Phase-based Gesture Motion Parametrization and Transitions for Conversational Agents with MPML3D

Klaus Brügmann
National Institute of Informatics
2-1-2 Hitotsubashi
Chiyoda-ku
Tokyo 101-8430, Japan
2-1-2 Hitotsubashi
mail@klausbruegmann.de

Hannes Dohrn
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Computer Graphics Group
Am Weichselgarten 9
91058 Erlangen, Germany
Computer Graphics Group
hannes.dohrn@gmx.de

Helmut Prendinger
National Institute of Informatics
2-1-2 Hitotsubashi
Chiyoda-ku
Tokyo 101-8430, Japan
2-1-2 Hitotsubashi
helmut@nii.ac.jp

Marc Stamminger
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Computer Graphics Group
Am Weichselgarten 9
91058 Erlangen, Germany
Computer Graphics Group
stamminger@cs.fau.de

Mitsuru Ishizuka
Graduate School of Information Science and Technology
University of Tokyo
7-3-1 Hongo
Bunkyo-ku
Tokyo 113-8656, Japan
ishizuka@i.u-tokyo.ac.jp

ABSTRACT

We present a method to produce smooth transitions between arbitrary pieces of character animation, which is based on the application of dynamic transition curves. Unlike other approaches, we achieve anytime interruptibility for body expressions, that is, gestures can be changed anytime during execution while maintaining naturalness of motion transition. To obtain highly natural skeletal movement, our approach is integrated with motion parametrization, as proposed in the "Verbs and Adverbs" technique [18], and further methods of fuzzy motion blending. We will demonstrate how the latest version of the Multimodal Presentation Markup Language (MPML3D) integrates parameterized agent behavior, and can support the incorporation of personality and emotional attentiveness in a straightforward way.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Animations, Artificial, augmented, and virtual realities*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Languages and structures*

General Terms

Design, Algorithms, Languages

1. INTRODUCTION

Using virtual agents in interactive applications imposes the need to satisfy users' expectations about the graphical quality of animation. In the past, cartoon-style agents were common due to low production cost. However, as high quality graphical interfaces and media footage are more affordable in recent years, a trend towards more realism has become apparent [10].

Display of emotion and personality is an important feature of conversational interface agents, since it raises the user's acceptance of a synthetic character as a communication partner [17]. Various projects have been investigating usage patterns for co-verbal gesture behavior and their impact on conversation outcome [8]. One task is to automatically determine which conversational gestures accompany speech [7]. Often conversational behavior is enhanced by parametrization for attenuation and variation in expression style. For example, the EMOTE system [1] proposes a movement style representation that adapts a subset of the LMA method (Laban Movement Analysis), by which inner states are associated with physical movement parameters. As pointed out in [6], the majority of co-verbal gestures are subconscious movements. Those movements also depend on emotional state [6], and thus might change in style and intensity whilst performing. Depending on the speakers' changing intention or other external events, gestures might also be interrupted or merge into other movements at any point in time.

Complementary to high quality graphical output, authoring languages for controlling agent behavior on higher levels of abstraction are needed. Various languages have al-

ready emerged that can represent and synchronize verbal and co-verbal communicative behavior of one virtual agent, including APML [5] and CML [2]. BML [11] is a recent initiative that targets at defining a standard for describing multi-modal behavior for human-like agents. As modeling the behavior of *multiple agents* demands a wider range of constructs, languages like ABL [13] have been developed. However, ABL require authors to possess a high programming skill level. By contrast, MPML has evolved as a simple but powerful tool for modeling interactive plot including overall scene setup, multimodal compound agent behaviors and user feedback channels [10][4]. MPML provides an easy-to-use XML based interface to edit interactive presentation content. The latest version, MPML3D [14], is suitable to be applied to realistic 3D character models, as it provides action synchronization on a fine-grained level. Since MPML3D relies on a reactive framework, it offers anytime responsiveness to scene events, which is a crucial feature for the synthesis of realistic conversational behavior.

The remainder of this paper is organized as follows. Section 2 shows how the latest version of MPML provides scripting of parameterized behavior controlled by character states. Section 3 describes our approach to realize parametrization and anytime transitions for gesture, and addresses further measures to increase motion naturalness. Section 4 concludes the paper.

2. MULTIMODAL PRESENTATION MARKUP LANGUAGE (MPML)

MPML is an authoring language that addresses content modeling for various agent-based scenarios [10]. MPML3D refers to the reactive version of MPML that focusses on the control of 3D agents [14]. Here we discuss the new features of MPML3D that are relevant for gesture parametrization. For a complete language specification and an introduction to MPML3D-authoring please refer to [9].

2.1 Action Parameters

MPML3D action tags now can include any number of behavior modulating parameters. For any given application the number and semantics of those parameters have to be clearly defined. For our presentation agents, there is a set of co-verbal gestures defined along with possible parameters. E.g. a propositional gesture depicting a certain size (Fig. 1) can be defined, which accepts the actual size to be shown as an action parameter as follows.

```
<Action>
    yuki.gesture("show_size", 0.55)
</Action>
```

The script starts with a listing of all scene entities. An entity description consists of the entities type, it's name (used for identification within the document), and other attributes like positioning information and entity state parameters.

```
<Scene name="SingleAgentSetting">
    <Entity type="human" name="yuki">
        <Resource type="model">Girl</Resource>
        <Property name="voice">Susan</Property>
        <Property name="position">
            70.0, 4.5, -20.0
        </Property>
    </Entity>
</Scene>
```

Figure 1: Propositional gesture: Showing a size

2.2 Agent State Parameters

Authors may equip scene entities with arbitrary state variables that can be addressed at runtime to support decisions concerning the scene flow, or to be used directly as parameters for actual actions. By employing this feature, any kind of content relevant properties such as personality traits, agent moods, or more discrete control mechanisms (counters), can be implemented easily.

The following example defines an entity of type "human being" equipped with state parameters "extraversion" and "arousal", which are constrained and initialized to certain values.

```
<Entity type="human" name="yuki">
    <StateParameter name="extraversion" type="float"
        min="0.0" max="1.0" default="0.6"/>
    <StateParameter name="arousal" type="float"
        min="0.0" max="1.0" default="0.3"/>
</Entity>
```

2.3 Using State Parameters

The advantage of *agent state parameters* is that they can be used as input for action parameters, and thus implement the connection between an agent's internal state and the way it performs actions. A frequently used gestures in human conversation is the so-called "beat" gesture [7]. A beat is typically a stroke with the hand. Although it does not carry any propositional content, it serves as a means to give emphasis to certain words in an uttered sentence or express the speaker's attitude. Beat gestures are aligned to passages of a sentence, if not to single words or even syllables. MPML3D supports the alignment between speech and gestures by sub-action synchronization. The style of the beat gesture may vary continuously with the flow of speech, e.g. increase or decrease in intensity. MPML3D allows authors to either control it's style directly (as described above) or to modulate it by agent state parameters.

The following example shows how the parameter "volume" of the "gesture" type action called "beat" is bound to the agent state parameter "extraversion". The "mapping"-attribute thereby specifies a polynomial used as a mapping function. This is particularly useful, when adverbs from different gestures or even different kinds of actions, which all interpret the value in their own way, are bound to the same parameter. Besides that, the embedded scripting functionality provides a more flexible usage of state parameters.

```
<StateParameter name="extraversion" type="float"
    min="0.0" max="1.0" default="0.5">
    <BindParameter type="gesture" name="beat"
        slot="volume"
        mapping="5, 4"/>
</StateParameter>
```

We use this technique to modulate several other metaphoric gestures besides the beat gestures. Furthermore, agent states can be modified via MPML3D commands during presentation flow.

```
<Action>
    yuki.setStateParameter("arousal", 0.8, 1.0)
</Action>
```

Here the action tag will change the state "arousal" of agent "yuki" to the value of 0.8 within 1.0 seconds. Since parameter state changes are actions themselves, they can happen in parallel to other dependent actions. In this way, an author can adjust the style of body motion while it is performed.

3. TECHNICAL REALIZATION

The particular features of MPML3D pointed out above require an animation engine that provides continuous variation of motion and that responds immediately to gesture rescheduling. Besides and corresponding to MPML3D's ease of use, adopting application dependent animation footage should demand little expert knowledge. The following sections show how such requirements can be met by an ECA animation engine. A key concept thereby are motion phases, whose utilization is twofold, addressing both motion parametrization and transitions. For a visual presentation of the salient features please consider watching the demonstration video ([3]) available for download at the project's homepage.

3.1 Gesture Parametrization

We introduce motion parametrization adapting an influential approach by Rose et al [18], "Verbs and Adverbs" (V&Adv). V&Adv is a method to generate new motions for virtual characters from existing ones on a frame-by-frame interactive basis. The technique requires that motion variants are similar in anatomy. However, it allows for samples with differing keyframe timing as well as differing overall duration. Unlike other approaches ([16]) it does not rely on the frequency domain. It is based directly on keyframes in the time-domain and thus facilitates processing of non-periodic motions, which nearly all conversational gestures are.

Typically a motion is to be modified according to a set of dimensions of emotional expressivity, the nature of which depends both on the application and the individual gesture. A multipurpose motion like a 'beat' might require a large set of dimensions, whereas the iconic gesture showing a 'V' for victory, being implicitly associated with certain emotions (joy, arousal) might provide only an intensity parameter for example.

According to V&Adv, we approach this issue with spanning a multidimensional expression space for each motion type (the 'Verbs') with each dimension referring to one attribute to be parameterized (the 'Adverbs'). These attributes refer to MPML3D action parameters (refer section 2.1) and can be associated arbitrarily with application-dependent motion characteristics. The expression space is populated with sample motions, each defining the motion style at some location. At each frame during performance, the agent's state yields the current animation's adverb parameters. In order to produce the final animation, the sample motions are blended according to the agent's parameter setting.

3.1.1 Blending Samples

An advantage of that approach is that with an appropriate interpolation technique, animators can freely choose number and location of samples for each motion, and locally refine the population where required. We choose a variation of Shepard's approach [19] for smoothly interpolating scattered data points. The algorithm yields a weight for each data point in the vicinity of the interpolated position P. We take the inverse distance as the main criterion, but also consider shadowing effects, which regard data point pairs that lie in similar directions as seen from P. As we use the scheme for blending motion samples, issues addressed by the original algorithm like computational error reduction or slope are of minor importance for our application. Our approach therefore focusses on distance and shadowing considerations. Non-normalized sample weights w_i are generated by equation (1)

$$w_i = (1/d_i)^2 * (1 + t_i) \tag{1}$$

$$t_i = [\sum_{D_j \epsilon C} \frac{1}{d_j} * [1 - \cos(D_i P D_j)]]/[\sum_{D_j \epsilon C} \frac{1}{d_j}] \tag{2}$$

, where t_i are the direction dependent weighting factors for each data point D_i and d_i are their respective distances to P. The t_i are determined using the angle between the considered data point (index i) and the other data points (index j), C being the set containing all data points. As we use euclidian distance and angles are computed via dot product, this variant is easily incarnated for any number of expression space dimensions required by the content creator.

By producing zero weights for sample distances above a dynamic distance threshold, sample motions gain only local control on the expression space, such that with equally spaced sample positions the algorithm evaluates zero weights for most of the sample motions and only few source animations have to be evaluated at each frame.

3.1.2 Time Normalization

To preserve semantically essential motion details over blending, all samples contributing to a motion must be structurally similar [18]. To give an example, all variations for a hand-waving gesture must have the same number of actual waves and start at the same side. However, to leave freedom in temporal articulation to the animators, no constraints regarding timing should be imposed upon the motion samples. This is solved by defining motion phases. Those are segmenting the motion of a each gesture into meaningful parts. For example, a beat gesture might consist of a "preparing", a "downstroke", a "rebound" and a "return" phase (figure 2). While the motion phases are the same for all samples, the temporal profiles may vary. When blending the samples it is essential that each one contributes with the same semantical part of it's motion. We therefore compute a normalized time index t_n similar to V&Adv (equation (3)). This time index is derived from the interpolated phase profile (and time index t_b) and determines the current time index for the sample

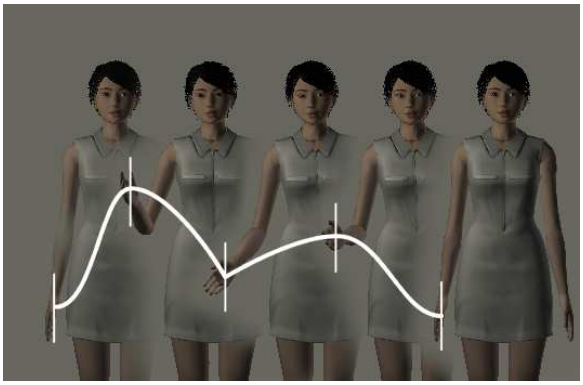

Figure 2: Possible phase profile of a beat gesture

motions t_i individually (equation (4)).

$$t_n(t_b) = \frac{((m) + \frac{t_b - s_{m_b}}{e_{m_b} - s_{m_b}})}{n} \tag{3}$$

$$\begin{aligned} t_i(t_n) &= s_{mi} + f * (e_{mi} - s_{mi}) \\ m &= \lfloor t_n * n \rfloor \\ f &= (t_n * n) - m \end{aligned} \tag{4}$$

In both equations m is the index of the current phase, s_{mi} and e_{mi} are the starting resp. ending times of that phase and n is the total number of phases.

3.1.3 Introducing New Samples or Parameters

Using this approach, specifying new gesture motions or refining the parametrization for existing ones requires only few steps. Introducing a new sample motion for an existing gesture is done by creating the desired animation footage and equipping it with phase annotation. This is done by specifying phase-boundary times (section 3.1.1) and setting interruptibility flags for each phase (section 3.2.6). Further, the sample has to be positioned by specifying an n-vector, with n being the number of dimensions of the expression space. The new sample will thus be considered when performing the motion and structurally refine the blended motion in the respective region.

Introducing new parameters to an existing motion means increasing the dimensionality of the expression space. When doing so, all that has to be done is to extend the position vector of each sample with the new coordinate. Of course, to make the new characteristics addressable by the control-framework, it must be given an (meaningful) identifier (such as "horizontal extend" or "intensity"). The semantics of the new parameter is usually what the author needs to be available for some new feature of the application.

Last, to introduce a whole new gesture motion, a non-empty set of samples and a (maybe empty) set of expressivity dimensions have to be specified. An author can freely define the phase profile for the motion, dependent on the motion details she needs to remain temporally aligned.

3.2 Gesture Interruptibility and Transitions

3.2.1 Desired Features

An aim of this approach is to provide anytime interruptibility for gesture motion. Unlike walking movements, which involve balancing requirements and thus in reality cannot be interrupted at any point, conversational gesture can. Gesture motion is generated along with and aligned to spoken words, both of which are driven by a continuously updated decision machine, the human mind.

To successfully simulate life-like gesture for virtual agents, any evolving motion has to be interruptible and transitable to other motions, according to changing "intention" of the agent. When interrupted, the intermediate motion produced must be smooth and maintain it's naturalness. An interrupted gesture should lose the impression of dedication immediately but not break in physical terms. Reflecting the change in the actor's mind, the motion must adapt the newly targeted gesture and create a transitional movement. The initial situation then is always two arbitrary configurations of the figure's skeleton for which a transition is needed (referred to as the "junction points" in the following). A configuration thereby means both the value and the velocity (1st order derivative) of the motion functions.

When synthesizing skeleton animation (in contrast to manual authoring by an artist) issues like inter-limb collisions and adherence to joint angle constraints have to be addressed. This constitutes a problem for researchers, because constraints are rarely included in keyframed animation footage and collisions are not trivial to handle with complex skeletons like the human one. However, for the majority of movements apparent in conversational body expressions, those issues are of little severity as the motion is performed mainly by arms and hands and naturally takes place in front of the body or, more precisely, in the shared space between the interlocutors. Also, interaction with the environment is not a major feature. Therefore we advocate a straightforward approach by applying dynamic transition curves for each degree of freedom (DOF).

3.2.2 Dynamic Transition Curves

A forward kinematics animation is given by motion functions addressing each DOF independently. A smooth transition between two skeleton configurations can be created by fitting a piecewise function that adheres first order derivatives at the junction points for each DOF. We choose Bezier curves of grade 3 as they combine C1-continuity at both end points with low computational cost. Transition functions may be fit in from any point of a motion - with some restrictions justified by natural circumstances (section 3.2.6). We therefore realize anytime interruptibility of co-verbal gesture.

3.2.3 Transition Timing

How much time should a transition take? For producing smooth motions, any time is applicable, however naturalness and emotive expression strongly depend on a transition's duration. As our control framework schedules motions as soon as they are to be performed, we assume that the speaker wants to perform the follow up gesture as soon as possible. So the transition duration should reflect the time required to reposition the limbs to start the new movement. We compute the transition duration dur_i for each DOF separately by averaging the distances of control polygon vertices b_k of the transition Bezier curve, as given in equation (5). We thus consider the distance to be bridged as well as the velocity at the junction points. To account for mass inertia, the length of the limb l_i is also considered. The largest dur_i

finally determines the actual transition time dur_{max}.

$$dur_i = \frac{\sum_{k=0}^{3} |b_{(k+1)mod3} - b_k|}{2} * l_i \quad (5)$$

To avoid angle constraint violation, this technique exploits the fact that those constraint intervals are implicitly given by the animation footage: For conversational gesture movements, if two configurations are compliant to those boundaries, a linear interpolation between them will also be. We however assume the footage does to not contain (unnatural) rapid motions towards the constraint intervals, for which the smooth blending might result in violation of the constraints.

3.2.4 Compound Curves

As all DOF-functions must reach the target configuration at the same time, most of the optimized curve shapes resulting from dur_i will be lost. To compensate for that we use blended transition curves. Those are pairs of piecewise functions (tr_0 and tr_1) each of which approximates the optimized Bezier curve shape near one junction point (C_S and C_E) for arbitrary durations. Equation 6 shows the formula of tr_0 for some DOF i, $F(t/f_i)$ being the Bezier part with two control points aligned to the linearly interpolating function $L_i(t)$ (equation 7), as shown in figure 3. The effective transition then is given by again linearly interpolating between tr_0 and tr_1.

$$tr_0(t) = \begin{cases} F(t/f_i) & \text{if } t < f_i, f_i = \frac{dur_i}{dur_{max}} \\ \text{L(t)} & \text{else} \end{cases} \quad (6)$$

$$L(t) = (1-t) * C_S + t * C_E \quad (7)$$

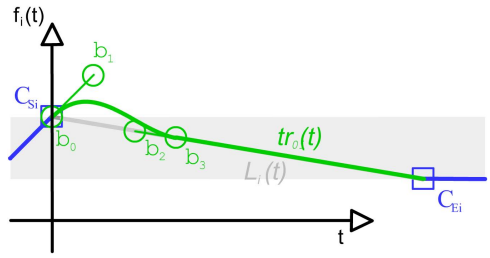

Figure 3: Piecewise transition curve function tr_0

3.2.5 Rotational DOF

With the human skeleton and forward kinematics, most of the DOF regard rotations. Using quaternion representation, each rotational DOF-function issues 4 values that cannot be processed independently as cartesian vector coordinates can. To compute transition curves for rotational DOF, we use quaternion-control points as shown by Shoemake [20]. As linear interpolation between quaternions does not produce linear angular changes, quaternion Bezier curves are evaluated by combining the DeCasteljau algorithm with the "Slerp" operator ([20]). Distance between orientations is given by the angle and scaling a rotation is done by scaling the rotation angle. Rotational derivatives are represented by quaternions as well.

3.2.6 Transitions and Phases

Our approach relies on phases to identify the essential part of a motion, per default all but the first an the last one. Those two phases are movements from and to the normal pose and are only used when a transition partner is missing. A conventional transition therefore will connect from the end of the next to last phase to the start of the subsequent motion's second phase. Similarly, in case of an interruption, the first phase of the follow-up gesture is skipped.

The problems described in section 3.2.1 we address with phases-based transition inhibition. Each motion may declare so-called "vital" phases, i.e. phases in which the skeleton is in a twisted configuration such that the transition curve approach would produce unnatural configurations. The engine then will delay a transition's begin until the end of that phase (and vital phases immediately following). For example, a posture showing the arms fold will define a vital phase lasting all the time that the arms wind around each other. An interruption during a vital phase will therefore not have immediate effect. This is a justified restriction to anytime interruptibility as it only reflects natural circumstances: A real human as well has to unwind his arms properly and before moving them freely again.

For further improving vital phase interruptions, a hold flag can be set for any phase, indicating that there is no actual movement taking place in it. Although the time that passes during a hold is semantically essential, it is not desirable to wait for it to pass even if that phase is declared vital. Hold phases have the property to have identical skeleton configuration an the beginning at the end regarding 0th and 1st order derivative. Thus, any hold phase can be skipped without destroying motion function's C1-continuity (figure 4). When a vital phase is interrupted, we perform all phases until the begin of the next non-vital phase but skip all phases that have the hold flag set. If the interrupted phase itself is a vital hold phase, the time index is proceeded to it's end immediately.

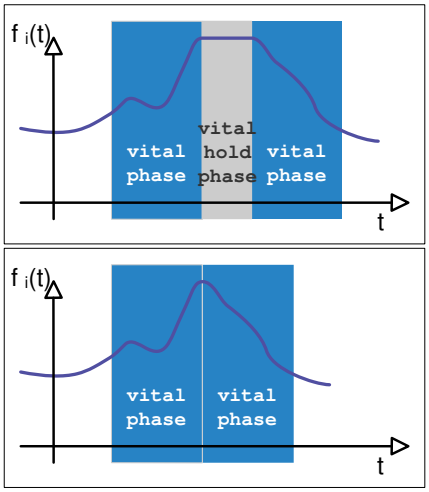

Figure 4: Skipping of vital hold phases

The motion phase annotation scheme presented here is highly adaptable. Possible extensions are repeatable phases, author-defined entry phases or hold phases with author-specified duration.

3.3 Secondary Motion

Even in static postures there is always a subtle motion of balance. In computer animation, as pointed out in [12], ignoring that kind of motion leads to an effect called "moving hold", i.e. the animated figure seems to freeze for a moment. To avoid this effect, fuzzy motion has successfully been applied by engaging noise functions [15]. Instead of mixing gesture with random motion, we introduce a secondary motion overlay, where the transformation of multiple animations is accumulated in each joint. Choosing only subtle movement like breathing and swaying, we avoid the moving hold effect while retaining more control compared to noise-based approaches.

4. CONCLUSION

We demonstrated how MPML3D supports current needs for control and quality of character animation. We described it's easy-to-use interface to model expressive behavior driven by dynamic agent- and action-specific parameters.Complementing the control interface features we presented a phase-based scheme for highly natural motion synthesis in real-time, providing the author with the non-restrictive yet intuitive expression space modeling paradigm. The salient features include parametrization of arbitrary dimensionality and anytime transitions, using traditionally key-framed animation footage with only minimal annotation requirements. The presented technique is applicable to any engine working on animation data applied to a joint hierarchy that typically resembles a creature's skeleton. Special focus is on animation of arms and hands which are the most significant limbs of communicative body expressions.

5. ACKNOWLEDGEMENTS

The research was supported by the Research Grant (FY1999–FY2003) for the Future Program of the Japan Society for the Promotion of Science (JSPS), by a JSPS Encouragement of Young Scientists Grant (FY2005–FY2007), and an NII Joint Research Grant with the Univ. of Tokyo (FY2006). The first author was partly supported by the JSPS Grant and a German Academic Exchange Service (DAAD) Scholarship. The second author was supported by the JSPS Grant.

6. REFERENCES

[1] J. Allbeck and N. Badler. Representing and parameterizing agent behaviors. In Prendinger and Ishizuka [17], pages 19–38.

[2] Y. Arafa, K. Kamyab, and E. Mamdani. Towards a unified scripting language. Lessons learned from developing CML & AML. In Prendinger and Ishizuka [17], pages 39–63.

[3] K. Brügmann. Graceful anytime interruptibility for virtual agents - demonstration video, 2007.

[4] K. Brügmann, H. Prendinger, M. Stamminger, and M. Ishizuka. Graceful anytime interruptibility for virtual agents. In *Proceedings ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE-07)*, pages 284–285. ACM Press, 2007.

[5] B. D. Carolis, C. Pelauchaud, I. Poggi, and M. Steedman. APML: Mark-up language for communicative character expressions. In Prendinger and Ishizuka [17], pages 65–85.

[6] J. Cassell. Nudge nudge wink wink: Elements of face-to-face conversation for embodied conversational agents. In J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, editors, *Embodied Conversational Agents*, pages 1–27. The MIT Press, Cambridge, MA, 2000.

[7] J. Cassell, H. Vilhjálmsson, and T. Bickmore. BEAT: the Behavior Expression Animation Toolkit. In *Proceedings of SIGGRAPH-01*, pages 477–486, 2001.

[8] N. E. Chafai, C. Pelachaud, D. Pele, and G. Breton. Gesture expressivity modulations in an ECA application. In *Proceedings 6th International Conference on Intelligent Virtual Agents (IVA-06)*, Springer LNAI 4133, pages 181–192, 2006.

[9] H. Dohrn and K. Brügmann. The mpml3d reference manual. http://research.nii.ac.jp /~prendinger/MPML3D/MPML3D.html, 2007.

[10] M. Ishizuka and H. Prendinger. Describing and generating multimodal contents featuring affective lifelike agents with MPML. *New Generation Computing*, 24:97–128, 2006.

[11] S. Kopp, B. Krenn, S. Marsella, A. Marshall, C. Pelachaud, H. Pirker, K. Thórisson, and H. Vilhjálmsson. Towards a common framework for multimodal generation: the Behavior Markup Language. In *Proceedings 6th International Conference on Intelligent Virtual Agents (IVA-06)*, Springer LNAI 4133, pages 205–217, 2006.

[12] J. Lasseter. Tricks to animating characters with a computer. *SIGGRAPH Computer Graphics*, 35(2):45–47, 2001.

[13] M. Mateas and A. Stern. A Behavior Language: Joint action and behavioral idioms. In Prendinger and Ishizuka [17], pages 19–38.

[14] M. Nischt, H. Prendinger, E. André, and M. Ishizuka. MPML3D: a reactive framework for the Multimodal Presentation Markup Language. In *Proceedings 6th International Conference on Intelligent Virtual Agents (IVA-06)*, Springer LNAI 4133, pages 218–229, 2006.

[15] K. Perlin. An image synthesizer. In *International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH-85)*, pages 287–296. ACM Press, 1985.

[16] K. Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):5–15, 1995.

[17] H. Prendinger and M. Ishizuka, editors. *Life-Like Characters. Tools, Affective Functions, and Applications*. Cognitive Technologies. Springer Verlag, Berlin Heidelberg, 2004.

[18] C. Rose, B. Bodenheimer, and M. F. Cohen. Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.

[19] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524, New York, NY, USA, 1968. ACM Press.

[20] K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, New York, NY, USA, 1985. ACM Press.