Direct manipulation like tools for designing intelligent virtual agents [⋆]

Marco Gillies[1], Dale Robeterson[2], and Daniel Ballin[2]

[1]Department of Computer Science, University College London, London, UK,
[2] BT plc, Adastral Park, Ipswich IP5 3RE, UK m.gillies@cs.ucl.ac.uk,
{dale.e2.robertson,daniel.ballin}@bt.com

Abstract. If intelligent virtual agents are to become widely adopted it is vital that they can be designed using the user friendly graphical tools that are used in other areas of graphics. However, extending this sort of tool to autonomous, interactive behaviour, an area with more in common with artificial intelligence, is not trivial. This paper discusses the issues involved in creating user-friendly design tools for IVAs and proposes an extension of the direct manipulation methodology to IVAs. It also presents an initial implementation of this methodology.

As computer graphics techniques progress from research result to wide popular adoption, a key step is the development of easy-to-use tools. A well known example in the last decade has been the development of HTML handling tools from simple text editors to graphical tools. These tools radically change the way in which graphical content is produced. They remove the need for programming ability and the concern with syntactic detail that is required by textual tools. They shift the focus to the purely graphical/artistic factors that are really central to graphics production. They allow professional artists (designers) to express themselves to the best of their abilities. They also enable amateurs access to the technology, enabling end-user content-creation and also content creation by non-artistic professionals such as educationalists or scientists. They remove the need for a programmer to be involved in the production process, thus putting more of the process in the hands of the artists or designer. Thus the development of easy to use tools is vital to any aspect of graphics research.

One of the most interesting and active areas of research in the graphics field in recent years has been in intelligent virtual agents. The distinguishing feature of intelligent agents is that they have proactive behaviour, and they respond autonomously to their environment. In research terms they lie on the boundary between graphics and artificial intelligence. They are used many in applications such as multi-user on-line worlds, computer games, health and therapy systems, interactive education environments and e-commerce[1]. IVAs can exhibit many types of behaviour, we focus on Non-Verbal Communication (NVC), and in particular posture and gestures, which are important expressive elements in social

[⋆] This work has been supported by BT plc, and the EU FET PRESENCIA project IST-2001-37927

interaction. Providing user-friendly tools for IVAs would greatly increase the ease of production of interactive graphical environments, reduce the cost and potentially increase the quality of the IVAs' behaviour by allowing more direct artistic input. Allowing end users to edit IVAs' behaviour would also be highly beneficial. This is particularly true of on-line environments where each user is represented by an IVA, called an avatar, with autonomous beheaviour as suggested by Vilhjálmsson and Cassell[2]. In existing multi-user virtual environments users are keen to personalize the appearance of their avatars[3], it is therefore likely that they would want to be able to personalize the behaviour as well, if user friendly tools were available.

IVAs have important features that affect design tools and make creating these tools an important research challenge. Most importantly they have autonomous behaviour. A character in an animated film will have their animation entirely specified beforehand by an human animator. By contrast, an IVA will generate new animation (or select existing animations) in real time based on its internal state and events in its environment. We will refer to this generation or choice of animation as the agent's *behaviour*. The major challenge in creating design tools for autonomous behaviour is that it depends on the state of the enviroment, what we will call context, and therefore this context must be taken account of whenever the behaviour is edited. The context can contain many features, the location of the IVA, the behaviour of other IVAs, or the behaviour of human participants in the environment (maybe captured through position tracking). The context therefore consists of large quantities of heterogeneous information, however, we assume that this can be reduced to a number of discrete and continuous variables. This reduction could be done by a number of pre-processing steps on the inputs. As we are dealing with NVC we mostly use information about social context, for example, whether the IVA is talking to an authority figure, or whether the topic of discussion is political. In this paper we assume the an IVA's behaviour is controlled by a fixed set of algorithms, which take a context and a set of parameters and use this to generate animation. The parameters of the IVA define its individual behaviour, how different IVAs behave differently in the same context. Parameters in our system mostly deal with how context maps onto behaviour, or onto intermediary internal states, for example, a parameter might be an IVA's tendency to become excited when discussing politics or its tendency to gesture when excited. We assume in this work that the design tools change only the parameters, while the algorithms remain the same. The conclusion will describe future approaches to editing the algorithms themselves.

Direct manipulation has been one of the most successful human computer interaction paradigms, particularly in graphics. The most important features of the paradigm is that it allows uses to edit the final visible result, rather than the, possibly difficult to understand, internal parameters. It seems particularly applicable to the animated behaviour of IVAs. However, traditional applications of Direct Manipulation rely on the ability to view the entire result at once, however, this is not possible for IVAs due to the highly context dependent nature of their behaviour. This paper presents tools that maintain the benefits of direct

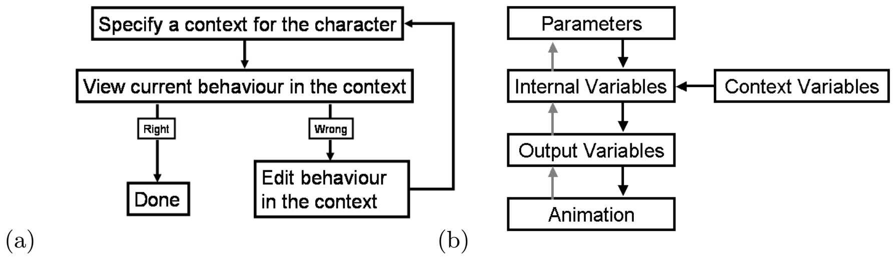
Specify a context for the character
View current behaviour in the context
Right
Wrong
Done
Edit behaviour in the context
(a)
(b)
Parameters
Internal Variables
Context Variables
Output Variables
Animation

Fig. 1. (a)An overview of the proposed interaction style for editing IVA behaviour. (b) 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.

manipulation. We propose an interaction style illustrated in figure 1(a). User may successively view the behaviour of the IVA in different contexts. The users set up the various variables that define the context and views the current behaviour within that context. They can then edit this behaviour if it is not correct and then pass on to the next context until the results are largely correct. Each edit made to the behaviour provides a new constraint on the parameters of the behaviour generation system. These constraints allow the user to successively refine the behaviour with each context viewed.

Our direct-manipulation like interface aims to allow end users to edit the behaviour of the IVA rather than its internal parameters. This could be achieved by allowing the users to directly animate the IVA, with a traditional 3D animation interface. This would given very exact control of the IVA's behaviour, and allow a great deal of nuance. We discuss this type of interface in section 2.2. However, this approach has a number of disadvantages, 3D animation can be very difficult for untrained end users. Also, the behaviour of an IVA is also often composed of a number of discrete possible actions (e.g. crossing arms, nodding, waving), rather than a continuous range of behaviour. Direct animation is unsuited to this sort of discrete action space, simply choosing actions from a list is a simpler interface. We therefore also provide an interface, aimed at untrained users and discrete action spaces, that provides buttons to select actions, described in section 2.1.

1 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[4, 5], Lee and Shin[6] or Popović and Witkin[7]. However, the more important contributions deal with 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[8] whose work on simulating birds' flocking behaviour has been very influential. Further important contributions include the work of Badler *et al.* on animated humans[9]; Tu and Terzopolous' work on simulating fishes[10]; Blumberg and Galyean's "Silas T. Dog"[11] and Perlin and Goldberg's "IMPROV" system[12]. 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[13, 14, 2]; Pelachaud and Poggi[15] and Guye-Vuilléme *et al.*[16]. The two types of behaviour we are using are gesture which has been studied by Cassell *et al.*[13] and posture which has been studied by Cassell *et al.*[14] and by Bécheiraz and Thalmann[17].

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 IVA behaviour, for example the APML language[18]. However, though markup languages are an important step towards making it easier to specify IVA 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[16], 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[19] have worked on specifying IVA behaviour by example. Pyandath and Marsella[20] 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 IVA 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[22] and Gain[23].

2 The Interface

This section describes the two user interfaces we have implemented, one based on specifying actions from a set, and the other based on directly animating the IVA's pose. This section also gives examples of their use. The remaining sections will then describe how the interfaces are implemented.

2.1 The Action based interface

The simpler of the two interfaces allows the user to specify an animation by selecting a number of actions. 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

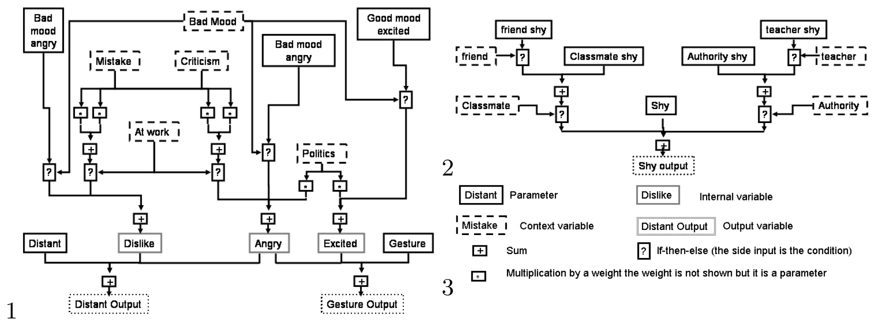

Fig. 2. The relationships between parameters and variables used in our examples (1) action based specification (2) direct animation (3) a key for the diagrams.

a greater or lesser degree, e.g. leaning backward). The interface contains button 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 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, which is then solved for to updated the parameters of the IVA.

Figure 3 gives an example of a sequence of edits. The example is based on editing gestures which tend to be discrete and therefore suited to action based editing. The behavioural control used is shown in figure 2. In it two types of behaviour are defined, *gesture* (beat gestures, which often acompany speech) and *distant* (more hostile gestures). These behaviours depend on a number of contextual paramters: whether the IVA is at work, in a bad mood, discussing politics, has made a mistake, or been criticised. These are used to generated a number of derived parameters which are used to calculate the behaviour parameters. These are: general tendencies to be distant or to gesture, how angry the IVA is, how excited the IVA is and whether it dislikes the person it is talking to.

2.2 The Direct Animation Interface

The other method for specifying behaviour is to directly animate the IVA. This leaves the IVA in a particular posture that must be solved for (currently direct animation is only supported on postures not full animations, extending it would not be too difficult). The user interface used for direct editing is similar to the previous example but the user directly edits the IVA's posture by clicking and dragging on its body rather than using buttons and sliders. Figure 4 shows an example that deals with head and torso posture. The space of these postures is more continuous than gestures and has far fewer degrees of freedom, making it more suited to direct animation. Figure 2 shows how behaviour is generated in this example. In this example there are three types of behaviour distant (turning head or body away), close (the distictive "head cock" posture with the head to the side) and shy (hunched over postures). Only the shy behaviour is shown but the other two have identical dependencies. The example is based on school children's relationships having two types of relationship, classmates (of which friends are a special case) and authority figures (of which teachers are a special case). Each of the three behaviour types can be exhibited differently with each type of relationship. There is also a general tendency to a behaviour type in all contexts, called the "shy".

3 Behaviour generation

The Demeanour architecture is used to generate behaviour for an IVA[24, 25], figure 1(b) shows the behaviour generation method. The basic components of the behaviour system are parameters and context variables, which can be combined together to form internal variables, and finally output variables that are used to animation the IVA. 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. For reasons described below we only allow parameters and variables that depend on parameters to be multiplied by variables that do not depend on parameters (a variable depends on a parameter if the parameter's value is used to calculate the variable's value, directly or indirectly). 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 parameters or variables depending on the value of a boolean condition variable, which can be a context variable but not a parameter:

$$\begin{aligned} x &= x_1 \text{ if } x_c = a \\ &= x_2 \quad \text{otherwise} \end{aligned}$$

Some of the variables produced are outputs that are passed to the animation system. The animated behaviour is generated using a set of basic pieces of motion. Each basic motion has a corresponding output variable that is used as a weight, with which to interpolate the motions, using a quaternion weighted sum technique similar to Johnson's[26]. Many motions can be continuously interpolated, for example 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.

4 Inferring Parameters from Behavior

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 IVA'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 IVA 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[27] to solve for the constriants. Linear programming mimimizes a linear expression subject to to a number of linear equality and inequality constraints:

$$\begin{array}{rl} \sum c_i x_i \text{ subject to} & \sum d_i y_i = 0 \\ & \sum e_i z_i \geq 0 \end{array}$$

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[27]. This minimization solves for the parameters while keeping their values as low as possible (to avoid extreme behaviour).

4.1 Constraints from action specifications

As described in section 2.1, the action based interface allows user to specify the IVA's behaviour using buttons and sliders which provide 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:

$$\begin{array}{rl} v_i - 0.5 \leq 0 & \text{if } w_i = 0 \\ \geq 0 & \quad w_i = 1 \end{array}$$

4.2 Constraints from direct animation

At a high level any posture produced by Demeanour is a weighted sum over the various possible postures as described in section 3:

$$p = \sum w_i p_i$$

As the value of the posture p is known the above formula can be added as a constraint on the values of the weights w_i. A posture is represented as a 3-DOF rotation for each joint of the IVA, so three constraints added for each joint. The weights are then used to add constraints on the output variables as above.

4.3 Constraints from internal variables

With this initial set of constraint we then start to form new constraint based on internal variables and parameters. Any variable will depend on other variables and parameters. If the 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:

$$v = \sum_i (u_j \prod_j k_j)$$

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 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.

5 Conclusion and further work

As described in the introduction this paper has provided three contributions:

1. It has highlighted an important problem for future research, building user friendly tools for designing IVA behaviour.
2. It has proposed an adaptation of direct manipulation editing as a methodology for solving this problem.
3. It has described an implementation of this methodology.

This is roughly the order of importance in which we rank these contributions. We have little doubt that tools for IVA design is an important area that deserves more research. The methodology we propose is a highly valuable one which we consider the most promising. Our own opinion is that different methods will be useful for different types of behaviour. Finally our implementation has shown

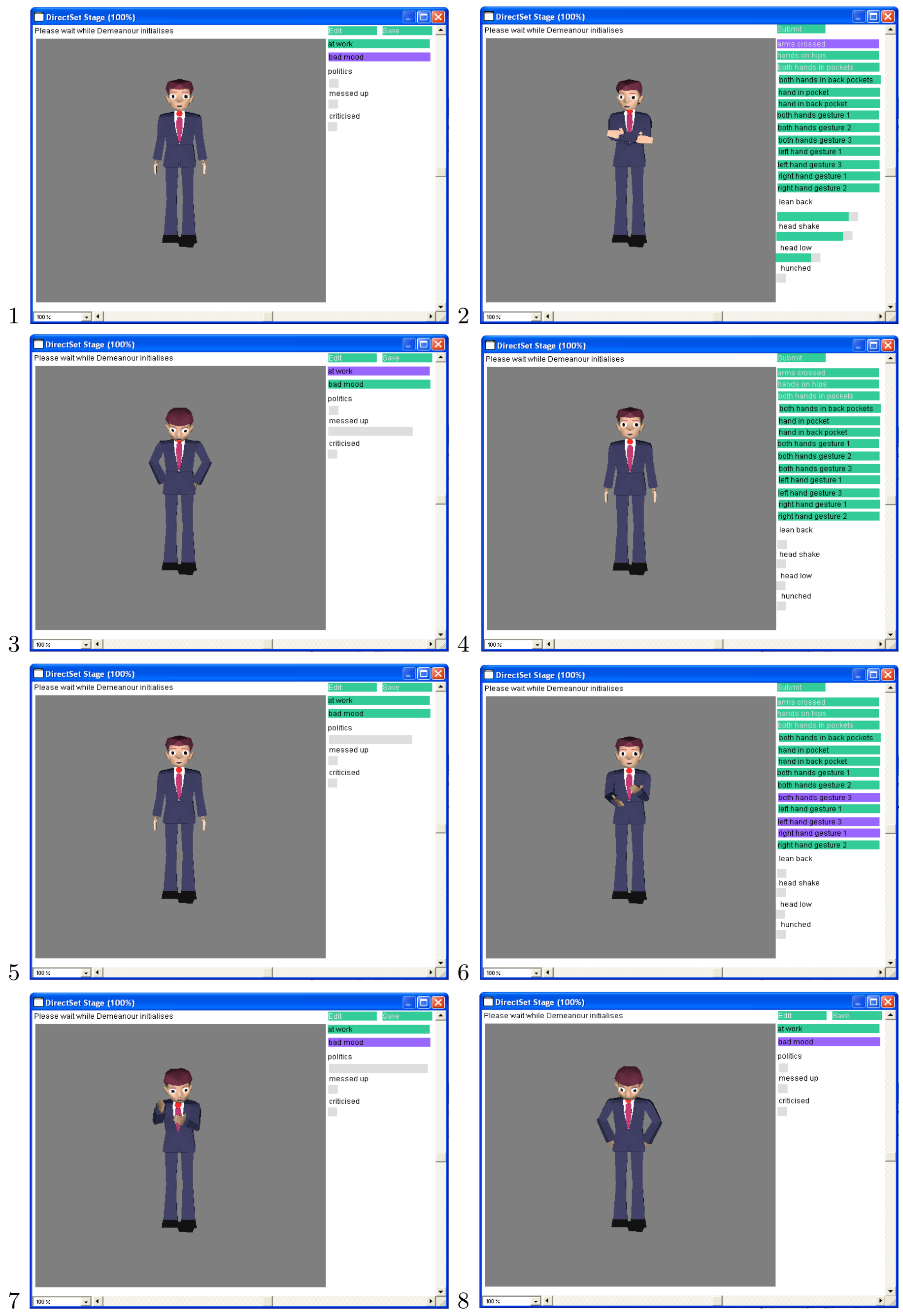

Fig. 3. A sequence of edits using the tool from the action based specification example. The the user initially specifies context (in this case that the IVA is in a bad mood). The initial behaviour (image 1) is neutral as there have been no edits (for clarity, in these examples neutral behaviour is merely a constant rest posture). The user then specifies some distant behaviour and submits it (2). The system has set the general Distant parameter so the IVA produces distant behaviour in a new context (3). The user removes this behaviour to specify a neutral context (4), thus reducing the contexts in which distant 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 IVA in a bad mood discussing politics produces both gesturing and distant behaviour (7). The final image has the same context as the original edit, showing that the same type of behaviour (distant) is successfully reproduced, but that the exact behaviour is different (8).)

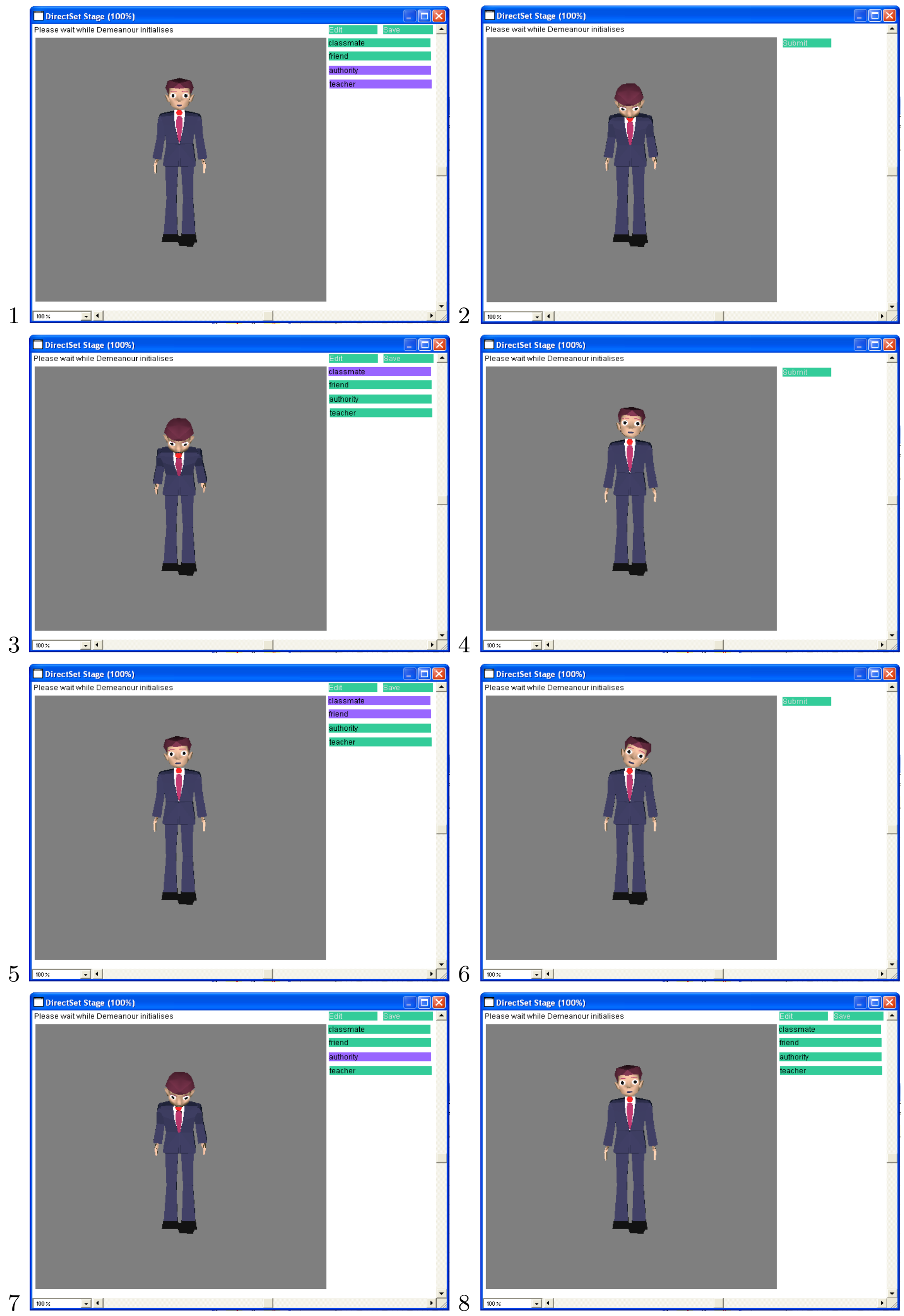

Fig. 4. A sequence of edits of the direct animation. The user initially chooses a "teacher" context (image 1) and creates a hunched over, shy posture (2). The system initially infers a general tendency to shyness (the "shy" parameter) and so displays the same behaviour in a classmate context (3). The user edits this posture back to a neutral one (4) and the system infers that the shy behaviour only occurs in authority contexts (the "authority shy" parameter). The user then adds a "head cock" in the "friend" context to add more close behaviour in that context (5, 6). The final two images show the resulting behaviour in different contexts. The system has generalized the shy behaviour from the "teacher" context to all "authority" contexts as shown in image 7, however, it is not displayed in a neutral context (8).

that the type of user interface we have described is possible in practice and has provided an important first step for research in this area. It is only a first step and more research is needed. It is important to extend our work to other types of behaviour, such as facial animation or speech. These extensions are likely to raise important new issues. Another issue is that our implementation only deals with setting the parameters of a behaviour system, we would also like to build tools that allow users to add new parameters and change the behaviour algorithms used. It is likely that this will require a different type of interface. One approach would be to use a machine learning method that is able to learn more than just parameters from behaviour. In fact a companion paper to this[28] describes initial experiments using reinforcement learning. Another approach is to divide the creation process into a number of stages, a more structural stage that defines the algorithms and one that defines parameters. Each stage could have its own interfaces. This has the benefit that each stage could have an interfaces that is well suited to it. Also, there is a natural division between experts who would perform the first stage and end users who could perform the second.

References

1. Schroeder, R., ed.: The Social Life of Avatars, Presence and Interaction in Shared Virtual Worlds. Computer Supported Cooperative work. Springer (2002)
2. Vilhjálmsson, H.H., Cassell, J.: Bodychat: Autonomous communicative behaviors in avatars. In: second ACM international conference on autonomous agents. (1998)
3. Cheng, L., Farnham, S., Stone, L.: Lessons learned: Building and deploying virtual environments. In Schroeder, R., ed.: The Social Life of Avatars, Presence and Interaction in Shared Virtual Worlds. Computer Supported Cooperative work. Springer (2002)
4. Gleicher, M.: Motion editing with space time constraints. In: symposium on interactive 3D graphics. (1997) 139–148
5. Gleicher, M.: Comparing constraint-based motion editing methods. Graphical Models (2001) 107–134
6. Lee, J., Shin, S.Y.: A hierarchical approach to interactive motion editing for human-like figures. In: ACM SIGGRAPH. (1999) 39–48
7. Popović, Z., Witkin, A.: Physically based motion transformation. In: ACM SIGGRAPH. (1999) 11–20
8. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. In: ACM SIGGRAPH. (1987) 25–33
9. Badler, N., Philips, C., Webber, B., eds.: Simulating Humans: Computer Graphics, Animation and Control. Oxford University Press (1993)
10. Tu, X., Terzopoulos, D.: Artificial fishes: Physics, locomotion, perception, behavior. In: ACM SIGGRAPH. (1994) 43–49
11. Blumberg, B., Galyean, T.: Multi-level direction of autonomous creatures for real-time virtual environments. In: ACM SIGGRAPH. (1995) 47–54
12. Perlin, K., Goldberg, A.: Improv: A system for scripting interactive actors in virtual worlds. In: Proceedings of SIGGRAPH 96. Computer Graphics Proceedings, Annual Conference Series, New Orleans, Louisiana, ACM SIGGRAPH / Addison Wesley (1996) 205–216

13. Cassell, J., Bickmore, T., Campbell, L., Chang, K., Vilhjálmsson, H., Yan, H.: Embodiment in conversational interfaces: Rea. In: ACM SIGCHI, ACM Press (1999) 520–527
14. Cassell, J., Nakano, Y., Bickmore, T., Sidner, C., Rich, C.: Non-verbal cues for discourse structure. In: 41st Annual Meeting of the Association of Computational Linguistics, Toulouse, France (2001) 106–115
15. Pelachaud, C., Poggi, I.: Subtleties of facial expressions in embodied agents. Journal of Visualization and Computer Animation. **13** (2002) 287–300
16. Guye-Vuilléme, A., T.K.Capin, I.S.Pandzic, Magnenat-Thalmann, N., D.Thalmann: Non-verbal communication interface for collaborative virtual environments. The Virtual Reality Journal **4** (1999) 49–59
17. Bécheiraz, P., 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
18. DeCarolis, B., Pelachaud, C., Poggi, I., Steedman, M.: Apml, a markup language for believable behaviour generation. In Prendiger, H., Ishizuka, M., eds.: Life-like characters: tools, affective functions and applications. Springer (2004) 65–87
19. Del Bimbo, A., Vicario, E.: Specification by-example of virtual agents' behavior. IEEE transactions on visualtization and Computer Graphics **1** (1995) 350–360
20. Pynadath, D.V., 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
21. Scerri, P., Ydrén, J.: End user specification of robocup teams. In: RoboCup-99: Robot Soccer World Cup III. Lecture Notes in Computer Science. Springer-Verlag (2000)
22. Hsu, W.M., Hughes, J.F., Kaufman, H.: 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
23. Gain, J.: Enhancing spatial deformation for virtual sculpting. PhD thesis, University of Cambridge Computer Laboratory (2000)
24. Gillies, M., Ballin, D.: Integrating autonomous behavior and user control for believable agents. In: Third international joint conference on Autonomous Agents and Multi-Agent Systems, Columbia University, New York City (2004)
25. Gillies, M., Crabtree, B., Ballin, D.: Expressive characters and a text chat interface. In Olivier, P., Aylett, R., eds.: AISB workshop on Language, Speech and Gesture for Expressive Characters, University of Leeds (2004)
26. Johnson, M.P.: Exploiting Quaternions to Support Expressive Interactive Character Motion. PhD thesis, MIT Media Lab (2003)
27. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C. Cambridge University Press (1992)
28. Friedman, D., Gillies, M.: Teaching characters how to use body language. In: Intelligent Virtual Agents. (2005) This volume.