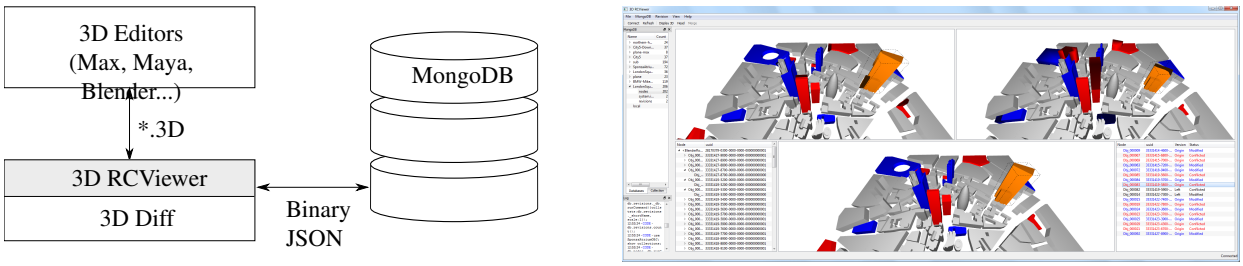
March 21, 2012

EUROGRAPHICS 2012
Poster

Revision Control Framework for 3D Assets

Jozef Doboš and Anthony Steed

Department of Computer Science, University College London, United Kingdom



(a) System Overview

(b) 3D RCViewer

Figure 1: In (a), 3D files from various modelling packages are uploaded into our viewer which stores scenes in a NoSQL database. (b) 3-way 3D diff supports selective merging from two different revisions (top) when combining into the common origin (bottom). Conflicts are highlighted in red, modifications in blue and current selection in orange.

Abstract

We propose a non-linear concurrent revision control for centralised management of 3D assets and a novel approach to mesh differencing. Large models are decomposed into individual scene graph (SG) nodes through an asset import library and become versioned as collections of polymorphic documents in a NoSQL database (DB). Well-known operations such as 2- and 3-way diff and merging are supported via a custom DB front-end. By not relying on the knowledge of user edits, we make sure our system works with a range of editing software. We demonstrate the feasibility of our proposal on concurrent 3D editing and conflict resolution.

Categories and Subject Descriptors (according to ACM CCS): I.3.4 [Computer Graphics]: Graphics Utilities—Graphics editors

1 Introduction

Storing 3D assets as files on a filesystem poses several drawbacks. Most of the time, the user loads a model into an editor, performs changes and saves it as a file again. When developing large models, it has to be decided how to spread the scene across various files and how to support collaborative editing. Either the scene is a single file limited by machine memory or the meshes are stored independently. The latter, however, introduces additional problems especially if edits involve distinct objects.

VisTrails Provenance Explorer for Maya [Vis12] is a commercial plug-in with a side-by-side differencing of selected 3D revisions. Despite capturing the edit history in its MySQL database, the 3D models have to be stored locally. Similarly, Denning et al. [DKP11] log edits in Blender. By label clustering via layers of regex, they create an interactive playback of modelling history. This, however, deals with single-user mesh edits only and tracks a subset of modelling techniques. Likewise, for images, Chen et al. [CWC11] propose an integrated revision control by logging actions in a graphical editor. Roupé and Johansson [RJ10], on the other hand, tackle a city wide 3D modelling via a hierarchical file structure stored in a Subversion repository.

We exploit the recent developments in database technology. Our non-linear revision control is built using a NoSQL database (MongoDB, `http://mongodb.org`) that stores SG nodes as well as their hierarchy. In addition, our DB front-end offers a conflict resolution interface that facilitates 2- and 3-way diff for meshes that enables the user to quickly select individual changes from any two revisions. We believe that, initially, the framework should not rely on any specific modeling tool but rather deal with 3D files external to the editor. We do not restrict the types of 3D models tracked and therefore a unified representation of scenes is stored.

2 3D Database

To our advantage, large models tend to encompass several meshes with their related semantic meaning. We exploit this natural partitioning and treat each part of an asset as a binary "blob" with its associated relationships. Zeleznik et al. [ZHC*00] use scene graph as a data format to inter-mediate between different applications. Following this example we convert files into SG components using the Open Asset Import Library (Assimp) [GSKN12] which can import and export various common 3D formats. These are subsequently encoded as Binary JSON (BSON) objects for storage and revision tracking in MongoDB.

To be non-restrictive, a scene graph is best described as a directed acyclic graph (DAG) where objects are stored in local coordinates with their associated global transformations. Non-linear history that allows for branching and merging is effectively a DAG, too. Hence, these DAGs reside in their respective collections (tables) inside DB. To identify SG nodes, we assign each a universally unique identifier (UUID). Even though NoSQL is schemaless, the `UUID` together with the `revision number` can be best described, in relational DB terms, as a composite primary key on the SG collection. Each SG node has to preserve this *revision metadata* during the round-trip from the modelling software to the DB and back. To retrieve a revision from the DB, the latest versions of each node are queried. On deletion, a node is replaced by BSON `NULL` document in the next revision and all of its children are recursively checked. If no other parents exist, these are also deleted. Implemented in C++, our revision control interface (3D RCViewer) facilitates aforementioned model handling and versioning, see Figure 1.

3 3D Diff

In order to detect conflicts during *commit* and *merging*, we perform an early reject byte-by-byte memory comparison on BSON objects that share the same `UUID`. If, for example, the counts of vertices differ, such meshes are already flagged as modified. Without a dedicated diff tool, conflicts would have to be exported into modelling software to be fixed and uploaded back to our 3D RCViewer. However, such packages do not support conflict resolution in similar 3D models, usually presenting the user with superimposed meshes and a list of reoccurring SG node names.

Head	Local	Result
○	○	○
⊕	⊗	conflict

Origin	Head	Local	Result
○	○	○	○
○	⊕	⊕	⊕
○	⊕	○	⊕
○	○	⊕	⊕
○	⊕	⊗	conflict

Table 1: Schematic representation of a 2-way (left) vs. a 3-way (right) diff with suggested merge results. Each SG node can be modified in `head` or `local/branch` revisions.

Our novel *3D diff tool* allows the user to selectively choose one or the other revision for each conflicting SG node. As shown in Table 1, discrepancies ($\oplus, \otimes$) in any two nodes (2-way) cannot be resolved automatically. However, adding extra information about a common `origin` based on the same `UUID`s (3-way) can further aid automated conflict resolution. Implemented within the 3D RCViewer, we present the user with either 2 or 3 model views and color-code conflicts, modifications and node selections. By using a change list (Figure 1b), the user can quickly decide on which modifications to preserve.

4 Discussion

We believe that our unification of versioning and storage can provide significant benefits over the common practice today. The smallest revision unit is a BSON document so if a single vertex has been changed, the whole mesh would need to be resaved. Nevertheless, we consider this a delta revision as it is only a subset of a larger scene. Also, it is possible that modelling software might not preserve our revision metadata. In such a case, nodes based on similar size, number of vertices etc. can be compared. We plan to make this work open source in the near future although it remains an open research question as to how to improve the user interaction in 3D diff. Possible avenues include automated camera navigation for better context understanding, bounding box conflict detection and vertex-level merging. Access to the DB as well as an independent 3D diff could be implemented via plug-in frameworks of many 3D modelling packages. What is more, MongoDB offers built-in geospatial indexing which would support spatial queries for large urban models.

References

[CWC11] CHEN H.-T., WEI L.-Y., CHANG C.-F.: Nonlinear revision control for images. In *ACM SIGGRAPH 2011 papers* (2011), pp. 105:1–105:10.

[DKP11] DENNING J. D., KERR W. B., PELLACINI F.: Meshflow: interactive visualization of mesh construction sequences. *ACM Trans. Graph. 30* (Aug. 2011), 66:1–66:8.

[GSKN12] GESSLER A., SCHULZE T., KULLING K., NADLINGER D.: Open asset import library (assimp), January 2012. http://assimp.sourceforge.net.

[RJ10] ROUPÉ M., JOHANSSON M.: Supporting 3d city modelling, collaboration and maintenance through an open-source revision control system. In *CAADRIA New Frontiers* (2010), pp. 347–356.

[Vis12] VISTRAILS, INC.: Vistrails for maya, January 2012. http://www.vistrails.com/maya.html.

[ZHC*00] ZELEZNIK B., HOLDEN L., CAPPS M., ABRAMS H., MILLER T.: Scene-graph-as-bus: Collaboration between heterogeneous stand-alone 3-d graphical applications. In *Eurographics* (2000).