Specifying MPEG-4 Body Behaviors

Anthony Guye-Vuillème, Daniel Thalmann

Computer Graphics Laboratory
Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland
{Anthony.Guye-Vuilleme, Daniel.Thalmann}@epfl.ch

Abstract

The MPEG-4 standard specifies a set of low-level animation parameters for body animation, but does not provide any high-level functionality for the control of avatars or embodied agents. In this paper, we discuss the required features for a script format allowing designers to easily specify complex bodily behaviors, and describe a system and its associated syntax - Body Animation Script (BAS) - which fulfills these requirements in a flexible way. The described architecture allows the organization and parametrization of predefined MPEG-4 animations and their integration with real-time algorithmic animations, such as pointing at a specific location or walking. This system has been implemented at EPFL in the framework of the EU SoNG project, in order to allow intelligent software agents to control their 3D graphical representation and end-users to trigger rich nonverbal behaviors from an online interface. It has been integrated into AML - the Avatar Markup Language.

1. Introduction

Shared Virtual Environments (SVE) using avatars and other artificial body representations are very popular nowadays. Such systems as the Blaxxun community platform[2] or Active World[1] allow users to meet at specific locations, recognize each other thanks to their embodiments and interact nonverbally by triggering predefined body animations, in addition to the classic chat functionality. The SoNG (portalS of Next Generation) project aims at developing a MPEG-4 platform for the next generation of such applications, allowing end-users to access diverse resources and services in an easy and natural way using 3D computer graphics elements, intelligent agents, new user-interfaces for rich human/agent interaction and real-time audio-visual communication. In order to demonstrate how these new technologies can be integrated, a sample e-commerce application is to be developed.
Our work constitutes one of the building block aiming at allowing intelligent virtual sales assistants to guide the users in a rich shopping metaphor, e.g. pointing at an object while describing its characteristics, and permitting customers to interact with each other in a multimodal way, e.g. managing their body language. What is missing in order to allow rich avatar and agent body behaviors, is an intermediate layer between the decision making ability of the intelligent agent and the low-level rendering capabilities of the MPEG-4 player. Though it would be possible for the agent to directly generate the low-level animation data, it would place on it a useless burden and would make difficult for content providers to finely design and control the behavioral outcome. The added value of an extra layer allowing the reuse of predefined animations by providing parametrization facilities is also to be taken into account.

Body animation in MPEG4 is defined in terms of 296 BAPs - Body Animation Parameters, which represent a certain Degree Of Freedom (DOF) of an anatomical articulation. DOF can be translations or rotations along an axis. These BAPs can be applied to any MPEG-4 compliant body and produce the same animation.

2. Previous work

Current shared virtual environments (e.g. [2][1][4]) provide very limited functions, typically predefined gestures, e.g. bow, which are directly triggered by a GUI or keyboard command. More advanced system relying on rule-based techniques, like IMPROV[10] or BEAT and BodyChat[3][13], have produced interesting results but they do not allow a clear separation between the purely deliberating function and the behavior specification itself. Using different approaches, the Motivate system[9] and ACE[8] also tackle the issue of behavior specification but have not been designed to be used in MPEG-4 systems.

The Virtual Human Markup Language - VHML[12], a system independent XML-based language which has been successfully used in MPEG-4 applications, encapsulates a markup language dedicated to body animation (BAML). In its current state of development, it is unfortunately missing any time synchronization information which is crucial to seamless animation. Moreover, it doesn't handle any reference to the environment (e.g. "look_left", "look_right" tags can't be used to direct the avatar's attention to a precise location in the scene, an object, etc.), which makes it more suited to nonverbal chat scenarios (talking head/trunk type) than to applications where avatars and embodied agents are immersed in rich shared virtual environments.

3. Requirements

We have identified a number of basic requirements that the system should meet in order to allow the specification of body behaviors in a powerful and convenient way :

Reuse of predefined body animations composed of gestures and postures and customization facilities - The ability for the script format to reference predefined animations and modify them on-the-fly is crucial since it facilitates reuse and makes possible to finely adapt the animations to the context. The speed of the animation is an important parameter, e.g. a fast (enthusiastic) head nodding is not interpreted in the same way as its slow motion equivalent (reluctant). In some cases, it is also important to allow a change in the gestures' amplitude by increasing joint angles, since the open/close dimension has been identified by psychologists as very important for non-verbal communication (e.g. open welcome gesture/closed shy posture). The use of masks to inhibit the animation of some parts of the skeleton is also valuable.

Organization of animations in the temporal dimension using timing information suited to synchronization with facial animation and Text-To-Speech - The number of animations referenced should not be artificially limited, and designers should be able to assign an individual starting time to each of them. The generation of a continuous signal cannot be presupposed, e.g. an avatar should be able to adopt a posture, be inactive for 2 seconds, and resume its activity. The tests we have conducted give good results with a relative time format based on minutes, seconds and milliseconds (mm:ss.mmm). Overlapping sequences should be allowed and in this case the parser should be able to automatically perform motion blending. A priority mechanism can be introduced. A synchronization mode which allows actions to be triggered relative to each other, e.g. at the same time as the preceding action or directly after the preceding action has completed, would also be useful for animation whose duration is not known at design time.

Real-time parametric animations - There are body actions that cannot be pre-generated because they are dependant on dynamic and unpredictable conditions, like the position or orientation of the avatar in the scene. Some behaviors, e.g. pointing or walking to specific coordinates in 3D space, must then be generated on-the-fly using appropriate techniques, from a set of specific parameters given as input in the script. The list of required parametric animations is application specific, but for SoNG's e-commerce scenario, facing (e.g. a specific client), pointing (e.g. at an object) and walking (e.g. to another part of the shop) are especially important.

4. Syntax

A description of our current script syntax for MPEG-4 body behavior specification - Body Animation Script (BAS) - based on the concept of animation track, is given in fig. 1. After that global settings are specified in the first section, an arbitrary number of body animation tracks can be given and each track can refer to several distinct .bap files. This reference specifies a start time or a synchronization mode syntax ('autosynch', 'autoafter'), as well as additional envelope parameters. The speed can be modified and an intensity value can be used to exaggerate or scale down the effect of the predefined animation. The priority parameter specifies which BAPs are to be used when several sequences overlap. An intermediate solution is automatically computed when the given priorities are equal. A mask input can also be used to indicate which BAPs must be processed or ignored in a specific track.

For gestures that cannot be predefined but need to be generated in real-time, a special syntax has been embedded in the script format. The references to these behaviors are given in the same way as predefined animations, i.e. in the scope of an animation track, with a start time/code, a speed value. The output animation is processed transparently : the mask is applied to it as it is to predefined animations, motion blending is applicable, etc. Facing, Pointing, Walking are the main basic behaviors yet defined (simple Waiting and Resetting are also provided). Each behavior has a target location specified by x, y, and z coordinates in 3D space. Additionally, the walking behavior can specify any number of control points through which the avatar must pass, by using one or more control points.

5. Architecture

Fig. 2 shows the main components of the architecture. The BAP DB contains a number of body animation files that are the basic building blocks used by the BAS Parser. These MPEG-4 animations have been generated using motion capture techniques and hardware (e.g. magnetic tracking unit) or manually designed using dedicated authoring

```
#BODY ANIMATION SCRIPT (BAS)
#A line starting with # will be treated as a comment
#A mask which does not inhibit any BAP can be replace by the
keyword all_set

BODY ANIMATION SCRIPT
[SETTING]
Project=<string name of project>
Fps=<int frames_per_second>
Duration=<mm:ss:mmm>
BodyAnimationLibraryPath=<string Local path of the folder
containing .bap files>
BodyAnimationTrackNumber =<int nt=number of body animation
tracks>

[BODY ANIMATIONS TRACK]
Name=<Name of track>
Mask=<int[296], the BAP indices that are affected by this track are
set to 1, the rest 0>
Number=<ne=number of basic body animations in this track>
<start time=mm:ss:mmm or autosynch or autoafter> [<file
filename.bap>] [ <speed =normal, slow or fast> <intensity=float>
<priority=int>……….ne such definitions]

…
nv such lines.

#Facing behaviour. x, y and z are the target's coordinates in meters,
starting_time, speed and priority are used in the same way as with
.bap files (speed=[normal,fast,slow], priority=[1..99])
<start time=mm:ss:mmm or autosynch or autoafter> [behaviour face]
[ x y z speed priority ]
#Ex: 00:00:000 [ behaviour face ] [ -10.5 -10 0 normal 0 ]

#Pointing behaviour. x, y and z are the target's coordinates in meters
<start time=mm:ss:mmm or autosynch or autoafter> [behaviour point]
[ x y z speed priority ]

#Walking behaviour. num_controlpoint is the number of control points
through which the avatar must pass, x, z are the control points'
coordinates in meters
<start time=mm:ss:mmm or autosynch or autoafter> [behaviour walk]
[ num_controlpoint speed priority ] [ x  z ] (… num_controlpoint such
blocks)
#Ex: autoafter [ behaviour walk ] [ 2 normal 1] [ 5 0 ] [ 5 5 ]

#Resetting behaviour. Reset the body to the default posture
<start time=mm:ss:mmm or autosynch or autoafter> [ behaviour
reset ] [ speed ]

#Waiting behaviour. Duration in seconds. Useful for synchro mode
<start time=mm:ss:mmm or autosynch or autoafter>[ behaviour wait ]
[ duration ]
```

Figure 1. BAS syntax

software (e.g. 3DS Max) and appropriate exporter plug-ins (also developed within the SoNG project). The BAS scripts can as well be stocked in a database, typically in the case of the Online Interface, or they can be dynamically generated by the MPEG-4 Agent according to its perception of the virtual environment (using Server Commands or scanning the scenegraph with EAI). MPEG-4BehaviorLib is a plugable dynamic library which is in charge of generating on-the-fly BAPs. It is called when a parametric animation reference has been identified in the script. FBAEncoderLib, another dynamic library, is sent the BAP data when one frame of the final behavior is ready. It creates a compressed FBA bitstream and prepares the encapsulation into a MPEG-4 BIFS-anim stream. The streaming server sends the data to the player, which separates the FBA content, uncompresses the BAPs, composes from them a rotation matrix for each modified joint before updating the scenegraph in a timely fashion.

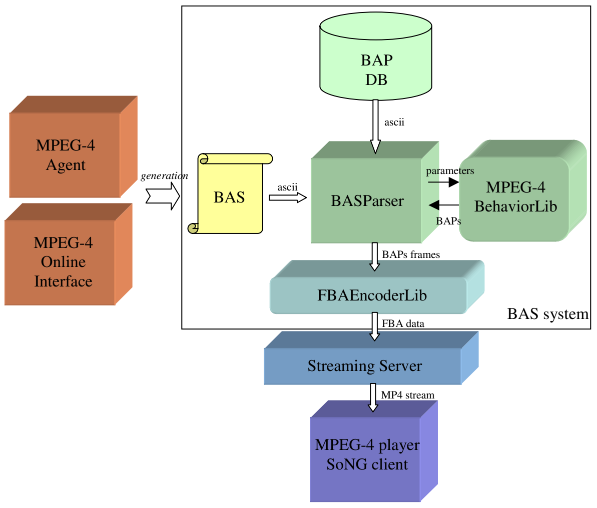

Figure 2. Architecture

6. Implementation

6.1. BASParser

All components have been developed in C++ and take advantage of powerful object-oriented features like polymorphism. Two main classes serve as a basis for the BASParser: BASBody, which directs the whole process at a high-level and is in charge of maintaining information about the current body state, and BAScript, which analyzes the scripts, handles all references to .bap files and dynamic behaviors, retrieves and blends the adequate BAP frames in an orderly fashion. Blending is based on a simple weighted averaging approach, maintaining a dilution factor for each BAP. Since the .bap file format defines a progressive animation, i.e. only the BAP updates are given, the signal must first be made continuous in order for the blending to give acceptable results. BAPFrame and its associated BAPMask class are used in most other classes. They efficiently handle one frame of indexed BAP data and provide some important low-level methods, such as computing masks union/intersection, averaging two frames of BAPs, etc.

BAPSource is the base class for every class providing BAPs to the parser, be it by reading and processing data from a file - BAPFile, or from an algorithmic source - SAFace, SAPoint, SAWalkTo. These three specialized action classes are also derived from BAPGenerator, which defines a set of matrix manipulation methods widely used in their implementation, e.g. quaternion interpolation. This architecture allows most of the code to be located in the base classes, thus keeping the derived classes very short and simple, which makes it very easy to add new behaviors. Moreover, the parsing and setup of the system become straightforward: each time a reference to a .bap file or to a behavior is met, a corresponding C++ object is instantiated and placed in a STL vector of BAPSources. Then, at each frame and for every object in the vector, the parser calls the GetNextFrame method, which is dynamically bound to the adequate implementation depending on the type of BAP source.

6.2. MPEG-4BehaviorLib

MPEG-4BehaviorLib is a C++ library which makes use of classic animation techniques based on matrix manipulation, like quaternion interpolation, and of inverse kinematics. IKAN[11], an inverse kinematics library developed at the University of Pennsylvania, is linked with MPEG-4BehaviorLib and computes, among other things, the final orientation of the arm in the Pointing action, given an elbow flexion parameter. It is also ideally suited to the implementation of new specialized actions like Touching or Grasping.

In order to exemplify the flexible nature of the architecture, we now provide a short example explaining how a simple walk engine (SAWalkTo) can be implemented. The first thing to consider is that most walk engines are based on a combination of a translations and postural shifts. Besides, it should be mentioned that every BAPSource holds a pointer to its containing vector, which allows it to trigger other actions and control their execution. Given these two elements, the subsequent steps can be followed in order to implement SAWalkTo : 1) At the beginning of the animation, a new SAFace action is to be triggered using the same timing and target parameters as the walk action. It will gradually modify the body and head orientations so that the target location is faced after the necessary number of frames 2) Simultaneously, a predefined animation representing a short cyclic walking sequence (2 steps, typically generated from motion capture) should be run. At each frame, the BAPFile state has to be checked and a new one instantiated when the former is completed 3) Simple translations have to be generated in the GetNextFrame method of the SAWalkTo object. When the target is reached, all child actions have to be marked as inactive and the posture should be reset by adding a final SAReset object to the BAPSource vector. Since all actions are processed in one single pipeline and every frame is automatically blended by the parser, such a simple procedure gives very good results in our architecture. As can be seen, the chosen approach is powerful and greatly facilitates code reuse.

6.3. FBAEncoderLib

The C++ FBA encoder that we have implemented is based on an arithmetic encoder coupled to a circular buffer. On top of it, producer and consumer threads, synchronized using semaphores, manage the input and the output of the BAP/FBA data. Since it was a requirement from the use-cases, it has been especially optimized so that the whole process is able to work in real-time.

7. Conclusion

We hope that this paper has reached the following main objectives:

- Adequate requirements identification for the task of specifying MPEG-4 body behaviors.
- Presentation of useful technology, including an efficient animation parser architecture, for shared virtual environments in general and particularly SoNG.
- Drafting of a proposal to address the need for a high-level syntax for MPEG-4 body animation.

The next step in the development of our solution is to integrate it, in a suitable and efficient way, with a MPEG-4 facial animation system and TTS engine. This is already well advanced. It will allow designers to create rich multimodal behaviors, including synchronization of verbal and nonverbal messages, which is crucial for the realistic modelling of human behavior.

Finally, it should be mentioned that we are currently working on a XML-based version of the BAS script format, which will facilitate validation, parsing and streaming of the behaviors.

8. Acknowledgements

The European IST Project 10192 SoNG has supported the work described in this paper. We are grateful to all the partners of the SoNG project for their direct and indirect involvement and help. Many thanks to Kaveh Kamyab and Sumedha Kshirsagar for providing inspiration and support, and Fabrice Vergnenegre for his work on the FBA Codec. Finally, special thanks are due to Beatriz Dias for proof reading the manuscript.

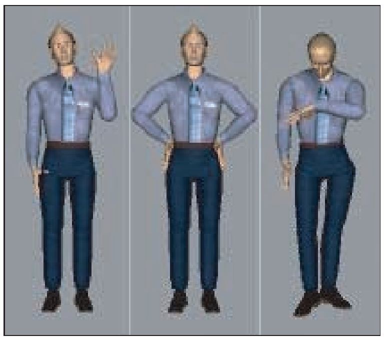

Figure 3. A sample from the BAP DB

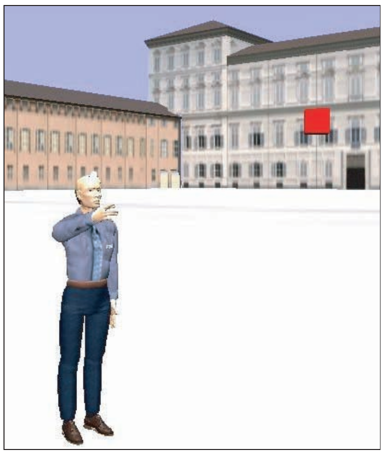

Figure 4. An example of Pointing

References

[1] Activeworld. http://www.activeworld.com/.

[2] Blaxxun Interactive. http://www.blaxxun.com/.

[3] J. Cassell, H. Vilhjlmsson, and T. Bickmore. Beat: The behavior expression animation toolkit. In A. Press, editor, *Proceedings of SIGGRAPH*, 2001.

[4] A. Guye-Vuillème, T. K. Capin, I. S. Pandzic, N. M. Thalmann, and D. Thalmann. Nonverbal communication interface for collaborative virtual environments. *The Virtual Reality Journal*, 4:49–59, 1999.

[5] H-anim. http://www.h-anim.org/.

[6] ISO/IEC JTC 1/SC 29/WG11 N2502, Information Technology - Generic Coding of Audio-Visual Objects, Part 2: Visual, October 1998.

[7] ISO/IEC JTC 1/SC 29/WG11 N2739 subpart 2, MPEG-4 Version 2 - BIFS, March 1999.

[8] M. Kallmann, J.-S. Monzani, A. Caicedo, and D. Thalmann. Ace : Aplatform for the real time simulation of virtual human agents. In *Proceedings of EGCAS'2000 - 11th Eurographics Workshop on Animation and Simulation*. Interlaken, Switzerland, 2000.

[9] Motivate product information, Motion Factory. http://www.motion-factory.com/.

Figure 5. Blending of two dances, charleston and disco

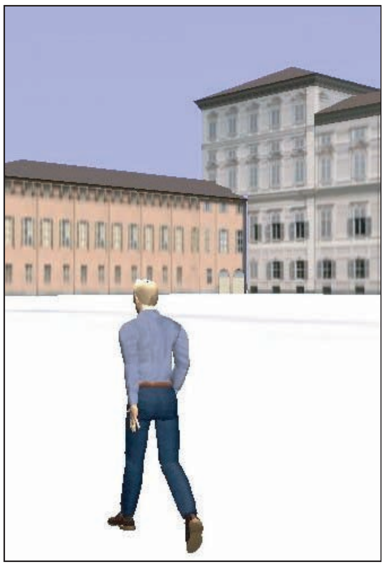

Figure 6. An example of Walking

[10] K. Perlin and A. Goldberg. Improv: A system for scripting interactive actors in virtual worlds. *Computer Graphics*, 30(Annual Conference Series):205–216, 1996.

[11] D. Tolani, A. Goswami, and N. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models*, 62(5):353–388, 2000.

[12] VHML. http://www.vhml.org/.

[13] H. H. Vilhjalmsson and J. Cassell. Bodychat: Autonomous commumicative behaviors in avators. In *Proceedings of International Conference on Autonomous Agents*, pages 269–276, 1998.