Towards 3D Model Interoperability in Games

Jia Sing Chen and Andrew Hogue
Faculty of Business and Information Technology
University of Ontario Institute of Technology
2000 Simcoe St, N
Oshawa, ON, Canada, L1H 7K4
jia.chen@mycampus.uoit.ca , andrew.hogue@uoit.ca

ABSTRACT

This paper presents an automatic method for enabling the interoperability of 3D models within different types of games. Character customization is an increasingly popular ability in games. Games for the Wii may take advantage of the built-in avatar customization (e.g. Mii avatars) yet few games to date have utilized this ability. Game designers are hesitant to allow players to incorporate creative freedom within the customization of their avatars for fear of invalidating the intended aesthetic experience. In this paper, we focus on developing a method to blend multiple 3D meshes together in order to maximize in-game usability of the user customized characters without sacrificing aesthetics of the original artistic style.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric algorithms, languages, and systems*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*

Keywords

3D model, mesh blending, interoperability, distance field, graph embedding

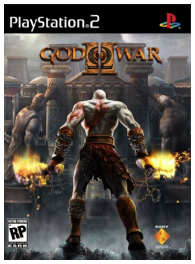
1. INTRODUCTION

Interoperability and passing data between games is becoming a reality. With the introduction of the Xbox Live Arcade (XBLA) on the Xbox360 a mechanism has been put in place to allow for the passing of data between XBLA titles and larger titles. For example, it was announced at the Game Developers Conference 2008 that Fable 2 would enable you to play "pub-games" that are available in the XBLA to increase your character's wealth in Fable 2. The data from the XBLA pub-game is transferred to your online presence accessible by Fable 2 when connected to the internet enabling the interoperability. This has numerous implications for upcoming titles. The type of data able to be transferred currently seems to be limited to numerical data (i.e. virtual currency), however one could imagine the possibility of incorporating a more general framework for the transfer of different types of data to be utilized in-game.

Character customization is an increasingly popular ability in games. Nintendo allows you to customize your own Mii avatar and utilize them in specific games. Currently only a few titles utilize the Mii avatar for in-game utilization. Similarly on other platforms, Sony's PlayStation Home allows you to create your own avatar and numerous games have added this ability for the character you control. Electronic Arts' Spore allows for extreme avatar customizability. The user is allowed to generate avatars using a vast set of tools and limbs increasing the amount of control the user has over the creature. Textures and animation are applied procedurally and tailored to meet the user-defined avatar. Online environments such as Second Life have become popular due to the built-in character customizability. Meez[1] is intended to be a chat/game website focused around avatar customizability. Most console games do not have a tremendous amount of flexibility and the user is quite limited to the amount of customization they can achieve. Notably, if the player painstakingly creates a character of interest for one game, the data cannot be transferred to another game.

Interoperability of avatar meshes has yet to be accomplished. Just imagine if you could play as Nintendo's Super Mario in Sony's God of War. What would that look like?

(a) Nintendo's Super Mario[2]

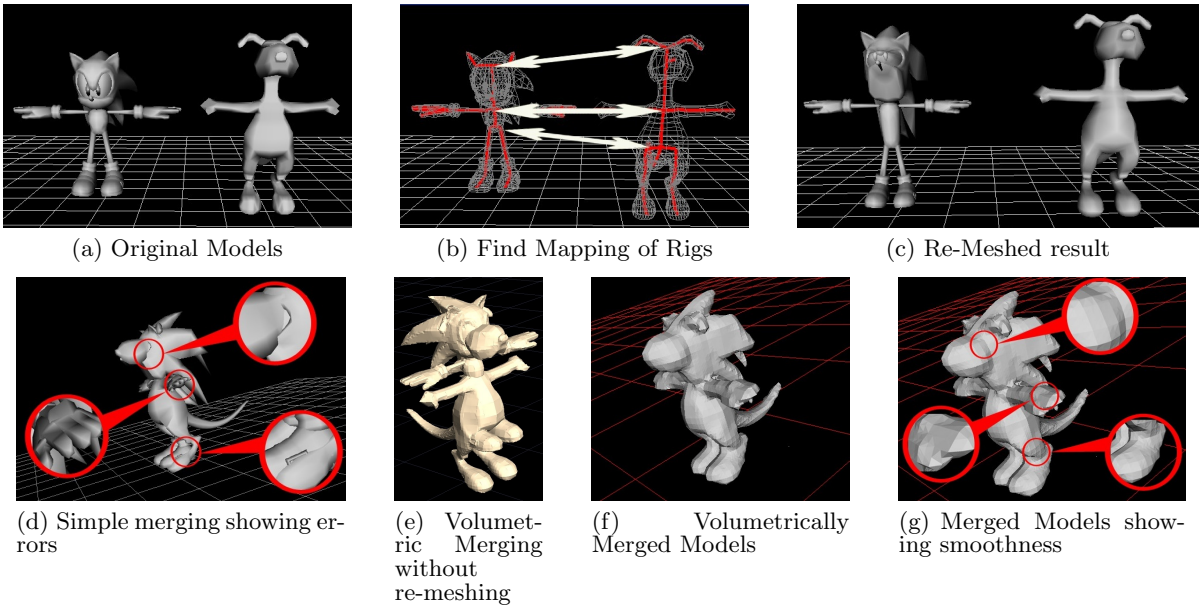


(b) Sony Entertainment's God of War 2[3]

Figure 1: Imagine what it would be like to play as Nintendo's Mario (a) in Sony Computer Entertainments God of War series (b).

[1]Meez.com

FuturePlay 2008 Toronto, Ontario, Canada. November 3-5, 2008



(a) Original Models (b) Find Mapping of Rigs (c) Re-Meshed result

(d) Simple merging showing errors (e) Volumetric Merging without re-meshing (f) Volumetrically Merged Models (g) Merged Models showing smoothness

Figure 2: Overview of the Algorithm. (a) First the source and destination models are loaded. (b) a mapping is found between the skeletons for alignment. (c) the source mesh is deformed to align the skeletons. (e-g) a volumetric approach is used to merge the meshes resulting in a smooth mesh ready for applying animation.

Few games allow for any sort of interoperability between them for good reasons. Copyright and, more importantly, security is of concern when allowing for data to be transferred to your game console or PC and a discussion on this is beyond the scope of this paper. Each game world has its own well-defined and unique artistic style. Incorporating mesh data not created by the game's artists and designers would compromise the intended look and feel of the experience. As in the example above, Nintendo's Mario has a primary color palette and has a cheerful, cartoon style while God of War's Kratos is quite the opposite, i.e. mean, muscular and monotone, which suits the game world and style of gameplay of their respective games. By directly incorporating the Mario avatar into the world of God of War, the character would look extremely out of place breaking the emotive intention of the game designers.

Other issues plague interoperability of characters between games such as texture, animation, size/scale, attributes and accessories make this a very complex problem to solve independently. This paper addresses the possibility of maintaining the intended artistic style of the game while automatically incorporating customized avatars between games.

2. MODEL REPRESENTATION

There are many ways to represent a 3D model. Perhaps the most flexible is to utilize an implicit function[2] or scalar field representation such as a distance field or a level-set[5]. This approach allows for flexibility in that it is not constrained to a particular mesh structure. For example, applying dynamics to a distance field or level set is possible and a new mesh can be extracted at each iteration[6]. Sculpting operations can be performed directly on the distance field[8].

[2]Super Mario is the property of Nintendo®

[3]God of War 2 is the property of Sony Computer Entertainment

In this work, we utilize the adaptively sampled distance field as our internal representation to merge the two meshes into a single mesh suitable for real-time animation.

3. METHOD

An overview of our method is illustrated in Figure 2. First we load in the two models we wish to incorporate and blend together (Figure 2(a)). We also load the skeletal rig if it exists. If no skeletal rig exists, we employ the autorigging approach from [1] to apply a generic skeletal rig to the mesh and compute the appropriate weights for each vertex. At this point we have two meshes with two distinct (and different) skeletal rigs. They can be animated individually but combining the meshes is problematic as their scales are much different. As can be seen in Figure 2(a), the Sonic the Hedgehog mesh is shorter and skinnier than the kangaroo mesh. We opted to deform the source mesh and bones into the same scale as the destination mesh. Currently we do this manually, however we propose using a graph embedding approach to find an embedding of the source skeleton in the second skeleton using the method from [1]. This provides us with a mapping of bones from the source to destination skeleton. Once the mapping is accomplished, we stretch the bones to be the same length as the mapped bones in the destination skeleton and re-mesh our model to apply the deformation (see Figure 2(c)).

Now that the meshes are in the same scale, we can merge them together. The simplest way to accomplish this task is to simply copy all of the triangles and vertices from one mesh into the other mesh. This creates problems where the meshes overlap as can be seen in Figure 2(d). This figure shows the sharp edges where the triangles overlap creating a mesh that cannot be animated aesthetically. We opt to perform a volumetric merge operation to ensure that the output mesh is aesthetically pleasing and able to be smoothly ani-

mated.

The volumetric meshing approach computes an adaptive distance field (ADF) using an Octree as described in [3] for both the source and destination meshes individually. The marching cubes algorithm [7] is used to compute the set of triangles for the distance field. Marching cubes samples a regular voxel grid and computes the distance to the surface (using both of the computed distance fields) defined as the minimum of the two distance samples. The results can be seen in Figure 2(f) showing a low resolution (64 subdivisions) merging of the two meshes. As can be seen in Figure 2(g), the resulting merged mesh does not suffer from the same breaks in surface as it did with the simple combination of the triangle sets. Since this new merged mesh corresponds to a known skeletal rig, the animations already defined by the game artists and animators will also properly animate this new mesh.

With this approach there are a few caveats. The resulting mesh detail depends upon the resolution of the voxel grid used in the marching cubes algorithm and also upon the maximum depth of the octree used for the distance field computation. In these examples, they were set to low values of 64 subdivisions each to illustrate the algorithm, however in practice the resolution should be increased dramatically to maintain overall mesh quality. For example in Figure 2(e), the voxel grid used was set at 512 subdivisions and much more detail such as Sonic's eye's can be seen more clearly in the resulting mesh (note that in this example the two meshes were not aligned showing the need for determining the appropriate mapping between skeletal rigs). A better approach would be to utilize the surface nets approach to mesh the ADF directly as in [4].

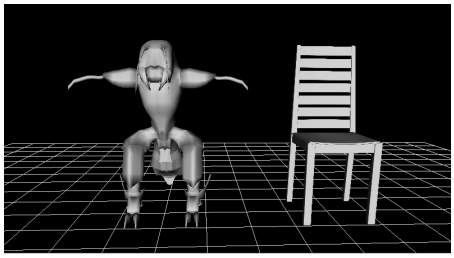
One advantage to this approach is the inherent level-of-detail encoded in the model representation. By utilizing suitable meshing techniques higher or lower resolution meshes may be extracted to conserve memory cpu intensity. The method can also be used to merge non-humanoid models as long as the models can be aligned in some fashion (See Figure 3 for an example). This is due to the usage of a non-mesh based model representation internally to perform the volumetric merging.

4. CONCLUSIONS

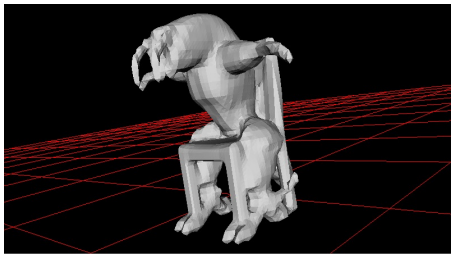
The method developed here is capable of merging two meshes while maintaining the ability to utilize pre-built animations for the destination model. This can be applied directly to characters from different games. In the current implementation, we focused on merging the mesh and skeletal data. Future work includes properly defining the animation blending factors to incorporate both the source and destination model animations; developing a more adaptive meshing approach to incorporate more detail into the resulting meshes; and developing proper texturing and shader support to bring the source mesh into the aesthetic realm of the destination mesh, i.e. to ensure that the Mario model has the "look and feel" of Kratos.

5. ACKNOWLEDGMENTS

We thank TurboSquid.com for use of the Sonic the Hedgehog, Kangaroo and Chair models.



(a) BugMan + Chair Models



(b) Merged Result

Figure 3: Results of algorithm on humanoid and non-humanoid models.

6. REFERENCES

[1] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26(3):72, 2007.

[2] J. Bloomenthal. Polygonization of implicit surfaces. *Comput. Aided Geom. Des.*, 5(4):341–355, 1988.

[3] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 249–254, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[4] S. Gibson. Constrained Elastic SurfaceNets: Generating Smooth Models from Binary Segmented Data. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 888–898, October 1998.

[5] J. Gomes and O. D. Faugeras. Reconciling distance functions and level sets. In *SCALE-SPACE '99: Proceedings of the Second International Conference on Scale-Space Theories in Computer Vision*, pages 70–81, London, UK, 1999. Springer-Verlag.

[6] B. Houston, M. B. Nielsen, C. Batty, O. Nilsson, and K. Museth. Gigantic deformable surfaces. In *Proceedings of the SIGGRAPH 2005 Conference on Sketches & Applications*. ACM Press, 2005.

[7] W. Lorenson. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[8] R. N. Perry and S. F. Frisken. Kizamu: a system for sculpting digital characters. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 47–56, New York, NY, USA, 2001. ACM.