Applying direct manipulation interfaces to customizing player character behaviour

Marco Gillies[1]

[1] Department of Computer Science, University College London,
Malet Place, London WC1E 6BT, UK
m.gillies@cs.ucl.ac.uk
http://www.cs.ucl.ac.uk/staff/m.gillies

Abstract. The ability customize a players avatar (their graphical representation) is one of the most popular features of online games and graphical chat environments. Though customizing appearance is a common ability in most games, creating tools for customizing a character's behaviour is still a difficult problem. We propose a methodology, based on direct manipulation, that allows players to specify the type of behaviour they would like in a given context. This methodology is iterative, with the player performing a number of different customizations in different contexts. Players are also able to continue customizing their character during play, with commands that can have long term and permanent effects.

1 Introduction

Avatars are a vital part of any online game. The graphical representation of a player is the essential element that presents their persona to the rest of the community. Players can develop a deep bond and association with their avatar. For this reason, creators of online games have dedicated a lot of attention to the appearance and animation of avatars. It has also recently been pointed out[29] that allowing avatars some autonomous behaviour can greatly enhance their realism, for example by giving them complex body language that would be too difficult for a player to control in real time. This autonomous behaviour allows the avatar to produce appropriate responses to the behaviour of other players without the player having to control every movement, for example, looking at another player's avatar when they talk. If a player is to truly form a bond with their avatar then they must be able to customize it to create the persona they want to project, this is one of the most popular features of on-line worlds[6]. Current games largely restrict customization to graphical appearance, however, if an avatar is to present a consistent persona it should also be possible to customize their behaviour to make it consistent with their appearance.

Creating user-friendly tools for customizing characters is a challenging problem. When customizing the appearance of a character the player can pretty much see the whole effect of their changes in a single view, maybe having to rotate the view occasionally. However, autonomous behaviour involves responding to different

events in the world and therefore requires the character to respond very differently in different contexts. This means that a player cannot simply judge whether they have created the character they want by quickly looking at a single view, or even a moderately sized sequence of views. What is needed is an iterative process of refinement of a character. We propose a methodology that involves iterative design of a character. Players may design their characters before joining a game by editing their behaviour in a number of different contexts. However, they can also refine the behaviour while playing using real-time customization.

Another problem with customizing behaviour is that autonomous behaviour systems are typically controlled by a large number parameters. The effect of these parameters on behaviour can be complex and, as described above, highly dependent on context. This means that directly editing these parameters can be highly unintuitive for players. To solve this problem we take inspiration from the highly successful "Direct Manipulation" paradigm of human computer interaction. Direct manipulation enables people to interact with software by directly editing the end result rather than the internal parameters that produce this result. Our methodology allows players to directly specify the behaviour that the characters should produce in a given context, while the software infers appropriate parameters. Typically specifying behaviour in a single context underconstrains the values of parameters. This means that players must edit behaviour in a number of different contexts, however, doing so in all possible contexts would be very time consuming, if possible at all, certainly not something that can be required of people playing games in their leisure time. This leads us back to the need for an iterative methodology that allows players to specify just as much as they feel they need at a given time, while allowing them to refine the behaviour at a given time.

2 Related Work

This work builds on a long tradition of character animation. The lower level aspects focus on body animation in which there has been a lot of success with techniques that manipulate pre-existing motion data, for example that of Gleicher[10,20], Lee and Shin[14] or Popović and Witkin[31]. However, we are more interested in higher level aspects of behaviour control. This is a field that brings together artificial intelligence and graphics to simulate character behaviour. Research in this area was started by Reynolds[26] whose work on simulating birds' flocking behaviour has been very influential. Further important contributions include the work of Badler *et al.* on animated humans[1]; Tu and Terzopolous' work on simulating fishes[27]; Blumberg and Galyean's "Silas T. Dog"[3], Perlin and Goldberg's "IMPROV" system[23] and the work of Gratch, Johnson and Marsella[12,19]. We mostly deal with non-verbal communication, which is a major sub-field of behaviour simulation with a long research history including the work of Cassell and her group[4,5,29]; Pelachaud and Poggi[22] and Guye-Vuillème *et al.* [11]. The two types of behaviour we are using are gesture which has been studied by Cassell *et al.*[4] and posture which has been studied by Cassell *et al.* [5] and by Bécheiraz and Thalmann[2].

Vihljàlmsson[28] has applied this type of autonomous non-verbal behaviour to avatars for on-line games.

Most of the work described above deals with the algorithms for simulating behaviour rather than tools for designing behaviour. Of the work on tools, most has focused on using markup languages to specify the behaviour of characters and avatars, for example the APML language[7]. However, though markup languages are an important step towards making it easier to specify behaviour they are a long way from the usability of graphical tools. There have also been tools for designing the content of behaviour, for example designing gestures[11], however, these tools do not address the autonomous aspects, i.e. how to decide which behaviour to perform in a given context. Del Bimbo and Vicario[8] have worked on specifying autonomous behaviour by example, but their work was restricted to vehicles and was not applied to human-like characters. Pyandath and Marsella[25] use a linear inference system to infer parameters of a Partially Observable Markov Decision Process used for multi-agent systems. This inference system is similar to ours, however, they do not discuss user interfaces. In the field of robotics Scerri and Ydrén[21] have produced user friendly tools for specifying robot behaviour. They use a multi-layered approach, with programming tools to design the main sections of the behaviour and graphical tools to customise the behaviour. They were working with soccer playing robots and used a graphical tool based on a coach's tactical diagrams to customise their behaviour. Their multi-layered approach has influenced much of the discussion below. Our own approach to specifying behaviour has been influenced by work on direct manipulation tools for editing other graphical objects, for example the work on free form deformations by Hsu, Hughes and Kaufman[30] and Gain[13].

3 Method Overview

As described in the introduction we are proposing an iterative methodology for customising and refining the behaviour of a character. A player starts by selecting a context and viewing the character's behaviour in this context. They may change this behaviour by selecting from a menu of possible actions (which can be combined and blended together). This choice of behaviour is used to determine a suitable choice of parameters for the character. The parameters must be such that they will produce the chosen behaviour in the given context. It is likely that many different sets of parameters will produce the same behaviour in that one context so the effect of choosing behaviour is not to identify a single set of parameters but to put a number of constraints on the possible parameter values. These constraints are solved using linear programming to produce a set of parameters. After choosing behaviour in a single context the system is likely to be highly under-constrained, and the parameters chosen might not be appropriate in different contexts. The player must, therefore, repeat the processes, putting the character in a number of different contexts. Each time a new behaviour is chosen for a new context more constraints are added and the whole set of constraints are solved for. As the number of contexts is potentially very large the player should not be expected to edit every single context

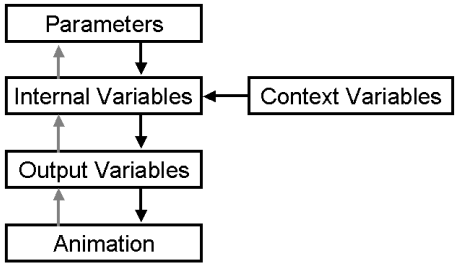

Figure 1. an overview of the behaviour generation process. The black arrows show the behaviour generation process and the grey arrows show the inference process that determines parameters from animation.

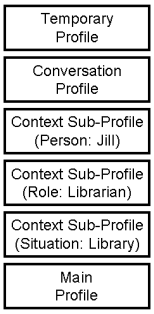

Figure 2. **The profiles stack containing a number of loaded contextual profiles**

or to completely finish the process of editing their character before starting play. Instead they should be able to update the character to correct any behaviour they find to be wrong during play. Players can do this by returning to the original customnisation tool after a session of play, but this looses some of the immediacy of correcting a problem as it occurs. We therefore provide a second interface, which allows players to continue customising during play in a way that is integrated with the real time control methods.

4 Behaviour Generation

The Demeanour architecture is used to generate behaviour for our avatars[17,18], figure 1 shows the behaviour generation method. The basic components of the behaviour system are parameters and variables. Parameters vary between characters and are the element that determines the difference in behaviour between different avatars. Examples might be, how friendly an avatar is or how often it nods when listening to another avatar. Variables, on the other hand, changes with different contexts. Some variables, context variables, are determined solely by external contextual factors, for example the behaviour of other characters. Parameters and variables are combined together to create new, internal variables, which represent the current state of the avatar, for example, whether it is angry or whether it likes the avatar it is talking to.

Some variables are used as outputs that driven the animation system. There are two mains ways of combining parameters and variables. The first is by addition and multiplication, which is often used to combine context variables with weighting parameters. Variables and parameters can also be combined by if-then-else rules that set the value of a variable to that of one of two option variables depending on the value of a boolean condition variable:

$$x = x_1 \quad if \quad x_c = a$$
$$= x_2 \quad otherwise$$

The animated behaviour is generated using a set of basic pieces of motion, each of which represents and action such as a gesture. The basic motions are interpolated using a quaternion weighted sum technique similar to Johnson's[15]. Each weight is determined by the value of an output parameter. Many motions can be continuously interpolated, e.g. leaning forward, however, others are more all-or-nothing, for example it makes no sense to cross your arms 50%. Therefore some motions are classed as discrete and can only have weights of 0 or 1. In this case the corresponding variable is thresholded so that values over 0.5 give a weight of 1.

5 Profiles

The behaviour of avatars can be controlled by using profiles[16]. A profile is a set of parameter values that are loaded together. Profiles are used as a means of customising a character, with the profile determining the behaviour of a character. They can also provide contextual variation, with different profiles being loaded in different contexts (see [16] for more details), and for regulating real time interaction, as described in section 6. This means an avatar will have a number of profiles loaded at any given time. They are stored in a stack as shown in figure 2. The base of the stack is always the main profile that contains the context independent customisations of a character, as described in section 5. Above this, a number of context dependent profiles are loaded. At the top are two profiles that are used to store results of user interaction, the temporary and conversation profiles, as described in section 6. When a new context profile is loaded it is added above all the previously loaded profiles in the stack but below the temporary and conversation profiles. Profiles higher up the stack will override profiles lower in the stack, so recently loaded profiles override older ones and user input overrides other profiles.

6 Off-line Customisation

This initial stage of customization will happen before the player starts playing the game through a first off-line editing state. The player can select contexts in which to view their avatars behaviour and then edit that behaviour. The behaviour is edited by selecting from a menu of actions such as gestures and head nodding. Action can either be discrete (you are either doing them or you are not, e.g. crossing your arms) or continuous (you can do them to a greater or lesser degree, e.g. leaning backward). The interface contains buttons which can select discrete actions and sliders to vary the degree of continuous actions. The user interface is shown in figure 3. The user first sets the context for a behaviour, which is itself expressed as

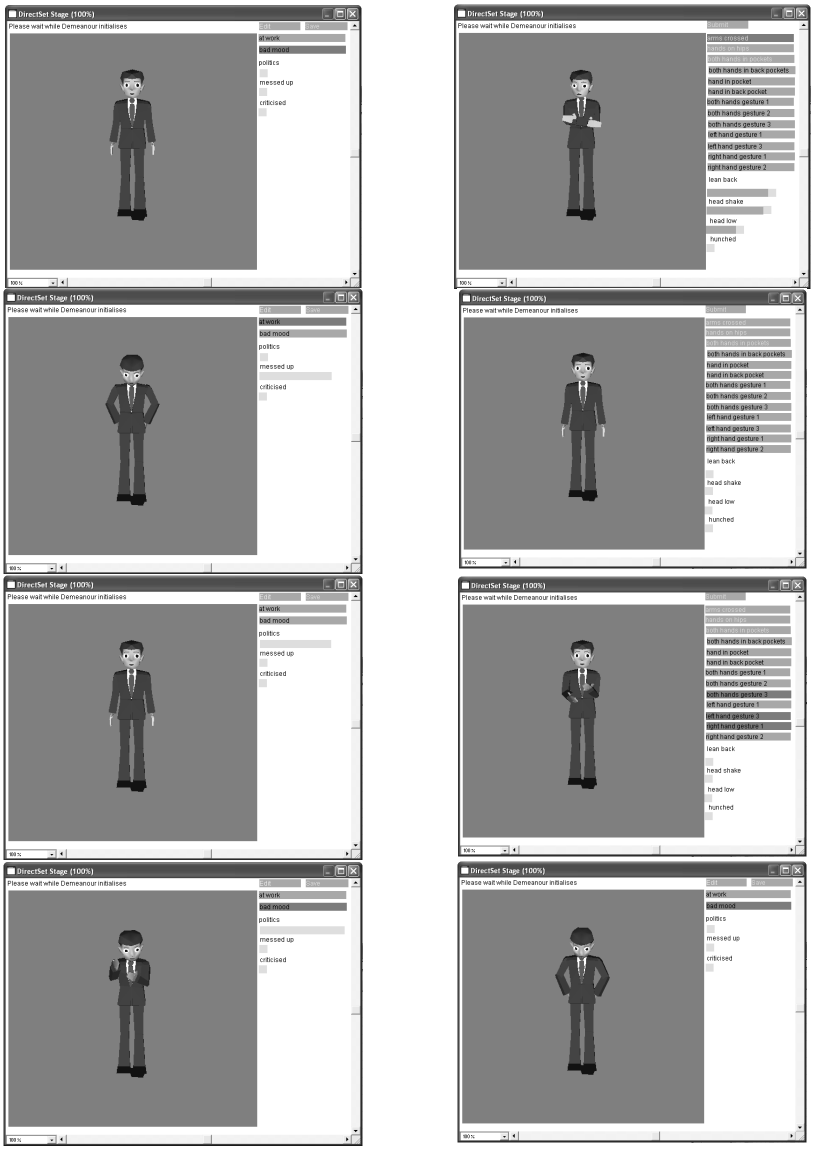

Figure 3. A sequence of edits using the tool from the action based specification example. The the user initially specifies context (in this case that the avatar is in a bad mood). The initial behaviour (image 1, left to right, top to bottom) is neutral as there have been no edits (for clarity, in these examples neutral behaviour is merely a constant rest posture). The user then specifies some hostile behaviour and submits it (2). The system has set the general hostile parameter so the avatar produces hostile behaviour in a new context (3). The user removes this behaviour to specify a neutral context (4), thus reducing the contexts in which hostile behaviour is produced, so in the next context (a political discussion) neutral behaviour is generated (5). The user adds gesturing and submits (6). The final two images show results after these edits, the avatar in a bad mood discussing politics produces both gesturing and hostile behaviour (7). The final image has the same context as the original edit, showing that the same type of behaviour (hostile) is successfully reproduced, but that the exact behaviour is different (8).)

discrete or continuous variables that are edited by buttons and sliders. The user may then view the resulting animation and if they are unhappy with it they may go to an editing screen to change the animation. When they are happy with this they submit the animation. The resulting set of actions is used to generate a number of linear constraints which are then solved for to updated the parameters of the avatar, as described in section 8. The player can then repeat the process by choosing an new context and editing the behaviour if it is not suitable. Figure 3 gives an example of a sequence of edits.

7 Real time customisation

The problem with using off-line customization for something as context dependent as character behaviour is that, after editing, the player cannot be sure that their avatar will behave as they want it too in all context. Even after a relatively long period of customization, they are likely to find incorrect behaviour when they actually use the avatar during play. For this reason we propose that players should also be able to continue customizing their character during play. This means that the initial effort is reduced, allowing players to start playing the game quickly. It also means that customization and correction of errors are situated in play so that players are able to specify the real behaviour they want at that time, rather than specifying behaviour for a hypothetical situation. It is likely to be easier for a user to know what behaviour is appropriate when actually engaged in a conversation than to think about it abstractly during an off-line customisation step.

Demeanour also contains a real-time control system where users can determine the affective state of their character through a number of commands as described in Gillies, Crabtree and Ballin[17]. So as to minimize the effort of controlling a character the control system should be well integrated with the rest of the game controls. As we are looking at conversational behaviour we have produced a control system that is integrated with the type of text-chat interface that is commonly used in on-line games (shown in figure 4). As well as typing the text that they are speaking players can also enter textual commands that control the behaviour of a character. These might be emoticons, e.g. :-) , which can control high level parameters such as the mood of a character, but they can also be direct requests for a particular action, enclosed in asterisks, e.g. *arms crossed*. When a player enters this command their avatar performs the action, but the action is also used to infer appropriate parameter for the character. A set of constraints is generated as described in section 8, these constraints are added to those from previous customizations, and solved for to generate new parameters.

When a player enters a command it not clear how long they intend a change of state to last, for example an increase in friendliness might have a very short scope, just the length of the current utterance or it might indicate a permanent positive attitude to the person being talked to. Demeanour uses character profiles to allow users to choose between four different scopes for a change of state:

- Temporary changes lasting for a limited period, disappearing after a time out.
- Changes lasting for the whole length of the current conversations
- Permanent changes to the attitude toward the conversational partner
- Permanent changes to the character's main profile.

Initially, when a player types a command that changes a parameter value, it is stored in temporary profile. This temporary profile is deleted after a short period of time and all the edits it contains are deleted. Thus the default scope for edits is that they are temporary. However, when a temporary profile is present (i.e. after the user sends a command) a button appears in the text chat interface allowing the user to save the profile. If the user clicks this button the temporary profile is saved into a

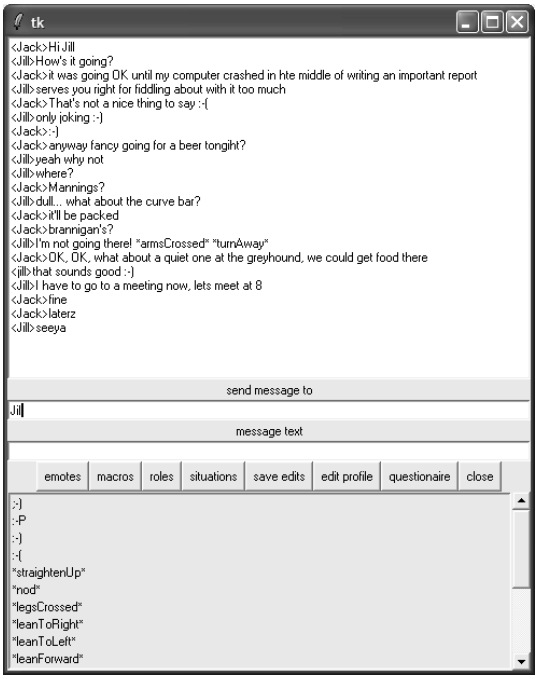

Figure 4 The text chat interface

conversation profile (as shown in figure 2), which has a longer scope lasting for the entire conversation. When temporary edits are saved into the conversation profile they are merged.

At the end of the conversation the user can merge the resulting edits into a permanent profile that is used in all future interactions. This can be to the character's main profile which controls its behaviour and is the chief method of customising a character. Users can also merge the conversation profile into a contextual profile for their conversational partner, thus developing the relationship between the characters during interaction.

8 Inferring Parameters from Behaviour

The main technical requirement for this user interface is the ability to use a number of examples of behaviour to generate constraints which are then solved for a suitable set of parameter values for the avatar's behaviour. To be more exact, each example is a tuple $<a_i,c_i>$ containing a context for behaviour c_i and an animation specified by the user a_i, which is the behaviour of the avatar in that context. The output of the method is a set of parameters. Each example tuple provides a constraint on the possible values of the parameters. We must solve for these constraints using a method that makes it simple to add new constriants, as the editing methods is iterative users will continually be solving and adding new constraints. The method must also be fast enough to solve in real time, if the tools is to be usable. This is simplified by the fact that the parameters and variables are combined together using linear summation, meaning that all relationships between variables, and therefore constriants are linear. This allows us to use Linear Programming[24] to solve for the constriants. Linear programming mimimizes a linear expression subject to to a number of linear equality and inequality constraints:

$$\sum c_i x_i \ \textit{subject to} \quad \sum d_i y_i = 0 \qquad \sum e_i x_i \geq 0$$

where the *x,y,z* are variables and the *c,d,e* are constant coefficients. We form constraints from the characters behaviour and internal parameters as described in the next sections. We then minimize the sum of all parameters values using a simplex linear programming method[24]. This minimization solves for the parameters while keeping their values as low as possible (to avoid extreme behaviour).

8.1 Constraints from Action Specifications

As described in section 6, the action-ased interface allows user to specify the avatar's behaviour using buttons and sliders to give weights for each action (0 or 1 in the case of discrete actions). When a animation is submitted these weights are used to form linear constraints. For a continuous motion the weight of the motion (w_i) should be equal to the corresponding output variable (v_i) so we add the constraint $v_i - w_i = 0$. In the case of discrete actions we are less certain: if the w_i is 0 we know that v_i is less than 0.5, otherwise it is greater, so we add an inequality constraint:

$$v_i - 0.5 \quad \leq 0 \ \textit{if} \quad w_i = 0 \qquad \geq 0 \quad w_i = 1$$

8.2 Constraints from Internal Variables

With this initial set of constraint we then start to form new constraints based on internal variables and parameters. Any variable will depend on other variables and parameters. A variable v_1 depends on another variable v_2 if v_2 is used to calculated

v_1 via addition and multiplication, v_2 is a condition or option in an if-then-else rule that is used to calculate v_1, or v_1 depdends on a third variable that recursively depends on v_2. If a variable only depends on context variables and not parameters it has a constant value in a given context so it is a *known* (k_i) variable in the current constraint. Parameters, and variables that depend on parameters, are *unknowns* (u_i). We must form constraints on all unknowns. We start with the constraints that are given by the animations, each of these contain at least one output variable. Each variable v may take one of 4 forms. If it is a parameter it is an unknown and no further constraints are added. If it is a constraint variable it is a known and has a constant value (this is not allowed for an output variable). If it depends on other variables and parameters by addition and multiplication we add a linear constraint. To ensure that it is soluble we ensure that in each multiplication, only one term is an unknown. Thus the equation for the variable is of the form:

$$v = \sum_i \left(u_i \prod_j k_j \right)$$

We can evaluate all knowns to calculate the coefficients of each unknown and rearrange to get a constraint:

$$\sum_i c_i u_i + c_0 - v = 0$$

If the variable depends on other variables by an if-then-else rule the condition variable is a known so we can evaluate it and know which the option variable v_i that v depends on, we can just add a constraint $v - v_i = 0$. The newly added constraints will have introduced new variables, and we recursively add new constraints for these until we are only left with knowns and parameters, at which point we perform the minimization as described above.

9 Conclusion

This paper has proposed a methodology for customizing avatars in on-line games. This methodology is based on Direct Manipulation in that players choose the concrete behaviour that they want their character to perform in a given context and linear programming is used to infer an appropriate set of parameters for the character from the chosen behaviour. This methodology is readily extensible to both off-line customization, and real-time customization during play. The second feature allows players to gradually adapt their character and to customize their character in a way that is situated in play. We are currently planning a user trial involving long term use of the system to evaluate its effectiveness.

We have used linear programming as it is a fast inference method. This means that it is usable in an interactive interface such as the one we are proposing, and remains usable for our real-time customization method. However, using linear programming does limit our behaviour systems to linear ones, which can limit the complexity of the behaviour produced. With the type of interface we propose, there is always likely to be a trade off between complexity behaviour and speed of infe-

rence, but more work is needed to determine the ideal balance, and therefore an appropriate inference method. More complex machine learning models, such as neural networks or Bayesian networks can give more powerful results at a greater computational cost. An even more powerful method would be to use an arbitrary parameterised algorithm to generate behaviour and then use a numerical optimisation method to determine parameter value. This would be extremely flexible but may well by computationally intractable.

It is also important to compare our method with other interface styles. We have also experimented with a reinforcement learning method in which users do not directly specify behaviour, instead they judge behaviour that is suggested by the system[9]. Our feeling, after initial experimentation, is that the task of judging behaviour is easier for an untrained user than specifying behaviour, but that reinforcement learning involves the user viewing a very large number of behaviours to produce a good result, which may make it impractical. We will conduct user trials to understand the issues better.

Acknowledgements

This work was sponsored by BT plc. I would like to thank the former members of the BT Radical Multimedia Lab for their help and support on this work, in particular Amada Oldroyd (in particular for the use of her avatars), Jon Sutton (for his advice on on-line chat worlds), Daniel Ballin and Barry Crabtree. I would also like to thank the members Virtual Environments and Computer Graphics Group at University College London, and in particular Mel Slater.

References

1. Badler, N., Philips, C., and Webber, B.: Simulating Humans: Computer Graphics, Animation and Control Oxford University Press (1993)
2. Bécheiraz, P. and Thalmann, D.: A Model of Nonverbal Communication and Interpersonal Relationship Between Virtual Actors In: Proceedings of the Computer Animation '96 IEEE Computer Society Press (1996) 58-67
3. Blumberg, B. and Galyean, T.: Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments In: ACM SIGGRAPH (1995) 47-54
4. Cassell, J., Bickmore, T., Campbell, L., Chang, K., Vilhjálmsson, H., and Yan, H.: Embodiment in Conversational Interfaces: Rea In: ACM SIGCHI ACM Press (1999) 520-527
5. Cassell, J., Vilhjálmsson, H. H., and Bickmore, T.: BEAT: the behavior expression animation toolkit In: ACM SIGGRAPH (2001) 477-486
6. Cheng, L., Farnham, S., and Stone, L.: Lessons Learned: Building and Deploying Virtual Environments (2002)
7. DeCarolis, B., Pelachaud, C., Poggi, I., and Steedman, M.: APML, a markup language for believable behaviour generation (2004) 65-87
8. Del Bimbo, Alberto and Vicario, Enrico: Specification by-Example of Virtual Agents' Behavior IEEE transactions on visualtization and Computer Graphics 1:4(1995) 350-360

9. Friedman, D. and Gillies, M.: Teaching Characters How to Use Body Language In: Intelligent Virtual Agents (2005)
10. Gleicher, Michael: Motion Editing with Space Time Constraints In: symposium on interactive 3D graphics (1997) 139-148
11. Guye-Vuilléme, A., T.K.Capin, I.S.Pandzic, Magnenat-Thalmann, N., and D.Thalmann: Non-verbal Communication Interface for Collaborative Virtual Environments The Virtual Reality Journal 4)(1999) 49-59
12. J.Gratch and S.Marsella: Tears and Fears: Modeling emotions and emotional behaviors in synthetic agents In: 5th International Conference on Autonomous Agents (2006)
13. James, Gain: Enhancing spatial deformation for virtual sculpting (2000)
14. Jehee, Lee and Sung, Yong Shin: A Hierarchical Approach to Interactive Motion Editing for Human-like Figures In: ACM SIGGRAPH (1999) 39-48
15. Johnson, M. P.: Exploiting Quaternions to Support Expressive Interactive Character Motion (2003)
16. M.Gillies, I.B.Crabtree, and D.Ballin: Customisation and Context for Expressive Behaviour in the Broadband World BT Technology Journal 22:2(1-4-2004) 7-17
17. Marco, Gillies, Barry, Crabtree, and Daniel, Ballin: Expressive characters and a text chat interface In: Patrick, Olivier and Ruth, Aylett (ed): AISB workshop on Language, Speech and Gesture for Expressive Characters (2004)
18. Marco, Gillies and Daniel, Ballin: Integrating autonomous behavior and user control for believable agents In: Third international joint conference on Autonomous Agents and Multi-Agent Systems (2004)
19. Marsella, S. C., Johnson, W. L., and LaBore, C.: Interactive Pedagogical Drama In: the proceedings of the 4th international Conference on Autonomous Agents (2000) 301-308
20. Michael, Gleicher: Comparing Constraint-Based Motion Editing Methods Graphical Models :63(2001) 107-134
21. Paul, Scerri and Johan, Ydrén: End User Specification of RoboCup Teams (2000)
22. Pelachaud, C. and Poggi, I.: Subtleties of facial expressions in embodied agents Journal of Visualization and Computer Animation. 13(2002) 287-300
23. Perlin, K. and Goldberg, A.: IMPROV: A System for Scripting Interactive Actors in Virtual Worlds In: Proceedings of SIGGRAPH 96 ACM SIGGRAPH / Addison Wesley (1996) 205-216
24. Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T.: Numerical Recipes in C Cambridge University Press (1992)
25. Pynadath, D. V. and Marsella, S. C.: Fitting and Compilation of Multiagent Models through Piecewise Linear Functions In: the International Conference on Autonomous Agents and Multi Agent Systems (2004) 1197-1204
26. Reynolds, Craig W.: Flocks, Herds, and Schools: A Distributed Behavioral Model In: ACM SIGGRAPH (1987) 25-33
27. Tu, X. and Terzopoulos, D.: Artificial Fishes: Physics, Locomotion, Perception, Behavior In: ACM SIGGRAPH (1994) 43-49
28. Vilhjalmsson, H.: Animating Conversation in Online Games In: M.Rauterberg (ed): Entertainment Computing ICEC Springer (2004) 139-150
29. Vilhjálmsson, H. H. and Cassell, J.: BodyChat: Autonomous Communicative Behaviors in Avatars In: second ACM international conference on autonomous agents (1998)
30. William, M. Hsu, John, F. Hughes, and Henry, Kaufman: Direct manipulation of free-form deformations In: Proceedings of the 19th ACM SIGGRAPH annual conference on Computer graphics and interactive techniques ACM Press (1992) 177-184
31. Zoran, Popovi and Andrew, Witkin: Physically Based Motion Transformation In: ACM SIGGRAPH (1999) 11-20