UnityVR (C#) and Agent (C++) Synchronization Protocol

Overview

The UnityVR could operate in two modes: synchronous and asynchronous. This can be changed in the configuration file with the *SyncMode* parameter (true = synchronous mode).

The purpose of the synchronization mode is that the agent and the VR use the same time scales. To ensure this, this simulation's time scale is independent from the calculation time of the agent and the VR. Such equal time scales are essential to simulate any scientific experiment.

The synchronization occurs in the following order. First, the agent receives its input and a *MsgStartSync* message indicating that the agent simulation could start. Then, the agent has enough calculation time to process input information as the VR is paused until the agent has processed all data and has sent all motor commands back. This is indicating by an *MsgStopSync* message sent by the agent to the VR.

The synchronization protocol is split over two frames of the VR (see figure).

- In the first frame, the VR sends the *MsgStartSync* message and all sensor data, like images and location, to the agent. The VR continues to the second frame. Now the agent should process the data and send motor actions back to the VR. The last messages, which the agent must send back, is *MsgStopSync*.

- The VR remains in the second frame and processes all motor messages from the agent until it receives the *MsgStopSync*,

The time scale is controlled by the value of *SimulationTimePerFrame* which can be changed in the configuration file. It determines the interval of sending images. In asynchronous mode, it is the interval in real time, hence the images are sent after this time period has passed. In synchronous mode, *SimulationTimePerFrame* determines the simulation time period. As the images are sent always in Frame 1, it is how much time is simulated over 2 frames in an experiment.
For the case that the *MsgStopSync* is lost, there is a timeout implemented in the second frame.

Design Ideas

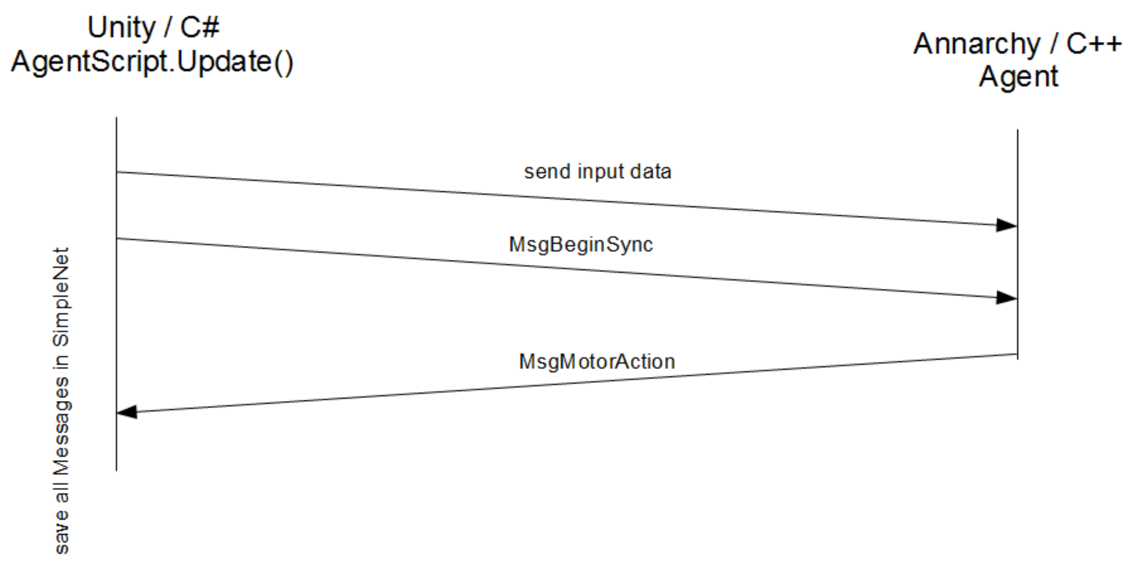
Why is it split up into two frames?

Unity processes everything in the VR sequentially. In the case of multiple agents, this implies that the sending and receiving of sensor and motor data is also sequential (sensor agent 1, motor agent 1, sensor agent 2, ...). This is due to the fact that each agent has its own *AgentScript* and all *AgentScript::Update()* functions are processed sequentially (these functions execute the sending/receiving).
By splitting the synchronization over two frames, the agents could be simulated parallel despite this sequential VR-functions. The first frame is used to send all sensor data to all agents in parallel and the second to receive all motor data to the VR. The end of each first frame serves as synchronization point. The time between the two frames should be used for simulations on the agent side.
Since all received commands are processed in the second frames, the end of such a serves as a synchronization point to update the internal status of the VR.

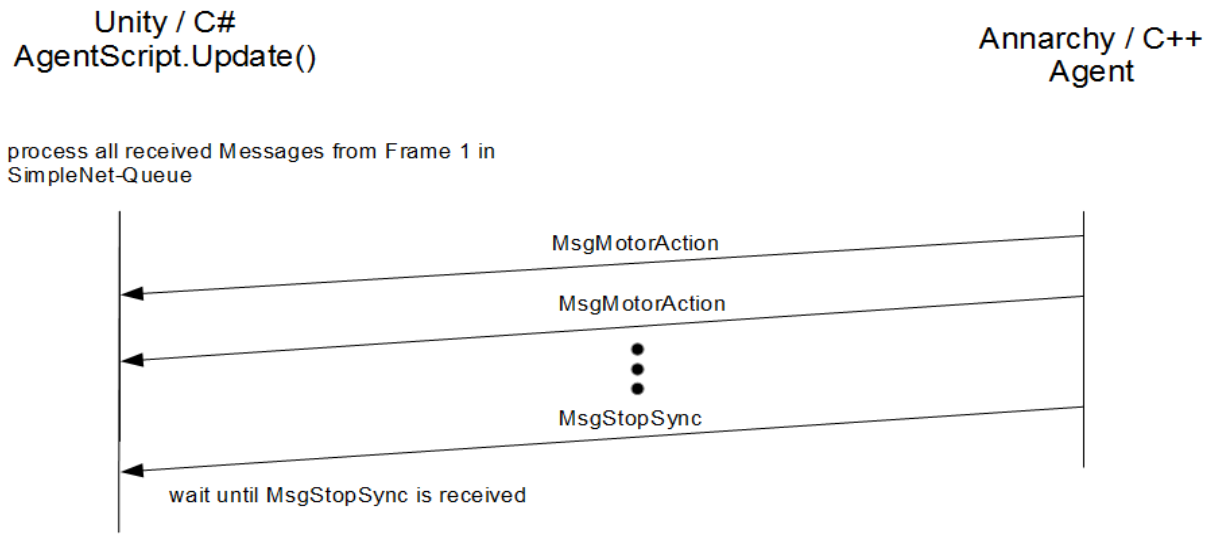
Pseudo code for Agent-C++

```
for(int time = 0; time < maxTime ; time++)
{
    //Wait until MsgStartSync is received
    while(!receiver->hasStartSyncReceivedAndResetState())
    {
        MSLEEP(1); // Sleep 1 millisecond
    }

    //Receive sensor data from tcp/ip network
    getImages();
    getOtherData();

    //do calculation
    calculateSomethingWithData();

    //send motor actions
    sender->sendSomeMsg();
    sender->sendSomeOtherMsg();

    //send stop signal back
    sender->sendStopSync();
}
```

Frame 1



Frame 2


Unity / C#
AgentScript.Update()
Annarchy / C++
Agent
process all received Messages from Frame 1 in SimpleNet-Queue
MsgMotorAction
MsgMotorAction
MsgStopSync
wait until MsgStopSync is received

VR internal world update