Multi User system for X3D scenes with or without simulation engines using XMPP, ECMA-script and C++ library.

Author: Sven-Erik Tiberg / Division of Engineering Psychology Univ. at Lulea of Technology Sweden. Jose Romero Miret sr. programmer / Avtech.aero.

Once up one time we needed a challenge, being former designer of motion simulators [ref 1] and interested in driver simulators, the idea came up to meet the challenge "car in a bag".

What could "car in a bag" mean, IMHO that's when you can travel with a simplified driver simulator in a cabin bag just using a portable PC, internet connection and a foldable steering wheel and pedals. Well the steering wheel and pedals could be in the checked in bag.

Sometimes we found that there is a lot of investments in driver simulators so why not try to use GPL based software as far as possible that are based on common standards.

One standard that appealed to me is VRML / X3D for the visualisation where it's possible to import scenery, houses, trees, pedestrian cars and even mousses and reindeers 3D models from other programs. Reindeers could seems exotic but they are in fact a real hazard in our roads. And when we are training drivers and observer their behaviour dynamic reindeers and mousses have to be a part of the scenery.

The discussion in this paper are about Multi User clients connected to a simulation server but in some applications there are no need for a simulation servers as when multiple users will interact with each other in for example discussions over a 3D representation of a house, mechanical unit or instructions on how maintain a engine. The system presented in this paper does of course apply to these MU systems to.

So why do we need internet, well either we put the simulation engine in the local computer or in a server, in both cases we need an exchange of info and states between the visualisation unit and the simulation engine and some Human Interaction Devices. It's possible to run everything locally but what if we would like to have more drivers on the virtual road or a trainer then we need a network connection between the actors. And if we want to have a common and central simulation engine with professional staff both in computers with applications and trainers then it would be easier using internet.

PS. why not apply the same approach for control room staff training or sea vessels or flight or...

As the setup of the 3D scene are fairly easy to set up and change why not use it for comparing different setup for dashboards / control panels.. If we would like to add more haptic functions in the user interface the H3DAPI-browser [ref 2] could be useful and if we like to add Augmented Reality the InstantReality - suit [ref 3] can be our choice.

Back to the challenge I'm addressing in this paper, network protocol.
In the beginning we set up this list of preferable criteria’s:

1. There should be open source clients for both 3D browsers and simulation engines.
2. The network clients should work on most common 3D browsers without recompiling the browser.
3. Transport should use Secure Layer both to prevent unauthorized intervention and bugging.
4. Nodes should use symbolic addresses so that the physical system could be reconfigured w/o altering the code in the node.
5. Authentication should preferable be done central.
6. If possible the transport should be wrapped into http to avoid firewall restrictions and restrictions in the 3D browser.

There are surely more preferable criteria like distributed protocol servers that uses a bidirectional tunnel trough intranet/internet firewalls but let us restrict us to those above.
Started with DIS-HLA [ref 4], Octaga [ref 5] and Bitmanagement [ref 6] but found that network protocol where mostly propertarian and therefore if was not easy to implement for simulation engines. Continued with Hawkey and twisted.python [ref 7] that would make it be possible to make a system that mostly meet the criteria but most of the system seems to be build from ground. About a year ago there was a discussion at the X3D-list about XMPP [ref 8] to be used for chat as text-nodes on X3D-sceen. Could XMPP be used for more than text, yes it seems so regarding to NPS thesis [ ref 9].

And by combining XMPP with jabber server most of the criteria above seems to be meet by using open source software that are well documented, supported and under continues development. Next step in the challenge, would it be possible to make the discussed system by using XMPP and jabber server, as stated above, we would like to say yes but being cautious I may say most probably as we at the moment do not have put together the tools available out there create a running system. But let us state that it's not impossible to present a running system by end of June.

Let us first point out about a, in our opinion, important constrain, the need for asynchronous handling of messages between network units as 3D browsers, HID (Human Interface Devices) and simulation engines. As all network units in a MU (Multi User/Unit) systems runs in real time with internal loop of calculation, presentation and event handling, they have it's own time step between loops. This time step can also vary depending on the extent of calculation, presenting and different routines used in event handling. By transmitting states in messages using UDP is a risk of loosing messages or the network stack overflow, those issues are surely addressed in other projects and reports and will not be covered in this paper.

Let's discuss the 3D client and in this paper X3D-browsers. All X3D-browsers are event driven either from using X3D defined nodes or by including script nodes in the X3D-sceen. Most X3D-browsers can interpret JScript/ECMA in a X3D-sceen, the downside with JScript are that it can only communicate with internet trough http connections.

The JavaScript library strophe.js [ref 10] are an event driven communication family of scripts made to handle XMPP messages. By using BOSH (“Bidirectional-streams Over Synchronous HTTP”) [ref 11] strophe.js can use http to communicate states from a X3D-browser script to another X3D-browser over a network communication server. The book "Professional XMPP programming with JavaScript and JQuery” [ref 12] covers well how to make a XMPP based network with BOSH using for example strophe.js.

If we include Strophe.js in a X3D-sceen and add custom code to interface with X3D ROUTE functions, we got a X3D-sceen that can send and receive states over the network using http protocol if we add translating functions between X3D field types and xml formatted string we got a sensor node that acts similar as the proposed Network Sensor Node. This will be discussed later in this paper.

On a typical simulation server the program used are mostly not fully adopted to real time mode so one of the task are to some how force the progress of calculation in RT by for example use a delay unit that synchronise the simulation clock with the real clock. The other issue are that the application must be built to run through all modules every loop of the calculation. There are mostly no limitation in including modules in applications that can communicate trough any internet port like http://code.stanziq.com/strophe/ [ref 10] has a C++ version called libstrophe for XMPP.

Now that we could create MU nodes to X3D browser with JScript and simulation engines with c++ code, how to connect them together. XMPP are defined to use a network server where each MU node connects with a unique address or JID (Jabber ID) like mu_node_name@jabberservername. mu_node_name are an account in the jabber server identified with the jabberservername.

At establishing of a XMPP/jabber connection from the client to the jabber server, the servers handle the IP-address of the client so if another XMPP/jabber client would send a message to the first client it only has sent the message to the server with a JID. If the JID are on another server, your server can be connected to the server hosting the receivers JID. This arrangement with JID makes it possible to address the right X3D-sceen wherever it's hosted no need to handle IP-addresses in the scripts. Secondly the MU nodes can be separated by firewalls with jabber servers on both sides of the firewalls and the MU node scripts can establish a connection to a X3D-sceen identified by a JID on the other side of the firewall.
The XMPP communication will be tunnelled through the firewall. We prefer to use ejabberd as it's a flexible and a fast jabber server that can handle both direct XMPP and streams over BOSH. Most XMPP communication takes place over Secure Socket Layer SSL or the newer SASL. A standalone ejabberd server takes about a day to set up from scratch and has a web interface for administration, you can setup a multi jabberd server connection through the administrative web interface.

Mentioned before about functions translating between X3D field types and xml formatted string. The X3D ROUTE node needs two parameters DEF (name of a X3D Node) and name of a type field used by the X3D node. The named type field are used to pass arguments to or from the X3D node. About the DEF parameter, it can point to a internal network connection script like XMPP connection or a group of XMPP connections within the same script.

A X3D scene can have multiple XMPP connection scripts each transferring one or more type field to one or more JID defined connectors. Meaning that a X3D-sceen can have more than one XMPP connection to one or more JID defined connector.

A XMPP network connection script can use xmlDIS to define the transferred content or a subset of xmlDIS or a single X3D field type defined xml string. It's also possible to use the location parameter in the JID address to define a field type name. Will publish examples on [http://wbsp.arb.ltu.se/Motionbase][ref 14] as they are created and tested.

A thing that made it possible to use JScript are the initialize() and shutdown() methods [ref 13]. Initialize() for establishing of connections with streams and initial values. shutdown() for a nice cleanup after a connection. It would be a good practice to create the XMPP connection script and it's interface to make it possible without a huge effort to rewrite the X3D-sceen for the coming Network Sensor Node.

The challenge to create a Multi User / Unit system are for us a thrilling challenge that probably will have many positive and negative surprises to tackle in the near future leading to more challenges.

Our present project under development a applications driver simulator with motion base 6 DOF platform:
REF list:


[2] H3DAPI is an open source haptics software development platform that uses the open standards OpenGL and X3D with haptics in one unified scene graph to handle both graphics and haptics. H3DAPI is cross platform and haptic device independent. It enables audio integration as well as stereography on supported displays.
http://h3dapi.org/

[3] The instantreality-framework is a high-performance Mixed-Reality (MR) system, which combines various components to provide a single and consistent interface for AR/VR developers. Those components have been developed at the Fraunhofer IGD and ZGDV in close cooperation with industrial partners:
http://www.instantreality.org/

The Use of DIS and HLA for Real-Time Virtual Simulation – A Discussion
Mr. Joseph Steel Royal Military College of Science, Cranfield University.
Royal Military College of Science, Cranfield University.


[6] Bitmanagement BS Collaborate designed for virtual reality applications from games up to cooperative work enables customer in combination with the basic visualization component BS Contact 7.2 (and higher) to realize real time communication between 3D objects like avatars in shared environments as well as collaborate working on 3D models with standard pc on the internet. http://www.bitmanagement.com/en/products/server-bs-collaborate


[10] Strophe is a family of libraries for writing XMPP clients. libstrophe is a C library with minimal dependencies. Strophe.js is a pure-JavaScript library. The implementations are production ready, well documented, easy to use, and easy to extend. http://code.stanziq.com/strophe/


[12] The book explains the XMPP protocol and its many uses as well as what types of problems it solves best. Nine applications are developed in increasing complexity, starting with an XMPP-based “Hello, World” example and ending with a real-time, multi-user gaming system for Tic-Tac-Toe. http://professionalxmpp.com/
