SWEETPEA: Software Tools for Programmable Embodied Agents

Michael Kaminsky[*], *Paul Dourish, W. Keith Edwards,*
Anthony LaMarca, Michael Salisbury and Ian Smith

Computer Science Laboratory
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto CA 94304 USA
first_surname@parc.xerox.com

[*]Laboratory for Computer Science
Massachusetts Institute of Technology
545 Technology Square
Cambridge MA 02139
kaminsky@lcs.mit.edu

ABSTRACT
Programmable Embodied Agents are portable, wireless, interactive devices embodying specific, differentiable, interactive characteristics. They take the form of identifiable characters who reside in the physical world and interact directly with users. They can act as an out-of-band communication channel between users, as proxies for system components or other users, or in a variety of other roles. Traditionally, research into such devices has been based on costly custom hardware. In this paper, we report on our explorations of the space of physical character-based interfaces built on recently available stock consumer hardware platforms, structured around an initial framework of applications.

Keywords
Interaction hardware, tangible media, augmented reality, ActiMates Barney, Mattel Talk-With-Me Barbie.

INTRODUCTION
Although Human-Computer Interaction, as a field, is focussed primarily on the needs of human users, it is also highly responsive to technological developments. Increasingly over the last few years, fuelled on the one hand by the wider availability of computational power in smaller and lower-power units, and on the other by a series of innovative and insightful reconsiderations of the nature of interaction with technology and the everyday world, a new form of interactive devices has emerged based on a very different interaction paradigm than the traditional desktop interfaces with which we are familiar.

In this paper, we report on our early experiences with the development and use of a general platform for developing these new forms of interactive device. We call these devices *Programmable Embodied Agents* or PEAs. A Programmable Embodied Agent is a portable, wireless, interactive device embodying specific, differentiable characteristics for interaction. The typical form these devices take is as recognizable embodied and often caricatured "characters"; toys and dolls that can be augmented with computational behaviors. Although specific individual devices of this sort have been explored as research prototypes in the past (e.g. Druin's Noobie [Druin, 1987]), we are interested in diversifying the forms of experimentation possible by exploiting a range of commodity platforms. Examples of these platforms are Microsoft ActiMates Barney, and Mattel's "Talk With Me" Barbie[1].

Programmable Embodied Agents lie at the intersection of three recent trends in interactive system exploration.

1. *Embodied Interaction*. A PEA device is not only a site for interaction, but is portable, outside the computer, in the world. Most of our applications focus on the boundary between the real world (including human activities) and the computational world. PEAs provide a natural way to move across this boundary, as well as providing an opportunity to exploit human skills (such as spatial discrimination, peripheral perception and haptic interaction) in a way that conventional devices cannot.
2. *Character-based Interfaces*. Since the interactive presentation of a PEA is some form of anthropomorphic character, the character of the agent can provide a context for the activity. The "helpful Barney" character provides a context for understanding why my Barney agent is telling me that the network is down. Being able to recognise and exploit individual characteristics can smooth interaction, and make it more compelling.
3. *Unified multi-purpose interactive devices*. The fact that we are used to the idea that individuals may have multiple concerns embodied in a single activity can be exploited in the design of PEA applications. The PEA may take on the role of advisor, assistant or gatekeeper, and so may comment on a wide range of different issues (the availability of software services, communi-

This is a draft paper in preparation for CHI'99. Comments are welcome, but please do not cite or distribute without the author's permission.

1. "ActiMates" is a trademark of Microsoft Corp. "Barney" and the Barney character are trademarks of The Lyons Group, L.P.. "Talk-With-Me" , "Barbie" and the Barbie character are trademarks of Mattel, Inc.

cation attempts by other individuals, meeting times, and so forth). The character-based interface helps make this "channel-switching" behaviour seem more natural.

The goals of the work outlined in this paper have been two-fold. First, we want to explore the use of Programmable Embodied Agents in our everyday work setting, using them to begin to move aspects of our computational environment into the real world that we all inhabit. We have been less concerned with the use of PEA tools for completely new forms of application functionality and more concerned with supporting existing activities. In general, we want to consider how to use these tools to smooth and assist in the accomplishment of tasks we already perform every day, to assist us in understanding the progress of on-line work, and to support workplace interactions. Second, we want to explore the use of new consumer devices as PEA platforms, rather than developing novel hardware devices. We hope that this will help make PEA tools more widely accessible, and support the development and deployment of PEA technologies more widely than has been possible previously.

The structure of the rest of this paper is as follows. In the next section, we consider related work from the research literature, and present the lessons we have drawn as the basis for the work presented here. We follow that with a discussion of our an exploration of ActiMates Barney as a platform for PEA development. Next, we introduce a framework for organising potential PEA applications, and explore a range of applications we have developed so far. Finally, we present some idea for future development.

RELATED WORK

Our work on Programmable Embodied Agents is inspired by recent developments in non-traditional interactive forms.

A variety of researchers have, in recent years, begun to explore the opportunities for computation in the world. Weiser's "Ubiquitous Computing" proposal [Weiser, 1991] was an early articulation of the idea that, since people already interact with artifacts in the everyday world, HCI should take the form of interaction with physical devices augmented with computing power, rather than interaction with computers that mimic or track the everyday world. Related work such as that of Wellner [Wellner 1991; Newman and Wellner, 1991], Fitzmaurice [Fitzmaurice, 1993; Fitzmaurice et al., 1995] and Cooperstock et al. [1995] has further explored the "augmented reality" design space.

More recently, Ishii at the Media Lab has been spearheading the development of a program of research into "Tangible Bits", which focuses on how interaction and computation can be brought into the real world and so can capitalise upon everyday human skills [Ishii and Ulmer, 1997].

Research efforts such as these have emphasised the power of computation harnessed to the world, and in particular the value of moving the site of interaction out of the "*box on the desk*" and into the world to which it refers. What we draw from this perspective is the argument that, since the site of the user's concern and activity is typically outside the computer, it makes sense to move the site of computational interaction outside the computer too, and to locate it along with the user, rather than forcing the user to interact with a keyboard, mouse and graphical display. At the same time, by moving out into the world, computational interaction can take advantage of the specialized context in which activity takes place, rather than adopting the "one size fits all" approach of graphical interfaces and UI widget sets. For us, then, Programmable Embodied Agents are sites of interaction that can be located "where the (inter-)action is."

In a variety of domains, the idea of characters as interactive proxies has attracted considerable interest. This idea has a long history – Laurel et al.'s work on "Guides" at Apple is an early example [Laurel et al., 1990; Laurel, 1990] – but lately it has come to considerably greater prominence as the computational power for interactive agents has become widely available (see, for example Adelson, 1992; Rist et al., 1997; Kurlander and Ling, 1995). Perhaps the best-known example of this style of interaction in current products is the "dancing paperclip" (the Office Assistant) of Microsoft Office, one of a variety of help agents available in Microsoft applications.

These explorations hold two lessons that have informed our work. The first is that these characters embody a stronger sense of identity and agency than can be conveyed by a typical graphical interface. As a result, then, they can serve better as channels for carrying particular sorts of information, since they take on the aspect of "messengers." Caricatures of conversational nuance (style of vocal delivery, movement, etc.) provide an opportunity to deliver both information and a context for understanding in a more natural style than could be done with textual messages, pop-up windows and so forth. This idea points us towards a set of opportunities for exploring the areas in which PEAs as specific communication channels, separate from the desktop interface and alongside it, can help manage the variety of information that we deal with in our everyday environments. So, for example, we can use a PEA to convey information about events outside our immediate view (e.g. in distributed systems); messages from other users; and so forth. We will later present a characterization of the styles of interactions for which we have built applications. First, however, we will explore the technical opportunities and challenges that consumer-unit PEAs present.

TECHNICAL BASIS

At the start of our project, we chose two potential platforms for PEAs – ActiMates Barney [Strommen, 1998] and Mattel's "Talk With Me" Barbie (figure 1).

Barbie is a free-standing Barbie doll who can be programmed to talk on a variety of topics. The necessary speech information is downloaded to Barbie via an infrared interface as Barbie sits at her workstation; Barbie's necklace is an IR receiver, while the screen on her computer is is an infrared transmitter). A child can use the supplied software to select a topic for conversation, and also provides names for the child and friends, which Barbie will drop into the conversation. Once Barbie has been programmed, she can be picked up and carried around like any other doll. Pressing a button on her back will cause her to speak one of the downloaded phrases.

.ActiMates Barney is an educational toy based on the children's television character. In addition to being a plush purple dinosaur, this Barney is a radio-controlled device, who can also be programmed to talk and to sing; the Barney toy can also move its arms and head, and has input sensors in his paws (as well as a light sensor in his eye). In "stand-alone mode," Barney can interact with a child in various ways, singing songs and playing games. Additionally, and more significantly for our purposes, Barney can receive instructions from a radio transmitter connected to a television set or personal computer, and controlled by suitably-enabled software or instructions encoded in the television signal. When operating "on-line" to a transmitter, Barney has a wider range of behaviours, which come from the software or television signal. We chose to start our experiments with Barney because of the wider range of functionality available, and in fact we have yet to turn our attention to Barbie

Barney and Barbie are both toys available in toy stores throughout the country. Indeed, this is what made them attractive to us; we were interested in the use of standard consumer platforms for research on Programmable Embodied Agents, rather than custom platforms which are not only costly to develop, but difficult to deploy in numbers. Developing on standard platforms such as these opens up new opportunities for exploring the use of PEAs in everyday settings.

On the other hand, there are two down-sides to the use of these standard platforms. The first is that we are constrained by the technology we are offered. Barbie can talk, but she can't move; Barney can move, but only in particular ways. Since our goal is to explore the "standard platform" approach, we chose not to modify the devices, although of course that is frequently appealing. (Actually, we introduced one minor modification. Since our Barney development was going on in a public space, we modified Barney so that his audio output could be sent to headphones rather than a loudspeaker. His cheery exclamations of "Let's have some fun!" are not greeted with quite the same joy and surprise when you hear them coming from the desk next to you seventy times a day.)

The second down-side is that the devices are sold in toy stores as end-user platforms, not development platforms. In line with our "commodity device" principle, our work involved discovering as much as possible about the remote operation of Barney without the benefit of developer documentation.

FIGURE 1: Barney and Barbie.

Exploring Barney's Technology

In "PC Mode," ActiMates Barney communicates with desktop computer software via a radio transceiver connected to the computer's MIDI port. Traditionally, these software titles are educational games, such as "Fun on the Farm with Barney" or "Barney Goes to the Circus," designed as learning experiences for children ages two and up. Building our own applications suitable to using Barney as a Programmable Embodied Agent required uncovering the details of the Barney MIDI protocol.

The challenge we were faced with, then, was to experimentally determine the protocol by which Barney is programmed and controlled. The details of our investigations are not relevant here; primarily we spent time watching how the existing Barney programs communicate with the doll. Using both hardware and software approaches, we were able to record "conversations" between the two entities that eventually we were able to study and understand. We could then use these mechanisms in our own applications.

Barney's Wireless Protocol

Essentially, the Barney protocol consists of various different packets, each encoded as MIDI messages. Barney packets range in length from three to thirteen bytes, each of which has a clearly defined structure. Examples of outgoing packets (to Barney) are system reset, flow-control packets, packets controlling the doll's movements, voice data, and protocol handshaking packets. In the opposite direction, Barney also uses flow control and handshaking packets, in addition to packets for describing the state of the sensors in his eye and paws.

In trying to decode the protocol to build our own application framework, one frustrating feature of the Barney protocol is that all packets, particularly those containing data (i.e., motion and voice) contain checksums. Making any changes to the recorded packets produces invalid ones that the Barney hardware simply ignores, so understanding the checksums was a necessary precursor to controlling Barney's behaviour. Furthermore, because the Barney checksums are different for each type of packet, we had to work out each one individually

Challenge/Response Handshaking

Perhaps the most interesting feature of the Barney protocol is a challenge-response that Barney initiates as part of the handshaking sequence that takes him from "stand-alone mode" to "PC mode." The controller device broadcasts a signal looking for Barney devices, and each Barney device within range responds with a 4-byte "challenge" sequence. When the controller responds with the correct response for the offered challenge, a connection is established between the two and the Barney doll is "on-line." Before we could develop software to drive Barney, we had to derive the correct algorithm for generating appropriate responses for the challenges our Barneys would generate.

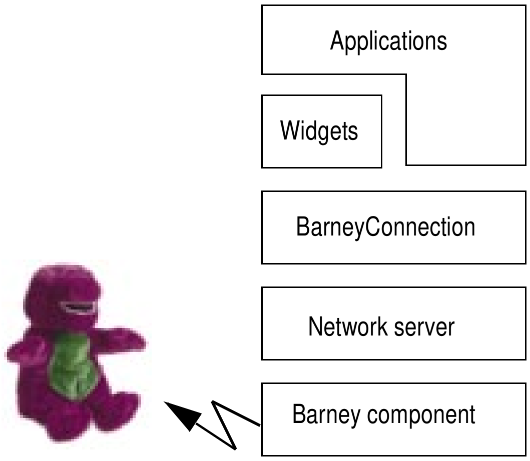

FIGURE 2: The Barney “Protocol Stack”

As it happens, calculating the 4 byte response is mostly a matter of reordering the challenge bits (also 4 bytes), a simple procedure. We speculate that the challenge/response unlikely meant as a security precaution, but rather as a way to allow multiple Barneys to coexist in the same room. The Barney software only completes the handshake with one Barney, so another child can continue to play with his toy in “stand-alone mode” (besides, all Barney’s responding to the software in unison, singing the same song and making the same motions, might be rather frightening).

Once Barney is “on-line” (in communication with a transmitter), a periodic “keep-alive” signal is required from the transmitter; if Barney does not receive this signal for a period of time, then he will revert to “stand-alone” mode.[2]

Voice Encoding

The voice encoding is the one feature of the Barney protocol that we were unable to understand fully. Various clues led us to conclude that the doll uses Linear Predictive Coding (LPC) for its speech; however, without knowing how the LPC parameters are encoded into the Barney protocol’s voice packets, we could not make Barney say arbitrary sentences[3]. Our solution for PEA applications is to use words and phrases that Barney already knows. One advantage of this is that it preserves the character – using the stock words and phrases means that Barney not only always sounds like Barney, but he always says the sorts of things that Barney would say.

Programming Barney

Based on what we learned about the Barney protocol, we built several layers of software infrastructure on which to construct PEA applications. The structure of the software system is shown in figure 2.

We implemented the low-level control protocol as a Borland Delphi 3 component. The component provides a simple interface to the application programmer who can move Barney’s arms and head, ask him to speak a pre-recorded sound file, and be notified of events such as Barney’s eyes being covered or a hand being squeezed.

One such application used for debugging exercises all the aspects of the protocol in the form of a Barney control panel (see Figure 3). The “Init” button performs the handshake, the sliders move Barney’s limbs, and the remaining buttons play sound files in various formats. Sensor events appear in the “Messages” text field as they occur.

Using the Barney component, we also wrote a “Barney server” which provides access to Barney’s functionality remotely through a TCP/IP network stream. Though the server allows direct connections through programs such as telnet, it serves primarily as a gateway for applications written in other high-level languages.

In fact, all of our PEA applications are written in Java using a Barney Connection class and listener interface, which speak to the Delphi server over the network. The BarneyConnection class encapsulates Barney’s control behaviour and allows applications to control Barney’s output functions (movement and speech), while the Listener interface allows applications to be informed of input activity (use of the paw sensors and covering or uncovering his eyes). With this framework in place, the PEA applications are lightweight, portable, easy to build, and seamless to integrate—like the agent itself—into one’s computing environment.

The Barney Widget Set

When we started building PEA applications for Barney, we encountered certain common idioms that could be applied across a range of interactive uses. This discovery should perhaps not have come as a surprise; since our intuition was that PEAs could be used as generic I/O devices, then it makes sense that certain application behaviours could be factored into “widgets.”

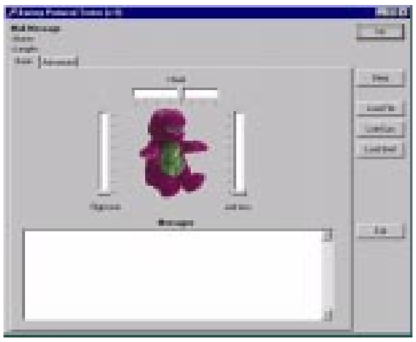

FIGURE 3: The Barney Control Panel

2.Conversely, as long as this signal is sent, Barney does *not* go into standalone mode. This means that an application can be written which explicitly prevents Barney from singing songs and playing games. People seem to find this “Barney Cone-of-Silence” a highly compelling application.

3.Linear Predictive Coding is a means of encoding speech information using a predictive mechanism that allows the next sample to be derived in part from the preceeding ones, resulting in a high compression factor. LPC coefficients are based on formant modelling. However, unravelling the precise encoding of the LPC coefficients into the speech control packet format was beyond the scope of this project.

Notifications

Frequently, our applications use Barney to notify the user that something has happened. In order to be sure to attract the user's attention, a number of our applications would use a generic behaviour that essentially caused Barney to move about randomly. Here, we are taking advantage of the limited range of motion in which Barney can engage. Since he can only move his arms and head, and since their motion and range is highly constrained, pretty much any motion looks "Barney-like" and maintains the "character" that we are attempting to exploit. The randomness does lend an air of agitation, however, which was the effect we wished to achieve.

Indicating Values

In some applications, we wanted Barney to be able to notify the user not only of some particular event, but also to give a sense of a quantity. To give a rough quantitative indication visually, we would use Barney's arms to indicate a value, moving them up and apart for large values, and down and together for smaller ones.

Counting

To use Barney to control devices, we frequently wish to use him as an input device. The sensors in his eye and paws can be used to trigger events, but we also needed to be able to group these events into more meaningful events. For instance, one widget is a counter for inputting numbers. As you squeeze Barney's left hand, he counts the squeezes out loud. Once you reach the number you want, you squeeze his right paw ("return") to select the number. Squeezing his foot will count back down again ("backspace").

Waiting

Sometimes the events we want to have Barney tell us about concern the completion of some task. Essentially, we delegate the responsibility of watching the task to see when it is complete to the PEA, who waits for completion and the signals it to the user. If the PEA is completely motionless while it's waiting, this can cause confusion. We wanted to give unambiguous feedback of a task in progress. Our "processing" widget involves moving Barney's from side to side slowly until the task is done; normally, when the task is completed there will be some explicit acknowledgment of success.

Exploiting the Character

Some common behaviours come not from our design, but from the character itself. This was an important aspect of our original approach. We wanted to be able work with and exploit the inherent and identifiable elements of the character the PEA presents. In the case of Barney, this means not only his generally cheery outlook on life (although none of our applications have so far involved singing songs), but also certain characteristic phrases. For instance, Barney likes to say, "Super-dee-duper"[4] and so a number of our applications use this phrase to indicate success.

Although these widgets emerged as an unanticipated side-effect of our implementation efforts, we have found them to be of considerable value in at least two sense. The first is that they provide a convenient encapsulation of application behaviour that makes it easier for application developers to get their applications integrated with Barney as an I/O device. The second is that they ease interactions for end-users by providing identifiable output behaviours and input interactions which are common across applications. These are, of course, the sorts of benefits which we would associate with conventional widgets in a UI toolkits (scroll bars, menus, etc.); what was unexpected to us was the extent to which these same notions would carry over to an unconventional interaction device like Barney.

	Channel (synchronous)	Proxy (asynchronous)
Person	*e.g. Barney Pals*	*e.g. Office Guardian*
Device	*e.g. Printer Monitor*	*e.g. Telephone Minder*
Event	*e.g. Build Manager*	*e.g. Meeting Minder*

FIGURE 4: An Application Framework

A FRAMEWORK OF APPLICATIONS

We have developed a range of applications of Barney as an example of a Programmable Embodied Agent. In doing this, we have been using an informal framework of applications which helps us organise and explore the space of applications. The framework is outlined in figure 4..

The framework is organised across two dimensions. The first concerns the style of communication in which Barney engages, either synchronous or asynchronous. The synchrony does not concern Barney's interaction with the user, which is clearly always synchronous, but with the occasion of the information conveyed. This distinction will be seen in more detail as we go through examples. We characterise those occasions on which Barney conveys synchronous, current information as ones when he acts as a *channel*, and those in which the information is presented asynchronously as ones when he acts as a *proxy*.

The table presented in figure 4 gives examples of applications that fit into each of these categories. We will encounter these and other applications as we step through the various cases.

Channel/Person

The "channel/person" category covers those occasions when the PEA acts as a proxy for another user in real-time. It can convey information directly from another person. For example, we have implemented two-way communication in the channel/person category in an application called "Barney Pals."

Two Barney PEAs, in separate offices, are connected via a network[5]. The software is configured so that input on one doll is mirrored as output on the other doll. For instance, if I squeeze the right paw on the doll in my office, the right arm on the doll in the remote office will move up; if I squeeze

4. "Super-dee-duper" is a trademark of The Lyons Group, L.P.. No, really, it is.

5. More accurately, two Barney radio controllers are driven by network-connected applications.

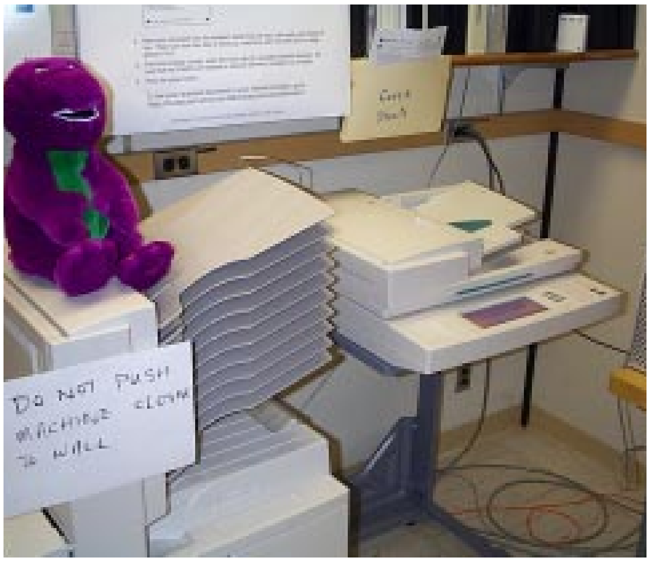

FIGURE 5: Barney as an interface to Document Services provides a link to otherwise invisible network processes.

the right foot, then the hand moves down. The left side can be controlled similarly, and eye covering can be used to cause speech generation at the remote side.

The result of this is an abstract communication channel, such as the "rollers" of Brave and Dahley [1997] or the "HandJive" device of Fogg et al. [1998] supporting synchronous communication between two users at a distance.

Channel/Device

Although my work is centered in my office and at my workstation, I make use of and rely upon devices spread throughout the building. Perhaps the most obvious is a printer, although many other devices, both physical and virtual, populate the working world. Barney can provide a way of making these remote and virtual devices accessible within my environment.

One simple case is the Print Monitor. The PEA software detects when I have sent a job to the printer, and then monitors the printer device to watch the progress of the job. Barney's head moves from side to side while the job is waiting in the printer queue, and a speech signal tells me when my print job is complete. Another signal warns me if a problem develops with the printer that requires my attention. Since our current infrastructure does not provide for arbitrary speech generation, we rely on digitised samples; in context, however, phrases such as "Why, thank you," "That was fun!" and "Please try again" function effectively as cues for submission feedback, success and failure.

As another example, we have configured a Barney to act as a feedback channel for part of our internal document management infrastructure ("Burlap"). One form of portal to this infrastructure is a digital scanner which will automatically scan and OCR input documents, placing the results in a file in my home directory. The virtual device – the OCR engine – has no local embodiment when I'm standing at the scanner. The "Burlap Barney" application acts as a feedback device, informing me about changes to the status of my scan job, so that I can know it is complete before I walk away. Since users generally interact with this system through a document device (rather than a desktop computer), we can use Barney as an extra I/O channel (figure 5). This provides a route into the otherwise invisible technology behind the device; covering Barney's eyes causes him to report on the state of the network of servers comprising the Burlap service.

Channel/Event

As an extension of the device model, a PEA can also act as a channel for becoming aware of events that happen at random in my computational environment – ones which are not associated with a particular device or action I have taken. One simple example of this is an email monitor ("Barney Biff") in which the PEA monitors my mailbox and informs me when email has arrived. This application uses the "arms-signal-quantity" widget behaviour, but also provides other notifications if any of the mail appears to be important. Another example is the "Build Master" application, which allows Barney to monitor a source code repository containing the up-to-date version of a large software project. In this application, Barney plays two roles. First, he can act as a channel to inform me of new changes that have been checked into the repository by my colleagues; whenever new code is checked in, Barney announces its arrival. Barney's second role is to perform automated tasks; once code has been checked in, the Build Master application attempts to verify the code that has been checked in, and reports success or failure to tell me whether the new code is good or not.[6] In terms of our framework, this application begins to blur the distinction between channels and proxies, since by informing me of activity on a device (the code repository), it also provides me with information about the activities of others, helping me to coordinate my activity with them [Dourish and Bellotti, 1992].

One feature of all of these "channel"-based applications is that they report essentially *out-of-band* information. In other words, the information that these applications convey does not arise in synchronous response to a specific action I took, but rather results from changes in the environment. These are the sorts of events for which pop-up windows on the computer screen are particularly annoying. A PEA, separate from my computer system but acting in concert with it, can provide a valuable "second channel" which need not divert me from whatever I am doing on my workstation at the time.

Proxy/Person

The "proxy" side of the framework refers to asynchronous information-passing. In these cases, the PEA will typically act as an interface to some kind of mechanism for relaying information across time.

As an example of the "proxy/person" combination, a PEA can be configured to deliver a message to people who visit my office while I'm away (acting as a proxy for me). In the case of Barney, this can be triggered by a change in light

---

6. This is another case where we can take advantage of the particular character of the PEA (in this case, Barney). When Barney tells you that the build has broken, he can sound really personally hurt.

level when people come to my office (since Barney has a light sensor in his eye for playing peek-a-boo). In our current framework, this use is limited by the fact that arbitrary speech generation is not yet supported.

Proxy/Device

As well as acting as a proxy for other people, a PEA can also act as a proxy for other devices. For example, one application allows Barney to monitor the telephone system. The telephones already maintain a log of calls that I missed while I was out of my office, although in practice I rarely remember to check it. However, under the “Telephone Minder” application, when I come back to my office, Barney’s raised arms indicate that I missed telephone messages; and if I squeeze his paw, then he speaks the phone numbers of the callers I missed while I was away.

Proxy/Event

The final component of the framework is the area where the PEA acts as a proxy for events, delivering them asynchronously. One opportunity here is for the PEA to report who has visited my office while I was away, or to let them record messages. These applications require integration with other components in our environment, such as location sensors or portable devices (such as an IR-equipped PDA to “beam” a signal to the PEA); portable wireless devices such as the PEA prototypes become more powerful when placed in an environment rich in them.

Our application focus has been on the “channel” applications, although some others (such as the Telephone Minder) have been prototyped. We are convinced, though, that the use of a PEA as a means to interact not only with people but also with other devices in the environment, and to act as a site of asynchrnous communication, is a rich source of potential new applications, and we are interested in exploring this area of the framework in more detail.

FURTHER WORK AND OPPORTUNITIES

The work described in this paper was conducted as a brief exploration of the opportunities offered by a set of newly available consumer devices. In the space of only a few months during the summer of 1998, we were able only to begin this exploration, especially since a sizeable amount of work was required to uncover the protocols by which Barney could be controlled, and assemble a software infrastructure over which applications could be constructed.

FIGURE 6: PEA applications exploit both tactile interaction and the use of a computational channel that is separate from the desktop workstation.

This leaves a number of avenues as yet unexplored. Our biggest disappointment was that, in the time available, we could not decode enough of the LPC mechanism to get Barney to speak arbitrary phrases. This is the single most significant potential advance for the development of future PEA devices, and we hope to be able to work more on this in the future, building on the groundwork laid so far.

At the same time, of course, we have yet to explore the control and use of the Barbie device. Although Barbie a less versatile device than Barney (both because she must be sitting at her workstation to receive signals, and because she cannot move her limbs), we are interested in the potential for a combination of devices. In particular, having multiple devices allows us to move further along the route of associating specific characters with different channels of information or different styles of interaction in a single workplace.

Our primary focus now, however, is on exploring the space of potential applications. The true test of this technology lies in its deployment in everyday working settings. We are investigating opportunities to deploy and study these devices in settings amongst our own colleagues, in particular as part of an ongoing research investigation into augmented reality and interfaces exploiting novel forms of direct, tactile interaction.

CONCLUSIONS

Although there has been considerable interest over the past few years in the opportunities for direct physical interaction with computational proxies and in the use of character-based interface agents, attempts to combine the two have been confounded by a variety of practical factors. Physical character-based interactive devices, or *Programmable Embodied Agents*, are expensive to develop and to deploy. However, recently a range of consumer devices have come onto the market that hold considerable promise as platforms for research into this new style of interaction.

We are interested in the potential uses of these devices as research platforms, and have been working to create a software infrastructure for the development of PEA applications. So far, we have been working particularly with Microsoft’s ActiMates Barney. We have developed a set of tools for controlling Barney, and for developing applications which exploit Barney as a generic interaction device. Using these tools, we have been exploring an initial framework of potential applications that can take advantage of the fact that PEA devices embody specific “personality traits,” afford direct physical interaction, and constitute a computational channel that is separable from the traditional desktop computer.

In this paper, we have introduced and explained the ideas behind this line of research, and presented Programmable Embodied Agents as arising at the nexus of two recent lines of HCI research, on tangible physical interaction and on character-based interfaces. We have demonstrated that we can build applications that capitalise on the values of each of these lines of investigation, integrating the immediacy and physicality of tangible interaction with the compelling interaction style of character-based interfaces. We have presented an initial framework for exploring the space of potential applications and have populated this space with a range of working prototypes.

We have begin to open up opportunities to exploit "consumer platforms" for PEA research. Although there is much work still be done, our applications show early promise. In particularly, they demonstrate that these cheap consumer devices can be used as research platforms for studies of embodied interaction, and hope that these results will provide a basis for a broader-based investigation of Programmable Embodied Agents.

ACKNOWLEDGMENTS

The work described in this paper was conducted while Michael Kaminsky was a summer intern in the Computer Science Lab of the Xerox Palo Alto Research Center. We would like to thank Karin Petersen and John White for their enlightened tolerance, Ron Frederick, Ralph Merkle and Roy Want for their contributions of expertise, and Melinda Stelzer for the inspired gift that started all this in the first place.

REFERENCES

1. Beth Adelson, "Evocative Agents and Multimedia Interface Design", *Proc. Human Factors in Computing Systems CHI'92* (Monterey, CA), ACM, New York, 1992.
2. Scott Brave and Andrew Dahley, "inTouch: A Medium for Haptic Interpersonal Communication", *Proc. Human Factors in Computing Systems CHI'97 – Extended Abstracts* (Atlanta, GA), ACM, New York, 1997.
1. Jeremy Cooperstock, Koichiro Tanikoshi, Garry Beirne, Tracy Narine and William Buxton, "Evolution of a Reactive Environment", *Proc. Human Factors in Computing Systems CHI'95* (Denver, CO), ACM, New York, 1995.
1. Paul Dourish and Victoria Bellotti, "Awareness and Coordination in Shared Workspaces", *Proc. Computer-Supported Cooperative Work CSCW'92* (Toronto, Canada), ACM, New York, 1992.
4. Allison Druin, "Building an Alternative to the Traditional Computer Terminal", Masters Thesis, MIT Media Laboratory, Cambridge, MA, 1987.
5. George Fitzmaurice, "Situated Information Spaces and Spatially Aware Palmtop Computers", *Communications of the ACM*, 36(7), 38–49, 1993.
6. George Fitzmaurice, Hiroshi Ishii and William Buxton, "Bricks: Laying the Foundations for Graspable User Interfaces", *Proc. Human Factors in Computing Systems CHI'95* (Denver, CO), ACM, New York, 1995.
7. BJ Fogg, Larry Cutler, Penny Arnold and Chris Eisbach, "HandJive: A Device for Interpersonal Haptic Entertainment", *Proc. Human Factors in Computing Systems CHI'98* (Los Angeles, CA), ACM, New York, 1998.
8. Hiroshi Ishii and Brygg Ulmer, "Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms", *Proc. Human Factors in Computing Systems CHI'97* (Atlanta, GA), ACM, New York, 1997.
9. David Kurlander and Daniel Ling, "Planning-Based Control of Interface Animation", *Proc. Human Factors in Computing Systems CHI'95* (Denver, CO), ACM, New York, 1995.
10. Brenda Laurel, "Interface Agents: Metaphors with Character", *The Art of Human Computer Interface Design* (ed. Laurel), Addison-Wesley, Reading, MA, 1990.
11. Brenda Laurel, Tim Oren and Abbe Don, "Issues in Multimedia Interface Design: Media Integration and Interface Agents", *Proc. Human Factors in Computing Systems CHI'90* (Seattle, WA), ACM, New York, 1990.
12. William Newman and Pierre Wellner, "*A Desk Supporting Computer-Based Interaction with Paper Documents*", Proc. Human Factors in Computing Systems CHI'91 (New Orleans, LO), ACM, New York, 1991.
13. Byron Reeves and Clifford Nass, "*The Media Equation*", Cambridge University Press, 1996.
14. Thomas Rist, Elisabeth Andre and Jochen Muller, "Adding Animated Presentation Agents to the Interface", *Proc. Intelligent User Interfaces IUI'97* (Orlando, FL), ACM, New York, 1997.
15. Erik Strommen, "When the Interface is a Talking Dinosaur: Learning Across Media with ActiMates Barney", *Proc. Human Factors in Computing Systems CHI'98* (Los Angeles, CA), ACM, New York, 1998.
16. Mark Weiser, "The Computer in the 21st Century", *Scientific American*, 256(3), 94–104, 1991.
17. Pierre Wellner, "The DigitalDesk Calculator: Tactile Manipulation on a Desktop Display", *Proc. Symp. on User Interface Software and Technology UIST'91* (Hilton Head, NC), ACM, New York, 1991.