Developing Educational and Entertaining Virtual Humans using Elckerlyc

D. Reidsma*, H. van Welbergen, R. Paul, B. van de Laar, and A. Nijholt

Human Media Interaction, University of Twente, the Netherlands
dennisr@ewi.utwente.nl http://hmi.ewi.utwente.nl

1 Introduction

Virtual humans (VHs) are used in many educational and entertainment settings: training and serious gaming, interactive information kiosks, tour guides, tutoring, interactive virtual dancers, and much more. Building a complete VH from scratch is a daunting task, and it makes sense to rely on existing platforms. However, when one builds a novel interactive VH application, one needs to be able to adapt and extend the means to control the VH offered by the platform, without reprogramming parts of the platform. This paper describes Elckerlyc, a novel platform for controlling a VH. The focus is on how to easily extend and adapt the system to the needs of a particular application, without programming.

2 The SAIBA Framework

The SAIBA framework [1] provides a good starting point for designing interactive VHs. The framework defines a three-stage process: *communicative intent planning*, multimodal *behavior planning*, resulting in a specification of the form synchronisation of the behavior that the VH should display, and *behavior realization*, to actually perform the behavior using the embodiment of the VH.

When designing a VH application, one should focus on the intent and behavior planning stages: given the goals of the VH and the context in the interaction with a human user, determine the communicative behavior that the VH should display. To ensure that this behavior can be executed transparently by any behavior realization platform, the emerging SAIBA standard Behavior Markup Language (BML) is being developed [1].

BML provides a general, realizer-independent description of multimodal behavior that can be used to (incrementally) control a VH. BML expressions describe the occurrence of certain types of behavior on an abstract level (gaze, speech, point and beat gestures, lexicalized gestures and facial expressions, and other types) as well their synchronisation.

Although this level of abstraction saves a lot of effort when building VH applications, BML Realizers should never completely be black box systems. One

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

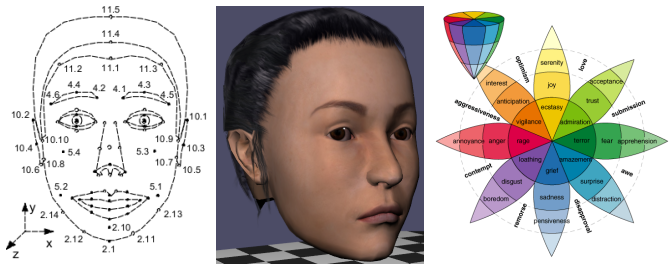

Fig. 1. Face editor: Define, for all MPEG4 control points, the effect on the mesh. Then, obtain subtle face expressions using MPEG4, FACS, or other means.

must still be able to specify for some of these abstract behaviors in exact detail how that behavior is performed – for example, by using a specific piece of mocap for a beat gesture, or a hand crafted animation for certain pointing gestures.

3 Elckerlyc

Elckerlyc is a state-of-the-art BML Realizer [2]. Elsewhere, we described its *mixed dynamics* capabilities, that allow one to combine physics simulation with other types of animation in a flexible way [3], and its focus on *continuous interaction*, allowing last moment modification of behavior plans with respect to content and timing, which makes it very suitable for VH applications requiring high responsiveness to the behavior of the user [4]. Here we will discuss how one can adapt Elckerlyc to suit the needs of a particular application without giving up the level of abstraction offered by its BML Realization interface.

Adding New Avatars. Often, VH applications require a custom-built avatar body. Elckerlyc's import facilities allow one to use standard modelling software to build the avatar body. Only two things are needed to control the resulting embodiment using Elckerlyc. Firstly, a *physical model of the avatar* needs to be created using Elckerlyc's physics editor, if one wants to use mixed dynamics. Secondly, in order to have full control of the avatar's face expressions, one needs to *make the face MPEG4 compliant* using Elckerlyc's face editor. For each of the standard MPEG4 control points of a face (see Figure 1), one needs to define their effect on the mesh. Then, all existing face expressions in Elckerlyc can be displayed on the new VH embodiment. Furthermore, the VH can then be controlled using BML Face Behaviors that apply (1) direct MPEG4 control of the face mesh, (2) the FACS Action Units of Ekman et al., which provide a more muscle-like approach, and (3) the basic emotion profiles of [5] using the method of [6].

Extending the Animation Repertoire. Elckerlyc can use motion units (animations) from any number of paradigms, such as motion capture, hand crafted animation, and physical simulation. Each of these paradigms can be seen as providing joint rotations over time, between $0 < t < 1$, with 0 and 1 being the start

and end of the animation. For each of the above paradigms, a file format is available from which Elckerlyc can read the specification of new motion units. For facial animation units, the same holds, but instead of joint rotations, a face unit provides displacements of MPEG4 control points over time, between $0 < t < 1$.

Mapping new motion units to specific BML behaviors. After adding new motion units for a specific VH application, the BML Realizer needs to know how to map BML expressions to them. This is achieved through two resource files: the gesture binding and the face binding. Each defines a mapping from BML behavior types to motion units (or face units) in one of the available paradigms, constrained by the value of certain parameters. For example, a <head> behavior is mapped to the procedural animation "nod.xml" when it satisfies the constraints action="ROTATION" and rotation="NOD", and another to motion unit when rotation="SHAKE" (see Figure 2).

```
<bml><head id="nod1" repeats="3" action="ROTATION" rotation="NOD"/></bml>
----
<gesturebinding>
 <MotionUnitSpec type="head">
  <constraints>
   <constraint name="action" value="ROTATION"/>
   <constraint name="rotation" value="NOD"/>
  </constraints>
  <parametermap>
   <parameter src="repeats" dst="r"/>
  </parametermap>
  <MotionUnit type="ProcAnimation" file="procanimation/smartbody/nod.xml"/>
 </MotionUnitSpec>

 <MotionUnitSpec type="head">
  <constraints>
   <constraint name="action" value="ROTATION"/>
   <constraint name="rotation" value="SHAKE"/>
  </constraints>
    ...
```

Fig. 2. Gesture binding: link BML behavior types to newly defined animations using the gesture binding, mapping BML parameters to parameters of the motion unit.

4 Conclusion

We globally described how one can use the combination of BML and Elckerlyc to control the behavior of a virtual human, leaving one free to focus on the *communicative intent planning* of the VH rather than the realization of the

movements, and we discussed how to adapt and extend Elckerlyc in a very easy way by simply adding new resources.

References

1. Vilhjálmsson, H., *et al.*: The behavior markup language: Recent developments and challenges. In: Intelligent Virtual Agents. LNCS, Springer (2007) 99–120
2. van Welbergen, H., Reidsma, D., Ruttkay, Z.M., Zwiers, J.: Elckerlyc: A BML realizer for continuous, multimodal interaction with a virtual human. To Appear in Journal on Multimodal User Interfaces (2010)
3. van Welbergen, H., Zwiers, J., Ruttkay, Z.M.: Real-time animation using a mix of dynamics and kinematics. Journal of Graphics Tools (2010, to appear)
4. van Welbergen, H., Reidsma, D., Zwiers, J.: A demonstration of continuous interaction with elckerlyc. In: Proc. Multimodal Output Generation. (2010)
5. Plutchik, R.: Emotion: A Psychoevolutionary Synthesis. Harper and Row, New York (1980)
6. Raouzaiou, A., Tsapatsoulis, N., Karpouzis, K., Kollias, S.: Parameterized facial expression synthesis based on MPEG-4. Eurasip Journal on Applied Signal Processing **10** (2002) 1021–1038