状态同步机制研究

以公司MRMxer为例子，研究如何应用。

是什么

状态同步机制是一种网络同步机制。它同步的是模板中的各种状态，它的流程简而言之就是客户端上传指令，服务端下发状态，客户端播放状态。体现在MRMixer为：客户端上传操作到服务器，服务器收到后计算出操作的结果，然后服务器以广播的方式下发模板中的各种状态，客户端收到状态后，更新自己本地的状态就可以了，即植入物的移动和渲染都由服务器运算。但是为了给玩家好的体验，减少同步的数据量，客户端也需要做很多的本地运算，减少服务器同步的频率以及数据量。

同步机制的目的在于通过同步信息能够模拟出其他用户的行为。在状态同步中，我们可以把每一个用户的行为方式抽象成多个状态，每一个状态都有定义好的行为方式。例如在MRMxer中有个操作是将房间的门打开，我们可以告诉客户端在某坐标的门处于打开状态，只要这个状态不变化，那么在没有接受到门的其他任何同步数据的情况下，就可以就知道门处于打开状态。状态同步中，植入物上面的每一个状态都相当于一个具有固定逻辑的行为模式，这个行为模式类似于黑盒，只需要给到相应的数据，就能表现出相应的行为，只要给到植入物相同的状态数据，植入物就会表现出相同的画面效果。当我们对某个植入物发出操作指令时，就是在告诉它应该先触发这个状态再触发那个状态，同时指令包含了状态需要的数据，当指令被广播到房间里的设备时，设备就会同步指令中的数据去模拟植入物的行为。

传输的是什么

在状态同步中，服务端中维护了多个场景，存储了多个完整的场景信息，而场景内部维护了场景中有哪些植入物，场景中的植入物都处于什么状态。当客户端上报操作指令时，服务器执行指令，在指令执行过程中就会引起植入物状态的改变，服务器就将这些改变了的状态信息转发给其他客户端，【下发时机可以是每执行完一个命令也可以是每隔一段时间【有50ms ,100ms】】。

对服务器来讲，它下发的是植入物状态【客户端不用运算就能渲染的那种，可以是场景的快照，也可以是场景的变化增量，变化增量以指令分组】，所以传输的是场景内各种植入物的各种状态，状态信息可能的结构体如下：

{“sceneID”:1,”objID”:2,”position”:”1,2,3”,”type”:”material”,”color”:”yellow”,”ratate”:”1” }

对客户端来讲，他上报的是操作指令，他可能的结构体为

{”sceneID”:1,”userID”:2,”command”:”mrtp://system/drag?id=SphereD”,”time”:”1356897524”}

如何保证服务端和客户端的状态一致性

会引起客户端和服务端状态不一致的原因有：1.到达客户端的状态信息指令顺序不对或者有缺失。2.客户端接收到相同的状态信息后渲染的效果不同。

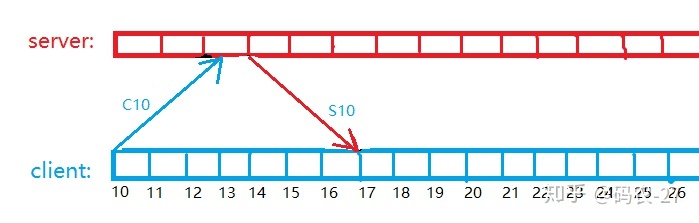
1. 采取TCP协议进行通讯，TCP协议保证了数据的顺序传输，避免后发先到的情况发生。[也引入了一个缺点，弱网环境下的用户体验贼差]
2. 客户端实现较好的兼容性，即相同的状态信息传入，在不同配置的多个设备上，植入物的行为都是一致的。

状态同步的关键问题及解决方案：

1. 输入预测

客户端上传指令涉及到网络延迟,服务端下发状态信息也涉及到网络延迟,如果没有输入预测的话,那么用户的操作,需要等到服务端的状态信息到了才生效，体验很不好,为了避免这样的体验,需要对客户端的指令进行预测。输入预测指的是:客户端生成了一个指令,在将放到上传指令队列的同时,客户端自身也执行这个指令。受网络延迟和模拟时间的影响，客户端又在预测自己，还需要解决预测失败的情况。

业内的解决方案通常是客户端缓存预测的命令和预测的结果，如图：



客户端模拟完自己的指令以后,缓存模拟过的指令和模拟结果,当收到服务器的快照包以后,将快照对应的指令和自己缓存的指令做对比,如果模拟结果不同,就需要使用服务器的快照,然后在此快照的基础上,模拟缓存的预测帧. 客户端在自己10帧的时候上传了当前的操作指令C10,同时客户端模拟操作,也继续采样操作模拟,从C10开始缓存模拟过的指令,当模拟到17帧的时候,收到了服务器的快照S10,这个时候客户端缓存了C10到C17的指令,这个时候,需要拿客户端C10预测结果跟S10的快照状态做比较,如果相等,那么没关系,继续从C17往前模拟,如果不等,那么就将S10设置当前状态,再从C11模拟到C17.

1. 延迟补偿

状态同步中，客户端中的植入物会分为两种，一种是自己可以操控的，一种不是自己操控的，对于不受控的植入物，会把植入物的状态直接设置为将服务器下发的状态信息中对应状态，对于受自己操控的植入物，由于输入预测的存在，就需要将状态信息中的对应状态和缓存中的做比对，然后预测执行指令。

对于用户而言，在客户端中现显示的能操作的植入物处于现在，而不能被操作的植入物出入过去，这就不适合应用在对时间或者空间比较敏感的场景中【例如在火灾现场灭火】，为了支持对时间和空间敏感的操作，引入延迟补偿机制。

机制核心在于服务器能够重建客户端在任何时间点看到的服务状态，使得服务器能够精确的知晓用户的操作目标【灭火究竟对准的物体是火还是其他】

帧同步机制：

简单来说，客户端发送游戏动作到服务器，服务器接收并汇总，然后直接转发给所有客户端，或者客户端直接通过P2P技术发送。客户端根据收到的游戏动作来做运算和显示。

帧同步的流程

同步随机种子：游戏中设计随机数的使用，通过同步随机数种子可以保持随机数的一致性。

客户端上传操作指令给服务器，操作指令包含游戏操作和当前帧索引。

游戏广播所有客户端的操作，如果没有操作也要广播空指令来驱动游戏帧前进帧同步主要依赖客户端的能力，服务器仅仅是做一个转发，甚至客户端可以无需服务器，通过P2P方式来转发数据。由于只是转发游戏的行为，所以广播的数据量比状态同步要小很多。

影响决定使用哪种同步机制的因素：

1. 掉线重连情况下，状态同步只需要获取服务端状态并还原就可以了，帧同步就需要从第一针开始将所有帧加速执行。

影响决定使用哪种同步机制的因素：

1. 使用状态同步的话，需要面临的问题有
2. 客户端生成合格的快照的难度【客户端能根据快照可以在非常短的时间内复原场景】
3. 快照文件的大小是否可控，快照文件过大则会给带来极大的服务器的带宽压力
4. 服务器上如何为每一个房间开辟一个伪客户端，即客户端能在linux平台上运行【制作MRMixer镜像，MRMixer兼容Linux】

二、使用帧同步的话，面临的问题有：

1. 帧同步技术最要的概念是”相同的输入 + 相同的时机 = 相同的显示“。游戏接收来自网络的多个客户端的操作，如果这些操作在各个客户端上都是一样的，那么多个客户端的显示也就是一样的，进而带来了”同步“的效果。客户端需要对一致性问题包括设置随机种子，审核静态变量等。

2. 它的逻辑性能优化有很大的压力。渲染帧和逻辑帧未分离，或逻辑帧计算不一致都可能导致不同步。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **植入物ID** | hash | 状态数据 | 当前帧数 | 房间ID | 命令ID |
| **1** | a | {“length”:2} | 7 | 5 | 8 |
| **2** | b | {“length”:4} | 9 | 5 | 9 |