

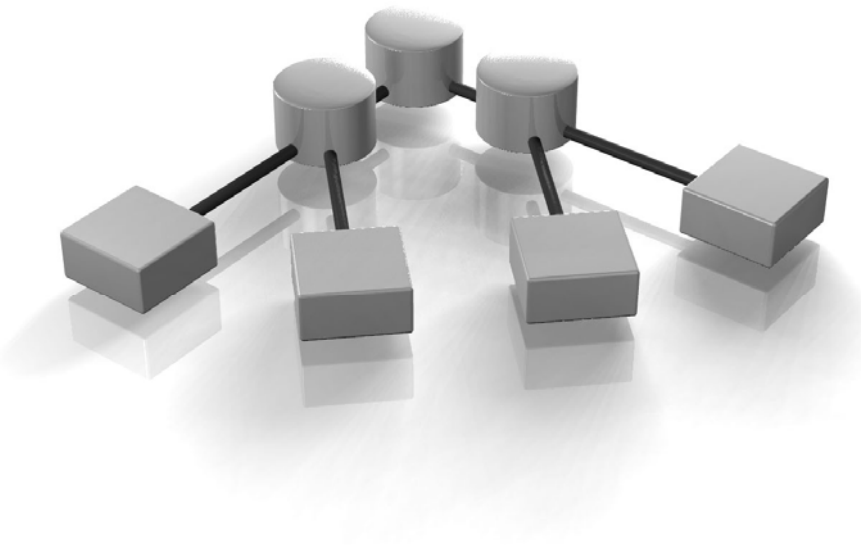
**SIXTH FRAMEWORK PROGRAMME
PRIORITY IST-2002-2.3.1.8
Networked Audiovisual Systems**



Uni-Verse Project

**WP 5.1 3D Modeler Plug-in – 3ds Max
March 15th 2006**

Distribution: Public



STREP project

Project acronym: Uni-Verse

Project full title: A Distributed Interactive Audio-Visual Virtual Reality System

Proposal/Contract no.: 002228

Table of Contents

Introduction.....	3
Part 1: How to use it	3
Overview	3
Connecting	3
Manipulating geometric objects	4
Handling other Verse object types	5
Light sources.....	5
Cameras	5
Sound sources	5
Part 2: How it's implemented.....	6
Overview	6
The Global Utility plug-in	6
The Utility plug-in	7
„VerseObject“ Geometry object plug-in.....	7
„VerseAudio“ Sound source Helper plug-in.....	7
How different objects are handled	7
Geometric Objects.....	7
Structure.....	7
Geometry updates	7
Materials	7
Bitmaps	7
Light sources	7
Cameras.....	8
Sound sources	8

Introduction

This document describes the Autodesk 3ds Max plug-in developed as Deliverable D 5.1, 3D Modeler plug-in, for the Uni-Verse project. Both how to use it and how it was implemented will be covered. The plug-in was written by Mattias Claesson of PDC, KTH, and the main reason for Uni-Verse to develop the plug-in is to lower the threshold for people working with 3d modelling to start using Verse. They are much more likely to try it out if they don't need to learn a completely new tool but can use the program they are used to working with. The goals for the capabilities of the plug-in is specified in the WP2.3 Uni-Verse Final Specification of System Capabilities, which defines three "levels" for the plug-in features: basic, good, and excellent. These levels will be referenced in this document when talking about the current status of the plug-in.

Part 1: How to use it

Overview

The design of the plug-in tries to be as close to what Max users are used to as possible. The main interface to the plug-in is a roll-up in the Utilities panel of Max. From there you connect to servers and get information about your connections and the servers. Apart from this roll-up, you shouldn't notice any major differences from how you would normally use Max, unless you want to. There are two modes of operation for the plug-in, automatic or manual. In automatic mode everything is done under the hood, invisible to the user. This is easier for the user but isn't as flexible as the manual mode. In the manual mode nothing is done without the user first saying so. The current implementation only has a working implementation of the automatic mode since it was deemed more important when getting people to try the system. If not otherwise noted, the rest of this document will describe the automatic mode.

Connecting

To get started you need to connect to a Verse server somewhere. You do this through the "Utilities" panel called "Verse". It looks something like this:

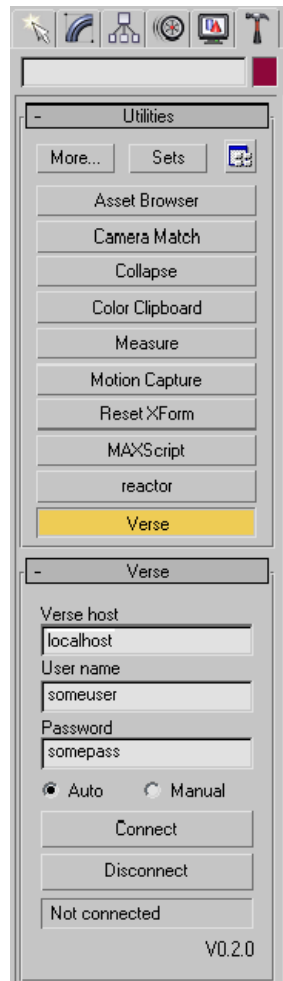


Illustration 1: Verse Utilities Rollup

In it you enter the server address or name to connect to and your username and password for that server. You also have the option to select what mode you want to use. This selection is currently disabled since the manual mode isn't working yet. When you have entered the necessary data you press the "Connect" button to try to contact the server. The status field at the bottom of the roll-up should tell you what is going on. When it has managed to log in to the server it should say "Connected". If you have chosen automatic mode and there are objects on the server, they should start to appear in the scene after this. You are now ready to collaborate with others through Verse.

Manipulating geometric objects

When you create a new object in Max, the plug-in will sense this and create a proper Verse object out of it. That is, an Object Node linked to a Geometry Node by a link called "geometry". It will also name the Object Node the same as the Max object and keep the names in sync. If someone else creates a proper Verse object on the server, the plug-in will insert a node of type "VerseObject" into the scene in Max and keep its geometry in sync with the geometry on the server. The "VerseObject" node type is one of the few visible additions by the plug-in. It represents geometric objects where new geometry has come from the Verse server. Regular objects in Max will be converted to VerseObjects when someone changes them on the server, including modifiers that had been added to the object, like this:

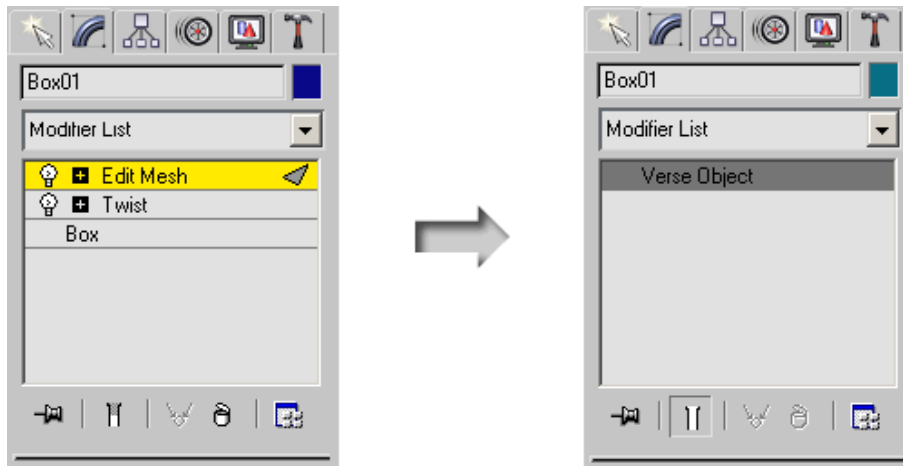


Illustration 2: Transformation to VerseObject

Also, if you add a material to an object in Max the plug-in will create and link a Material Node to the object in Verse, and vice versa. If the material in Max contains bitmaps as attribute maps, those bitmaps will be stored in Bitmap nodes in Verse.

Handling other Verse object types

The plug-in also has support for some types of non-geometric objects that Verse has, namely light sources, cameras and sound sources:

Light sources

When you create some kind of light source in Max the plug-in creates an Object Node in Verse and keeps the light intensity and position in sync between Verse and Max. The light source in Verse will always be a point source, no matter what kind of light source you create in Max, and the plug-in will always create light sources of type "Omni" in Max.

Cameras

If you create a camera in Max, the plug-in will create a camera object in Verse. The attributes that are supported are ortho mode, FOV, focus distance, near and far clipping distances only. All other attributes will be ignored and left at their default values, if available.

Sound sources

Verse sound sources are handled through a Helper object in Max called "VerseAudio". The only thing you can do with sound sources in Max, besides transforming it, is attach a URL to it describing where to find the actual audio for it. This can, for instance, be used to tell audio streaming clients which audio file to use for which sound source. This feature is used by the Uni-Verse Audio Simulation system, UVAS.



Illustration 3: VerseAudio

Part 2: How it's implemented

Overview

When "Verseifying" an application there are two distinct cases. You either have the source and can change the core of the application to better suit the Verse model, or you don't. In this case we don't and are stuck with the public API for writing plug-ins to Max. Since Max isn't designed as a collaborative tool, neither is its plug-in API. Fortunately Max exports quite a lot of its core functionality through the API and the fact that it is already a heavily multi threaded application probably helped a bit in the end. Mostly because you need to make sure that some part of your program gets to run periodically to keep the connection to the Verse server from timing out and to receive new data from the server. In our case this is handled by a so called Global Utility plug-in, a GUP, which runs in its own thread and a couple of other plug-ins. So the "Verse plug-in" really consist of several Max plug-ins working together. Currently there are four of them, the GUP, a Utility plug-in, a "Geometry Object" plug-in and a Helper plug-in. The GUP is the main plug-in, though. The other ones are basically there just to support it. They all run inside Max's own threads. The plug-ins talk to each other both by shared data and through the two event mechanisms in Max. The connection to Verse is handled by a C++ library called Ample which gives the programmer a higher level of abstraction of Verse then does the standard Verse API. The development of Ample was initiated by our needs in developing the plug-ins but written to be a general library useable for all kinds of Verse applications. It was developed mainly by Camilla Berglund at PDC, KTH.

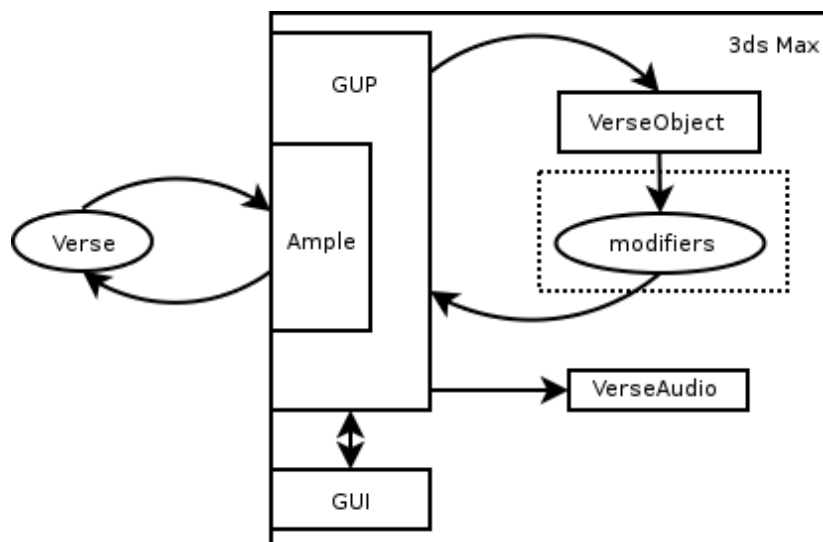


Illustration 4: System Overview

The Global Utility plug-in

There is only one Global Utility plug-in and it runs in its own thread. It handles most things that don't need to be in the object structure of Max. Among other things, the connection to the server and the synchronization of data

to and from Verse. It relies on Ample for the handling of connections and the sending and receiving of Verse data. It registers with Max to get notifications when things happen, like the user creating a new object in the scene or adding a modifier to one. When an object is added to the scene, the GUP also makes a reference to it so that it will get notifications from Max when something happens to that particular object, like the user adding a material to it or changes its geometry or name. Through these two mechanisms the plug-in can keep good track of what's happening inside Max and what the user is doing.

The Utility plug-in

This one handles the user interface found in the Utilities panel in Max. It's also a singleton plug-in like the GUP, but this one runs completely inside Max. It communicates with the GUP through regular method calls and shared data, to give it orders from the user and to get status information about the Verse connection.

“VerseObject“ Geometry object plug-in

This plug-in represent geometric objects that has received data from the Verse server. These can either be created when someone else creates an object on the Verse server or when someone changes an object created in Max. It inserts a copy of the geometry seen on the server into Max's geometry pipeline.

“VerseAudio“ Sound source Helper plug-in

A Helper object in Max is an object in the scene that doesn't represent anything when rendering. In this case it is used to represent a Verse sound source that the user can position in the scene.

How different objects are handled

Geometric Objects

Structure

The plug-in creates proper Verse geometric objects on the server and only imports those that it can understand. Currently this means an Object Node linked to a Geometry Node with a link called "geometry". It does not yet properly support hierarchies of Object Nodes or multiple Object Nodes linked to the same Geometry Node. One Material Node linked to the Object Node is also supported, but if there are more than one only the first is used.

Geometry updates

If the user creates an object in Max it will remain a normal Max object as long as possible. This means that the user can use all the features of Max's construction history for that object until someone else modifies that object on the Verse server. When this happens, the object's construction history will be collapsed to a single VerseObject with the geometry as seen on the server. This will not happen while the user is editing that object, though. During editing all changes to the edited object coming from the server is ignored. In the current implementation the user has to not change the object for a fixed amount of time before its construction history can be collapsed. When the object times out, the geometry on the server will contain both the changes made in Max and those by other users. This means that no changes will get lost and it's still up to the server to decide the ordering of the updates if there are conflicts.

The plug-in also throttles the changes sent to the server so that the network isn't flooded with updates. Currently this is implemented by waiting a minimum amount of time before sending another update of the same object. To minimize the network load the plug-in also only sends the vertices and polygons that has actually changed compared to its local copy of the geometry. This check is only done trivially and the way Max works, a modifier might create a topographically equal object but with polygons and vertices in a different order, in which case the plug-in will consider the whole object changed and send all the data to the server. Fortunately most modifiers in Max seem to behave nicely in this respect, i.e. they preserve the ordering if possible.

This behaviour corresponds to the "basic" level for the Geometry, Object and Material node types in Verse. Handling links between Verse nodes corresponds to level "good" for Object nodes.

Materials

Material Nodes will be created and linked to the Object Node with a link called "material" when the user adds a material to an object in Max. Material Nodes created on the server will currently map to a standard material with default values in Max. Thus Material Nodes are supported at the "basic" level, according to the specification.

Bitmaps

If the user has added a bitmap to some of the attribute maps of a material in Max, Bitmap nodes will be created in Verse for those and the bitmaps will be sent to the server. This fulfills the requirements for the "basic" level of Bitmap nodes.

Light sources

The simplest light source in Verse is just an Object Node with a non-zero "light intensity" attribute. This is fully supported in the current plug-in and generates an "Omni" light source in Max. The plug-in does not handle more

advanced types of Verse light sources, like one with geometry. It will simply behave as one without geometry. The current functionality fulfills the "good" level for Object nodes in Verse.

Cameras

A camera in Verse is an Object node with a Tag Group called "camera". Inside this Tag Group the camera stores attributes that describe it. The Tags used by the plug-in are `field_of_view`, `focus_distance`, `clip_near`, `clip_far`, and `orthogonal`. The type of camera created in Max is one called "Free". Since cameras weren't defined in Verse when the project started, this is implemented in addition to the specification, by request from users in the project.

Sound sources

Sound sources are represented as Object nodes linked to Audio nodes, with a link called "audio". Since Max notion of sound consists of playing a single audio file synchronized to an animation, we had to add a new plug-in to Max called "VerseAudio" to be able to handle multiple sound sources which has a position in the scene. It is considered a Helper type of plug-in by Max, which means that it doesn't affect the rendering of a scene. It is visualized in the editor as a loudspeaker. In addition to transforming the sound source, the user can also specify where to get the actual sound from. This is done by attaching a URL to the Audio node in a Tag called "url" located in a Tag Group called "uvas". This URL is used by the Uni-Verse Acoustic Simulation system, UVAS. Since Audio nodes were added to Verse after the project started, they weren't mentioned in the specification either. It was considered important for the audio part of the project to have some simple way of manipulating sound sources, so they were added to this plug-in.