To appear Proceedings of CHI '99

Sympathetic Interfaces: Using a Plush Toy to Direct Synthetic Characters

Michael Patrick Johnson[♦], Andrew Wilson[♣], Christopher Kline[♦], Bruce Blumberg[♦] and Aaron Bobick[♣]

The Media Laboratory, MIT

Synthetic Characters Group[♦] & Vision and Modeling Group[♣]

20 Ames St.

Cambridge, MA USA 02139

{aries, drew, ckline, bruce, bobick}@media.mit.edu

ABSTRACT

We introduce the concept of a *sympathetic interface* for controlling a synthetic animated character in a 3D virtual environment. A plush doll embedded with wireless sensors is used to manipulate the virtual character in an iconic and intentional manner. The interface extends from the novel physical input device through interpretation of sensor data to the behavioral "brain" of the virtual character. We discuss the design of the interface and focus on its latest instantiation in the *Swamped!* exhibit at SIGGRAPH '98. We also present what we learned from hundreds of casual users, who ranged from young children to adults.

Keywords

Sympathetic interface, plush toy, synthetic character, physically-based interface, virtual world.

INTRODUCTION

Our group's main research goal is to make interactive *synthetic characters*, 3D virtual creatures whose simple intelligence and emotion make them seem sentient and alive. This paper presents one aspect of the *Swamped!* research testbed being developed under the direction of Prof. Bruce Blumberg. The research problem that this paper addresses is how a user can interact with such a character in the most compelling, immersive manner possible. In particular, we wanted to allow a child to take on the role of one character in an interactive cartoon experience

Any interface for controlling a complex virtual character needs to address several important problems:

- Many degrees of freedom need to be controlled.
- Context-specific actions should be simple to do (same input can map to different actions).
- The character should always remain "in character" and believable in its role.
- Navigation through the environment should be easy.
- The control mapping needs to be open-ended and easily scalable to new interactions

The characters we build have many degrees of freedom (such as arms and legs and heads) which can be animated, all of which potentially can be controlled. The interface must allow this complexity of motion. It must also allow actions that are context-specific in a simple manner. For example, if we have a chicken character, the same input from the user could mean "fly around" or it could mean "fly onto that raccoon's head and scratch him." The interface should use context to disambiguate the input rather than forcing the user to do so. To keep the illusion of life and personality, the characters need to remain "in character," moving and behaving in a compelling and believable manner as designed by the artists. For example, if the character is sad, he should respond to user input in a sad manner. Also, the user simply needs to be able to get around in the virtual world. Finally, the interface should easily allow new types of interactions to be added as simply as possible without adding cognitive complexity to the user (learning new controls, e.g.).

Since we want the interface to be used by children, it also needs to be friendly and simple to learn. Additionally, we want it to be haptically engaging since children enjoy touching objects. Previously, we discovered that touch was important when users complained about not being able to "hug" a character in a project that used computer vision input [8].

We argue that a plush toy[1] embedded with sensors is a natural and novel solution to this problem. Children are often already familiar with plush toys and acting out stories

Permission to make digital/hard copy of all or part of this material without fee is granted provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication and its date appear, and notice is given that copying is by permission of the Association of Computing Machinery, Inc. (ACM). To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or fee.

[1] A soft doll for children

with them. They are enjoyable to hold and are cuddly. They allow many degrees of freedom to be sensed, and the mapping from the doll to a similar virtual character is simple and often obvious. By augmenting this input device with gesture recognition technology and the character's "brain," we can allow the character to remain in character and allow it to disambiguate user input based on its perceptual and motivational context. We call this collection of technology a *sympathetic interface,* which we describe shortly.

We demonstrated this interface in an interactive cartoon experience called *Swamped!* which was used by hundreds of users at SIGGRAPH '98 and internally within our lab. The user takes on the role of a chicken that is trying to protect its eggs from a hungry raccoon in a barnyard setting. The chicken has various behaviors such as squawking to get the raccoon's attention and make him angry, scratching his head, kicking him and setting a trap for him. The raccoon is fully autonomous, choosing what actions to take based on his desires, perceptions, and emotional state. The chicken is semi-autonomous and is controlled by the user. The user stands in front of a projection screen showing the virtual world and the virtual chicken and directs the chicken by making appropriate gestures with the doll (see Color Plates 1 and 2). For example, wobbling the doll back and forth makes the virtual chicken walk and flapping the doll's wings will make him fly. The user's attention is meant to focus on interactions in the virtual world and not on the doll itself.

The goal of this paper is to introduce the concept of a *sympathetic interface* and one implementation --- using a plush toy to control a virtual character. We will also describe design decisions in developing the interface and our lessons from many users experiencing the *Swamped!* exhibit.

SYMPATHETIC INTERFACE

We use the term *sympathetic interface* to describe this type of physical interface. The plush toy interface is sympathetic in several senses of the word:

- the effect of an action resembles its cause (sympathetic magic, sympathetic vibration)
- it is inviting and friendly
- it tries to help the user by understanding what they are trying to do given the context

The first aspect describes the coupling between the physical and the virtual instantiations of the character. Sir James Frazer used the term *sympathetic magic* to describe one of the common ritual magic principles he discovered in various primitive cultures [5]. In sympathetic magic, an effect can be created by performing an iconic version of it. The classic, well-known example of sympathetic magic is that of the *voodoo doll*. A voodoo doll is often a wax or cloth effigy that is somehow associated with a person. The voodoo practitioner then manipulates the doll to cause an effect on that person, such as sticking pins in the doll's arm to cause pain (theoretically). We often use this "voodoo doll metaphor" to describe the iconic nature of the interface – "Do to the doll what you would like the virtual character to do."

Secondly, a plush doll is inviting and friendly to a child. Children are not afraid to pick it up and manipulate it. Also, they can develop an emotional contact with the doll, which is not the case for a traditional input device such as a mouse or keyboard.

Furthermore, we designed the interface to be sympathetic to what the user is trying to do. It tries to understand the *intentions* of the user's input in the current context of the virtual environment and tries to help them achieve it. For example, if the user is clearly heading for the henhouse in the virtual world, the chicken should realize this and help navigate there rather than forcing the user to make fine control adjustments.

Intentional Control

We call this iconic mapping *intentional control*. Intentional control is an interface control mapping from the input device to the character where there is an interpretation layer between the raw data and character response which tries to infer the user's intent of the input rather than mapping it onto the character directly. The user is influencing the character at the *behavioral* level as opposed to the *motor* level.

Intentional control offered us a solution to the problem of "direct drive." In an early tethered version of the doll (see below), the sensor data were hooked directly to a virtual character's animated degrees of freedom, leading to a direct, continuous mapping of doll motion to character motion. Although this mapping made the nature of the interface immediately obvious to a user, the generated motion tended to be jerky due to noisy data or people moving the sensors quickly. Also, if the virtual character did not do *exactly* what the doll did, users complained. This fact makes the sensor problem much harder since you need to sense every available degree of freedom in the doll.

Furthermore, a major motivation for researching intentional control was our need to let the character "remain in character." Our experience with the direct drive version made us aware that animating a character well was hard. We wanted to keep artistic control over the animation of the character so that it moved believably according to the artist's vision of the character and maintained the illusion of life. Relying on the raw sensor data produced very robotic and lifeless motion, even in the hands of skilled puppeteers. Finally, we did not want to force users to have to make very fine control adjustments with the interface when the task they wished the virtual character to do was obvious in context. For example, the chicken will fly onto the raccoon's head if the wings are flapped anywhere near the

raccoon rather than making the user laboriously steer onto his head. This fact also allows the virtual character to act out the action in a dramatic fashion, much like a real actor being directed by a director.

The facts that our characters have a behavior system, or "brain," and are embedded in a virtual world allow them to disambiguate potentially conflated inputs by using perceptual, motivational and emotional context. In the absence of behavioral disambiguation, the perceptual problem is much harder since the gesture recognition needs to do the disambiguation out of context. We discuss this further shortly.

RELATED WORK AND INFLUENCES

Physically-based user interfaces allow a user to manipulate a familiar physical object to achieve a desired effect in a virtual environment, be it a 2D desktop or a 3D application. The concepts of graspable user interfaces [4] and tangible user interfaces [7] center on this idea. To use the parlance of Fitzmaurice, we leverage the affordances of both the physical and virtual instantiations of the character. The physical character affords passive haptic feedback (the cuddly nature of the doll, the feeling of actually moving the parts) and a simple, iconic way to manipulate many degrees of freedom. The fact that the character exists in a virtual world, however, means that anything is possible. We can do anything that we can imagine and program.

Applications demonstrating physically-based interfaces have focused mostly on drawing and data manipulation tasks, however, and often involved an extremely direct coupling between the physical and virtual [4,7,6]. Our work differs from these in that we avoid a *direct* coupling of device input to output in favor of an *intentional* coupling.

Alison Druin's Noobie was influential in our research [3]. Noobie was a large stuffed animal with a computer monitor in its stomach and touch sensors around its body which children could use to design animals. Our work differs in that Noobie was meant to be a softer computer terminal interface, not a synthetic character interface.

Microsoft's ActiMates Barney doll is a plush toy embedded with pressure and light sensors, a wireless data connection, voice output and simple arm motors [11]. Barney is designed to be a social agent, not a transparent interface to a character. The focus of the interaction is on the Barney doll itself, not a virtual representation of him. Barney's main role is to comment on computer media at crucial moments to facilitate learning. Our work is designed to be an interface for controlling a virtual character where the focus of the interaction is on the virtual world.

We initially evaluated the *Monkey 2* kinematic keyframe input armature from Digital Input Design, Incorporated (www.didi.com). A physical skeleton structure consisting of links and sensor-equipped joints is carefully manipulated to pose a corresponding 3D model in animation software. The *Monkey* was not suitable for our purposes since it is too large and unwieldy to manipulate simply in real-time by an experienced adult puppeteer, let alone a child. It has many degrees of freedom, and it is hard to keep it upright without making the joints very stiff to move.

EARLY TETHERED PROTOTYPE

An early prototype of the plush toy interface was a beaver plush doll which had flex sensors in the legs, tail and arms and an Ascension Flock of Birds magnetic sensor in its head and torso which gave accurate position and orientation data. These sensors were all tethered to the computer, making it hard to move the doll in certain ways, like upside down. The application we tested involved swimming around in a lake with fish and other items to see, but no explicit goal. The user controlled the direction and speed of swimming and could make the character stop and tread water in order to look around by manipulating the head and limbs in a direct drive mode. Many users tried this system and enjoyed using the plush doll to steer the beaver through the water saying it was a "great idea," but most complained about the wires. Several users thought the character was robotic and lifeless at times. We feel that this was because he was controlled with a direct mapping from sensors to animation during these times (direct drive). Many tired quickly of the scenario, also, asking "what else can I do?" or "what's the goal?" The *Swamped!* exhibit was designed to have a wireless interface, an intentional control mechanism and a more compelling interaction in response to these criticisms.

WIRELESS INTERFACE DESIGN

This section describes the design and functionality of the latest wireless version of the interface as demonstrated in the *Swamped!* exhibit at SIGGRAPH '98. The raw sensor data is first interpreted using gesture recognition techniques. The behavioral system then interprets these gestures in the context of the character's current environment and state (see Figure 1).

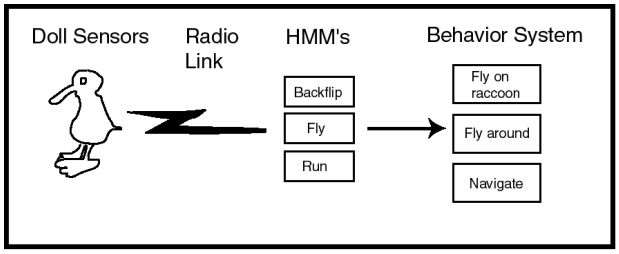


Figure 1: Data flow from sensors over wireless link to gesture recognition (HMM's) which are inputs to the character's behavioral system (brain).

This section is divided into several parts: physical construction, sensing technology, gesture recognition, and behavioral interpretation.

The Doll

The physical doll was fabricated to match the virtual character, which was modeled first. We feel that this

similarity is important for making the sympathetic connection between the doll and character clear to the user.

An armature made of plastic, brass tubing and wire holds a sensor package, sensors and provides an articulated structure (see Color Plate 4). The doll's exterior is fleece fabric and molded latex (see Color Plate 3). The latest incarnation of the doll embeds a variation of a wireless sensor package designed by Joe Paradiso *et al* [9]. The sensor package includes an array of 13 sensors, including those relating to the doll's attitude and various configuration aspects:

- two pitch and roll sensors
- one gyroscope sensing roll velocity
- three orthogonally mounted magnetometers sensing orientation with respect to magnetic north
- two flexion (FSR) sensors for wing position
- three squeeze (PVDF) sensors embedded in the body and beak
- one potentiometer to sense head rotation about the neck

See Color Plate 4 for a photo illustrating how the sensor package is embedded in the doll's armature. On board the sensor package, a PIC micro-controller with analog to digital conversion packetizes the sensor values which are then transmitted via radio frequency at a rate of at least 30 Hz. The receiving station then relays the sensor records via a serial connection to the host computer. The commercially available radio transmitter/receiver pair uses the 418 MHz or 433 MHz frequencies which do not require a license from the FCC. Depending on the type of antenna used, the transmitter has a range of several hundred yards. The sensor package does not produce any radiation known to be harmful. The on-board electronics are powered by a 9 volt lithium battery, lasting several hours.

When the *Swamped!* setup is moved, magnetometer readings must be calibrated to account for any change in the direction of magnetic north. To read heading information from the magnetometer, it is important that magnetic north lie somewhat along the plane of the floor; this can be an issue in some buildings, where the earth's magnetic field is distorted in many ways.

Unlike the Ascension or Polhemus unit, the wireless sensor array does not give true 6 degree of freedom position and orientation data: there is no world-coordinate position data, and the orientation data from the magnetometers is complete only up to a roll about magnetic north. At first this fact may appear to be a significant drawback; however, we do not necessarily need accurate position and orientation since we are not using direct control. The gesture recognition techniques for implementing intentional control do not necessarily require complete 6 DOF data. In fact, the variety of sensors on board permits the interpretation software great flexibility in detecting events. For example, many different styles of "walking" may be detected with the gyroscope alone.

Gesture Interpretation

Raw data from the doll is processed in real-time on the host computer to recognize gestures that are taught to the system in a learning phase as described in this section.

Action Primitives

Our goal is to provide the user with a high level of direction over the synthetic character. It is undesirable from aesthetic and usability points of view to have the user explicitly plant one foot in front of the other to make the character walk. For example, the system should respond to a motion that most users agree evokes "walking" without concern for how the motion would look if rendered in a literal fashion. *Swamped!* uses machine learning and gesture recognition techniques that complement the wireless sensor array to provide a high-level iconic, or intentional, style of control, as described above. The system can detect a variety of actions of the doll under user control, such as *walk*, *run*, *fly*, *squeeze-belly*, *hop*, *kick* and *back-flip*. Each of these action primitives has at least one associated gesture recognition model. Multiple models are used in cases where users tend to use one of a number of styles to accomplish an action. For example, users tend to have two ways of making the character walk: rolling the doll back and forth in a cyclic fashion or a similar motion about the vertical (yaw) axis. By using machine learning techniques, we have avoided the difficulty of writing *ad hoc* routines to identify each of the action primitives. Models of the actions are computed in an automated fashion from a set of examples collected using the actual doll. In this training phase, we use a footswitch to indicate to the system the exact start and duration of a new example. In this way, examples can be collected quickly and easily.

Hidden Markov Models

We use hidden Markov models (HMMs) to learn and recognize action primitives. HMMs were originally developed for speech recognition applications [10] and have been applied to automatic gesture recognition problems [12]. HMMs provide a sound probabilistic framework for modeling time-varying sequences for later recognition. We omit the details of how HMMs work (see [10]) but note that in practice, the designer must specify a Markov model which describes the overall temporal structure of the action (e.g., A to B to C, then back to A in the case of a cyclic gesture). Given this Markov model, the HMM training algorithm takes as input the set of example sequences and associates each state of the Markov model with a particular region of the feature space (the sensor readings). The resulting HMM may be used during runtime to determine the similarity of an input sequence to the set of training sequences. Once the HMM is trained, the training sequences are no longer needed. The set of training sequences must be chosen to span the variation of how

different users execute the gesture. In *Swamped!* typically no more than 10 to 20 examples were necessary to train an HMM for a given action primitive. During runtime, each of the HMMs are fit to a sliding window of the past 2 seconds of data returned from the sensors. The HMM testing algorithm returns a continuous value indicating how well the model fit the data. If this value exceeds some threshold, the system concludes that the user performed the action corresponding to the HMM. These thresholds are chosen empirically so as to not allow too many false positives. The testing process is computationally efficient and is well within the computational power of today's computers.

Representational Choices

In the application of HMM's for gesture recognition, it is important to pick features that are appropriate to the action. One approach in the application of HMM's for gesture recognition is to provide all features to the HMM and hope that the set of examples covers the variation that is reasonable to expect. In the case of *Swamped!*, it is sometimes difficult to guess what people are going to do with the doll to indicate some action. For example, a given user may hold the doll differently than the person that trained the system, and so the trained model may not generalize to the new user. While the system allows the addition of training examples and subsequent retraining, it is often easier to restrict the model to a subset of features so that the models generalize appropriately. For example, the *run* primitive is an HMM built solely on the vertically mounted accelerometer. When the user shakes the doll up and down in a large and deliberate manner, the *run* HMM fires, regardless of the value of the other sensors.

HMM's must also be supplied with an initial Markov model which has a topology that is appropriate to the action to be recognized. In *Swamped!* there are two basic topologies used: a periodic or cyclic topology used for *walk*, *run*, and *fly* and a two-phase topology (out and away from some rest position and then a return) for all other primitives. We suspect that because of the physical constraints of holding a doll these are the two most useful topologies in describing how users interact with the doll.

Character Behavior System

Intentional control requires that the character designer decide beforehand which behaviors that the character will perform and program the character to perform the associated sequence of sub-actions when the appropriate gestural and/or contextual signals are received. This allows the character designer to change the complexity of the character's behavior without changing the interface or burdening the user.

Motivated by these concerns, we chose to implement this component of the interface as a reactive behavior system similar to the work of Blumberg [1,2]. In *Swamped!*, the chicken's behavior system treats the gesture threshold value from each HMM as a proprioceptive sensory input to a corresponding consummatory behavior. For example, when the user flaps the chicken's wings, the HMM for the flying gesture surpasses its threshold and stimulates the flying behavior. If it is the most appropriate behavior at the time, the flying behavior will become active, which will cause the virtual chicken to begin flying. Similarly, when the user ceases to flap the wings on the doll, the feature information from the doll no longer matches the gesture for flying, the HMM will fall below threshold, and the flying behavior will become inactive. This type of system also has the advantage that it can handle noisy sensor data robustly and not "dither," or switch between two behaviors very quickly.

At any moment during the interaction the incoming feature data may closely match more than one gestural model and therefore several behaviors may wish to become active. We resolve such ambiguities by organizing mutually exclusive behaviors into groups within which they compete for dominance in a winner-take-all fashion. While the primary discriminant for determining the dominant action (and consequently the "intention" of the user) is the magnitude of the HMM, context can also play an important role, as described above. In *Swamped!*, the dominant action is dependent on context when the gesture is active. For example, when the user specifies the *fly* gesture near the raccoon, the chicken attempts to land on the raccoon's head. Otherwise, he flies in a navigational manner. Also, the *kick* behavior will not fire unless the chicken is near the raccoon. This disambiguation was useful for filtering out spurious *kick* gestures that were often conflated with *run* or *hop*.

To aid navigation, the chicken will also try to infer which object in the world the user is trying to steer towards. The chicken will then bias his heading towards this object to ideally reduce the complexity of navigation.

DISCUSSION

The *Swamped!* system currently runs on a dual processor Pentium computer, with graphics being rendered on a Silicon Graphics Onyx2 Infinite Reality system. Sound and music are also rendered on separate machines. The entire system is written almost exclusively in Java, including the gesture recognition and behavior system with the exception of the underlying graphics and sound rendering which use C++. Rendering speed was the bottleneck, but we achieved framerates of 30Hz.

Over 400 users interacted with the *Swamped!* installation in the *Enhanced Realities* exhibit at SIGGRAPH (see Video Figure). Users were told the cartoon scenario and that the goal was to keep the raccoon busy so he did not eat any eggs by engaging in the chicken's various behaviors. The main behaviors available were:

- Squeeze the torso or beak to squawk and make the raccoon angry
- Fly around the world or on the raccoon's head to make him angry. Continuous direction control was provided by pointing the chicken left or right to steer

- Walk into various buildings to set a trap for the raccoon or get catapulted to the other side of the world
- Kick the raccoon to make him angry
- Stand on head to do backflips

When the raccoon was angry he would chase the chicken, keeping him away from the eggs.

This section will discusses problems we discovered and lessons we learned during this experience.

User Classification

In general, we encountered three categories of users: teachable, ideal and skeptical, in order of approximate group size. The ideal users were often children who would pick up the doll, start manipulating it and immediately understand the concept of the interface. One girl of about six years old became an expert user within minutes (better than the designers, in fact) and played for a half hour. The teachable users were by far the largest group. The typical member of this group would pick up the doll and try to manipulate one part, such as one wing or a foot, expecting a direct mapping. After we demonstrated a *walk* gesture and explained the "voodoo doll" metaphor, many of these users quickly could learn to use the doll and enjoyed the experience. Several users, however, never understood how the doll controlled the character and were convinced that there was no connection.

We informally asked users what they thought of the interface. Most responses were positive. Users said that it was "very cool," "magical," "a great idea," and "beautiful." Several users asked "how much?" thinking we were selling prototypes for videogame systems. Children, in particular, loved playing with the system and many came back to try again. Although adults would often tire of woggling the doll continually, children thought that this was great. One excited girl said, "you mean I can keep moving him like this?" One four-year old child tried to control the character with a plush toy he had with him and was disappointed that it did not work. We feel that the haptic feedback and normal play style associated with a plush toy allowed people to form a more emotional contact with our characters, making them seem more alive.

Most of the complaints involved steering and navigational problems (see below) and wanting more actions available to perform, which is not a function of the interface design.

Navigation

Navigation to desired locations in the scene often proved difficult to users. One problem we noticed is that the turning radius of the character is fairly large, making it difficult to make sharp turns. Also, there was no continuous way to adjust the walking speed of the character in order to compensate for this. The chicken either walked at constant speed or ran at a higher constant speed. We discovered that there was no simple way of inferring the user's desired speed for the chicken using the gesture-based input. We plan to look at the frequency at which the user makes the walking gesture as a possible parameter to infer speed. Another navigation problem was over-steering, which we describe below.

Camera

The automatic camera control algorithms were originally written for fully autonomous characters with no user inputs and were designed to show off the content of a scene, including action and emotions. When the interaction was added, however, it was often difficult to navigate the chicken based on the camera angles that were chosen and impossible when the chicken was off-screen. The camera system was redesigned to take into account the need of the user to navigate by trying to keep the viewpoint behind the virtual chicken so that the user could steer left and right easily, while also cutting to show important actions that the autonomous raccoon was taking off-screen.

Users were still often frustrated when the camera "cut when I was about to do something." This problem of stealing control from the user in order to display relevant information about the narrative is still an open research question.

Users often would "over-steer" the chicken since they could not see enough of where the chicken was headed when he was taking a sharp turn. The virtual character's heading at this point was somewhere off-screen. By the time the desired location appeared on-screen, the user had already steered too far. This problem also occurs with novice airplane pilots who eventually learn to reduce control input before reaching the desired attitude. We noticed that some users became better at this, but we need to look at ways to avoid this problem, such as wider angle of view or more anticipatory camera angles.

These experiences argue that the camera algorithms for a virtual world cannot be made separately from the input and user interface decisions. The camera is intimately coupled to the user interface in the same way that the behavior system is.

Gesture Recognition

The gesture recognition system uses models automatically derived from training examples recorded while the designer manipulates the doll. It is important that the examples exhibit the same kinds of variation that the system is likely to encounter with naive users, or the system will sometimes miss the user's gesture. While it is very easy to collect training examples and retrain a gesture model with the current system, the designer can never be sure that the training examples will work for all users. Furthermore, it can be difficult or impossible for the designer to reproduce a variant on a gesture seen during runtime.

With *Swamped!* we found that it was indeed difficult to span the entire range of variation of some gestures. Much of this variation is due to differences in the user's style of control. For example, some users manipulate the doll with

small, subtle movements, while others make broad, quick gestures. The tendency of users to repeat the misunderstood gesture more boldly and deliberately (a habit borrowed from speech?) will sometimes take the user further away from the gesture model. We found it very difficult, for example, to train a model of the *kick* gesture to satisfy all users. One approach is to simply raise the gesture acceptance threshold; unfortunately, the gesture may then mask the activity of some other gesture. As previously mentioned, we addressed this problem by using mutual exclusion groups in the behavior system. Longer gestures such as *walk* were less problematic. This is most likely because long gestures include more movement information and are thus more unique among the set of gestures.

Wireless vs. Tethered

The users who had experienced the previous tethered interface were much happier with the wireless version, saying that it seemed "magical" without the wires. It was unencumbering and could be carried around the room and used as a normal plush doll away from the context of the screen. It also allowed a wider set of gestures, such as an energetic shaking, which we interpret as *run*.

Strengths and Weaknesses of Sympathetic Interfaces

We feel that our initial evaluation of the plush toy as a sympathetic interface was a success. Users responded positively to the device and intentional control concept. One of the biggest weaknesses with an intentional control mechanism was navigation. Most people are used to more direct control from videogames and have little patience when it is difficult to navigate. This is an issue we will address in future work, perhaps by mixing some direct control with intentional control.

Many people, including children, learned to use the multiple degrees of freedom of the doll very quickly. This implies that we succeeded in making the device easy to learn and use. The designers found it fairly easy to add new gestures and behaviors, suggesting that the system will be scalable. We demonstrated some simple context-dependent disambiguation of gesture, but we cannot make any strong claims yet about the success of this until we add many more such examples.

CONCLUSIONS AND FUTURE DIRECTIONS

We demonstrated the concept of a sympathetic interface in the form of a novel plush toy control interface for a virtual character. Over 400 participants successfully used the system and offered positive comments. Our initial success with sympathetic interfaces suggests that they are a fruitful new research direction for animated character control.

We plan to add actuators to the doll to create simple active haptic feedback. Rather than trying to animate the doll in a robotic sense, we will choose actuators that will convey the emotional and physical state of the virtual character to complement the visual and audio cues. Simple examples of this are a variable-rate breathing actuator to convey exertion, a heater to make the doll warm, and a motor that can make the doll shiver when the character is afraid or being attacked.

ACKNOWLEDGEMENTS

We would like to acknowledge the significant contributions made by the entire *Swamped!* team, without which this paper would not be possible: Michal Hlavac (art design and chicken model), Ken Russell (graphics engine), Bill Tomlinson (camera control and armature design), Song-Yee Yoon (sound), Dan Stiehl (doll fabrication), Zoe Teegarden (electronics), Jed Wahl (animation) and Teresa Marrin (music). Special thanks to Dr. Joe Paradiso and his group for the wireless sensors package. Additional thanks to Bill Tomlinson for photos and video. This work was partially supported by the Toys of Tomorrow and Digital Life Consortia.

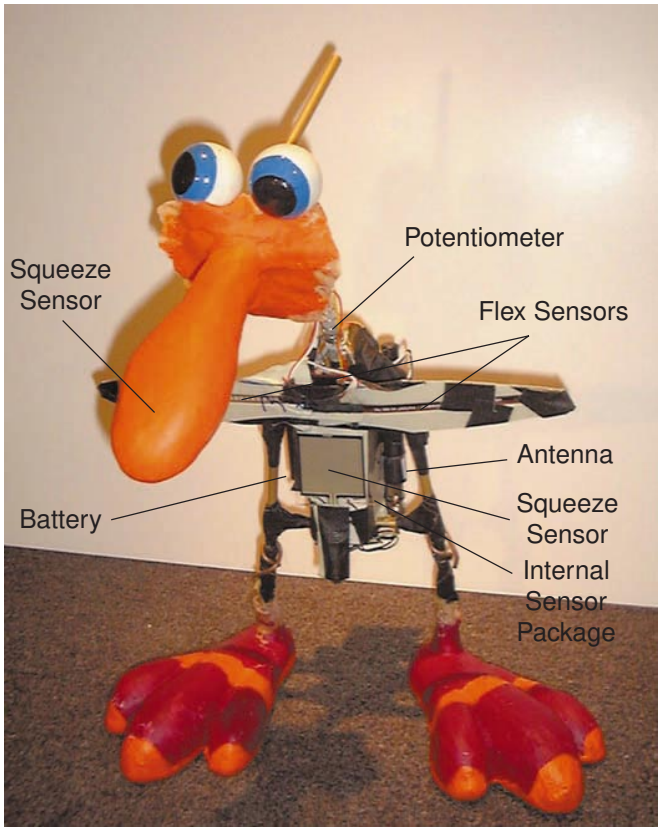
REFERENCES

1. Blumberg, B. and Galyean, T. Multi-level Direction of Autonomous Creatures for Real-Time Virtual Environments. In *Computer Graphics*, SIGGRAPH '95, (Los Angeles, CA, August 1995), ACM Press, 47-54.
2. Blumberg, B. *New Dogs, Old Tricks: Ethology and Interactive Creatures*. PhD Dissertation, MIT Media Lab, 1996.
3. Druin, A. Building an Alternative to the Traditional Computer Terminal. Master's Thesis, Media Laboratory, MIT, 1987.
4. Fitzmaurice, G.W., Ishii, H., and Buxton, W. Bricks: Laying the Foundations for Graspable User Interfaces, in *Proceedings of CHI '95* (Denver, CO, May 1995), ACM Press, 442-449.
5. Frazer, Sir J. G. *The Golden Bough*. The MacMillan Company, New York, 1922.
6. Hinckley, K., Pausch, R., Goble, J.C., and Kassell, N.F. Passive Real-World Interface Props for Neurosurgical Visualization, in *Proceedings of CHI '94* (Boston, MA, April 1994), ACM Press, 452-458.
7. Ishii, H. and Ullmer, B., Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms, in *Proceedings of CHI '97*, (Atlanta, GA, March 1997), ACM Press, pp. 234-241.
8. Maes P., T. Darrell, B. Blumberg, A. Pentland, The Alive System: Full-body Interaction with Autonomous Agents. In *Proceedings of Computer Animation '95 Conference*, Switzerland. IEEE Press, April 1995.
9. Paradiso, J., Hu, E. and Hsiao, K-Y. Instrumented Footwear for Interactive Dance. To be presented at the XII Colloquium on Musical Informatics, Gorizia, Italy, September 24-26, 1998.
10. Rabiner, L.R. and Juang, B.H. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, 1993.
11. Strommen, E. When the Interface is a Talking Dinosaur: Learning Across Media with ActiMates Barney, in *Proceedings of CHI '98* (Los Angeles, CA, April 1998), ACM Press, 288-295.
12. Wilson, A.D. and Bobick, A.F. Nonlinear PHMMs for the Interpretation of Parameterized Gesture. *Computer Vision and Pattern Recognition*, 1998.

Color Plate 1: The user's view of the scene
The chicken, raccoon and henhouse with eggs is visible.

Color Plate 2: The installation setup. Here, one of the authors controls the chicken as the raccoon makes a diving leap for him.

Color Plate 3: The external view of the doll



Color Plate 4: The chicken armature with attached sensors