EG 2017 - Short Papers, pp. 29-32. The definitive version is available at http://diglib.eg.org/ (DOI: 10.2312/egsh.20171007).

AnimDiff: Comparing 3D Animations for Revision Control

George Madges[1] and Idris Miles[1] and Eike Falk Anderson[1]

[1]The National Centre for Computer Animation, Bournemouth University, UK

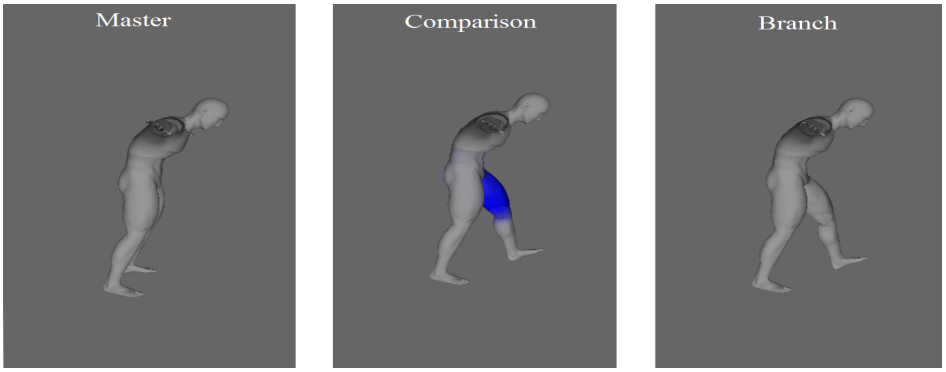

Figure 1: *Example of a comparison made by our animation differencing system.*

Abstract

The process of animating a complex 3D character can be a time consuming activity which may take several iterations and several artists working in collaboration, each iteration improving some elements of the animation but potentially introducing artifacts in others. At present there exists no formal process to collate these various revisions in a manner that allows for close examination of their differences, which would help speed up the creation of 3D animations. To address this we present a method for equivalence checking and displaying differences between differing versions of an animated 3D model. Implemented in a tool that allows selective blending of animations, this provides a first step towards a 3D animation revision control system.

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

1. Introduction

In a feature animation production environment it is likely that several artists will be working on different aspects of the same 3D asset to speed up production, but this collaboration – although intended – can be a hindrance if not managed properly. Similar problems exist in software development where they are addressed by so-called revision or version control systems (VCS's). These enable collaboration, allow inspection of code history and provide tools for merging code contributed by different developers, streamlining the software development process. We address this by introducing VCS techniques into the 3D computer animation domain, demonstrated through an artists' tool that enables the visualisation of the differences between multiple design iterations of the same animated 3D model. This is achieved through a novel method for equivalence checking and differencing of 3D computer animation data ('diff' algorithm).

This paper is organised as follows: In section 2 we discuss related work in the field of VCS's for computer graphics. Our approach and prototype system are presented in section 3. This is followed with a discussion of results, limitations and potential extensions of our method in section 4. In section 5 we provide our conclusions and highlight outstanding issues that we intend to address in the future.

2. Version Control Systems for Computer Graphics

Over the past decade there have been several attempts to provide VCS techniques for computer graphics to aid the creation of 3D computer generated models in the creative industries. Currently available systems, e.g as presented by Doboš and Steed [DS12a, DS12b], allow designers of 3D computer generated models to successfully compare scene and static (non-animated) 3D models with one another to highlight differences and modifications.

EG 2017 - Short Papers, pp. 29-32. The definitive version is available at http://diglib.eg.org/ (DOI: 10.2312/egsh.20171007).

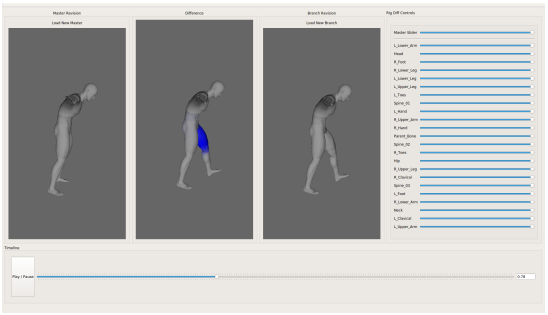

Figure 2: *Our system in action: The left view is the original animation and the right is a new iteration (modification) of the animation. The middle view highlights the differences between these and allows blending between left and right. Sliders enable the user to fade each joint from the original to the new animation allowing for inspection on a joint by joint basis.*

Denning and Pellacini [DP13] discuss a range of applicable matching algorithms for calculating the edit distance between 3D model versions. Other systems have been inspired by the collaborative text editing tool Google Docs (`https://docs.google.com`) to allow real-time collaborative mesh editing [SSTP15]. Different from these systems, Silva et al. [dSJCM15] present a massively parallel approach to VCS's for computer graphics assets, primarily focussing on binary files, which allows fast and efficient revision control but does not provide the user with semantic details about the differences between revisions. While these systems are rapidly maturing, they still lack the capability to effectively compare 3D animations (animation data) and animated 3D models.

2.1. Visualizing Differences

Visual difference data is only usable when presented to the user in a useful manner [Zam15], i.e. relevant to its purpose. While this has not yet been addressed for animation data revision control, there exist a number of solutions in the context of 3D models.

Denning and Pellacini [DP13] discuss colour coding operations for use with their MeshGit algorithm to enable users to quickly and easily comprehend changes to a 3D model through visual inspection. In their implementation differing parts of a mesh are coloured red to indicate deletion, green to indicate insertion and blue to indicate modification of the geometry (e.g. moved vertex positions).

Similarly, the repository hosting service Github provides an experimental feature to compare 3D model revisions that takes both versions of a model that, by using binary space partitioning, computes added, removed, and unchanged parts [Git13]. This system also follows the convention of colour coding the model to indicate the operations performed and uses similar colour coding as the MeshGit [DP13] implementation, i.e. red for deletion and green for addition, however, Github's tool does not consider modification of the geometry and displays unchanged parts of a model as wireframe instead of shaded. Additionally the Github tool includes a technique for visualising differences in the form of a revision slider that tweaks the opacities of the original and new models superimposed on top of each other, acting as a time-line of revisions with old model versions fading into new ones.

Somewhat related to this is Wei's method for comparing the differences between 2D images [Wei10], which compares the history of operations performed on the image through DAG's (Directed Acyclic Graphs), allowing the history of an image to be recognized with a single glance.

The most important difference between static 3D models and animated meshes is the inclusion of time as a variable in the animation data, as difference data for animations can change over time and is by definition non-linear. These non-linear differences between keyframes provide important information for the animation designer and require the use of methods that show these changes over time. Balakrishnan et al. [BDG15] attempt to solve this for video clips by using overlays. During playback of the video, outlines (detected edges) of shapes of an edited video clip are overlaid onto the original video with the colour of the outlines indicating the edit distance to the original.

2.2. Comparing Animation

Existing systems for comparing animation (motion) data are mostly concerned with motion capture data, where application areas are motion classification and analysis [Val16]. In the context of animation synthesis and identification from motion capture data, Kulbacki and Bak "partition sets of primitive motions into appropriate groups according to similarity between motions" [KB02] using Dynamic Time Warping (DTW) [MÖ7] for computing differences between animations, which they then discretize into a value that could be used as a similarity metric in a VCS. There do not appear to be existing solutions for presenting animation difference data visually to an animator to help them better understand changes and thus improve their work.

3. A Differencing Method for Animation Revision Control

We propose introducing version control techniques for 3D computer animation systems, specifically focusing on the 'diff' (differencing) aspect found at the core of VCS's and applying this to animation data. Our system highlights the differences between differing animations (Figure 2) in a manner that allows users (animation designers) to inspect changes, e.g. to identify problems with the animation.

3.1. Comparing Animated 3D Meshes

Our system loads two differing animations of a 3D mesh, which are then pre-processed (see section 3.1.1) to prepare the animation data for differencing (see section 3.1.2). Differences between the animations are stored in a difference data structure that is passed to a viewer application for visualisation of the 3D mesh animation and difference data. Our system currently has the following requirements:

- The 3D model animation must be skeletal based, i.e. employ a hierarchical control-rig of joints connected by bones for animating a skinned mesh.

EG 2017 - Short Papers, pp. 29-32. The definitive version is available at http://diglib.eg.org/ (DOI: 10.2312/egsh.20171007).

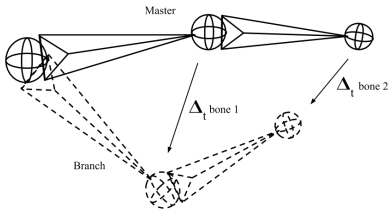

Figure 3: *Each joint in the rig structure contains the transformations for each key-frame. We calculate the difference values for position, scale and rotation for the branch (modified animation) from the master (original) animation's respective values.*

- The skeletal rig must be consistent across differing animations, i.e. the rigs used in these animations must have an identical joint hierarchy. Differences in the animation rigs themselves (different numbers of joints or different connections in the rig hierarchy) are not currently supported.
- Interpolation between key-frames is assumed to be linear.

3.1.1. Pre-processing

To import animated 3D models we use the Open Asset Import Library (ASSIMP) [ASS16], which stores the 3D model's control rig in a tree structure [ASS12], separate from the animation data. In a first pre-processing step we convert data from ASSIMP's internal format into our own data structure that embeds the animation into the rig by storing key-framed transformations for each joint, to simplify the differencing operations.

3.1.2. Computing animation differences

We analyze the data from two different animations and synchronize their key-frames (Alg. 1) by iterating through each joint of the rig and subsequently iterating through each key-frame of both animations (*MstrFrms* and *BrnchFrms*), comparing their time-stamps (intervals between key-frames). When key-frames are missing in one animation but are present in the other, corresponding key-frames (*NewFrm*) are created by linear interpolation of the nearest neighbouring key-frames (*Frm*) to either side of the missing key-frame.

These key-frames are then compared (Alg. 2) to produce difference values between master and branch position, rotation and scaling information (Figure 3).

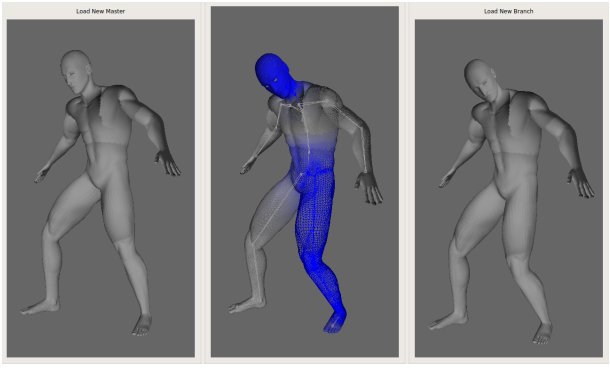

Figure 4: *Left is the original animation. Right is the new animation. In the middle we display the differring version in wireframe to show that the colour coded rig matches the mesh.*

Algorithm 1 Computing Differences

```
1:  procedure COMPUTING DIFFERENCES
2:      for each Joint j do
3:          m ← 0
4:          b ← 0
5:          while m < MstrFrms || b < BrnchFrms do
6:              if m == MstrFrms then
7:                  DIFF(Frm_Mjm, Frm_Bjb)
8:                  b ← b + 1
9:              else if b == BrnchFrms then
10:                 DIFF(Frm_Mjm, Frm_Bjb)
11:                 m ← m + 1
12:             else
13:                 if Frm_Mjm.Time < Frm_Bjb.Time then
14:                     Δt ← Time_Mjm − Time_Bjb−1
15:                     NewFrm ← Lerp(Frm_Bjb, Frm_Bjb−1, Δt)
16:                     DIFF(Frm_Mjm, newFrm)
17:                     m ← m + 1
18:                 else if Frm_Bjb.Time < Frm_Mjm.Time then
19:                     Δt ← Time_Bjb − Time_Mjm−1
20:                     NewFrm = Lerp(Frm_Mjm, Frm_Mjm−1, Δt)
21:                     DIFF(NewFrm, Frm_Bjb)
22:                     b ← b + 1
23:                 else
24:                     DIFF(Frm_Mjm, Frm_Bjb)
25:                     m ← m + 1
26:                     b ← b + 1
```

Algorithm 2 Diff

```
1:  procedure DIFF
2:      ΔPosition ← Position_B − Position_M
3:      ΔScale ← Scale_B − Scale_M
4:      ΔRotation ← Rotation_B Rotation_M^−1
```

3.2. Visualising Animation Differences

In our system we play back animations in a similar fashion to most computer animation systems, i.e. by using linear blend skinning to deform the mesh according to the skeletal rig and blend weights. In order to visualise the 'diff' we highlight the differing rig bone by changing its colour to blue and also changing the colour of the mesh affected by the differing bone to blue (Figure 4), determining which mesh vertices are affected, and thus shaded, by using the corresponding mesh vertex blend weights. Due to the rig being consistent across animations the differences in animation data can be considered as modifications. This is in contrast to VCS's such as Git, that typically consider differences as deletions and insertions as they often cannot track changes as such.

EG 2017 - Short Papers, pp. 29-32. The definitive version is available at http://diglib.eg.org/ (DOI: 10.2312/egsh.20171007).

4. Discussion

The system described here – a tool to aid artists and animators – only handles linear history edits, i.e. branching is not supported, as it only implements a '2-way diff'. A production environment would benefit from the ability to create different branches and provide a non-linear edit history – where several modifications branch off from an original (parent) version. An extension of the system to handle branching and merging of revisions would allow several animators to work in parallel. This 3-way differencing and merging – where several modifications are compared – is a common operation in text based VCS's, and the incorporation of this into our system would greatly enhance its application potential for animation production environments.

Storage of 'diffs' instead of complete edit histories, as found in some VCS's to optimize disk usage, would not work particularly well with our 'diff' algorithm as there would be no cost reduction – 'diffs' could include additional interpolated key-frames, so the storage requirements for these might even exceed the storage requirements of a complete history.

Our differencing technique also has potential uses in other domains. It could be used for animation education, e.g. in a massive open online course (MOOC) where students could be tasked with replicating a reference animation, for which our approach, with some minor modifications, could be used for automated assessment, determining the similarity of a student's submission to the reference animation. Related to this our method could be used in a tool for plagiarism detection, determining if a submitted animation is identical to other submissions or animations from a repository.

5. Conclusions and Future Work

We have presented a method for equivalence checking and the display of differences between differing versions of an animated 3D model. This introduction of version control techniques into the field of 3D computer animation not only provides a first step towards a 3D animation revision control system, but may also be applicable in other related domains.

Adapting our method to handle 3-way differencing – assuming the existence of a common parent or ancestor [Men02] and two differing branches – would be fairly straightforward. It could be achieved by comparing each branch with the parent and possibly an additional comparison of the results, where differences could indicate a merge conflict. At this point the user could then be prompted to choose which modification of the joint to use or interpolate between the two, creating a new version.

Additional future work includes a user study with potential users, especially artists in an animation production environment, to evaluate the effectiveness of the user interface and to explore the system's potential for integration in pipeline tools for animation asset creation.

Other than implementing a '3-way diff', we hope to extend our algorithm (Alg. 2) to also compare differing mesh vertex weights and changes in the rig structure. Eventually, a full version control solution for 3D animation would not only need to identify differing animation but also changes in the mesh, requiring the inclusion of existing methods for static mesh revision control (as discussed in section 2), the creation of a repository for storage of revisions and front end integration into off-the-shelf animation production software.

6. Acknowledgements

We would like to thank our colleague and mentor Valery Adzhiev who provided valuable advice. His support, encouragement and suggestions have made this project possible.

References

[ASS12] ASSIMP TEAM: Open asset import library documentation, July 2012. URL: http://www.assimp.org/lib_html/. 3

[ASS16] ASSIMP TEAM: Open asset import library website, 2016. URL: http://www.assimp.org/. 3

[BDG15] BALAKRISHNAN G., DURAND F., GUTTAG J.: Video Diff: Highlighting differences between similar actions in videos. *ACM Trans. Graph. 34*, 6 (2015), 194:1–194:10. 2

[DP13] DENNING J. D., PELLACINI F.: MeshGit: Diffing and merging meshes for polygonal modeling. *ACM Trans. Graph. 32*, 4 (July 2013), 35:1–35:10. 2

[DS12a] DOBOŠ J., STEED A.: 3D Diff: An interactive approach to mesh differencing and conflict resolution. In *SIGGRAPH Asia 2012 Technical Briefs* (2012), pp. 20:1–20:4. 1

[DS12b] DOBOŠ J., STEED A.: 3D revision control framework. In *Proceedings of the 17th International Conference on 3D Web Technology* (2012), Web3D '12, pp. 121–129. 1

[dSJCM15] DA SILVA JUNIOR J. R., CLUA E., MURTA L.: Efficient image-aware version control systems using gpu. *Software: Practice and Experience 46* (2015), 1011–1033. 2

[Git13] GITHUB: 3D file diffs, 2013. [Online; accessed 29-December-2016]. URL: https://github.com/blog/1633-3d-file-diffs. 2

[KB02] KULBACKI M., BAK A.: Unsupervised learning motion models using dynamic time warping. In *Intelligent Information Systems 2002* (2002), pp. 217–226. 2

[MÖ07] MÜLLER M.: Dynamic time warping. In *Information Retrieval for Music and Motion*. Springer, 2007, pp. 69–84. 2

[Men02] MENS T.: A state-of-the-art survey on software merging. *IEEE Transaction on Software Engineering 28* (2002), 449–462. 4

[SSTP15] SALVATI G., SANTONI C., TIBALDO V., PELLACINI F.: MeshHisto: collaborative modeling by sharing and retargeting editing histories. vol. 34, pp. 205:1–205:10. 2

[Val16] VALČÍK J.: *Similarity Models for Human Motion Data*. PhD thesis, Masaryk University, 2016. 2

[Wei10] WEI L.-Y.: *Nonlinear Revision Control for Images*. Tech. Rep. MSR-TR-2010-105, Microsoft Research, 2010. 2

[Zam15] ZAMAN L.: *User Interfaces and Difference Visualizations for Alternatives*. PhD thesis, York University, Canada, 2015. 2