

Bjoern Hartman  
Advisor: Dr. Norm Badler  
Applied Senior Design Project - Final Report

# Human Character Animation in 3D-Graphics: The EMOTE System as a Plug-in for Maya

## Introduction

Realistic animation of human or human-like characters within a 3D computer graphics environment is a complex task. Beyond the issue of correctly simulating the physical aspects of movement lies the deeper problem of rendering this motion expressive. Three major strategies to approach this matter can be distinguished:

1. In traditional methods of **keyframe animation**, an artist manually adjusts key figure poses between which a system then interpolates. Keyframe animation can yield both realistic and vivid results but it is time and labor intensive and more importantly dependent upon the artist's permanent personal exertion of control over all aspects of the process. These constraints render keyframe animation unsuitable for many applications where such talent and resources might not be available, especially in the area of entirely computer-generated animation exclusive of human intervention.
2. **Procedural animation** does not require such user participation. Instead, animation paths are computed according to some underlying algorithm. Thus far, procedural methods have often lacked naturalness since parameters modeling the qualitative aspects of human movement are not well understood or agreed-upon.
3. Finally, animations translated from a real environment into a computer graphics system by means of **motion capture** require no artistic involvement or complex computation, but their utility is ultimately very limited since, without presence of an interpreting computational model, no generalizations can be made from a particular recorded sequence to a more universal class of movements.

Recently, progress has been made towards building a procedural system that takes advantage of the strengths of the other two approaches. Specifically, the EMOTE system, to be described below, allows for realistic procedural gesture synthesis based on both motion capture data and keyframe-definable qualitative parameter information. However, since EMOTE, like most other research in the area, was born out of an engineering context within an academic setting, its visualization system is unlikely to be found in a typical artistic 3D graphics production environment. Encapsulating the EMOTE system into a plug-in for an industry standard 3D modeling and animation package would extend its range of potential users and applications as well as allow the system to take advantage of advanced animation and rendering features not available in engineering-centered applications.

The objective of this Senior Design project, then, was threefold:

1. To analyze and understand a sophisticated, realistic model of procedural human movement animation.
2. To gain familiarity with the plug-in authoring process for a widely used 3D graphics system.
3. To subsequently combine insights from the previous two steps by translating such an animation system into a flexible plug-in format that takes advantage of the strengths of the 3D application.

## The EMOTE System

The computational model used in this project is the **EMOTE (Expressive MOTion Engine)** system developed at the University of Pennsylvania's Center for Human Modeling and Simulation (HMS). Focusing exclusively on upper body limb movement, EMOTE procedurally synthesizes gestures based on principles of movement observation science, namely Laban Movement Analysis (LMA) and its Effort and Shape components. Effort and Shape variables represent a convenient set of high-level qualitative parameters to control the form and execution of movement. Prior to this project, EMOTE was solely available as an extension to the Jack toolkit. While Jack excels in simulating human factors, its models are not deformable and its rendering capabilities are limited compared to other commercially available, more artistically oriented packages. For a detailed description of EMOTE, see [2].

## Alias | Wavefront Maya

The chosen visualization environment for this project is Maya version 3.0 [Figure 1]. Maya is the industry standard modeling, animation, rendering and special effects application for broadcast, film, multimedia and game development. It is extremely powerful and flexible - notably, it features a full range of deformation functions, some of which will be applied to a human model in this project. Furthermore, Maya has an open architecture and can thus easily be extended by third-party developers. The two main ways to customize Maya are through the **Maya Embedded Language (MEL)** and the **Maya Application Programmer Interface (API)**. MEL is an interpreted scripting language that runs within the Maya environment. It provides high-level access to Maya nodes and allows for easy creation of custom graphical user interfaces (GUIs) through its **Extended Layer Framework (ELF)** interface commands. While MEL supplies the user with commands that can control most of Maya's features, it suffers from the performance disadvantage inherent to interpreted languages and also fails to offer important programming elements such as pointers. By accessing the Maya API, custom compiled shared objects (written in C++) can be added to Maya for low-level access to internal data structures. Compiled plugins also offer significantly faster execution times than MEL scripts.

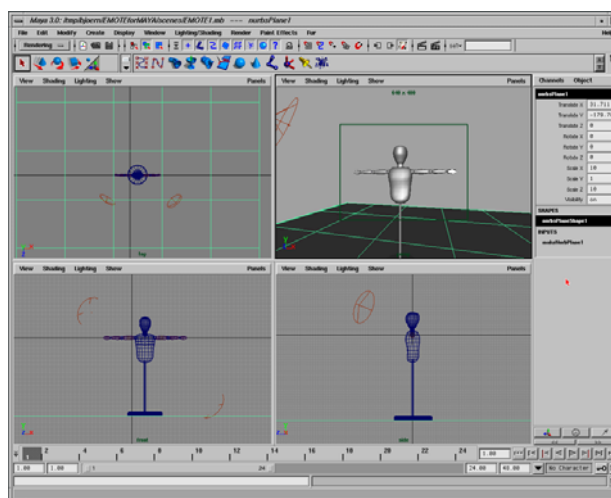
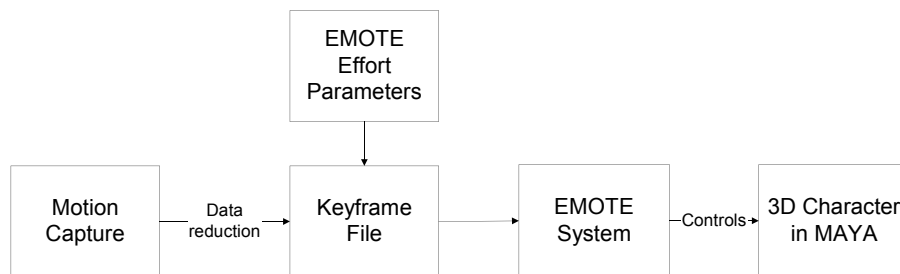


Figure 1 - The Maya 3.0 Environment

## System Outline

The basic structure of the system implemented for this project can be summarized as follows [see Figure 2]: First, primary data is acquired from a live actor through motion capture. Specifically, the actor's wrist positions in XYZ-space are recorded. In the next step, data density is reduced by eliminating all but the extreme keypoints of the captured motion (those points where maximum limb extensions and/or changes of direction occur)<sup>1</sup>. Furthermore, the recorded wrist position coordinates are transformed from world space to a coordinate system relative to the actor's shoulder joints. The data points thus obtained are then stored in a keyframe phrase file along with EMOTE Effort parameters that control the qualitative aspects of the animation to be generated. A sample keyframe phrase file for a right arm gesture is reproduced in Figure 3. The phrase file is provided as input to the EMOTE plugin which interpolates between keypoints using Tension-Continuity-Bias (TCB) spline curves whose parameters are derived from the user-defined Effort keys. The generated spline curves are then used as animation paths to control the limb trajectories as well as additional deformation parameters of a 3D human model within Maya.



**Figure 2 - EMOTE for Maya Block Diagram**

<sup>1</sup> In the sample gesture used for the project, this process reduced the number of data points from roughly 170 down to 11.

```

emacs@buzz.cis.upenn.edu
Buffers Files Tools Edit Search Mule Help
/* PHRASE FILE
/* 0 - global, 1 -> local
/* right hand
IkFrame 1: 0 33.7440 25.9920 32.3760 -1.3600 0 1
IkFrame 10: 0 24.6240 30.3240 -27.3600 -0.6300 0 1
IkFrame 16: 0 38.5320 41.0400 6.8400 -1.3000 0 1
IkFrame 36: 0 24.6240 30.3240 -22.5720 -0.6300 0 1
IkFrame 39: 0 38.5320 41.0400 6.8400 -1.3000 0 1
IkFrame 66: 0 24.6240 25.0800 -34.2000 -0.6300 0 1
IkFrame 83: 0 38.5320 45.3720 6.8400 -1.5400 1 1
IkFrame 100: 0 38.5320 50.3720 10.8400 -1.5400 0 1
IkFrame 127: 0 25.3080 -2.7880 10.8400 -1.1700 0 1
IkFrame 140: 0 35.1480 -2.7880 13.6800 -1.1700 0 1
IkFrame 168: 0 39.9000 -4.5760 26.6760 -0.9000 0 1
/* Effort parameters
/* space direct +----- indirect
/* weight strong +----- light
/* time quick +----- sustained
/* flow bound +----- free
IkEfframe 1: 0 0 0 -0.5 0 0 -0.3
IkEfframe 10: 0 0 0 0.5 0 0 -0.3
IkEfframe 16: 0 0 0 -0.5 0 0 -0.3
IkEfframe 36: 0 0 0 0.5 0 0 -0.3
IkEfframe 39: 0 0 0 -0.5 0 0 -0.3
IkEfframe 66: 0 0 0 0.5 0 0 -0.3
IkEfframe 83: 0 1 0 0 0 0 0
IkEfframe 100: 0 1 0 0 0 0 0
IkEfframe 127: 0 1 0 1 0 0 1
IkEfframe 140: 0 1 0 1 0 0 1
IkEfframe 168: 0 1 0 1 0 0 1
-:*** test2.kf (Fundamental)--L1--Top-----

```

Figure 3 - A sample keyframe phrase file

## Implementation Details

As mentioned above, Maya provides two different toolsets for extending the functionality of its system: MEL scripts and the Maya C++ API. For the EMOTE adaptation for Maya, an integrative approach was taken, incorporating both an administrative MEL script and a computation plug-in written in C++ [see Figure 4]. The MEL script controls the GUI and handles the loading, execution, and unloading of the plug-in within Maya, while the compiled C++ object, the plug-in proper, performs the actual interpolation of animation parameters. To understand the interconnection between script, plug-in and geometry objects, it is necessary to introduce the Maya node system.

A scene in Maya is internally represented in the form of a **Dependency Graph (DG)**, a collection of connected entities called DG nodes. An important subset of DG nodes are **Directed Acyclic Graph (DAG)** nodes. DAG nodes can be further divided into two subgroups: shape and transform nodes. Shape nodes contains the actual geometry present in the scene, while transform nodes manipulate position, orientation and scale of the shape nodes. Directed edges in the DAG correspond to a parent-child relationship. In non-DAG DG nodes, connections (which can be cyclic) allow data to move from one node to another. The data interface for each node is defined by its **attributes**. Information flows in and out of a node through its **data plugs**, instantiations of the attributes. Within a node, data received on its inputs plugs can be manipulated and computation results can be made available at its output plugs. Maya requires nodes to be self-contained in that they are not allowed to acquire or manipulate any additional external scene data beyond what is available through its plugs.

The EMOTE plug-in was implemented as a DG node. The accompanying MEL script creates an instance of the node, called `emoteNode` as well as a GUI to supply data to the node's input attributes [see Figure 5]. Once the user has specified the filename of the key phrase file to be used and the object names of the deformers and arm joints to be animated, the script connects the `emoteNode`'s output attributes to the corresponding DAG transform node attributes which in turn manipulate the shape nodes that store the geometry.

Specifically, EMOTE's interpolated wrist position outputs are connected to the inverse kinematics handles of the human figure's arm bone structure. Thus EMOTE can take advantage of Maya's built-in inverse kinematics solvers.<sup>2</sup> Additionally, EMOTE can output arm deformation parameters to change the shape of the arm geometry as will be discussed below.

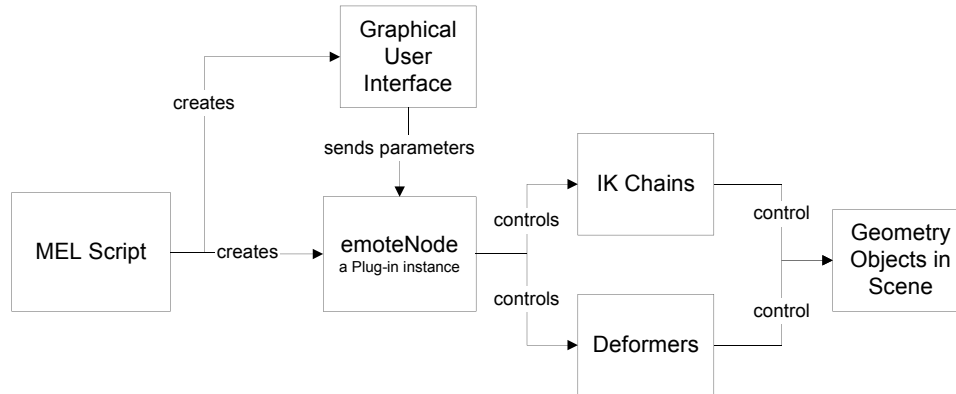


Figure 4 - Detailed EMOTE for Maya Architecture

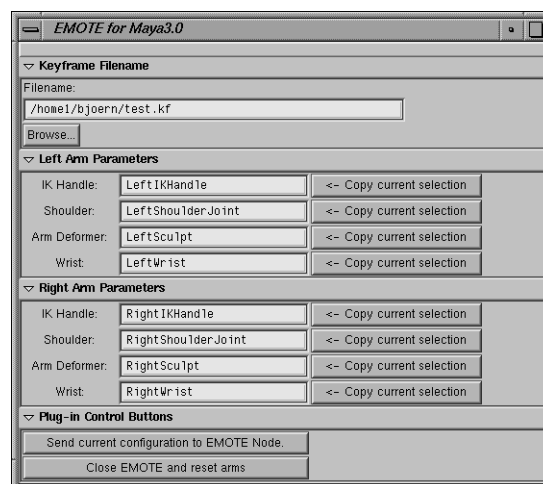


Figure 5 - EMOTE for Maya Graphical User Interface

## Animating the figure

Because of performance limitations of the available Maya installation<sup>3</sup> it was infeasible to work with a detailed, high polygon-count human model. Instead, a simplified upper-body model was developed based on a wooden artist's mannequin [Figure 6]. The figure was fitted with a simple shoulder-elbow-wrist bone structure in each arm to which the geometry

<sup>2</sup> After specifying wrist position, one rotational degree of freedom remains – the elbow swivel angle, calculated as the angle between the plane defined by shoulder, elbow and wrist joint, and a “down” vector pointing in the negative y-direction. EMOTE calculates this angle independently of Maya's IK and makes it available as another output attribute which is then connected to the IK chain's corresponding twist attribute.

<sup>3</sup> Maya was not installed locally on the SGI machines, but rather ran remotely on GRAPHICS.CIS and had to pipe all graphics through the network, making it impossible to manipulate complex objects in real-time.

nodes were parented. By using spherical joints in the model, the traditional difficulties in maintaining coherent geometry and avoiding gaps or polygon interpenetration around the shoulder and elbow areas could be averted. Connecting the figure's arm joints to the EMOTE plug-in successfully yielded the anticipated animation sequence [see Figure 7]. Comparing the generated gesture to a sequence created by the original EMOTE system for the Jack toolkit yielded only minor discrepancies chiefly attributable to scale differences in the two figure models employed.



**Figure 6 - The mannequin - a simplified human upper body model**

To increase the realism of the performed motion, some preliminary experiments with deformers were carried out. In particular, different methods of animating limb volume were tested to simulate human biceps contraction. Limb volume was defined as a linear combination of elbow angle and the Effort parameter Weight: the stronger a movement and the further bent the elbow, the larger the resulting deformation. Three separate renderings of the test gesture were generated using different deformation methods. First, a control version was created without any limb volume changes. In the second approach, the entire upper arm was scaled in the YZ-direction by a variable factor greater than 1 (the local X-axis runs through the arm lengthwise, so YZ-scaling increases the cross-section area of the arm). Finally a more sophisticated technique was considered: an ellipsoid sculpt deformer was placed into the upper arm that pushed arm vertices in its vicinity outwards, thereby mimicking the shape of an actual human muscle. The amount of deformation in this strategy was controlled by varying the envelope (the amount of influence the deformer has on geometry vertices) of the deformer. Figure 8 shows corresponding frames for the three approaches in a position of near-maximum deformation (large elbow angle and maximum Weight setting)



Figure 7 - Three still frames taken from the generated animation



Figure 8 - Limb volume deformations: none, ellipsoid sculpting, YZ-scaling

## Future Work

Human upper body movement is far more complex than can be captured by the two variables examined in this project. For the arm, parameters for elbow twist, wrist bend and wrist twist should be added. Animating realistic hand animations is yet another complex challenge. Furthermore, it is important to recognize that most human gestures are not executed by the arms alone. Corresponding torso movements and deformations frequently accompany arm movements. Building an expressive torso should therefore be considered to further enhance realism. Finally, the animation system should be applied to a more detailed, life-like model to be able to judge whether the results are truly convincing or not.

## Acknowledgements

I would like to thank Liwei Zhao, one of the original authors of the EMOTE system, for providing me with the source code and test keyframe phrase files for EMOTE as well as for being an indispensable source of answers for my many questions.

---

## References

1. Alias|Wavefront. *Maya 3.0 Unlimited Online Library*. (Online).
2. Diane Chi, Monica Costa, Liwei Zhao, and Norman Badler. The EMOTE model for Effort and Shape. In *Proceedings of SIGGRAPH '00*, pages 173-182. ACM Press, July 2000.
3. Liwei Zhao and Norman Badler. *Learning Motion Qualities from Observation*. 2001 (forthcoming).