Steps Toward Child-Designed Interactive Stuffed Toys
Yingdan Huang and Michael Eisenberg
U. of Colorado Dept. of Computer Science
Boulder, CO USA
duck@cs.colorado.edu

ABSTRACT
Within the past decade, computationally-enhanced toys have become a staple of children's environments. In large part, this is due to the small size, robust operation, and low cost of embedded computing that enables computers (and associated electronic devices) to be included within toys of all descriptions. More recently, a variety of powerful technologies have emerged so that children can design their own computational artifacts: that is, small (and inexpensive) processors, sensors, and actuators have been developed that are well-suited to combination with "soft" materials such as textiles. This paper describes Plushbot, a system-in-development that allows children to create their own plush toys and stuffed animals, and to include computational enhancements within the toys that they create. Thus, Plushbot represents a step toward expanding children's creative design of their own interactive, computationally-enhanced characters. The paper describes the current state of the Plushbot software, shows a sample project created with the system, and describes plans for upcoming pilot tests with the system.

Categories and Subject Descriptors
K.3.1 [Computers and Education]: Computer Uses in Education – computer-assisted instruction.

General Terms
Human Factors.

Keywords
Plushbot, LilyPad Arduino, plush toys.

1. INTRODUCTION
The world of children's toys, over the past decade or so, has become closely interwoven with the development of small, affordable, embedded computation. There are numerous examples along these lines–among them, computationally-enhanced "talking books", globes, construction kits, and the like–but for the purposes of this brief introduction, it is worthwhile to focus on the inclusion of computation within "character" toys such as dolls, robots, and stuffed animals. A pioneering representative along these lines was the Furby toy [Cf. 10] which permitted children to interact in complex ways with an appealing creature whose programmed responses to handling were not immediately predictable. In a similar genre, toys such as the Aibo robotic dog [1] and (more recently) the "Dinosaur Train" interactive dinosaurs [8], among many others, reveal the possibilities of toys that sense different sorts of movement, handling, or external objects and respond in sophisticated ways.

The addition of computation to "character toys" reflects the work of an active and creative research community, and likewise reflects a rapidly evolving technological environment. Johansson's "Sniff" toy, for instance, is a delightful stuffed dog with an embedded RFID reader (as well as a microprocessor and motors) that can respond to objects with appropriate RFID tags. [7] The Topobo system [11] is perhaps better described as a construction kit than a character toy, but it permits children to assemble their own object (generally an animal-shaped object) from computationally-enhanced pieces, and to "program" the resulting structure by handling the toy and demonstrating movements that can be "remembered". Probo [12] is a remarkable "huggable robot", something like an elephant in appearance, with a wide range of sensory abilities and readable facial expressions. The Birdwatcher toy is a prototype stuffed duck equipped with an Aniomagic Schemer device (as well as a motor, LED light, and speaker); the toy can be programmed in complex ways without wires, merely by watching a pattern of flashing lights sent from a designated website [6].

These examples are drawn from a large, and expanding, realm of computationally-enhanced toys, both in the commercial and experimental realm. Still, the "traditional" toys of this sort (if such a term can be used) are, in effect, pre-built: that is, the child does not actually create the toy from craft materials or a design of her own choosing. The time, and technological environment, are now right to permit children to combine elements of craft creation with computational devices: that is, it is now possible for children to have the means to design, build, and program their own interactive toys.

This paper describes the current state of the Plushbot software system. Plushbot is conceived as an application with which children can plan and design their own computationally-enhanced stuffed toys. The system is arranged to work with the LilyPad Arduino kit [3], a collection of computational devices suitable for use with fabrics and conductive thread. An early version of the Plushbot system was described in [6]; this paper describes the current Plushbot (significantly advanced since the earlier description), presents a sample project done with the system, and concludes with a brief description of our upcoming initial pilot tests with middle school children.

2. PLUSHBOT: THE CURRENT SYSTEM
The basic idea behind Plushbot is that it supports the creation of computationally-enriched plush toys. Traditionally, there has been a thriving (though not huge) subculture of craft practitioners who design and create fabric patterns suitable for the creation of plush
toys; over the past several years, an application named Plushie [9] has been employed for this process as well. The Plushbot system is not nearly as extensive as Plushie in its treatment of the relationship between a 3D form and the set of 2D pieces that can be sewn into that form. Rather, Plushbot is designed with an eye toward the incorporation of computational elements into plush toy designs.

In outline, the Plushbot system is composed of two Web-based interfaces, both written in JavaScript. We describe these two interfaces in turn, in the course of a sample scenario for the creation of an interactive stuffed cat toy.

2.1. The Pattern Interface

The pattern interface is entirely new to this version of Plushbot, and focuses on the initial stages of pattern creation for a plush toy project. There are two ways for the user to create a pattern: (a) by using the various drawing tools provided (rectangle, circle, polyline and spline tools), or (b) by tracing (using the polyline tool) the outlines of a pre-existing pattern. Once a closed shape for a pattern piece has been created, the user can edit and modify it. For our sample scenario, we use an existing plush toy pattern named "Cute Kitty" [4], and we will add LilyPad Arduino components to this design to make an interactive version of the toy.

For a project such as this, in which we base our pattern on some pre-existing design, the interface supports the metaphor of "tracing paper": that is, we can imagine that the pre-existing pattern is underneath a piece of transparent paper. The user can move the pattern around on the screen to trace a particular shape in an unused area of the screen. Once a shape has been traced, Plushbot provides an "offset tool" to generate the markings for the seam threads within the outer perimeter of the shape (several of these outlines can be seen in Figure 1–for instance, within the large shape toward the left). A marker tool can also be used to annotate shapes with additional information–these marks can be employed to help guide the eventual sewing together of the pieces. Finally, once a pattern is completed within the pattern interface, it can be saved to the system's database for the next step (i.e., circuitry design).

![Figure 1: The pattern interface for creating or tracing a plush toy pattern.](image)

2.2 Playground Interface

The playground interface (see Figure 2) focuses on the synthesis of physical forms with computing. The first step in using this interface is to load a saved pattern; the pattern pieces may now be rearranged on the screen to facilitate the placement of computational components. Toward the right of the screen, one can see a variety of icons representing these components, and these icons can be positioned within the pattern; for instance, the user might choose to drag a LilyPad component onto one piece, and an accelerometer sensor onto another piece, as seen in the example of Figure 2. The two pieces can now be connected by either a polyline or spline tool, which (in this interface) produces a line that represents conductive thread. As can be seen in the figure, the conductive thread lines are drawn in chosen colors to facilitate identifying the continuation of a thread line from one piece to another; additionally, in the area between pieces, the thread line is drawn in very light gray. (This is done merely to highlight the continuity between pieces; only the darker lines on the pieces correspond to the actual sewing of physical thread on the eventual construction.)

Once the circuitry design is complete, an additional tool has been developed to output a HPGL file for a laser cutter, which will cut outlines of all pattern shapes and engrave traces on them. Traces include the bounding box of a LilyPad component, paths of conductive thread, the seam allowance of each pattern shape, and any informational marks that a user has made. These engraved traces are of tremendous utility during the physical construction stage.
Figure 2: The playground interface for combining a plush toy pattern with computational pieces (shown in icon form at right). Toward the top of the figure, one can see sets of buttons. The five red buttons are for retrieving and saving files of various sorts (including the HPGL file mentioned in the text); the eight blue buttons are for manipulating and editing pattern shapes; the five orange buttons are for placing and manipulating thread lines; the three purple buttons are for zooming (in and out) and clearing the view in the large canvas at left. In the figure, one can see that six computational elements have been placed on five distinct pieces; and plans for connecting these elements with conductive thread have been drawn on the pattern as well.

2.3 The Physical Construction Phase

Once the HPGL file for a project has been generated and pieces cut out, conductive threads are now sewn onto the pieces according to the markings produced in the playground interface. Figure 3 shows several of the pattern pieces for our cat project, with the threads sewn onto them. In some cases, as in the large piece shown in Figure 4, it is necessary to sew patches onto the piece between layers of conductive thread (this is to avoid allowing lines of conductive thread to intersect, causing a short circuit). The photograph of Figure 4 also demonstrates the next step in construction—i.e., sewing on the computational elements.

Figure 3. Several of the newly-cut-out pattern pieces with conductive thread sewn onto them.

Figure 4. A pattern piece with sewn-on patches to avoid allowing conductive threads to cross. For this particular piece, the next step has also been taken of beginning to sew on the computational elements themselves (the LilyPad processor, for example, is seen toward the center of the large piece).

The final phase of construction is the assembly of the pieces into a complete figure. (In point of fact, this last phase must be handled somewhat more carefully than usual for a computationally-enriched toy; for example, as pieces are sewn together and fiber stuffing added to the interior of the toy, it is important to ensure that interior conductive threads from adjacent pieces do not touch).
Figure 5 shows the final result—an interactive cat figure that plays music when its two paws are squeezed (the cat plays "Mary had a Little Lamb" when the left paw is pressed, and "You Are My Sunshine" when the right paw is pressed). The flower on the cat's head has a three-color LED that changes color when the figure's tail is shaken (an accelerometer sensor is attached to the tail). The LilyPad processor and power are inside the toy, but can be easily accessed by opening snap buttons toward the bottom of the figure; this permits the user to reprogram the toy.

3. ONGOING AND FUTURE WORK

Perhaps the most important, and immediate, plans for our work with the Plushbot system is to conduct initial pilot tests with children. During the spring semester, we will introduce the application to approximately 60 middle school children (ages 12-14); briefly, our plan is to have the children work in small groups of three to pursue their own projects similar in spirit (and scope) to the one described in the previous sections. Our first-order goals are simply to see whether Plushbot can be used and understood by these students, whether the activity of creating stuffed toys is motivating for the children, and what aspects of the interface need most urgently to be altered or extended.

In the somewhat longer term, one of the central questions in the development of Plushbot is how, and to what extent, to integrate LilyPad Arduino programming into the interface. Our current feeling is that it would be helpful to combine a programming environment into the application; very likely this would not be an environment of our own devising, but rather an existing or separately-developed programming environment that could then be integrated with the Plushbot system. The goal of this integration would be to allow children to write (at least simple) LilyPad programs and to simulate those programs within Plushbot (e.g., by seeing representations of their effects on the screen).

4. ACKNOWLEDGMENTS

The work described in this paper has been funded in part by the National Science Foundation under grant CNS0940484. We wish to thank Leah Buechley, Mark Gross, Jane Meyers, and Nwanua Elumeze for helpful conversations.

5. REFERENCES