NetworkedVR Milestones

Simplifications throughout:

- Peers do not acknowledge that messages have been received
- Do not do delta compression
    - Send peers your state update in full
    - Do not delta encode state updates based on the last acknowledged state
- Do not do any area of interest filtering, so secondary replicas are kept on every peer other than the one holding the primary copy (i.e. every peer has a reference to every object, even if it is not the primary owner of that object)

V0: Proof of concept [DONE]

Goal

Two instances of the application on the same machine can sync one object in one direction.

Simplifications

- Don't worry about time-stamping
- Send position updates even if the object did not move
- Statically assign GUIDs to the two in-game objects so that GUIDs don't have to be synced
- Secondary owner's interactions with the object are not synced back to the primary owner

Strategy

- New instance acts as client for fixed time period (say, 5 sec) and searches for server; if server not found, instance becomes server
    - Therefore, one peer can be started, then the second can be started >5 seconds later and they will connect properly
    - Whichever is the "server" is assumed to be the primary owner of the mutable game object
- Server sends position encoded as a Flatbuffer to client
- Client receives position every frame and sets the object's position by looking it up in it's LocalObjectStore (the GUID is still statically set)

V1: Two way syncing

Goal

Secondary owner's interactions with the object are synced back to the primary owner as actions.

Simplifications

- Don't worry about time-stamping
- Statically assign GUIDs to the two in-game objects so that GUIDs don't have to be synced

Unsimplified

- Primary should not send position updates if the object did not move

Strategy

- Set a flag in the local object store when the secondary owner player is interacting with the object
- Set position updates to the primary owner, who modifies its position of the object and rebroadcasts the positions to everyone but the client that sent the positions

V2: Send updates on a fixed interval [DONE]

Goal

Send updates every 11ms (90 fps). Try every 33ms (30fps) if 11ms is too fast.

Simplifications

- Statically assign GUIDs to the two in-game objects so that GUIDs don't have to be synced

Unsimplified

- Send updates on a fixed interval

Strategy

- Peers send their update across the network every 11ms

V3: Secondary copy interpolation [DONE]

Goal

Eliminate lag caused by network latency by keeping secondary copies one snapshot behind at all times.

Simplifications

- Statically assign GUIDs to the two in-game objects so that GUIDs don't have to be synced

Strategy

- Updates for secondary copies are buffered 11ms + network latency and interpolated between position updates

V4: Multiple objects with different primary owners

Goal

Both peers can spawn objects that are added to their own primary store.

Simplifications

- Do the simplest thing possible for spawning new objects: hook up a button that, when clicked, spawns a box at a fixed location
- Every peer receives a secondary copy (no area of interest)

Unsimplified

- Dynamically assign GUIDs as objects are created and broadcast the new object and GUID to other players

Strategy

- When a new object is spawned, broadcast the new object and its GUID to peers so they can register it as a secondary copy.

V5: Allow for many instances of the application

Goal

Many instances of the applicationcan sync many objects in many directions.

Simplifications

- Make every peer receive a secondary copy (no area of interest)

Unsimplified

- Dynamically assign new objects to the LocalObjectStore

Strategy

- Create a LocalObjectStore class which, every frame, broadcasts the positions of all its primary objects to clients which have registered that they maintain secondary copies
- On primary, register object in the LocalObjectStore class, which assigns the object a GUID

Important note: Once this stage is reached, we can test a simple “client-server” approach against our peer to peer approach by only spawning objects on one peer vs. spawning a few on each peer.