Networked Physics Overview

Overview of networked physics in Unreal Engine.

Overview

Networking, or replication in games, refers to the ability to communicate gameplay information between multiple machines over an internet connection. Unreal Engine features a robust networking framework that helps developers streamline the creation of multiplayer games.

Networked physics is part of the networking framework and enables physics-driven simulations to work in a multiplayer environment. In Unreal Engine, physics replication refers to Actors with replicated movement that simulate physics. These simulations run inside the local client (player's machine) during gameplay.

The biggest challenge with traditional networked physics is interactions between simulating objects when they collide. Replicating such interactions becomes more difficult the faster an object travels and the higher the network latency (speed of the network connection). These challenges set a limit to what can be practically simulated in games.

For a more detailed explanation of networking in Unreal Engine, refer to the Networking and Multiplayer section of the documentation.

Default Replication Mode

The **Default** replication mode is Unreal Engine's legacy physics replication mode. This mode is active on Actors that **replicate their movement** and their **root component** is set to **simulate physics**.

This mode replicates the simulation by altering each object's velocity on the client to match the object's velocity on the server at that time. It takes half a Round Trip Time (RTT) for the client to receive data from a server-authoritative object. To replicate the object where it's currently located on the server we forward predict the received state-data by half the round trip time.

This form of replication takes two factors into account:

- The positional offset between the object's position in the client and the predicted location on the server.
- The velocity of the object on the server.

The positional offset for the object is then divided by a desired correction time. This produces a velocity (distance / time) which is added on top of the object's server velocity. This results in the correction of the object's position over the desired correction time.

The Default replication mode does not handle interactions gracefully, as any local alterations to the object’s velocity, position, or rotation is overwritten by the object’s velocity in the server. This results in the object’s position being corrected in the client, despite its local interactions.

Predictive Interpolation Mode

The **Predictive Interpolation** replication mode is designed for server-authoritative Actors. It works by altering each object’s velocity on the client to match the object’s velocity on the server at that time, similarly to the Default mode. However, this mode is designed to better handle interactions and local physics alterations on the client, as long as they are done predictively (expected to be applied the same on the server and the client).

This mode takes into account the object’s local (client) velocity in addition to the object’s server velocity when calculating the object’s final velocity. In addition, it handles corrections based on both, the round trip time (RTT) (latency) and the current send-rate time interval (time interval when the client receives data from the server).

The end result is a physics simulation that allows the client to apply a predicted force onto the object before the server sends back the results of that same force down to the client. The client can also interact with the object by pushing it, for example.

It's important to note that higher latency and / or higher velocity will degrade the quality of the physics replication. However, this can be balanced via the replication settings to better suit your project.

Resimulation Mode

The **Resimulation** replication mode is designed for server-authoritative Pawns and Actors.

Resimulation is fully forward predicted on the client. This means that the client is half a Round Trip Time (RTT) ahead of the server.

This mode is designed to make physics Pawns possible and to handle interactions with better accuracy in multiplayer by allowing full client prediction of physics. The client is half a Round Trip Time (RTT) ahead of the server and a full RTT ahead of confirmation data received from the server. This requires the physics simulation to cache a history of the physics state each physics tick for at least a RTT amount of time, which consumes CPU and memory.

When the client receives state information from the server, it compares them with the cached physics state in its history for the corresponding physics frame. If the state information differs enough, it triggers a physics resimulation.

A resimulation is done by rewinding the physics simulation back to the cached history state, correcting the errors based on incoming server states, and then simulating physics again up until the current physics tick.

At the end of a resimulation, an object might be at a different state than before the resimulation, which could cause snapping of the object's position. In this case, the mode interpolates the rendering of the object from its current state to the new state.

Resimulation can be used on Actors and Pawns while other Actors run Predictive Interpolation at the same time. The goal is to gracefully handle interactions between Actors running the two different replication modes. Actors replicated through Predictive Interpolation are more performant and less network intensive compared to Actors replicated through Resimulation.

Physics Pawns work by having the player inputs drive the movement of the Pawn through physics forces. These inputs are applied during resimulation by the **Network Physics** component.

The Network Physics component handles networking, history caching, resimulation of inputs, and custom states for Pawns and Actors. The Network Physics Component is a low-level system and needs to be implemented manually through C++.

Networked Physics Settings

You can modify several networked physics settings in the **Project Settings** under the **Physics** category.

You can also add the **Network Physics Settings** component to any Actor that is using networked physics replication modes to override its physics replication settings.

In addition, you can further configure networked physics settings via **console commands** by typing **np2.PredictiveInterpolation** in the console to see all the options for **predictive interpolation**, or **np2.Resim** and **p.Resim** to see all the options for **resimulation**.