Scripting Choreographies

Stefan M. Grünvogel and Stephan Schwichtenberg *

Laboratory for Mixed Realities, Institute at the Academy of Media Arts Cologne
Am Coloneum 1, D-50829 Köln, Germany
{gruenvogel, schwichtenberg}@lmr.khm.de

Abstract. We present a system for the simple and fast creation of character animation which is used within an augmented reality environment. Instead of using fixed sequences of animation, the motion of the characters are created by subtasks which are able to change the motion dynamically according to the environment. Dynamic motion models are used to produce the animation data. These dynamic motion models are controlled by the subtasks to create their appropriate behaviour.

1 Introduction

We are developing the character animation system for the augmented reality project mqube (http://www.mqube.de). The aim of this project is to build a prototype of a multi-user environment, where several people work together (directors, stage and light designers) to create the stage set and to place the lights on a miniaturised stage scaled down by factor 4. The user group also tests the interplay between actors, dynamic light and scenery. For the simulation of actors on the stage virtual character are used. The users are not interested in the low level editing of character animation. Instead a simple and fast way for the creation of complex character behaviour (e.g. let the character walk along a given path and wave with its arms at a certain time) has to be provided. Furthermore a characters should react on the properties on the stage, e.g. jump independently over obstacles which occur on its path.

There are only few approaches where the interaction with virtual characters in augmented reality is examined (e.g. [1],[2]). There the behaviour and the animation of the characters is created automatically by the underlying system. Our aim is to give the user herself the possibility to edit the movements and the behaviour of the character within the environment.

2 System Architecture

The CEManager is the interface to the AR-System (cf. Figure 1). It creates and deletes the scene graph nodes of the characters in the render engine and passes commands to the the choreography editor.

* This work was supported by the German Ministry of Education and Research (BMBF Grant 01 IR A04 C: mqube - Eine mobile Multi-User Mixed Reality Umgebung). test

The choreography editor controls the behaviour of all the characters in the scene. A character is represented within this level by a CECharacter object. The CEChoreographyEditor creates or deletes CECharacter objects and sends them CEDirectives to control their behaviour.

To each CECharacter belongs a unique AEMotionController. The CECharacter controls the production of the animation of the character by sending AECommands to the AEMotionController. The AEMotionController is responsible for the real-time creation of the animation. The AEBuffer and the AESubmitter connect the output of the AEMotionController with the corresponding node in the scene graph.

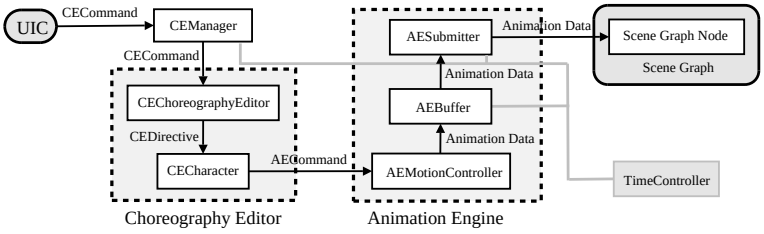

Fig. 1. The System Architecture.

CEManager, the AESubmitters and the AEBuffers are separate threads synchronised by the TimeController. The TimeController also transforms the time information from the AR-system (which has its own virtual time) into the internal virtual time of the character subsystem. The current time within the internal virtual time can be moved backward and forward with arbitrary speed.

3 Choreography Editor

3.1 Creation of Choreographies

Non-linear animation tools (e.g. Maya^{TM} or Filmbox^{TM}) create animations by blending and merging animations layered on a time line together. We adopt this paradigm, but use instead of fixed animation sequences so called subtasks for the dynamic creation of the character motion.

Subtasks model reflexive behaviour exhibiting a fixed behavioural pattern in response to given stimuli. They can be classified as level 0 in Brooks' subsumption architecture (cf. [3]). Examples for subtasks are waving with hands or to walk along a given path. The resulting motions of the character can be influenced by the user or by objects within the virtual environment during their execution (cf. Figure 2).

For the user and the AR-system these subtasks are hidden behind the CEManager. The user creates a choreography script by spooling forward or backward

in virtual time and sending CECommands to the CEManager. These CECommands are used to create new subtasks at the current time or to change the behaviour of a subtask which is active at this time. The overall choreography of the character is given by playing the CECommands at the virtual time they are received by the CEManager.

3.2 Character Model

The CEChoreographyEditor is responsible for the creation of CECharacters and their animation engine (cf. Figure 1). It also interprets the CECommands received from the CEManager and sends the appropriate commands (CEDirectives) to the CECharacter.

Within a CECharacter the subtasks are realized as state machines. The subtasks are using dynamic motion models (cf. Section 4) for the creation of their animation. The advantage of using dynamic motion models is, that they also provide a high level interface for the creation and manipulation of animation data, hiding their actual implementation.

Each CECharacter administers its choreography script of the character, which is a ordered list of CEDirectives. They are used for the creation, deletion and manipulation of subtasks. If the choreography is played, CEDirectives are send to the subtask according to their time stamp. CEDirectives hold information like the addressed subtask and a set of parameters which may change the behaviour of the subtask.

The CECharacter also resolves conflicts between different active subtasks which can occur if two subtasks need to control the same parts of the body. There are three methods to resolve this conflict: ignore one subtask, suspend one subtask until the other has finished or abort one subtask.

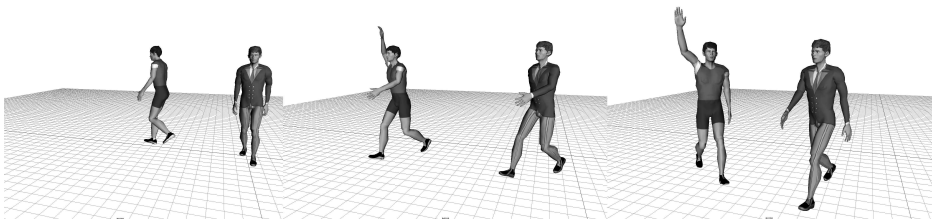

Fig. 2. Result of two simple choroegraphies. Two characters are walking along a path with different walk styles, one is waving.

4 Animation Engine

The overall real-time creation of the animation data of a character is controlled by its AEMotionController. The AEMotionController uses dynamic motion mod-

els for the creation of animation data. The term motion model first appeared in the offline animation context (Grassia [4]). They resemble the lowest level in the Improv system by Perlin and Goldberg [5] (cf. [6] for a discussion on the differences). In [6] we describe *dynamic* motion models, which can be used for interactive real-time environments.

Dynamic motion models are models for simple movements like waving, pointing or walking, each having motion specific sets of parameters (e.g. walk speed and style for the walk motion or the left or right hand for waving), which can be changed while playing the animation. The dynamic motion models are implemented as state machines. Their state can change according to the passed time of the animation or the received new parameters or commands. All motion models have four states in common: *idle*, *prestart*, *running* and *stopping*. In *idle* state the motion model does not produce any animation. If the motion model receives the command to start it turns into *prestart* state where it adjusts the characters' pose to start the actual motion. After having reached this pose, it switches into *running* state, where the actual motion is synthesised. If the goal of the motion is reached, it switches into *stop* state where the character is brought into a neutral position.

The characters are represented within the animation engine as articulated figures, where the rotation and translation values of joints are updated in a fixed frame rate. Dynamic motion models create their motion by combining animation clips with clip operators. Animation clips are abstract objects producing for a specific time span on the time line animation data. Clip operators have animation clips as operands and take the animation data of their operands to produce new animation data. Because clip operands are likewise animation clips, they can also be the operands of other clip operators. The actual source of animation data are clip primitives which store pre-produced animation data. To create the animation of a character which starts to walk from a standing posture we construct e.g. the operator tree in Figure 3. *WalkStart* and *WalkCycle* are clip primitives where *WalkStart* is the animation where the character makes his first step from a standing posture and *WalkCycle* is a walkcycle. By the *TimeShift* clip operator the start frame of the *WalkLoop* is shifted to the last frame of the *WalkStart* clip on the time line. The *Loop* repeats the underlying clip for an infinite time. Then finally the overall animation is produced by blending the *WalkStart* with the result of the *Loop* together. The advantage of dynamic motion models is that e.g. by exchanging the clip primitives with walk animations in a different style a whole new animation is produced easily.

Motion models produce their animations by constructing operator trees like in Figure 3 with the help of *base motions*. These are clip primitives together with annotations further describing the clip primitive. E.g. for a walk cycle the frames where the heel hits the ground or the specific style of the animation is stored in the annotations. This information is used by the motion model for correctly blending the different clip primitives together. If a motion model receives some new parameters (e.g. change the style of the walk movement) it destroys its current operator tree and creates a new one according to the new parameters.

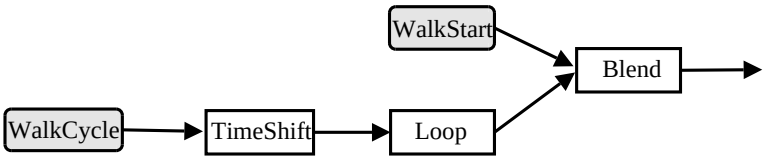

Fig. 3. The operator tree for a walk animation.

Thus it is possible to change the characteristics of a motion in real-time. If more then one motion model is active, then the AEMotionController creates the overall animation by blending the operator trees of the motion models together (cf.[6]). It is also responsible for resolving conflicts between motion models which want to use the same parts of the body.

5 Current State and Future Research

The animation engine and the choreography editor have been integrated in the augmented reality system. The user can place commands on the time line by moving the current time on the time line and choosing some commands from a graphical menu. Currently we have implemented commands like *pathwalking* (let the character walk along a path), *wave* or *walkstyle*, where within the latter we can choose between different styles of walking. The first impression is that our approach for scripting character animations works and is intuitive for the user. Within a minute a character is created walking along a path, starting to wave and changing its walk style at user-defined time stamps. Currently we work on improving the possibilities to edit the choreography and enlarge the animation and the interaction possibilities of the character within the virtual environment.

References

1. Torre, R., Fua, P., Balcisoy, S., Ponder, M., Thalmann, D.: Augmented reality for real and virtual humans. In: CGI 2000, IEEE Computer Society Press (2000) 303 – 308
2. Tamura, H.: Real-time interaction in mixed reality space: Entertaining real and virtual worlds. In: Proc. Imagina 2000. (2000)
3. Brooks, R.A.: Intelligence without representation. Artificial Intelligence **47** (1991) 139–159
4. Grassia, F.S.: Believable Automatically Synthesized Motion by Knowledge-Enhanced Motion Transformation. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh (2000)
5. Perlin, K., Goldberg, A.: Improv: A system for scripting interactive actors in virtual worlds. Computer Graphics **30** (1996) 205–218
6. Grünvogel, S.M.: Dynamic character animation. International Journal of Intelligent Games & Simulation **2** (2003) 11–19