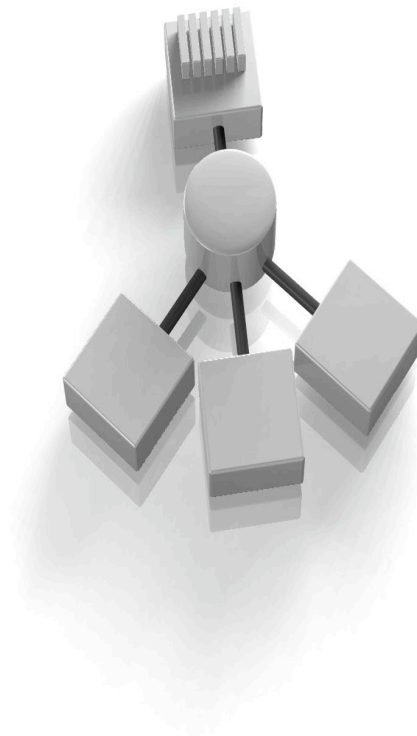


**SIXTH FRAMEWORK PROGRAMME
PRIORITY IST-2002-2.3.1.8
Networked Audiovisual Systems**



**Uni-Verse project
Deliverable D 5.1.2
Blender-Verse
March 2007**



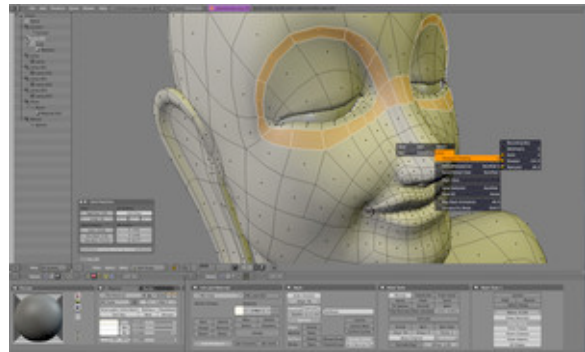
STREP project
Project acronym: Uni-Verse
Project full title: A Distributed Interactive Audio-Visual Virtual Reality System
Proposal/Contract no.: 002228
Distribution: Public

Content

1. Content	2
2. Summary	3
3. Specs for integration with Verse Protocol	3
4. Internal database upgrading	4
5. Real-time event handler system	5
6. UI tools to visualize access to (Verse)	5
7. Modeling tools	6
8. Character animation upgrade	6
9. HDR rendering and compositing	7
10. Releasing, validating and conclusion	7

2. Summary

The 3D creation suite Blender is the largest open source, cross platform, and freely available 3D tool on the web. Thanks to its open source nature, we can develop a fully integrated support in Blender to connect to all Uni-Verse components. This work is tightly coupled to redesign and upgrading work on Blender's internal database and modeling/animation and rendering systems, targeted on optimal support for the Uni-Verse system and making Blender a profitable and viable choice for 3D artists, engineers, architects, and game developers.



The Blender Foundation will partially coordinate this online, facilitating the community of developers, and partially internally with own development.

As special focus and development areas, the following activities have been executed:

- Define specifications for integration with Verse Protocol
- Internal database upgrading (undo system, library system)
- Real-time event handler system
- UI tools to visualize access to (Verse) data and online connections
- Modeling tools (subdivision surfaces, advanced selection tools, modifiers)
- Character animation upgrade (Bones, inverse kinematics, editing tools)
- High definition color rendering and compositing.
- Releasing, validating and testing, also with content oriented project

3. Specifications for integration with Verse Protocol

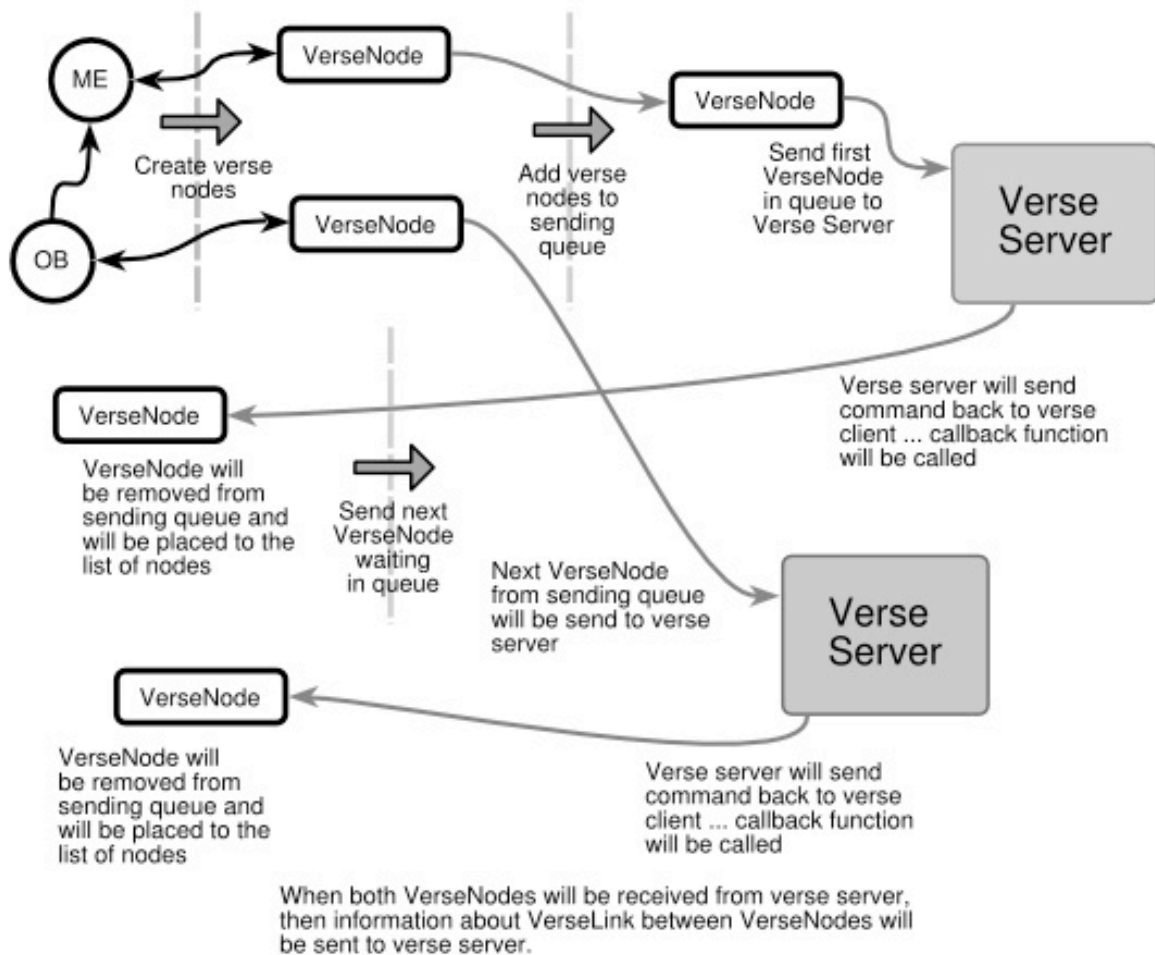
Main reference:

<http://wiki.blender.org/index.php/BlenderDev/VerseIntegrationToBlender>

In order to have the Verse protocol integrated in Blender several aspects had to be tackled inside of Blender's core functions and design.

Summary:

- A two-way dynamic list with array (index based) access has been developed to match Verse and Blender.
- Blender Object and Mesh data and API has been expanded on kernel and on UI side (both kept separated)
- Synchronization and event handling integrated inside of core Blender window manager.
- Python API access
- Image access



(Image: Graph demonstrating sending Mesh object to verse server)

4. Internal database upgrading

a) Global undo

Main reference:

<http://www.blender.org/development/release-logs/blender-235a/global-undo/>

With Blender's internal database being designed to save and read files incredible fast, tests were done to verify if this would give acceptable undo for the rest of Blender editing. Main problem is that this quickly becomes a scaling issue; with complex projects in Blender nowadays easily going to dozens of megabytes. Disk access also can become a bottleneck, where using memory is much faster.

The solution was found by streamlining and altering the core file saving routines.

When a Blender file is being saved, it already gathered fixed 'chunks' of minimum 50 kB, this to minimize disk and network overhead. These chunks now, for an 'undo save' can be copied to memory directly. That will result in a list of chunks that can be easily fed to the reading code for doing an 'undo'.

The chunks can also be quickly binary compared, to efficiently store incremental editing steps

Another crucial improvement was to do an undo-push *after* an editing command (instead of before, as in current mesh undo). This efficiently covers the small intervals between user actions, keeping it an interactive user experience.

And last but not least; this systems gives on each 'undo level' a full available set of chunks which can be immediately read. Each undo level can be easily removed or merged with the previous one.

b) Referencing, Library system

Main reference:

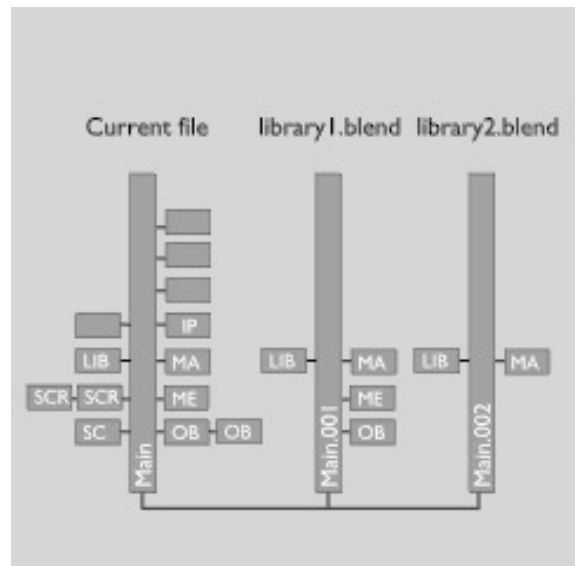
<http://www.blender.org/development/release-logs/blender-242/library-linking/>

General Blender architecture document about the database:

<http://www.blender.org/development/architecture/>

One of the most powerful Blender features is to be able to share data from Blender files with other files, to create libraries of objects or animations. This is a very important feature in a production environment, so it was important to become verified to work and be fixed in many places

This only is for static (non-real-time) updating and storage, and therefore not a replacement but an addition to using the verse protocol.



5. Real-time event handler system

Timeline reference:

<http://www.blender.org/development/release-logs/blender-237a/timeline/>

In order to have continuous and realtime - but non-blocking - updates in an Interface, a timer based handler system has been implemented. An internal clock can check every 'n' ticks for an event or state being changed, calling the appropriate updating functions, and then signalling parts of the UI to be redrawn.

Blender already had a good event caching mechanism, ensuring that event floods (like 100s of redraw commands) are gathered first, combined where possible, and then executed.

6. UI tools to visualize access to (Verse) data and online connections

Main reference:

<http://wiki.blender.org/index.php/BlenderDev/VerseIntegrationToBlenderUserDoc>

The Blender interface has been expanded in several areas with tools and visualization for establishing and controlling verse synchronization and connections.

- Main pulldown menu options
- Button panels with data visualization
- Outliner graph display with connections and nodes

Once configured, all settings save in Blender project files as usual. This also means the verse connection gets re-established on loading projects.



7. Modeling tools

Modifier reference:

<http://wiki.blender.org/index.php/BlenderDev/Modifiers>

Subdivision surfaces:

<http://wiki.blender.org/index.php/BlenderDev/CCGSubSurf>

http://wiki.blender.org/index.php/Manual/Subsurf_Modifier

Array modifier (example):

<http://wiki.blender.org/index.php/BlenderDev/ArrayModifier>

A crucial feature for modern 3d tools is modeling using subdivision surfaces, which is also the default 3d model definition in Verse. Blender now has a much improved and very fast CCGSubsurf, derived from the Catmul-Clark subdivision surface.

By implementing this as a generic modifier, more modifiers (= procedural modeling tools) became available in Blender. One of the most striking and unique examples was the Array modifier.



8. Character animation upgrade

Main reference

<http://wiki.blender.org/index.php/BlenderDev/AnimationUpdate>

Results:

http://www.blender.org/cms/How_Armatures_work.634.0.html

Highlights of this work included:

- Redesign of data structures and APIs, to enable linear-time non-recursive evaluation of hierarchies.
- Dependency graph solver to tie all animation systems together
- Transform handlers and drivers
- Display lists speedup
- Armature & Bone editing and UI visualization improvements
- IK and FK combination

9. HDR rendering and compositing

Technical references:

<http://wiki.blender.org/index.php/BlenderDev/RenderPipeline>

<http://www.blender.org/development/release-logs/blender-242/generic-node-system-for-blender/>

End user docs:

http://www.blender.org/cms/Render_pipeline.747.0.html

<http://www.blender.org/development/release-logs/blender-242/blender-composite-nodes/>

Without a doubt, compositing is currently one of the hot topics in 3D computer graphics creation, particularly because it enables efficient management and creative control of complex scenes and images. A typical 3D graphic can consist of many individual layers, each having a special filter or effect applied and combined into the final result. By pre-rendering such layers an artist can work much faster on fine-tuning the final result of an image.



In Blender, the Compositor is tightly integrated and aligned with the rendering pipeline. For this reason it is part of the Blender Scene, meaning there's only one "Composite" possible per Scene (but each file can have unlimited Scenes).

Compositing can also be used 'stand-alone; with only images read from disk as input, allowing you to render the Composite without having a 3D rendering invoked.

By default the Blender render system and the Compositor works in HDR, with 4 byte per component floating point accuracy.

10. Releasing, validating and testing and conclusion

In the course of this project, the following releases have been done:

Dec 2005, Blender 2.40 (Animation system., modelling)

January 2006, Blender 2.41 (Bug fixes, Game engine updates)

July 2006, Blender 2.42 (HDR Render, first release with Verse)

February 2007, Blender 2.43 (Tested and final release of all the D 5.2.2 features)

Each Blender release got at least 300,000 downloads from the blender.org servers, not counting all mirrors world wide, nor distributions by third parties and magazines.

Testing and bug reporting happens online. Over 1500 reports come in each year.

Most important was the Elephants Dream 3d animation short, which not only gave a lot of publicity, but also credibility among professionals to integrate Blender in their workflow.