An Animation Framework for Continuous Interaction with Reactive Virtual Humans

H. van Welbergen, D. Reidsma, J. Zwiers, Zs. Ruttkay, and M. ter Maat
Human Media Interaction - University of Twente
P.O. Box 217, 7500 AE Enschede
{welberge|dennisr|zwiers|ruttkay|maatm}@ewi.utwente.nl

Abstract

We present a complete framework for animation of Reactive Virtual Humans that offers a mixed animation paradigm: control of different body parts switches between keyframe animation, procedural animation and physical simulation, depending on the requirements of the moment. This framework implements novel techniques to support *real-time continuous interaction*. It is demonstrated on our interactive Virtual Conductor.

Keywords: Virtual Humans, Interactivity, Gesture Generation, Conducting Motion

Figure 1: The Virtual Conductor, Photo: Henk Postma, Stenden Hogeschool

1 Introduction

Virtual Humans (VHs) often interact with users using a combination of speech with gestures in a conversational setting. They tend to be developed using a turn-based interaction paradigm in which the interlocutor and the VH take turns to talk (Thiebaux et al., 2008). If the interaction capabilities of VHs are to become more human-like and they are to function in social settings, their design should shift from this turn-based paradigm to one of *continuous interaction* in which all partners in an interaction perceive each other and express themselves continuously and in parallel (Nijholt et al., 2008). We present in this paper the design and implementation of a framework for building Reactive Virtual Humans (RVHs) that are capable of exhibiting this kind of continuous interaction. Continuous interaction needs an immediate adaptation to external events (in the environment and from the user). This requires re-timing of already planned behavior to match with these events, and re-planning or re-scheduling of the planned behavior on short notice.

In our previous work we have introduced mixed paradigm animation using procedural motion on selected body parts and physical simulation on the remaining body parts (van Welbergen et al., 2009), which allows us to combine the physical integrity of physical simulation with the precision of procedural animation. In this paper we show the applicability of physical simulation for *secondary* motion, that is, motion such as balancing or eye blinking that one wants the VH to display, but does not want to have to specify in detail. We present the implementation of *switches* between physical or kinematic control of motion on different joints, depending on the focus of the animation task at any moment.

We explain the design and implementation of the architecture using our implementation of a Virtual Conductor (Reidsma et al., 2008) that can interactively conduct an ensemble of human musicians, listen to the music they play, and reactively adapt its conducting behavior and the timing thereof when the musicians need to be corrected (See Figure 1).

2 Architecture of Our Animation Framework

We base our architecture (Figure 2) on the SAIBA Framework (Kopp et al., 2006), which contains a three-stage process: *communicative intent planning*, multi modal *behavior planning*, resulting in a BML stream, and *behavior realization* of this stream. Our architecture encompasses the behavior planning and realization stages. A feedback loop between these two stages allows flexible (re)planning of behavior. We zoom in on our implementation for the planning and realization of animation of our system (see (Nijholt et al., 2008) for its initial design).

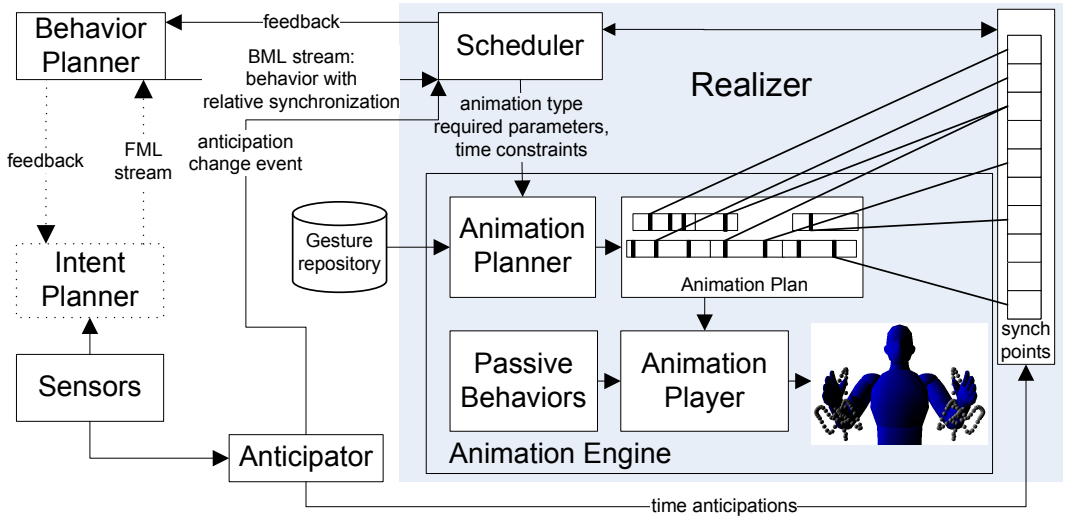

Figure 2: Architecture

The realization stage starts with a specification, in a BML stream, of a set of 'behaviors' on different modalities, synchronized to each other and possibly to predicted/anticipated world events. For example: the BML stream could specify that the timing of conducting gestures is initially determined by *synchronization points* derived from the tempo indications in the music score. Interaction with the world is achieved through *Anticipators*. An Anticipator is a module that takes as input perceptions of events in the real world, extrapolates them into predictions of the timing of future events, and uses these predictions to update the timestamps of synchronization points. The conducting anticipator, for example, sets – and dynamically updates – the exact timing with which the conducting beats should actually be executed, making use of the score of the piece (intended tempo), the detected tempo of the music played by the musicians and knowledge on how to make musicians play on time (Reidsma et al., 2008). BML$^{\mathrm{T}}$, our extension of BML, allows the specification of alignment to events from an Anticipator.

2.1 The Organization of Motion

We organize motion in *motion units*. A motion unit has a predefined function (for instance: a 3-beat conducting gesture, a walk cycle) and acts on a selected set of joints. A set of parameters can be used to adapt the motion unit (for instance: amplitude for the conducting motion unit, or desired pelvis height and joint stiffness for a physical balancing motion unit). Motion units contain one or more *motion phases*, separated by *keys*. Each key is assigned a predefined canonical time value $0 \leq \alpha_i \leq 1$ that indicates where it is located within the motion unit. Given the current set

of parameter values and some canonical time $0 \leq \alpha \leq 1$, a motion unit can be executed, typically by rotating some joints of the RVH.

Procedural motion units rotate joints over time as specified by mathematical expressions that take α as well as a vector $\mathbf{a} \in \Re^n$ as parameters. These expressions can be used to directly steer Euler angles components of joint rotation, to position the root or to position the wrists and ankles using analytical inverse kinematics. The parameter values $\mathbf{a}$ can be changed in real time, changing the motion shape or timing. All mathematical expressions are evaluated using the Java Math Expression Parser.[1] Arbitrary custom function macros can be designed. We defined such macros for hermite splines, TCB splines (Howe et al., 2006) and perlin noise. Our design allows arbitrary mathematical formulas and parameter sets to be used for motion specification and is therefore more flexible than, but still compatible with traditional procedural animations models that define motion in terms of splines or other *predefined* motion formulas and use *fixed* parameter sets (Chi et al., 2000; Howe et al., 2006). In our work, keyframe animation is regarded as a specialized procedural motion unit.

Secondary motion units, such as eye blinking or balancing, are activated activated using the BML stream. Secondary physical motion units are executed by physical controllers. We have implemented several balancing controllers, that offer different trade-offs between balancing stability and movement naturalness (van Welbergen et al., 2009). Our pose controllers loosely keep body parts in their desired position, while still being effected by the forces on the body. Controllers use techniques from control theory to steer the VH's 'muscles' in real time. The input to such a controller is the desired value of the RVH's state, for example desired joint rotations or the desired position of the VH's center of mass (CoM). Such controllers can, to a certain extent, cope with external perturbation and move the body using Newtonian dynamics, taking friction, gravity, and collisions into account.

Transition motion units are used to automatically create movement from the pose at which they are activated to a predefined pose. Currently this is done using a simple slerp interpolation.

2.2 Motion Planning, Re-planning and Execution

The animation planner creates instances of motion units (called *timed motion units*) and inserts them in the *Animation Plan*, as specified by the scheduler. Timed procedural motion units are instantiated from a gesture repository. Secondary motions are enabled and disabled as prescribed by the BML stream.

We infer the continuously changing mix of kinematically and physically steered joints from the active procedural motion units and secondary motions. A *switch* from kinematical to physical control on a body part is implemented by setting up the appropriate physical representation and applying the current velocity and position to the matching body parts in the new physical representation. A switch from the physical to kinematic control simply removes the physical representation of the body part from the physical body of the RVH. For example, when the conductor just indicates the beat, he conducts with his right hand lets the left hand hang down loosely. This is implemented using a physical representation of the lower body and left arm, steered by respectively a balancing controller and a pose controller. The right arm is steered by a procedural conducting animation. To use the left arm for an expressive conducting gesture, we disable the pose controller and plan the expressive gesture as a procedural motion. This automatically executes a switch, removing the physical representation of the left arm from the physical body.

The keys of the timed motion units are linked to the synchronization points in the realizer, as specified in the BML. Synchronization between keys in different timed motion units is achieved by linking them to the same synchronization point.

The Anticipator notifies the *Scheduler* whenever its predictions change, and updates the synchronization points within the Realizer. Many of such updates are minor and do not require a change in the Animation Plan. Since the keys of timed motion units are symbolically linked to

[1] Singular Systems, `http://sourceforge.net/projects/jep/`

the synchronization points, the timing update is handled automatically. More significant prediction updates might require an update of the Animation Plan, which is handled by the Animation Planner and the Scheduler. Such an update typically involves re-timing of behavior on several modalities to generate a more natural behavior execution plan, as suggested in Nijholt et al. (2008). If the Scheduler can not generate a (multi modal) execution plan that satisfies the new time predictions, the Scheduler omits the behaviors that cannot be executed and notifies the Behavior Planner. It is then up to the Behavior Planner to update the behavior plan.

The animation player executes the active timed procedural motion units. The generated motion is then combined with the currently enabled secondary motions, using our system that mixes motion on physically steered body parts with (procedural) motion on kinematically controlled body parts, taking the forces generated by the kinematically steered joints into account (van Welbergen et al., 2009).

3 Results and Discussion

We presented a complete framework for animation of Reactive Virtual Humans that implements novel techniques to support tightly synchronized real-time continuous interaction using a mixed animation paradigm that switches the control of different body parts between procedural animation and physical simulation, depending on the requirements of the moment. The system offers an adjustable balance between ease-of-use and flexibility by allowing motion specification through both high level behavioral primitives and (at the same time) a detailed specification on those aspects for which the user needs it. Some demonstration movies of our mixed paradigm animation and procedural motion system can be found online.[2]

Acknowledgments

This research has been supported by the GATE project, funded by the Dutch Organization for Scientific Research (NWO) and the Dutch ICT Research and Innovation Authority (ICT Regie).

References

Chi, D. M., Costa, M., Zhao, L., and Badler, N. I. (2000). The EMOTE model for effort and shape. In *SIGGRAPH*, pages 173–182, New York, USA. ACM Press/Addison-Wesley Publishing Co.

Howe, N. R., Hartmann, B., Leventon, M. E., Mancini, M., Freeman, W. T., and Pelachaud, C. (2006). Implementing expressive gesture synthesis for embodied conversational agents. In *Gesture in Human-Computer Interaction and Simulation*, volume 3881 of *LNCS*, pages 188–199. Springer.

Kopp, S., Krenn, B., Marsella, S., Marshall, A. N., Pelachaud, C., Pirker, H., Thorisson, K. R., and Vilhjálmsson, H. H. (2006). Towards a common framework for multimodal generation: The behavior markup language. In *IVA'06*, volume 4133 of *LNCS*, pages 205–217. Springer.

Nijholt, A., Reidsma, D., van Welbergen, H., op den Akker, H. J. A., and Ruttkay, Z. M. (2008). Mutually coordinated anticipatory multimodal interaction. In *Nonverbal Features of Human-Human and Human-Machine Interaction*, volume 5042 of *LNCS*, pages 70–89, Berlin. Springer.

Reidsma, D., Nijholt, A., and Bos, P. (2008). Temporal interaction between an artificial orchestra conductor and human musicians. *Computers in Entertainment*, 6(4):1–22.

Thiebaux, M., Marshall, A. N., Marsella, S., and Kallmann, M. (2008). Smartbody: Behavior realization for embodied conversational agents. In *Autonomous Agents and Multiagent Systems*, pages 151–158.

van Welbergen, H., Zwiers, J., and Ruttkay, Z. (2009). Real-time animation using a mix of dynamics and kinematics. *Submitted to Journal of Graphics Tools*.

[2] http://hmi.ewi.utwente.nl/casa09_files/demo_conductor/mixed.html