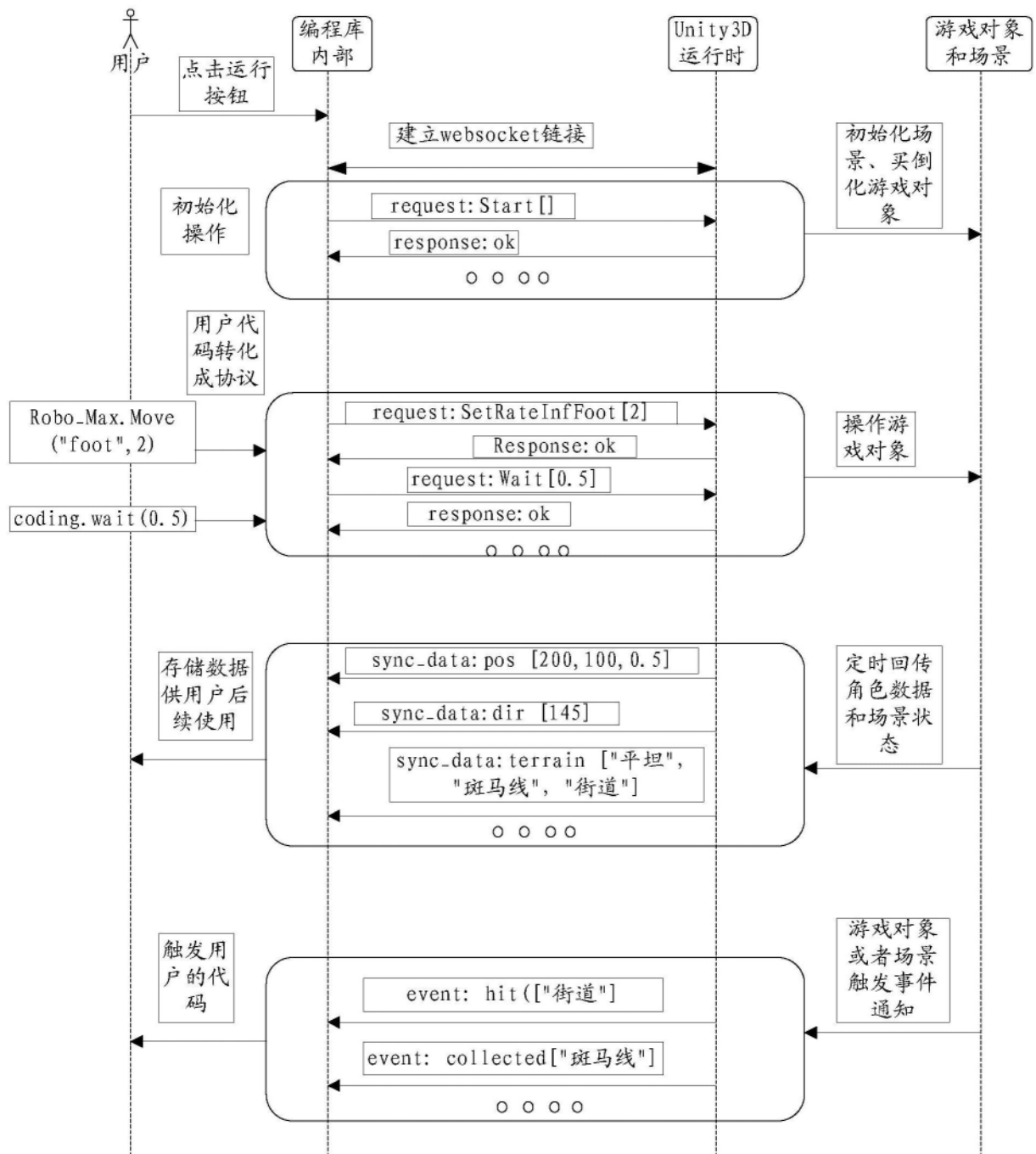


图19

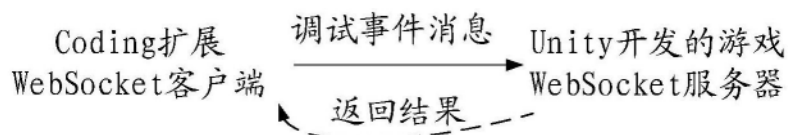


图20

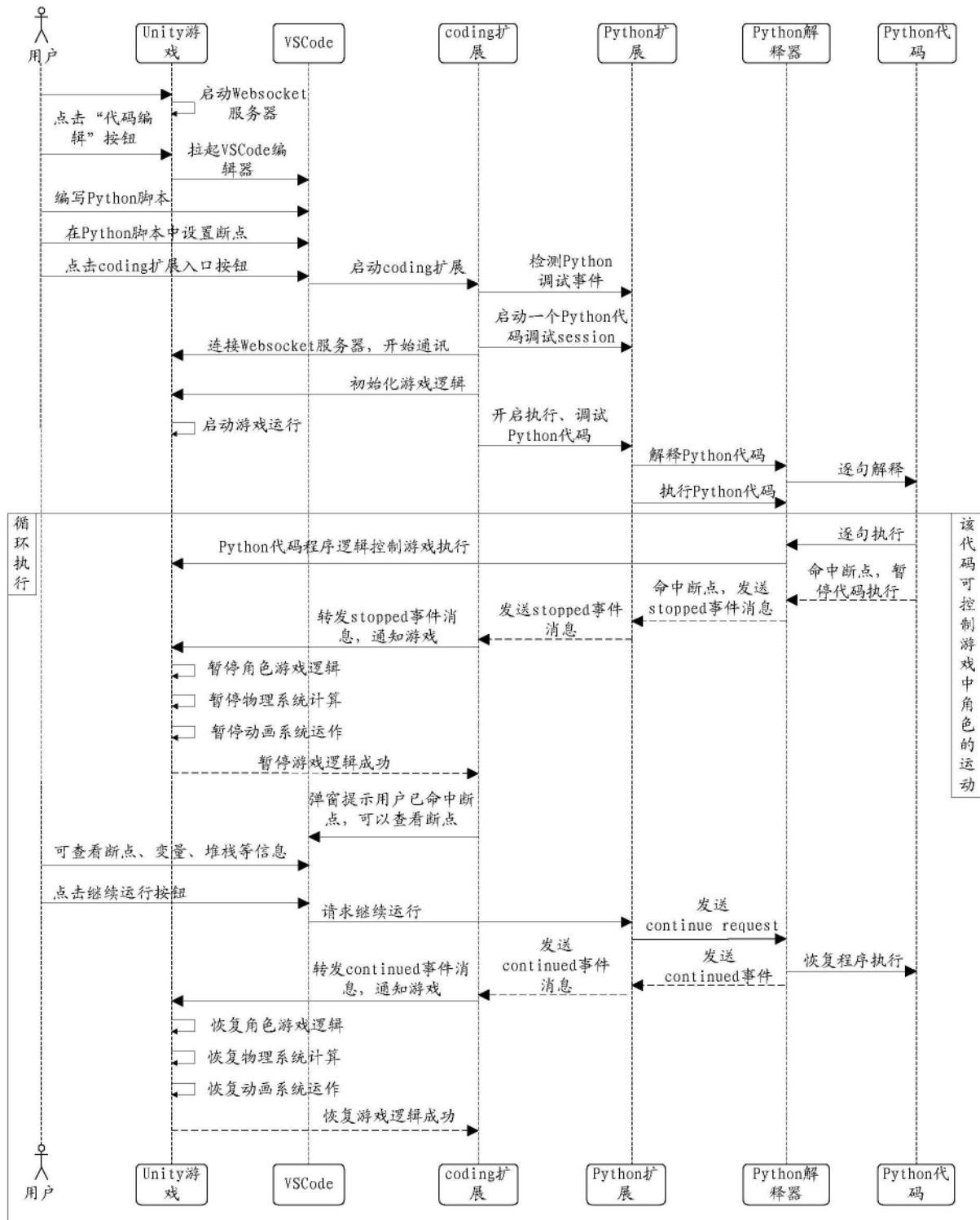


图21



图22

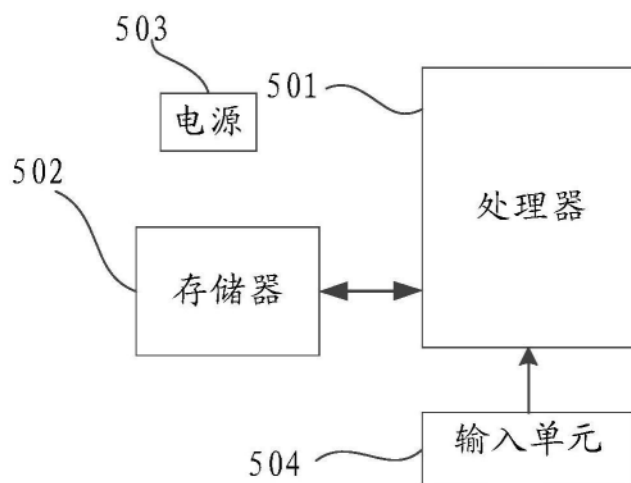


图23