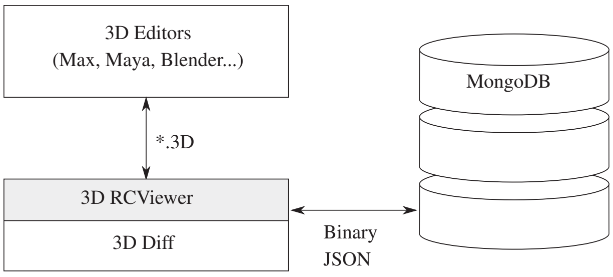
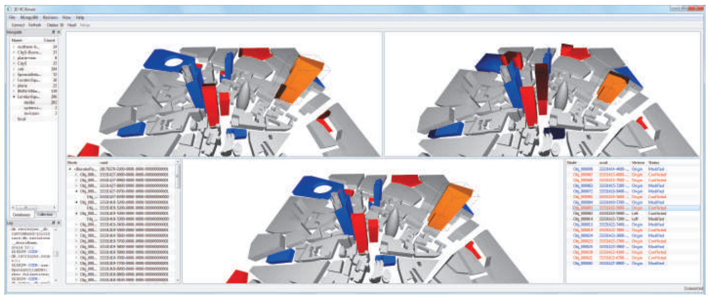
Revision Control Framework for 3D Assets

Jozef Doboš and Anthony Steed
j.dobos@cs.ucl.ac.uk



(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.

Introduction

Previously for images, Chen et al. [CWC11] proposed an integrated revision control by logging actions in a graphical editor. We believe that, initially, our framework should not rely on any specific modeling tool but rather deal with 3D files external to the editor.

- Non-linear revision control system is built using a NoSQL database (MongoDB).
- Scene graph nodes as well as their hierarchy are stored.
- Our DB front-end offers conflict resolution that facilitates 2- and 3-way diff for meshes, see Figure 1b.

3D Database

Zeleznik et al. [ZHC*00] used scene graph as a data format to inter-mediate between different applications. Similarly, we convert 3D files into scene graph components using the Open Asset Import Library (Assimp). These are subsequently encoded as Binary JSON (BSON) objects for storage and revision tracking in MongoDB, see Figure 1a.

- Scene graph is described as a directed acyclic graph (DAG).
- Non-linear history that allows for branching and merging is a DAG, too.
- To identify scene graph nodes, we assign each a universally unique identifier (`UUID`) and a `revision number`.

3D Diff

As shown in Table 1, discrepancies (⊕, ⊗) in any two nodes (2-way) cannot be resolved automatically. Our novel *3D diff tool* allows the user to selectively choose one or the other revision for each conflicting scene graph node.

- We perform an early reject byte-by-byte memory comparison on BSON objects that share the same `UUID`.
- Adding extra information about a common `origin` based on the same `UUID`s (3-way) can further aid automated conflict resolution, see Table 1.

Discussion

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.

- The smallest revision unit is a BSON document.
- If a single vertex is changed, the whole mesh would need to be resaved.
- Modelling software might not preserve `UUID` and `revision numbers`.

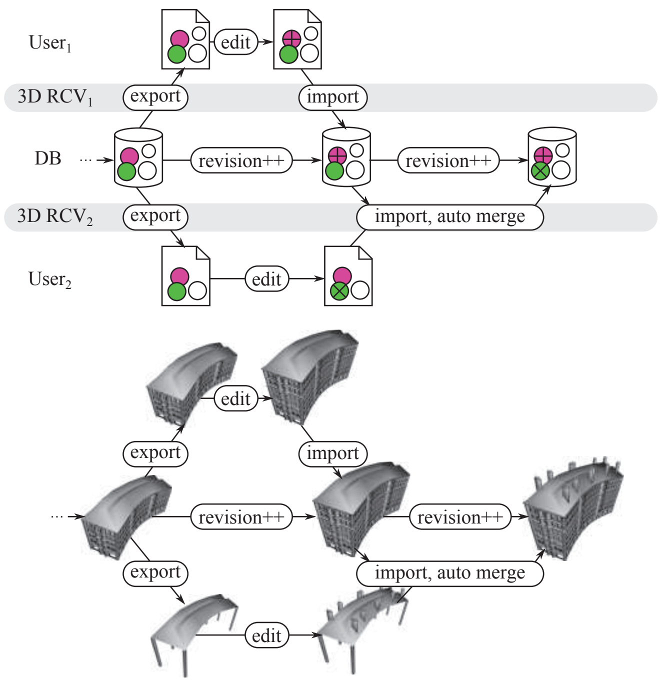


Figure 2: Diagram of non-conflicting edits resolved by automatic merge. Because modifications to the `roof` (green) are independent of its position (purple) within the scene, User$_2$ can commit using the implicit auto merge functionality.

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 scene graph node can be modified in `head` or `local/branch` revisions.

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.

[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).