Multimodal Programming Environment for Kids: A "Thought Bubble" Interface for the Pleo Robotic Character

Kimiko Ryokai
School of Information
Berkeley Center for New Media
University of California Berkeley
Berkeley, CA 94720 USA
kimiko@ischool.berkeley.edu

Michael Jongseon Lee
School of Information
University of California Berkeley
Berkeley, CA 94720 USA
michael_lee@berkeley.edu

Jonathan Micah Breitbart
School of Information
University of California Berkeley
Berkeley, CA 94720 USA
breitbart@ischool.berkeley.edu

CHI 2009, April 4 – 9, 2009, Boston, MA, USA
ACM 978-1-60558-246-7/09/04.

Abstract
We introduce a mixed physical and digital programming environment for children to control robotic characters. We present our design rationale, our initial prototype, report the results from our initial evaluation, and discuss ongoing work.

Keywords
Tangible, programming, robotic toys, children

ACM Classification Keywords
H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—artificial, augmented, and virtual realities.

Introduction
Robotic toys and animatronics are gaining popularity and starting to be widely available on a consumer level (e.g., by WowWee, Sony, Hasbro, Omron). Ugobe's Pleo [5] is one such consumer level robotic dinosaur toy targeted for children aged 8 and up. Pleo is designed to be a friendly and curious baby dinosaur that exhibits a variety of life-like movements. Out of the box, Pleo responds to touch and gives an impression of learning by reacting to an individual

figure 1. UGOBE's robotic baby dinosaur, Pleo.

owner in unique ways. However, in reality, Pleo is not equipped with any learning mechanism but simply runs complex combinations of canned responses that emulate intelligent behavior.

Our informal observations of children (age 5-10) playing with Pleo showed that children cuddled with Pleo, and like with a real pet, they wanted to teach Pleo special tricks. Instead of passively responding to what is already programmed into Pleo, we want to give children the opportunity to actively create and control Pleo's behaviors.

Pleo is an open source platform, allowing technically capable hobbyists to customize and program their original behaviors beyond the preprogrammed actions

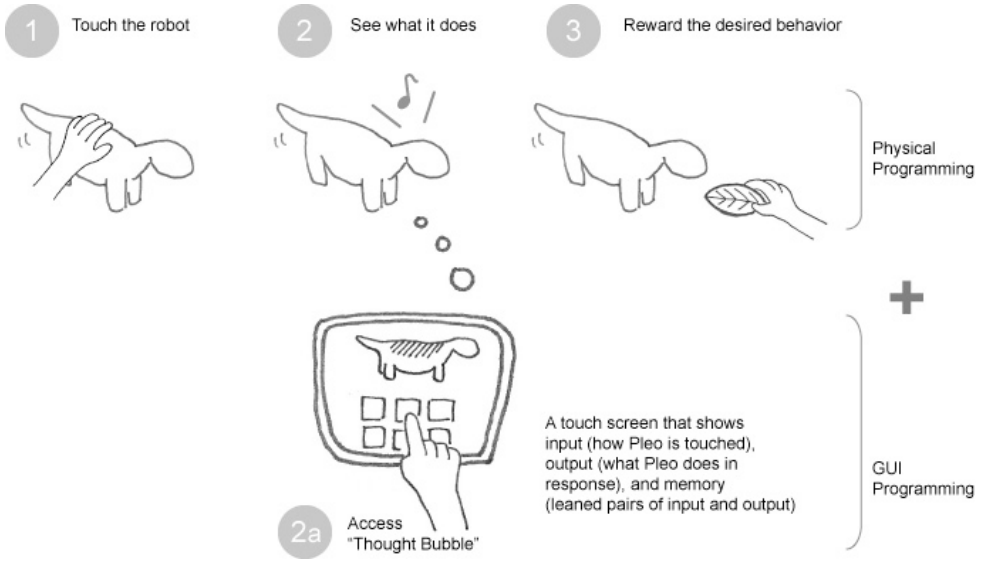

figure 2. Combining the Physical and the GUI programming using the "thought bubble" of the robotic character as a metaphor.

(e.g., singing original songs or performing customized dances). However, in order to produce such customized expressions, one needs a relatively high level of technical competency (e.g., knowledge of the C programming language and PAWN scripting). In this regard, our goal is to create an environment that allows children to easily program and control the creature's behaviors.

Pleo "Thought Bubble": Combining the Physical and Virtual

During our informal play sessions with an out-of-the-box Pleo, children noticed that Pleo was doing something in response to their physical touch. However, it was not so clear to the children which part of their touch was recognized by Pleo, and what Pleo would do in response. Therefore, we wanted to give children real-time access to what goes on in the robotic dinosaur's head so that they could better understand the process as well as what behaviors they can change or control.

Illuminating the thought process: Thought Bubble

The metaphor we built on is a "thought bubble" of the robotic character, through which children tap into the thought process of the character. We use a touch screen in combination with Pleo to show what is happening to Pleo in terms of *input* (how Pleo is touched), *output* (what Pleo does in response), and *memory* (learned pairs of input and output), in real time.

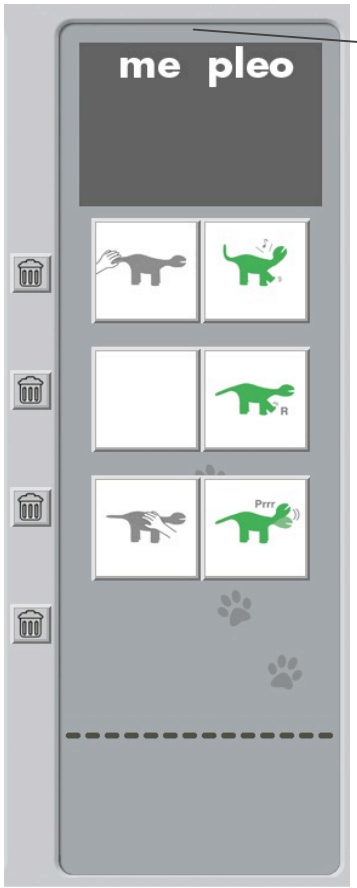

figure 4. A close up of the "memory bank" showing some behaviors have a conditional statement (e.g., "when the tail is touched"), and some have not.

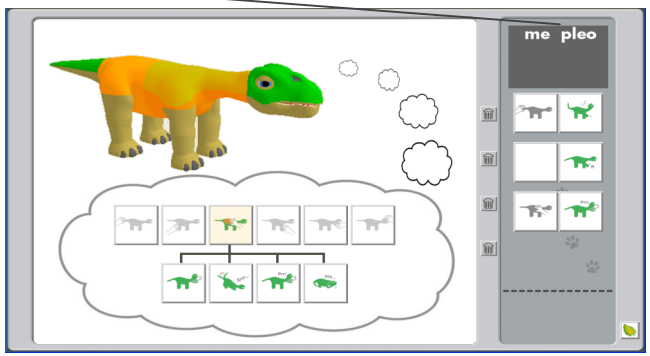

figure 3. Touch screen interface to Pleo's Thought Bubble

MONITORING THE STATUS AND POSSIBLE ACTIONS

Touching different body parts of the robotic Pleo immediately highlights the chosen area of the Pleo image on the screen and shows available actions tied to that area (see figure 3). At the same time, Pleo starts to physically perform the listed actions in sequence. For example, if Pleo's chin is touched, Pleo goes through purring, tail wagging, mooing, and singing in sequence, as long as the chin is being touched. These actions performed by Pleo are also highlighted on the "thought bubble" screen. The desired behavior may be positively reinforced by either feeding the robotic Pleo its physical leaf, or by pressing the virtual leaf icon on the touch screen interface. For example, if Pleo's singing behavior while being scratched on its chin is rewarded with the leaf, Pleo associates the chin scratch with the singing behavior. Therefore, the next time Pleo's chin is touched, Pleo associates that input as the cue to start singing.

PROGRAMMING SEQUENCES

"Learned" behaviors are saved in Pleo's "memory bank" (see figure 4). This memory bank serves as a repository showing which tricks Pleo has been taught. The left column labeled "me" represents input, i.e., what the user does (e.g., touching Pleo's chin, head, legs, back, or tail) and the right column labeled "pleo" represents output, i.e., what Pleo does in response to the input (e.g., singing, wagging tale, stretching, etc.) Touching a body part associated with the top-most pair will cause the last learned trick to execute.

The items on the left "me" column can be removed by pressing the trashcan icon. Once the input part of the pair is removed, the action is automatically played directly after the preceding pair. For example, figure 4 shows the topmost pair in the queue would initiate by touching the tail. If the tail is touched, Pleo will start to sing. Immediately after singing, Pleo will wiggle his right leg, as the "me" column is missing and Pleo does not have to wait for any input from the user. After wiggling its right leg, Pleo will wait for the user to touch its back to execute the next pairing. Using the trashcan tool on a lone action sequence will "undelete" the associated body part, returning the pair to its original state. As such, the memory bank allows basic conditional (procedural) as well as sequential behavior programming.

Combining Physical Interaction and GUI

We wanted to allow multiple entry points to interaction with the robotic toy by providing both physical and virtual interfaces. The child may choose to program 1) with the physical toy only, ignoring the screen interface altogether and focusing on physical interactions with Pleo, 2) on the screen interface only, directly

controlling Pleo's behaviors through the GUI only, or 3) using a combination of the physical interface and GUI.

Related Work

A variety of programming environments for robotic creatures have inspired our work. Crickets [9] and commercially available LEGO Mindstorms [3] are systems of physical LEGO blocks, sensors, actuators, and programming environment that allow children to create their own programmable robotic creations. They invite creators to move between the physical world of model creation with blocks and the virtual world of programming. Topobo [8] is a new construction kit with kinetic memory that invites young children to build 3D creatures and program their movements by directly twisting and turning the physical model. Guo and Sharlin presented a system that allows a person to control a robotic character, Aibo, via Nintendo Wii game controllers [1]. By combining the physical and the virtual, our approach is to have children decide where to focus their actions and allow them to easily move between physical interaction and virtual control.

System implementation

The "thought bubble" interface was developed in Python using the pyGame library [6]. Communications with Pleo are achieved by sending serial commands over the USB port using the pySerial libraries [7].

For our initial prototype, we chose four different possible behaviors that Pleo can perform for each of the six stimulus points on Pleo (tail/backside, body/back, back legs, front legs, top of head, chin), for a total of 24 possible behaviors. We chose behaviors that would be memorable yet relatively brief (between four and ten seconds) that include a combination of movements and sounds. In identifying actions to include, we used the Dino-MITE [2] and MySkit [4] software applications. Dino-MITE allows monitoring of Pleo's COM port connection and joint positions, as well as sending commands to Pleo. Dino-MITE can also list Pleo's built-in behaviors. MySkit is a performance editing program that allows users to create "skits" by manipulating Pleo's joint positions. The behaviors we chose to include are a combination of the built-in behaviors identified through Dino-MITE, complete and modified behaviors from the MySkit library, as well as custom designed behaviors built in MySkit.

The custom and modified MySkit behaviors were loaded onto Pleo's SD card. Each behavior has a custom command that our program uses to execute behaviors through the serial USB connection to Pleo. After identifying the behaviors used in our system, we created icons corresponding to each behavior for the "thought bubble" interface. Our program associates each behavior command with the corresponding icon.

Evaluation

We were interested in investigating whether or not children understand the "thought bubble" interface as a tool to access and control Pleo's behaviors. Another interest was in learning how children's focus shifted back and forth from the physical interface to the GUI.

Participants and Methodology

Nine children between the ages of 5 and 8 participated in our study. Three groups of children played with our system in dyads. One group of children played with the system in a triad. The investigator first briefly introduced Pleo and the thought bubble screen to the children and then left the system for the children to

figure X. Two children interacting with both the Pleo robot and the "thought bubble" interface on the touch screen. In general the children started with the robot interaction, but then moved to a combination of physical interaction and GUI.

play by themselves. Each group played for 20-30 minutes with our system. Since it was difficult to gain ready access to Pleo's touch sensors, for our initial observation, we decided to use a "Wizard of Oz" approach where one of the investigators watched the children's interactions and filled in the gap by sending the touch screen interface the relevant information. A simple key press by the wizard or touch of an icon by a child triggers the program to send the corresponding behavior command to Pleo which in turn causes Pleo to act out that behavior.

Results

The children understood the relation between the "thought bubble" screen and Pleo's action with respect to which part was being touched. They also understood that touching appropriate parts of the screen could actively control Pleo's behaviors. Teaching Pleo behaviors had the children very engaged throughout the process. The children also remembered how to access certain types of behaviors, e.g., touching the head to access and activate the "Moo" sound of Pleo. The children eagerly showed each other different tricks Pleo could perform, "Look what he can do! [as they touched the thought bubble screen to navigate and activate desired behaviors]". Many pairs started by focusing on physically touching Pleo and gradually moved on to interacting with the GUI once they had a better understanding of the system. None of the children completely ignored the GUI screen to focus solely on physical play with Pleo. The children did seem to understand the right hand region of the screen to be the "memory bank." When asked by the investigator to explain what they thought the right hand region represented, the children answered, "That's what Pleo knows." However, the children did not use the editing function of the "memory bank" to create sequences of behaviors. Instead of waiting for the "sequence in memory," the children used the left side of the screen to directly control and reinforce Pleo's behaviors. They seemed to interpret the "memory bank" as a log or history, and not something to be acted upon. Therefore, no procedural programming was observed. The function of the memory bank should be made clearer in the future and should perhaps include a "go" function to cycle through the list without having to touch the body part.

At the beginning of the play session, the children seemed to understand the leaf as a reward they give to Pleo to reinforce desired behavior. However, the children quickly wanted to use the leaf to feed Pleo. In a future version of the system, we could provide two types of food: one to feed Pleo and the other to positively reinforce learned behaviors (like a "treat").

Technical Limitations

We did observe the children touching different body parts at once or in rapid succession. Since Pleo is designed to complete an action before it can execute anything else, it will be unable to support commands in rapid succession. To avoid complications with Pleo's communication buffer, we plan on implementing a behavior monitor that allows for interruptions and can quickly update new behaviors.

Discussion and Future Work

The evaluation of our initial design and prototype provided us with ideas for future design and implementation improvements. Specifically, the design of the "memory bank" needs refinement. The interface should invite children to edit and manipulate behaviors.

One idea is to make the list more like pages of a storybook, showing a chain of events (e.g., and then this happens, and then this happens after this, etc.).

Even with some technical glitches, the general premise of having a physical robot and accessing its thought bubble to control the character seems promising. In response to the investigator's question, "What does this [pointing at the "thought bubble" screen] do?" one child responded, "You can *jump to it* [pointing at different behaviors on the screen] rather than [gestures touching of Pleo robot]." When the investigator asked, "Do you need both the screen and Pleo?" a couple of children answered, "You can just take Pleo with you, but [without the screen] it would be harder to see what Pleo is thinking."

Conclusion

We have presented a mixed physical and digital programming environment for children to control robotic characters. We have given children real-time access to what goes on in the robotic dinosaur's head so that they could better understand the process as well as what behaviors they can change or control. Children knew that they could cuddle and interact with Pleo just like a regular stuffed animal, but it was also clear to the children which part of their touch was recognized by Pleo, and what Pleo could do in response. The "thought bubble" interface offered children an extra lens through which they could tap into the process of the robot's activity, and also direct its behavior. We are continuing to improve the interface and plan to conduct a longer study to investigate types of storytelling play children may engage in with programmable robotic characters.

Acknowledgments

We thank the children and their parents, Caleb Chung, John Sosoka, and Barbara Barza from UGOBE, and Matt Bauer for his help with Dino-MITE.

References

[1] Guo, C. and Sharlin, E. Exploring the use of tangible user interfaces for human-robot interaction: a comparative study. In *Proceeding of CHI '08*. ACM Press (2008), 121-130.

[2] Dino-MITE http://www.bauerindependents.com/SUBMAIN/dinomite.htm)

[3] LEGO MindStorms http://www.mindstorms.lego.com/

[4] MySkit http://www.dogsbodynet.com/myskit/index.html

[5] Pleo. http://www.pleoworld.com/

[6] pygame game development library http://www.pygame.org/news.html

[7] pyserial http://pyserial.sourceforge.net/

[8] Raffle, H., Parkes, A., Ishii, H. Topobo: A Constructive Assembly System with Kinetic Memory, in Proceedings of *CHI '04*. ACM Press.

[9] Resnick, M., Martin, F., Sargent, R. and Silverman, B. Programmable Bricks: Toys to Think With. IBM Systems Journal, vol. 35, no. 3-4, pp. 443-452.