Reusing Motions and Models in Animations

Akanksha, Z. Huang, B. Prabhakaran, and C. R. Ruiz, Jr.

School of Computing
National University of Singapore
{akanksha, huangzy, prabha, conradod}@comp.nus.edu.sg

Abstract. Computer animated 3D models have been increasingly used in multimedia presentations because they can be viewed and manipulated directly in 3D. In this paper, we propose a database approach for this problem. Our main objective is to help authors create multimedia presentations by reusing existing animations. In our approach, animations are stored in databases. We define a set of metadata that describe the animations. Authors can search on these databases by issuing queries. A set of motion editing operations are defined and used to manipulate the query results to create new ones according to the requirements of the presentations. We have developed an animation toolkit implementing the approach and the results are promising.

Keywords: Multimedia presentations, animation, database, reuse, metadata

1 Introduction

Computer animated 3D models have been used increasingly in multimedia presentations. One main reason for this is that the computer animated 3D models can be viewed and manipulated directly in 3D.

However, users who require animations in the presentations may find it difficult to produce the desired motion sequences. Creating animations of good quality needs considerable effort of skilled animators, actors, and engineers. Thus, reusing the existing animations to create new ones becomes an appealing problem.

The major idea of reusing animations is to adjust the existing motion sequences for the new requirements while at the same time, retain the flavor of the original motion. One method was proposed by Bruderlin and Williams [3], who treated motions as signals and applied techniques of signal processing to adapt them. A variant of their method, motion-displacement mapping, was introduced by Popovic and Witkin [10]. Other works include Gleicher [4], Hodgins [5], and Monzani et al.[9]. All these methods address the reuse for a specific type of animation and manipulate on low level kinematic or dynamic structures of geometric models.

Using database in computer animation is not a new concept. Thalmann et al. [12] presented an *Informed Environment* that creates a database dedicated

to urban life simulation. Using a set of manipulation tools, the database permits integration of the urban knowledge in order to simulate more realistic behaviors. Ayadin et al. [1] used databases to guide the grasp movement of virtual actors. Their approach creates a database based on divisions of the reachable space of a virtual actor. Kakizaki [7] proposed a multimedia presentation framework using animation databases. It uses a scene graph and an animated agent in multimedia presentations. The motion of the agent is categorized into three classes: pointing, moving, and gesturing. The reuse of animation has not been addressed in any of these three database methods.

Our Approach: We propose a database approach for using animations in multimedia presentations. The major contributions include the following technical aspects:

- Augmented scene graphs (with the incorporation of Interpolator Nodes) as data structure for representing animations of 3D models.
- A set of metadata for describing the animations.
- Similarity measures for computing the relevance factor of an animation to resolve queries on the databases.
- A set of operations applied on the query results that help in reusing animations.
- An animation toolkit implementing the approach.

A specific area, where animations can be applied in a multimedia presentation, is a document produced for the hearing impaired. The toolkit can be used to generate animations of sign language. The sign language can be translated from the audio clips.

We will focus on the database aspects of the approach without the details of specific animation techniques such as inverse kinematics [6, 13]. We will present them separately. The remaining part of the paper is organized as follows:

Section 2 describes the storage of animations as databases. Section 3 discusses the operations the user can invoke to retrieve, insert, and adjust geometric models and motion, to create an animation. Section 4 shows, by means of examples, the different ways in which the query results can be manipulated. Section 5 briefly describes the implementation. Section 6 provides a conclusion of the work done and discusses possibilities for future work.

2 The Animation Databases

Now, we describe how we organize animations as databases. The major idea is to use the *augmented scene graphs*. A standard scene graph is a hierarchical structure to describe geometric models. We augment it in order to represent motion sequences of the geometric models by introducing a new node for the motion. We specify a set of metadata to describe motion sequences and geometric models that are associated with the augmented scene graph model. The metadata is then organized together with animations as a relational database. We provide a set of operations that can be used to query the animation databases. Finally, the query results can be adjusted in temporal and spatial domains according to the new specifications of the query metadata.

2.1 The Scene Graph

The scene graph was originally designed for real-time 3D graphics system Performer [11]. The Performer scene graph is a tree structure. The nodes in the tree have different types. Typically, leaves in the scene graph represent geometric models and interior nodes represent transformation objects. It also contains data about the geometric models in a scene, such as their shape, size, color and position. Each piece of information is stored as a node in the scene graph.

An animation object consists of two fundamental elements: geometric models and their motion sequences. A geometric model is a mathematical representation of 3D scenes in the world. It can be as complex as a human body model comprising sophisticated shapes with hundreds of degrees of freedom (DOFs). The motion sequences represent the kinematic information of the geometric models. A typical representation is keyframe.

To represent a motion sequence of a specific geometric model we incorporate the motion node: *Interpolator*. An *Interpolator* node is associated with a geometric model of multiple DOFs. In an Interpolator node, keyframes are defined for all DOFs: {key $[k_1, k_2, ..., k_i, ..., k_m]$, keyvalue $[v_1, v_2, ..., v_i, ..., v_m]$}, where k_i is the *key frame number* or *key frame time*. v_i is a vector in the motion configuration space: $v_i = [u_{i,1}, u_{i,2}, ..., u_{i,j}, ..., u_{i,n}]$, where $u_{i,j}$ is a *key value* of DOF j (in displacement for translational DOFs and angle value for rotational DOFs) at k_i, and n is the number of DOFs of the model. The key and keyvalue together form one motion sequence for a model.

In our implementation, the scene graph is represented in VRML format. VRML is a text-based language used to model virtual environments [14]. One VRML file can be decomposed into many smaller VRML Text Descriptions which represent individual geometric models and motion sequences. These *Text Descriptions* are then stored in the VRMLText table of the database. The animation objects are linked to the VRMLText table by a Has-A relationship. In the case of geometric models the table stores the prototypes and nodes which are used, where as, for motion sequences it contains the Interpolator nodes and Event Routing.

One simple example of a scene graph representing an animated blue cube is shown in Figure 1 whose VRML description is given below.

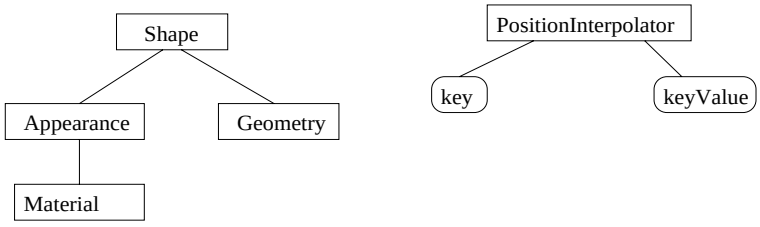

Fig. 1. An example of the scene graph of an animated blue cube. Three key frames are defined in the PositionInterpolator node.

```
Shape{
   geometry Box { size 1 2 3 }
   appearance Appearance { material Material { diffuseColor 0 0 1 } }
}
PositionInterpolator {
   key [ 0.0 0.5 1.0 ]
   keyValue [ 2.0 0.5 2.0, 2.0 0.5 3.0, 2.0 0.5 3.0] }
```

2.2 Animation Metadata and Database Organization

Metadata is widely used in the query of multimedia databases [2]. We have defined content-descriptive metadata for geometric models and their motion sequences. Their storage and querying are dealt with in a similar manner. The queries are performed based on the metadata and result in animation object(s) best matching the user requirements. We consider the following types of animation metadata.

- Metadata describing the geometric models: Type, Size, Position, Color, and Orientation.
- Metadata describing motion information: Motion Type, Motion Speed, Start Time, End Time, and Style.

For database organization, we use a relational database management system, *MS Access*, to store the animation databases. Scene graphs are stored and indexed. The metadata are stored in the fields of the database. (In the current implementation, the metadata for each animation object are manually added to the scene graph). By doing it in this manner, we can use SQL-like queries to retrieve animation objects. The fields of the database for the geometric models and the motion sequences are shown in Table 1 and Table 2, respectively. The geometric models and motion sequences of each animation object are stored in two separate tables of the database and are linked by the Object ID being the primary key.

Table 1. Database fields for a geometric model.

Field Name	Description
Object ID	A unique ID for the geometric model and its motion sequences
Category ID	A unique ID of the category the geometric model belongs to
Model Name	The name of the geometric model
Model Type	The type of the geometric model
Size	The scaling factor of the geometric model
Position	The location of the geometric model
Color	The color of the geometric model
Model TextID	The unique ID linking to the VRML Text Description
Proto TextID	The unique ID linking to the VRML Prototype

Table 2. Database fields for the motion sequence.

Field Name	Description
Object ID	A unique ID for the geometric model and its motion sequences
Motion Name	The name of the motion sequence
Motion Type	The type of the motion sequence
Motion Speed	The speed of the motion sequence
Start Time	Motion starting time sequence
End Time	Motion ending time sequence
Style	Motion style, e.g., brisk
Motion TextID	The unique ID linking to the VRML Text Description

Besides the geometric model table and the motion sequence table, the scene also has its table in the database, which is linked to the geometric model table by a Has-A relationship. Models are also linked to a Model_Component table which links records on the model table to other models if it is a complex model. A complex model is made up of simple models (or atomic models). Here also there exists a Has-A relationship. We also have a category class and each model belongs to a certain category. In our database the model table is linked by an Is-A relationship to the category table. Generally relational databases are designed based on an Entity-Relation (ER) diagram. The ER diagram of our framework is given in Figure 2.

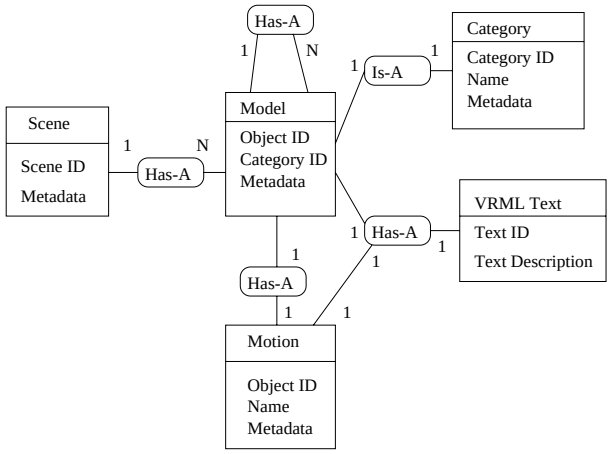

Fig. 2. The ER diagram of RDBS for the scene graph.

Using this design we can index the database by propagating the metadata of the animation objects upward, thus a scene would have all the metadata for its child animation objects. The searching would first be conducted at the higher

levels of the scene graph and will descend to lower levels only if a match is found. An alternative is by using categories as indexes, where animation objects or scenes are also assigned categories.

A query on the animation database is resolved by making use of a generic multi-attribute equation to compute the relevance of each motion sequence. A record satisfies the query on the condition that the relevance is equal to or higher than a predefined threshold.

Similarity measures are specified for the Motion Type, Style, Interval, Speed, etc. as well as metadata of the Geometric Models. Conceptually, a similarity measure $SIM(i, q, k)$ (normalized) is the similarity of an animation A_i, with query q, based on metadata k. It will be 0 if k does not occur in the metadata of A_i. The higher the occurrence rate and the closer the values are, the higher the similarity measure.

The overall similarity measure is the weighted sum of the individual similarity measures. The weights have a significant effect on the outcome of a query. For an animation, default weights are assigned based on the relative difficulty of readjustment of the metadata, such that the summation of weights is 1. The user has an option to change the weights if required.

Table 3 depicts an example of query results when a user performs the following query: Motion Type: running; Motion Style: normal; Motion Interval: 0-5; Motion Speed: 5. Given a threshold value of 0.5, only the top three records will be retrieved.

Table 3. The relevance scoring for a multiple result query.

Rank	Type	Style	Start Time	End Time	Speed	ID	Score
1	Run	fast	15	25	5	Nancy	0.60
2	Run	happily	0	20	2.5	Girl	0.50
3	Run	merrily	0	5	2.5	Nancy	0.65
4	Walk	normal	0	10	10	Andy	0.45
5	Wave	normal	6	18	15	Andy	0.42
6	Walk	sluggish	5	10	12	Man	0.24

3 Operations for Reusing Animations

This section discusses the operations a user can invoke to retrieve, insert, and adjust animation objects to scene graphs in order to create a new motion sequence. These operations are performed on the output of a user query. The metadata are used to adjust the queried results for a new animation. The operations can be divided into three categories: spatial operations, temporal operations, and motion adjustment operations.

3.1 Spatial Operations

Spatial operations on motion sequences involve changing the position, size, and orientation of geometric models. The operations used to query and manipulate the spatial attributes of an animation, are as follows:

1. *Insert Operation:* this operation inserts a geometric model into a scene under specified conditions. The user can include the new x, y, z coordinates of the model, and/or the time frame when the model will be shown.
2. *Delete Operation:* this operation deletes a geometric model from a scene. This can help the user eliminate unnecessary models from a retrieved result.
3. *Extract Operation:* this operation extracts a geometric model from a scene. It could also be used to extract an atomic geometric model from a complex model.
4. *Edit Operation:* this operation edits the metadata of a geometric model. The user can edit the position, size, and orientation of the model.

3.2 Temporal Operations

Motion is an integral part of any animations. Temporal operations help in retargeting motion for new characters and scenes. The operations for manipulating motion are use, get, and disregard.

1. *Use Operation:* this operation utilizes a motion to a character. We have adapted the motion-mapping approach to re-use existing motion to other models. The user maps the interpolator points of the motion to the joints of the geometric model to be applied to.
2. *Get Operation:* this operation extracts motion from a character. In a scene graph, motion is defined by keyframe animation and by the nodes: OrientationInterpolator and PositionInterpolator, among others. We simply search for the interpolators that are linked to the specified model and copy the keyframes.
3. *Disregard Motion:* this operation deletes the motion of a geometric model. This can be done by searching for the keyframes that are used on the model and by deleting them, including the timer node that was used.
4. *Project Operation:* this operation projects a specific time interval from a motion sequence based on the user's specified conditions. For example, if the user queried and found an animation matching all the requirements, except for time constraints, the user may opt to project a portion of the animation.
5. *Join Operation:* the operation allows the user to join two key frame sequences together. This provides the user with the ability to generate more complex motion from standard motion. The time base will be automatically adjusted.

3.3 Motion Adjustment Operations

In most cases, when a motion of a geometric model is reused by another one, it is necessary to adjust the motion to the new scene. Motion is usually very specific and hence we propose a set of operations to help the user adjust the motion to the new scene.

1. *Crop Motion:* operation cuts the motion into a smaller motion sequence by deleting the keyframes and calculating the appropriate substitutions.
   The crop operation edits all the Interpolator nodes (described in Subsection 2.1) for a particular motion, based on the crop value α. If α falls between two consequent keyframes k_i and k_{i+1}, the key frames from k_{i+1} to k_m will be deleted. The new ending key value k'_m can be computed using the key frame interpolation if it is not an existing one.
2. *Duplicate Motion:* operation duplicates the key frames to expand the duration of the motion. This helps if the user requires the same animation as what is retrieved but with a longer period.
   The duplicate operation adjusts all the Interpolator nodes for a particular motion, based on the duplicate value α. The old key frames will be duplicated α times.
3. *Change Speed:* operation changes the speed of the motion. It is implemented by scaling the key time base, i.e., the period represented in {key $[k_1, ..., k_m]$} of a PositionInterpolator node. So the duration of the motion will be changed. In order to keep the same duration, the new motion can be used in conjunction with the previous two operations.
4. *Retarget:* operation allows the user to adapt an existing motion sequence to meet the new constraints. Our current implementation uses inverse kinematics [6, 13] for articulated 3D models. The position and orientation of the end effectors are specified automatically by the new constraints or interactively by users. The inverse kinematic algorithm can compute the new position and orientation values of other internal joints automatically in real time.

To facilitate ease in using the system, a series of operations is automatically generated, depending on the specifications of user in the Animation Toolkit. The series is then applied to the query results to meet the requirements. The users do not need to know the operation syntax. They interact with the system, via the toolkit.

4 Examples of Manipulating the Query Results

Now, we present examples of using the operations described in Section 3 to manipulate the query results for reusing animations. Using the operations, three different types of reuse are possible: adjusting the motion sequence of a model (Subsection 4.1), applying motion to a different model with a similar structure (Subsection 4.2), and generating complex motion sequences for a model (Subsection 4.3). Using our database approach, the resulting motion will take the flavor of the original ones. Together with the operations defined in Section 3, the following clauses are also used.

FROM : used to specify the file in which motion/model is present
TO : used when applying a motion to a model or inserting a geometric model in a scene
WHERE : used to specify the condition of the query
WHEN : used to specify the duration in which the model is presented

4.1 Adjusting Motion of a Model

In this type of motion reuse, the motion needs to be modified. For example, the user can change the speed of motion and the duration. The modified motion is applied to the same geometric model that the motion was applied to previously. In a more general manner, the user can adjust motion to meet the new constraints. For example, the user wants to adjust an existing wiping motion to fit the new table in the scene. The following operations can be applied: EXTRACT woman FROM scene; GET wiping FROM woman; EXTRACT table FROM scene; INSERT table; EDIT table translation(10,5,8); USE wiping TO woman; RETARGET wiping. The result is shown as a snapshot in Figure 3.

(a) (b)

Fig. 3. Retargeting an existing motion to the new geometric model in the scene: (a) wiping a table; (b) bending to wipe a lower table.

4.2 Applying Motion to a Different Model

This type of reuse is needed in the case when the motion and model required, in the presentation, are not available in the same scene of the database.

For example, the user intends to create an animation of a man walking in a room. The query results are a room, a walking woman, and a model of man from three scenes respectively (Figure 4).

To generate the animation, according to the requirements, from the query results, the following steps will be conducted: (1) INSERT the model of man into the room, (2) EDIT to make the model of man well scaled and located in the room, (3) EXTRACT the walking motion sequence from the walking woman, and finally (4) USE the motion sequence to the model of man (Figure 5).

The motion mapping technique we use is based on the work of Lee et al. [8] and Gleicher [4], without the restriction that the two models should have the same structures. Since the interpolator nodes of one model can be connected by the user, using the GUI, to another model it offers much more versatility. For example, the walking woman described in the previous example can be remapped to an animal such as a cow. Since the scene graph provides a hierarchical

(a) (b) (c)

Fig. 4. Query results from three scenes: (a) a room; (b) a walking woman; (c) a man.

Fig. 5. Applying a walking sequence of a woman to man.

structure of both the human and the cow, the user can connect the respective segments of the cow with the interpolator nodes of the walking woman (Figure 6).

4.3 Generating Complex Motion Sequences

Complex motion can be derived by combining two or more motion sequences into a single sequence. The motion sequences may be from a single model or multiple models. The resulting complex motion may be applied to one of the original models or to a different model. An example of complex motion could be a man "walking" (lower body) and "waving" his hand (upper body) at the same time. In Figure 7, the snapshots show the individual motion sequences and the new motion sequence created. The following operations are applied to reuse the animations: GET walking FROM Andy; GET waving FROM Andy; JOIN Andy.walking WITH Andy.waving.

5 Implementation

We have developed an animation toolkit based on the proposed database approach using Visual Basic 6.0 (VB). Access 97 was used as the primary database

(a) (b) (c) (d)

Fig. 6. Mapping the motion of a walking woman to a cow: (a) frame 1 of woman; (b) frame 1 of cow; (c) frame 2 of woman; (d) frame 2 of cow.

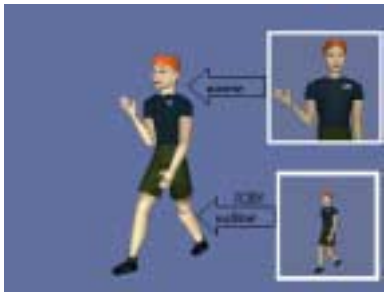

Fig. 7. An animation reuse example combining two sequences.

for the storage of the animations represented in scene graph. The scene graph can be rendered by Open GL Optimizer, IRIS Performer, and VRML browser. To parse and browse the animations we used the ActiveX object Web Browser, which invokes the default web browser of the system with the installed VRML browser, such as Cosmo Player or MS VRML viewer.

The GUI of the system uses a Multiple Document Interface with menus and toolbars. The major GUI components are the Query, Scene Graph, VRML Text Document, and VRML Browser windows. The user can create a new animation by choosing the new operations through the menu or the toolbar.

6 Conclusion and Future Work

Using animations of 3D models has been receiving more interest in multimedia presentations. In this paper, we have addressed this problem by a database approach. We have implemented an authoring toolkit using the approach and our experimental results are promising.

We are still working on a unified multimedia presentation system to integrate the animation toolkit with those of other types of media to support authoring

a more complex multimedia presentation. We are considering more applications for usability study.

Presently the toolkit is in the early stages of development. It is possible to incorporate a standard form of metadata description, e.g., XML at a later stage. We also will do the comparison study between our framework and MPEG-4.

7 Acknowledgments

The 3D models of the woman, the man, and the barmaid are available from http: //www.ballreich.net, http: //www.seamless-solutions.com, and http: //www. geometrek.com respectively. We thank the anonymous EGMM 2001 reviewers for the suggestions to improve the paper.

References

1. Ayadin, Y., Takahashi, H. and Nakajima, M. Database Guided Animation of Grasp Movement for Virtual Actors. *Proc. Multimedia Modeling '97.* (1997) 213-225
2. Bohm, K. and Rawok. T. C. Metadata for Multimedia Documents. *SIGMOD-Record Special Issue on Metadata for Digital Media.* Vol. 23, No. 4, Dec. 1994.
3. Bruderlin, A. and Williams, L. Motion Signal Processing. *Proc. ACM SIGGRAPH '95.* (1995) 97-104.
4. Gleicher, M. Retargeting Motion for New Characters. *Proc. ACM SIGGRAPH '98.* (1998) 33-42.
5. Hodgins, J. and Pollard, N. Adapting Simulated Behaviors For New Characters. *Proc. ACM SIGGRAPH '97.* Los Angeles, CA. (1997) 153-162.
6. Huang Z., Boulic R., Magnenat-Thalmann N., and Thalmann D. A multi-sensor approach for grasping and 3D interaction. *Proc. CGI '95.* Leeds, UK. (1995) 235-254.
7. Kakizaki, K. Generating the Animation of a 3D Agent from Explanatory Text. *Proc. ACM MM '98.* (1998) 139-144.
8. Lee, W.M. and Lee M.G. An Animation Toolkit Based on Motion Mapping. *IEEE Computer Graphics International.* (2000) 11-17.
9. Monzani J. S., Baerlocher P., Boulic R., and Thalmann D. Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting. *Proc. Eurographics 2000.*
10. Popovic, Z. and Witkin, A. Physically Based Motion Transformation. *Proc. ACM SIGGRAPH '99.* (1999) 11-19 .
11. Rohlf J. and Helman J. IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics. *Proc. ACM SIGGRAPH '95.* 550-557.
12. Thalmann D., Farenc N., and Boulic R. Virtual Human Life Simulation and Database: Why and How. *Proc.International Symposium on Database Applications in Non-Traditional Environments (DANTE'99).* IEEE CS Press, 1999.
13. D. Tolani, A. Goswami, and N. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models* 62(5), Sept. 2000, 353-388.
14. The VRML Consortium Incorporated. The Virtual Reality Modeling Language. *http://www.vrml.org/Specifications/VRML97/.* International Standard ISO/IEC 14772-1: 1997.