Reducing lag in a distributed physics system through ahead-of-time simulation

An initial implementation

Abstract

Lag in Virtual Reality can be devastating to the enjoyment of participants. This paper reduces the lag in a physics system by implementing a subset of the architecture proposed by Loren Peitso and Don Brutzman[1]. This subset contains an event-based physics system together with a networking system. The collision system of the physics system has been heavily reduced in size, as to speed up development of the architecture.

This architecture seems promising and handles predictable cases with almost no noticeable lag. This architecture does not handle external events well, as the lag is noticeable. The 'healing' process for unexpected external events should be researched further to make this architecture viable.

Background

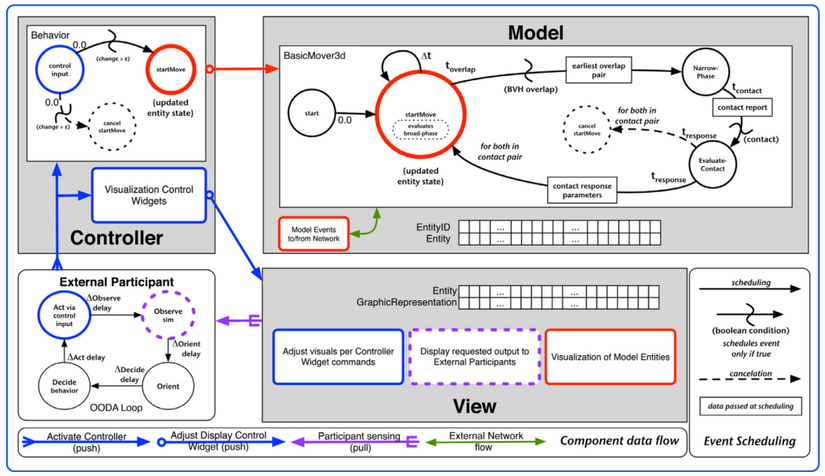
In multiplayer game development, client-side prediction is key to reduce the lag of the system. This is where the client predicts the state of the game, before the server returns the actual state of the game to the user.

Physics engines use numerical methods to calculate ordinary differential equations. More specifically this paper has used the Runga-Kutta of the fourth order to calculate the following functions:

$$\mathbf{a} = \mathbf{F}/m$$

$$\Delta\mathbf{v} = \mathbf{a} * \Delta t \rightarrow \mathbf{v}_n = \mathbf{v}_{n-1} + \mathbf{a} * \Delta t$$

$$\Delta\mathbf{s} = \mathbf{v} * \Delta t \rightarrow \mathbf{s}_n = \mathbf{s}_{n-1} + \mathbf{v} * \Delta t$$



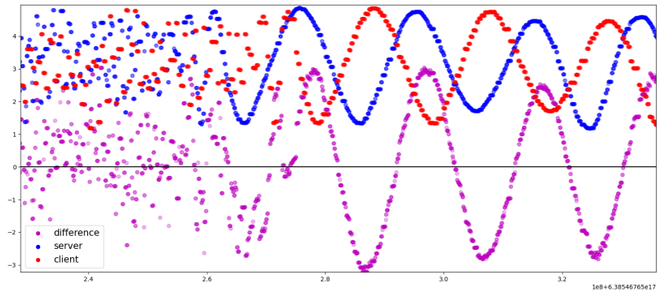
The architecture proposed by Loren Peitso and Don Brutzman[1]

Method

To test this architecture two tests were conducted. The main difference between the tests was the amount of latency. This was achieved by testing on either one computer or two separate computers. During both tests two spheres where dropped, and their height was measured. Afterwards the difference was calculated.

Results

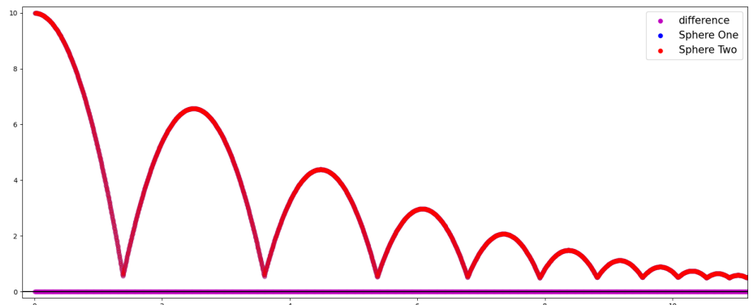
As can be seen in the results, when there is no lag both spheres drop with barely any error. But once latency is added though the separation of the computers, some small error is visible. This error is not very noticeable in the application. The last graph shows what happens with a lot of external events. The latency in that case is very noticeable for the user.



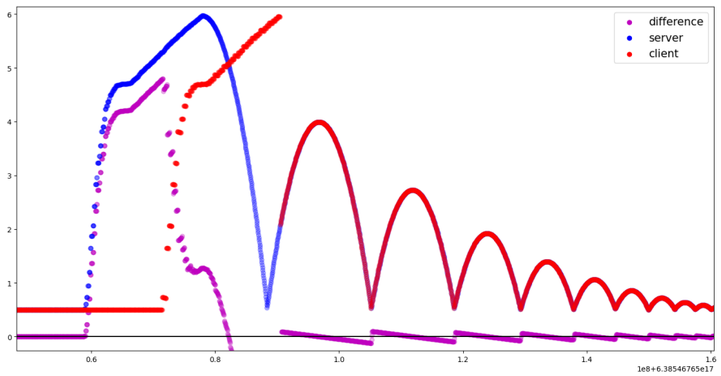
Graph 3: Second test, some latency. Only external events

Conclusion

The method proposed by Loren Peitso and Don Brutzman[1] makes latency unnoticeable unless external inputs are present. While this is useful for non-interactive applications, Virtual Reality applications or other games can expect external inputs often and would mostly rely on the 'healing' process to mask lag. The paper of Loren Peitso and DonBrutzman[1] does not fully explain how to do this. Even still, further research should be made to implement those features, since the architecture seems promising.



Graph 1: First test, almost no latency



Graph 2: Second test, some latancy. The first part is due to an external event, the last part is the physics system

References:

[1] Loren Peitso and Don Brutzman. Defeating lag in network-distributed physics simulations. Graphical Models, 111:101075, 2020.

TUDelft

Author: Timothy Zonnenberg

Responsible professor: Ricardo Marroquim

Supervisor: Amir Zaidi