CTRL_SPACE: Using Animatronics to Introduce Children to Computation

Andrew Sempere and Bakhtiar Mikhak
Grassroots Invention Group, MIT Media Laboratory
{andrew, mikhak}@media.mit.edu

Abstract

In this paper we present hardware and a software environment called CTRL_SPACE, specifically designed to introduce preliterate children to basic computational concepts. This is done through the control of an animatronic face whose individual components are used as an analogy for a programmable object. The software environment is visual, and the entire work is grounded in a critical historical view of computers and computation.

1. Introduction

With few notable exceptions [1,2,4,7], our notion of programming and computation belongs to a bygone era. The fairly recent availability of cheap, powerful computation allows us to spend more computational cycles on interface. In the process of rethinking what an interface to computational ideas means we uncover two critical points:

1. The historical trend in promoting "computer literacy" has maintained a focus on learning how to communicate in "computer language." The true power of computation, the ability to use computational thinking to solve problems, has taken a back seat to learning how to co-exist with technology.

2. This state of affairs is the result of a series of interface design decisions, many of which rely on historical precedent that is largely accidental. Rethinking what is truly important about computation while taking into account the possibilities offered by the access to surplus computational resources by designers of educational systems, we arrive at the conclusion that computation and computers are fundamentally different things. At worst, computers as the instantiation of computational ideas become blocking factors to understanding those ideas.

It is possible to reconsider completely what computation "looks like" and thus reconceive what it means to introduce children to computation. CTRL_SPACE attempts this by rethinking what it means to discuss the idea of "programming." CTRL_SPACE is used in conjunction with an animatronic head, called ALF: Acrylic Life Form [7], which allows us to leverage the inherent familiarity children have with face making and the similarity this has to several basic computational concepts such as objects, parameters and command sequencing.

2. Traditional approach to computation

Computational ideas existed long before the computer on our desks, and yet it is the object that we interact with and that which is most often the focus of so called computer literacy programs. In evaluating and creating new interactive systems for children it is important to recognize that the character of computers owes more to the history of computer use than to any principle of computation.

The primary user interface of the computer (the keyboard and screen), while extremely powerful, is the result of a historical convergence of technologies originally developed for different purposes. Text based programming with its lists of sequential instructions has served us well and is likely to continue to do so, but there is little *inherently* computational in these systems of instructional communication.

The primary series of questions asked in the early stages of this research were the following: Is the dominant method of manipulating computation (text based programming) which serves traditional modes of use well (quantitative analysis) truly appropriate for all uses of computation? Does it provide the most direct access to computational ideas? Is this method appropriate in developing systems for children?

3. Alternatives

There are many existing examples of visual programming languages or programming systems that incorporate visual/spatial elements. Visual Basic, MAX/MSP and Macromedia Director are a few such systems. None are appropriate for preliterate children.

One example that comes close is *LogoBlocks*, a graphical programming language designed for use with the programmable brick (precursor to the Lego RCX microcontroller) [1]. LogoBlocks uses the Logo

language for programming, but adds color and shape to code the commands as a way of assisting young children and novices in programming tasks. This coding functions as a substitute for linguistic syntax, but ultimately with LogoBlocks the user is still working with text. Why not eliminate the need for written language entirely?

3.1 Visual programming & the Deutsch Limit

In the project proposal for LogoBlocks, Andrew Begel examines some of the problems of adopting graphical programming languages, the foremost being the so-called Deutsch limit: "Deutsch originally said something like 'Well, this is all fine and well, but the problem with visual programming languages is that you can't have more than 50 visual primitives on the screen at the same time. How are you going to write an operating system?'"

With regards to the development of an environment for preliterate children the answer to this objection is simple: We aren't. Writing an operating system in a fully graphical programming environment is not straightforward, efficient, or useful.

The purpose of this work is to find the space where visual and physical programming is maximally useful, something that seems to be the case only in particular domains. The challenge remains in allowing for a seamless transition to more "advanced" techniques when the time comes. In part, the answer to this is found in recognizing that for many people and many cases, computation serves a particular task. While such people will undoubtedly benefit from an understanding of computational problem solving skills, general-purpose programming is far from necessary.

3.2 Physical programming and imitation

In their work, Allen Cypher, Henry Lieberman and others describe programming by example [3]. While Cypher, et al were not explicitly concerned with children, the idea of imitation as a method of programming is shared by the research described here. Imitation, especially with young children, is an excellent way to communicate information. Young children are highly self-focused and are quite good at expressing what they want to do "like this." This characteristic is similar to that which Papert leverages when he discusses body syntonicity and "playing turtle" [6] (although body syntonicity remains first person egocentric, while in the case of CTRL_SPACE we are asking the children to project themselves onto an other).

3.3 Physical, virtual, and the intermediate

There have been several physical programming environments developed to leverage children's affinity for imitation. In particular, it is worth mentioning *Dr. Legohead*, an animatronic head that is programmed by direct physical manipulation. *Dr Legohead* was a product of Rick Borovoy's thesis work [9].

More recently, the Tangible Interface Group at the MIT Media Lab has developed *Topobo: Physical Programming Instantiated* [11], a "constructive assembly system" which allows users to build an object and program it by physically moving its parts. Topobo records these movements and replays them.

In both cases, computation is attached directly to physical objects, removing the intermediate layer between the programmer and the programmable object. While Borovoy and others outline the reasons that introducing physicality is an improvement over purely screen based systems, eliminating the intermediate layer entirely does away with a host of possibilities.

It remains a basic tenet of computer science that given enough time and space, the analog world can produce the same results as the digital. Even so, there are particular classes of problems and actions that are utterly impractical to model in the analog world. Code re-use is difficult if one has to literally construct multiple instances of the same object. Recursion is nearly impossible. Dr. Legohead and Topobo are excellent and necessary steps in breaking from the tradition of requiring a complex syntax for programming. The next logical step is careful reintroduction of an abstract intermediate software layer to enable better access to the rich power of computation

4. Facemaking, containership, debugging

CTRL_SPACE interfaces a general purpose input device with ALF: Acrylic Life Form [5], an animatronic head (shown in figure 1), designed and built by Chris Lyon, a member of the Media Lab's Grassroots Invention Group. ALF has six features that are controlled by the Tower modular computer system [5]. As an object, a face can be used to represent a kind of computational containership. A face can be easily broken down into component parts and easily sequenced to create actions. It is not difficult to discuss

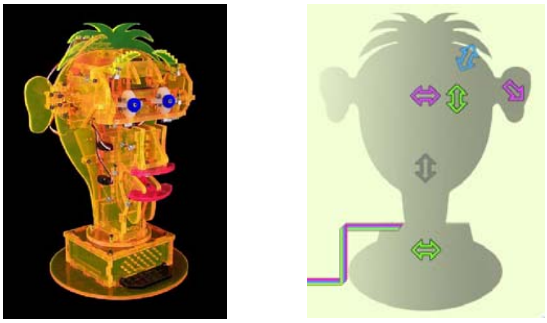

Figure 1. ALF (left) and mapping (right)

the face as a single object and also to refer to its parameters (eyes, ears, mouth). One can issue a command to the object (make a sad face) and then adjust individual parameters (now raise one eyebrow) and the outcome is immediately visible. Considered this way, the potential for addressing a wide range of computational concepts using a face is readily apparent. For example, one could imagine presenting the idea of a state machine with a face. Debugging is made simple by virtue of the fact that the wrong sequence of commands results in a face that is immediately visually recognizable as "wrong."

Perhaps more important than the fact that faces exhibit containership is the fact that faces are intimately familiar objects to all of us. There is a great deal of research that indicates how significant our brains consider facial recognition to be. Piaget discusses the fact that children as young as eight months use imitation (of sounds, as well as physical actions) to explore their world. More recent research by Stern [9] and Tronick [10] highlights the specific importance of facemaking to early development. By the age of four, children are fully capable of understanding how to control their own faces and are intimately interested in the notion of representation on the face (what indicates sad, happy, angry). Therefore, the face provides us with an object that is readily understood by a four year old, has a very familiar analog (one's own face) and at the same time demonstrates a kind of containership that is useful for accessing a number of computational ideas.

5. Storytelling and sequencing

While it is the face robot that allows for "object-oriented-ness," it is the nature of storytelling that allows for children to establish a rule set. The story becomes a script and can be thought of as programmatic sequencing. The introduction of sensor data as an event trigger provides a mechanism to introduce logic structures and conditionals. The addition of multiple ALFs or similar objects would allow for multiple characters and parallel rule sets.

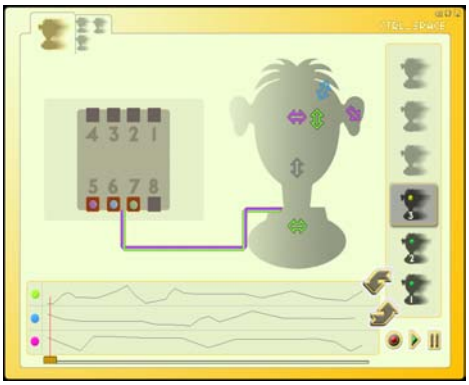

Figure 2. Action creation mode

6. Representation of actions

The CTRL_SPACE environment is an attempt at leveraging the power of physical interface *and* software abstraction. The system is fully graphical, contains no text, and centers around the idea of action. The environment supports two modes of use: action creation and action sequencing. Figure 2 and figure 3 show screenshots from the action creation mode and the action sequencing mode respectively.

Actions are represented by two related fields: the timeline, which shows a visual representation of change over time, and the mapping of these values to particular

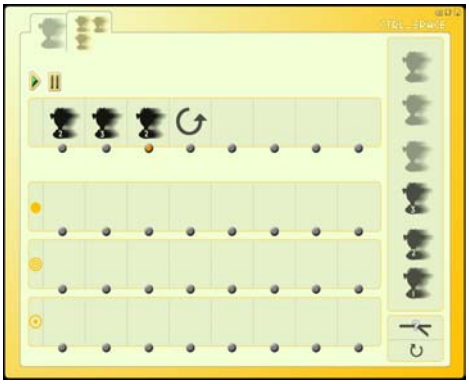

Figure 3. Action sequencing mode

features of ALF, as shown by the color coding of the arrows on the ALF head (figure 1.)

Creation mode allows for the creation and editing of actions, which may be stored for later use. Saved (or minimized) actions are represented by the "ALF in motion" icon and are stored in the action palette on the right hand side of the screen.

Once a child has built up a library of actions, they may use the action sequencing mode to define a "program" consisting of a sequence of a number of actions. Sequencing mode also introduces basic logic structure and branching on the basis of conditionals.

7. Representation of conditionals

When users drag the conditional branch icon onto a frame (or click on a frame which contains a conditional) they are presented with a dialog box that allows them to adjust the type of conditional. There are two types of conditionals, blocking and non-blocking, which correspond roughly to *wait…until* and *if…then...else* statements respectively. Blocking conditionals are indicated by a red question mark and cease program execution until the condition is true, at which point the program branches as indicated. Non-blocking conditionals are indicated by a yellow question mark. With non-blocking conditionals, the condition is tested once when the frame is executed. True evaluation branches the program to the indicated subroutine. False evaluation continues the program on the next frame.

The destination of a program branch is indicated by an icon which corresponds to one of the three optional sequences specified below the main sequence. By using a loop or another conditional in the frame following a non-blocking conditional, the user can create more complex logic structures, as shown in figure 4 along with a more traditional textual representation in pseudocode.

8. CTRL_ARM physical interface

Early on, it became apparent that a device that would bring the act of issuing commands to ALF closer to the physical act of puppeteering could prove useful. At the same time, it seemed important to create an interface that lent itself to the use and discovery of computational abstraction. This requirement seemed to call for an interface that was *not* a replica of the ALF.

To that end, a two axis armature, called CTRL_ARM, was constructed. CTRL_ARM uses analog potentiometers to measure hand movements, sending data in real time to the CTRL_SPACE software. CTRL_SPACE allows a child to map sensor inputs in real time to one or more of ALF's features. The software also allows users to record the sensor input and play it back at any point in time.

The act of mapping the world to digital space is itself a computational idea and is supported most directly by allowing the CTRL_ARM motions to be mapped arbitrarily to one or more of ALFs features. Motion occurs in real time, but computation allows it to be manipulated in any number of ways.

CTRL_ARM provides concrete access to ALF in the sense that it involves physical motion, but abstract access through computation in that it allows for arbitrary mapping of sensors.

```
if CONDITION A
{
        subroutine ⊙
}

else
{
   if CONDITION B
   {
        subroutine ●
   }

   else
   {
      waituntil(CONDITION C)
      {
        subroutine ●
      }
}
```

Figure 4. Logic Structure and Equivalent

9. The power of augmented reality

In terms of introducing children to computation, the physical world is a wonderful starting point because of its familiarity and children's natural inclination to explore the way objects move, bend and break. *Augmented reality* marries the familiarity of our analog world and our natural inclinations to hold, shape, poke and prod with our hands to the infinitely malleable world of computation.

The design of a physical interface for use by children is in no way a trivial task. CTRL_ARM is presented as one example of a possible physical interface (and a very simple one at that). The topic of design of physical interfaces for children deserves a thorough investigation and may even serve as a site of learning for the children, who may benefit from designing their own interfaces. In an effort to better

support this, CTRL_SPACE was built around the idea of generalized sensor input rather than "CTRL_ARM input," allowing any number of existing or future input devices to be built and used in conjunction with animatronic objects.

10. ALF represents a class of objects

While it should be clear by now that an animatronic head with discrete movable parts is a good choice for this environment, it is important to note that ALF is presented here not as the ideal object, but as a representative of a class of objects.

ALF is a head-only robot with a limited number of movable parts. While it has proven more than sufficient as a proof of concept device, much can be gained by using other animatronic objects. The CTRL_SPACE software environment is ultimately intended to be multi-purpose in this sense: able to control a wide range of motor driven programmable objects.

In order to encourage this, ALF's control structure (based on the Tower modular computing system) may be completely removed and re-used with CTRL_SPACE to manipulate whatever objects one wishes. In addition to using other existing objects, the possibility of designing one's own animatronic character in this way is quite compelling, opening up an entirely different and interesting set of ideas which cover engineering, materials science, physics, electronics and control feedback.

11. Conclusion

In CTRL_SPACE, the use of the face robot as an analog to one's own face enables access to computational ideas in a familiar manner. The act of imitation allows the child to teach ALF what to do and the incorporation of sensors for input allows one to literally program ALF by example. An intermediate software environment provides a layer of abstraction that allows access to powerful computational concepts, but remains text free, sacrificing generality of purpose for specificity of task that eases understanding. A careful balance is maintained by virtue of the fact that the CTRL_SPACE software is deliberately limited in scope.

A number of choices have been made in CTRL_SPACE which, while they allow for easier access to complicated concepts for very young children, may prove frustrating for more experienced users. In such cases, it is important to note that the choice has been made deliberately in an effort to make concepts more accessible. The problem of growth is mitigated by the fact that CTRL_SPACE should be seen as one of a family of projects. The hardware is based on the Tower modular computing system; there is little to prevent (and much to help) them to continue their work using a high level language of their choice.

Finally, the concentration of this paper has been entirely about the environment, but it is important to point out a crucial and often overlooked component of children, technology in education: namely, children! CTRL_SPACE represents the result of a participatory design process that led to a series of design guidelines [8]. The experience of the children and these guidelines are both critical. While technology affords us new ways of communicating ideas to learners, education begins and ends with the people involved; people whose learning process must never take a backseat to any technology.

12. References

[1] Begel, A. (1996). LogoBlocks: A Graphical Programming Language for Interacting with the World (AUP). MIT Media Lab.

[2] Borovoy, R. D. (1996). Genuine Object Oriented Programming. MIT Media Lab, MIT.

[3] Cypher, Allen, Et al (1993). Watch What I Do: Programming by Demonstration. Cambridge, MIT Press.

[4] Hancock, C. (2003). Real-time programming and the big ideas of computational literacy. MIT Media Laboratory. Cambridge, MIT.

[5] Lyon, C. (2003). Encouraging Innovation by Engineering the Learning Curve. Electrical Engineering and Computer Science, MIT.

[6] Papert, S. (1993). Mindstorms: Children, Computers, and Powerful Ideas - Second Edition. New York, BasicBooks

[7] Raffle, Hayes, Amanda Parkes, Hiroshi Ishii (2004). Topobo: A Constructive Assembly System with Kinetic Memory. CHI 2004, April 24-29.

[8] Sempere, A. (2003). Just Making Faces? Animatronics, Children and Computation . MIT Media Laboratory. Cambridge, MIT

[9] Stern, D. (2002). The First Relationship. Cambridge, Harvard University Press.

[10] Tronick, E. Z. (1986). Maternal Depression and Infant Disturbance. New Directions for Child Development, no.34. San Francisco: Jossey-Bass, Winter.