Just Making Faces? Animatronics, Children and Computation

Andrew Sempere
B.F.A.
School of the Art Institute of Chicago, 2001

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences at the
Massachusetts Institute of Technology
September 2003

Author Andrew Sempere
Program in Media Arts and Sciences
August 1, 2003

Certified By Dr. Bakhtiar Mikhak
Research Scientist of the Media Laboratory
Thesis Advisor

Accepted by Andrew B. Lippman
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Just Making Faces? Animatronics, Children and Computation

Andrew Sempere

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences at the
Massachusetts Institute of Technology

ABSTRACT
Computation is a powerful way of knowing and exploring the world that finds its application in a broad range of human activities, from art making to mathematical modeling. Historically, this way of knowing has been taught in a canonical, top-down abstract fashion. This thesis presents a critical historical analysis of computers and computation in order to arrive at a framework for design of spaces for introducing computational concepts. Existing work is revisited before presenting a new system called CTRL_SPACE, specifically built to as an alternate method of conveying computational concepts to young children ages four to seven.

Bakhtiar Mikhak
Research Scientist of the Media Laboratory
Thesis Advisor

Just Making Faces? Animatronics, Children and Computation

Andrew Sempere

The following people served as readers for this thesis:

Reader Edith Ackermann
Visiting Professor of Architecture
Massachusetts Institute of Technology

Reader Glorianna Davenport
Principal Research Associate of the Media Laboratory
Massachusetts Institute of Technology

Acknowledgements

This document would not exist without the help and guidance of Bakhtiar Mikhak, whose depth of thought, dedication, patience, integrity and kindness have shown me what it means to truly care about education. Thank you for believing in me and showing the way by example.

Glorianna and Edith, thank you for your willingness to spend the time with my sometimes very rough drafts. Your insights and encouragement were invaluable.

A special thanks to my crack-team of beta testers: “Alex,” “Cindy,” “Jeff” and their parents. Your generosity of time truly helped build CTRL_SPACE. Thanks also to Chris Hancock, Cynthia and the kids from Milton Academy for sharing your time and your thoughts.

Thanks to my family: Mom, Christian, Seth, Chloe and Violet for supporting me through these long two years. You kept me sane, even if I sometimes drove you crazy.

Chip and Melissa Paliocha for their support, friendship and enduring tolerance of my idea of scheduling.

Michael, Sara, Tim, Scooby, Margarita and Dan for the thousand small ways you made me feel welcome when I arrived at the Grassroots Invention Group. Thank you for continuing to inspire me with your work and friendship. It’s been a great year.

To Tangentlab: Dima, Mary, Silvia and Ben (yip yip yip!), thanks for reminding me that there’s *always* time to put on a multi day international art event.

Ranjit, thank you for allowing me to pester you with Lingo questions.

To the many family members, friends and colleagues who offered me words of encouragement, cold beer, prayers, hot coffee, technical advice, joy, laughter and the occasional kick in the pants these last two years: You are too numerous to name but you are the giant shoulders I stand on. Thank you.

Finally to Anindita, my reader, editor, sanity check, partner and love of my life. I don’t know how you put up with me and my pronoun choices, but I am truly lucky. I would never have made it through without you. I love you forever!

Table of Contents

List of Figures

A Guide For Readers

In general, readers who wish to know only **what** was done should read section one and then skip to section three. Readers who are interested in a critical historical analysis that explains further the **why** of this thesis should also read section two.

Section one provides a basic overview of the ground covered by this thesis, as well as a brief look at the material to follow.

Section two provides historical and intellectual background by way of explaining the reason this work is important.

Section three talks about the conceptual and political motivations, presents a set of guidelines for developing educational environments for children with computation and then revisits existing work with this framework in mind.

Section four presents a particular software and hardware environment developed for this thesis.

Section I Introduction

PRELUDE TO SECTION I

This section serves to introduce the concepts, history and ideologies that form the basis of the work presented in this thesis. We begin with a chapter outlining the core ideas, especially the notion of *computation* as intimately tied to but distinct from the idea of *computer*. This chapter also sets the stage with a brief overview of the philosophy of education and the body of work on which this thesis rests. Finally, we briefly touch upon some of the issues to be examined in depth later, such as the problems educators may encounter as a result of the current computer interface to computational ideas. Section one ends with a brief summary of the material covered in section two. This may function either as an extended introduction for those who intend to read section two, or else a summary for those who intend to skip directly to section three.

1 COMPUTATION A WONDERFULLY POWERFUL IDEA

THE IMPORTANCE OF WONDER

Eleanor Duckworth was a student of Piaget and is currently Professor of Education at the Harvard Graduate School of Education. Duckworth writes in her book *The Having of Wonderful Ideas* that the having of wonderful ideas is the essence of intellectual development, and the essence of pedagogy is to create occasions where students may have such ideas and feel good about having them.[1]

Duckworth's "wonderful idea" is a concept similar to that which Seymour Papert refers to when he speaks of "powerful ideas" in his seminal *Mindstorms* and in other writings. This idea is also resonant with what educators refer to as an "a-ha moment," the point at which the solution to a problem becomes clear as the result of a particular piece of information that just *fits.* Often this particular piece of information is unassuming in and of itself, perhaps trivial, certainly forgettable in isolation, but at the precise moment that it arrives in the thinker's consciousness it becomes *the* key that unlocks a puzzle, collapsing the problem at hand like a house of cards. The truly powerful ideas, the best of these moments, can be characterized by the applicability to cross traditionally set boundaries between domains. For example, while we may experience a powerful idea in the context of mathematics, we may find it changes the way we experience music or poetry.

These moments are the holy grail of education, something we have all experienced on our own or with the guidance of a good teacher. Unfortunately it is the case that we tend to consider "wonderful ideas" a mysterious force, happy occurrences attributed to chance (much like seeing a rainbow or bumping into an old friend) that are profound and yet fundamentally uncontrollable. If we are lucky enough to have experienced a teacher who seemed to inspire these ideas in ourselves we remember them fondly as one of those great teachers who just seemed to have a knack for teaching.

The body of work built on the foundation laid by the genetic epistemologist Jean Piaget tells us that this is not the case. Wonderful ideas are certainly full of wonder, but they are not mysterious. Duckworth writes: "Wonderful ideas do not spring out of nothing. They build on a foundation of other ideas."[2] And with due respect to the great teachers of the world (who know this fact already): education is not magic, it is work. Wonderful ideas and the teachers that inspire them do come from somewhere, and that somewhere can be discussed. Interpersonal human interaction plays a vital role in this process and no software program, book or article can ever replicate or replace this. Even so it is possible to discuss the qualities of interaction, process, educational environments, successes and failures, and to develop guidelines for their use and development.

[1] Duckworth, 1.
[2] Duckworth, 6.

EPISTEMOLOGY, CONSTRUCTIONISM AND CONSTRUCTIVISM

The work of Piaget ultimately evolved into a methodological approach[3] known as *constructivism* (with a V). Edith Ackermann explains: "Piaget's constructivism offers a window into what children are interested in, and able to achieve, at different stages of their development. The theory describes how children's ways of doing and thinking evolve over time, and under which circumstances children are more likely to let go of—or hold onto— their currently held views."[4]

Piaget, then, provides us with an understanding of epistemology, of how we come to know things, and in particular how children come to know things. Piaget tells us that the acquisition of knowledge is an iterative process of building and altering a *construct*, or worldview. This construct is a lens through which we view the world, and at all levels is highly robust. This latter point is significant because in some cases, especially with young children, the nature of one's construct may be considered inaccurate but is in no way unsubstantiated. If education is mistakenly seen (usually by an adult) as the quest to correct another's (usually a child's) "faulty" worldview, the experience will be at best frustrating for both and at worst will become an intellectually violent experience for the learner. In many cases the learner will even come to consider themselves deficient or worse develop a hatred for a particular knowledge domain, a state of affairs common to mathematics education, defined by Papert as *mathophobia*.[5]

Seymour Papert, a mathematician who worked with Piaget, developed these ideas further into a concept he called *constructionism* (with an N). Again, Ackermann explains: "Papert's constructionism, in contrast [to Piaget's constructivism], focuses more on the art of learning, …and on the significance of making things in learning. Papert is interested in how learners engage in a

[3] I hesitate to label any of this work a "theory" because in the world of education this invites constructivism and its descendants to be compared and contrasted with other "educational theories," a highly distracting and ultimately unfruitful exercise. The distinction I make is a temporal one. Piaget and the "two Cs" operate on a *fundamental* basis: Piaget's explanation of the way the human mind works. Accepting Piaget's work, one can naturally (though not without effort) determine the best way to convey knowledge. Compare this to the basic notion of a theory, which is by definition an explanation or model based on observation. The implication in this is that a theory is constructed *after* the fact. In a rough sense Piaget is looking for the a-priori explanation of the way the mind works, out of which any number of educational theories might emerge. By way of example I offer the comparison of constructionism to instructionism (a comparison we see frequently) as something of a non sequitur: Constructionism is a different class of thinking entirely, which just might call (at learner's request) for a bit of instructionism. Far better that constructionism be compared to the unnamed body of ideas that lead some to believe instructionism is the best teaching methodology than to instructionism itself.

[4] Ackermann, *Constructivism: One or Many*.

[5] Papert, *Mindstorm.*

conversation with (their own or other people's) artifacts, and how these conversations boost self-directed learning, and ultimately facilitate the construction of new knowledge."[6]

Papert, then, suggests that the constructs identified by Piaget can be expressed and mediated through the creation and manipulation of *artifacts*. These artifacts may be literal, physical objects, or else mental or "virtual" constructs. In his seminal 1980 work *Mindstorms: Children, Computers and Powerful Ideas* Papert further outlined the possibilities of virtual constructs by describing the ways in which computation might provide an ideal, or *rich*, environment in which to create and manipulate mathematical artifacts on screen.

In *Mindstorms* Papert is primarily concerned with the learning of geometry. But how do computers enable one to better understand geometry? The answer is not to be found in computers per-se, but rather in the application of computational concepts to geometric concepts, made possible by virtue of the fact that computers *enable this to happen*. This application, of course, requires a support structure, boundaries and rules for exploration. This structure, usually embodied in a literal computer application, forms a rich exploration space known as a *microworld*. In the case of Papert and mathematics, the microworld described in *Mindstorms* was the combination of approach, turtle object and Logo programming language. The language itself was invented at BBN, but became a microworld with Papert's careful choice of primitives and the addition of a turtle object.

The role of the educator is to provide a structured environment in which it becomes possible for learners to encounter wonderful ideas on their own, by way of creating, manipulating and comparing artifacts. As it turns out, computers provide an ideal space for one to engage in this type of activity. But while the possibilities are compelling, it is a short step and a long fall from supporting wonderful ideas with computation to the technocentric assumption that "computers in schools make things better." This brings us to a distinction of crucial importance: that between *computer*s and *computation*.

COMPUTER VS. COMPUTATION

The *computer* is the box on your desk, while c*omputation* is a way of thinking, using *computational concepts*, to solve problems. Computational concepts include but are not limited to ideas such as sequencing, formalized logic, iteration and recursion. These concepts are not embodied in every (or even most) aspects of using or programming a computer, but they are nevertheless computational in

[6] Ackermann, *Constructivism: One or Many*.

nature. Similarly, while computational concepts are evident in programming and in the use of computers, their usefulness is not limited to that domain. In fact, examples of the use of computational concepts for problem solving, with and without the use of a computer, abound in fields as diverse as mathematics, engineering and the fine arts.

It is a central premise of this thesis that the computer as an embodiment of computation is a historical artifact and that, in light of what computation might be used for, it is both possible and necessary to rethink this embodiment. In particular, this thesis will address the various ways in which we currently communicate computational concepts to machines and to one another, and the ways in which this communication might be improved.

In practice it becomes quite difficult to maintain a clean division between the object and the objectified. The embodiment of computational ideas in the form we call the computer strongly affects the way we think about computational ideas. Nevertheless, It will serve us well to make this point now, as we will return to it frequently: while intertwined, *computers* and *computation* are not the same thing.

THE CANONICAL APPROACH: TRADITIONAL USES OF COMPUTATION

"The prevailing image of the computer represents it as a logical machine and computer programming as a technical, mathematical activity. Both the popular and technical culture have constructed computation as the ultimate embodiment of the abstract and formal."[7]

Computers have long been perceived as tools in the service of math and science. From their birth as the multipurpose grandchildren of "mechanical calculators," these machines have been designed and used to crunch numbers and model complex mathematical systems. From census counting to fractal drawing, there can be no doubt that computers do this task extremely well – enabling humanity to model systems and calculate large quantities of data with relative ease.

While it is easily demonstrated that the use of computers for math and science is positive, the overwhelming popular perception of computers as glorified calculators has done more harm than good, especially with regards to the roles computers might play in education. In *Mindstorms*, Seymour Papert addresses this point with respect to the way computers are typically used in physics class. First, the primary blocking factor to student understanding as Papert sees it is that physics is

[7] Papert and Turkle, *Epistemological Pluralism*, 1.

taught largely as a quantitative discipline. Rather than attempt to connect students' own understanding of the world to the laws of physics, instructors ask students to memorize formulas and run through equations. These formulas purport to explain the way the universe functions. While this may be the case in fact, they simultaneously resemble nothing the students interact with on a daily basis.[8]

Computers *could* be used to alleviate this. For example, one could imagine a graphical environment that allows students to model and modify the rules of gravity. Done well, this type of activity not only leverages the power of the computer as a modeling tool (it is hard to imagine a real time simulation of alternate rules for gravity accessible to students without a computer) but also provides a more solid connection between the mathematically modeled world and the physical world we live in. Despite this potential and the example of a few brave teachers, it is unfortunately the case that computers in the realm of physics class are used largely to solve more difficult quantitative problems. In this way they not only fail to help the student to understand the material, they actually compound the initial problem by confronting the confused student with increasingly complex formulae.

EXPRESSION AND COMPUTATION

While general purpose programming is a worthy goal and quite useful in its own right, the underlying skills used in programming are applicable in other domains, and computation has proved useful for a wide range of activities with which it is not traditionally associated.

Perhaps the most interesting "alternate" use of computation is as a tool for personal expression. In this mode computers and computation can augment traditional methods of expression, or else themselves become the medium and the mode of expression. In a later chapter we will explore specific examples of both of these types of activities.

In general, and unlike the natural marriage of computation with quantitative analysis, the convergence of computers and expression is not always a happy one. In his book *Being Digital*, Media Lab founder Nicholas Negroponte put it this way: "Computers and art can bring out the worst in each other when they first meet. One reason is that the signature of the machine can be too

[8] Papert *Mindstorms*, 139.

strong. It can overpower the intended expression …Technology can be like a jalapeno pepper in a French sauce. The flavor of the computer can drown the subtler signals of the art."[9]

In designing occasions which involve computation and expression, where one hopes students will have a wonderful idea, the designers must continually ask themselves two related questions.

1.What does the inclusion of computation add to this activity?

2. Is this an improvement over the status quo?

These questions are of particular importance in the case of computation used for expression. In such cases the computer is often used to justify expression as a worthwhile activity (as technological and therefore relevant) or else is introduced in such a way that encourages the inevitable self-referential "expression" consisting of a demonstration of software.

SO WHAT'S THE PROBLEM?

There exist many examples of computation used for personal expression, but the ready availability of these examples belies the fact that that the current set of tools we have to access computation are hardly *ideal* for expression. Far from it! The traditional tools used to access computation have a steep learning curve, as they were never originally intended for this purpose. For every individual who has composed electronic music, written a piece of software or created a web page, there are a dozen who have found the process frustrating enough to abandon the pursuit. Worse, there are those who walk away from computation entirely, denying themselves access to powerful ideas and mistakenly attributing this to a deficiency of self, or of the machine. In fact, the problem and the solution lie between the two, at the level of adequately designed microworlds or authoring tools.

HELP! MY COMPUTER SPEAKS A FOREIGN LANGUAGE!

"…It is possible to design computers so that learning to communicate with them can be a natural process, more like learning French by living in France than like trying to learn it through the unnatural process of American foreign-language instruction in classrooms."[10]

Unfortunately, it is the case that the method of communicating with a computer has historically followed the form factor of the device rather than informing its design. First patch cables, then

[9] Negroponte.
[10] Papert, *Mindstorms,* 6.

switches and punch cards, then text: the nature of programming has been guided by the nature of the computational object itself. Despite the passage of years, computer programming today means essentially the same thing as it did in the 1960s: text-based lists of instructions entered via a keyboard.

It should be acknowledged that the ability to communicate with a computer via a keyboard and a high-level programming language represents a massive improvement over the methods used before this. Nevertheless, learning to program a computer in this manner requires one to learn a new language. Computer code is not natural for humans, nor is it natural for machines. Rather, it is an intermediary piece of technology developed to allow analog humans and digital machines to communicate with one another. Computers must first compile or translate almost all code before it can be run, while humans must continually bend over backwards to formalize their logic and clarify ambiguities in order to prevent confusing the machine.

When learning a programming language or solving a programming task using a high level language, some time is spent on looking up commands, but by far the bulk of one's time is spent on syntax: puzzling out the best way to translate human desire into computer readable form. An easily stated objective (IE "I want the computer to write my name in green on a blue background") becomes a task, which must be broken down into component parts. While the act of breaking down a task into component parts is in fact an integral part of thinking computationally, the nature of the parts themselves rarely makes sense to the uninitiated. Furthermore, as any programmer worth his salt will tell you, there are certain methods of accomplishing programmatic tasks that are decidedly better than others. These methods – usually the most flexible and efficient means of accomplishing a task – are frequently the most obscure in terms of linguistic syntax, in part because they disregard human comfort in order to rely more heavily on the *computer's* preferred method of accomplishing tasks.

LANGUAGE, ILLITERACY AND COMPUPHOBIA

There is a good reason text-based computer programming persists - it has certainly served us well in the last forty years - but it is not at all clear that this is the only or the best way to communicate with a computer.

One of the problems with human computer interaction based on text is the degree to which the metaphor of language and writing utterly permeates the notion of how one *ought* to express one's

desires to a computational device. One need look no further than the langauge surrounding computers in schools: a focus on computer *literacy*, programming *language*, *syntax*, *expression.* The focus on computer use as a problem of literacy produces a label for those who find the process of working with computers frustrating: "computer illiterate."

The preoccupation with literacy as a measure of development and the force of the anxiety and shame associated with being labeled illiterate should not be underestimated. Coupled with the perception that access to computation requires rigid conformance to "computer thinking," this creates a state of affairs remarkably similar to *mathophobia* as described by Papert. This state of affairs is well expressed in the following poem by a high school senior, as quoted by Sherry Turkle in *Life on the Screen:*

> *If you could say in numbers what I say now in words,*
> *If theorems could, like sentences, describe the flight of birds,*
> *If PPL [a computer language] had meter and parabolas had rhyme,*
> *Perhaps I'd understand you then,*
> *Perhaps I'd change my mind…*
>
> *But all this wishful thinking only serves to make things worse,*
> *When I compare my dearest love with your numeric verse.*
> *For if mathematics were a language, I'd succeed, I'd scale the hill,*
> *I know I'd understand, but since it's not, I never will.* [11]

The message here could not be clearer: As far as the poem's author is concerned, mathematics and computation are not tools for expression. More interestingly, the author makes this point by turning the language of computer literacy back on itself. If computer languages really *were* languages, the author argues, they would have that space for ambiguity that allows for expression in the same way verse does. This last point is issued as a challenge: demonstrate that computation has the expressive abilities of human language and the author will not only make use of computation, they will "love it dearly."

[11] Turkle, *Life on the Screen,* 53.

COMPUTATIONAL FORM AND FUNCTION

This thesis can be seen as an attempt to rise to this challenge by occupying itself with the following question: What do we need to do to give PPL its meter and parabolas their rhyme? In the stone age of computing, when it took great feats of engineering to get a computer to add two numbers, meeting the computer halfway made sense. The flexibility and the intelligence of the human mind, coupled with the perceived benefit of using computers to solve problems outweighed the shortcomings or utter lack of user interface. Recently, with the growing availability of powerful and cheap computation, this approach is appearing more and more an unnecessary artifact of history. By closely examining the foundations and assumptions of computation in our culture, we may find a way to push great ideas even farther and make the wonderful power of computational thinking available to everyone.

2 MODERNISM FUTURISM AND INDUSTRIALIZED EDUCATION

MODERNISM

Modernism is a term which has been only slightly less abused than it's successor, the dreaded *postmodernism*. In general, modernism refers to a general movement of the arts, literature and architecture, occurring during the whole of the 20th century, away from the dominant academy-established views of what art was and how it ought to function in society. Early modernist work is characterized by a focus on industrial materials and methods of production; It is interested in science and the possibilities of using scientific equipment, such as photography, for expression. It is frequently optimistic, often politically motivated and is largely populist, something that is often completely misunderstood. As an aesthetic, modernism favors dynamic line, action and edges, synthetic shapes and colors.

While the modernist aesthetic is easily recognizable today, modernism was far from a strictly formal approach. Modernist art and design embodied a widespread cultural view towards technology, industry and engineering that was at best concerned with forging a better life for humankind through technology and at worst viciously anti-historical, seeking to centralize power and create efficiency through rigid adherence to standards. Futurism, an Italian expression of modernism, reflects the latter.

FUTURISM

Founded by the Italian magazine editor F.T. Marinetti, Futurism was one expression of the modernist movement launched with the publication of the Futurist Manifesto in 1909. Futurism sought to resolve the contradiction between the pleasure of a modernized society and the inevitability of its side effects (pollution, noise, conformity) by fully embracing both. Futurism saw beauty in industry, speed and technology and also in riots, danger, violence, war and ultimately Fascism. It is perhaps understandable that Marinetti's contemporaries initially seem to have turned a blind eye to the latter. Faced with the growing reality of Fascism, however, this rhetoric became far less excusable and far more frightening. It quickly became apparent that when Marinetti suggested "that the word *Italy* should prevail over the word *Freedom"* and that the only things worth admiring were the *"…*mad sculptures that our inspired artillery molds among the masses of the enemy" that he quite literally meant it.[12]

[12] Marianetti, *War, the World's Only Hygiene.*

FRANZ MARC AND JAN TSCHICHOLD

The events of WWII perfectly describe the inevitable collision of the joyous and violent approach to modernism. On one hand the war was largely responsible for catalyzing industrialization, in particular the adoption of standardization which eased manufacturing and the creation of digital computation to replace analog. Standardization also brought us anthropometry and ergonomics, which may function to improve the safety and comfort of our environment. On the other hand the attempt to apply standardization to the human mind gave rise to eugenics and the belief that the intellect can be quantified as easily as the number of fingers on the average human hand. In Europe, the rise of Fascism and the events of the Holocaust embody the worst case scenario of these ideas applied to humankind: complete dehumanization through industrial process.

Franz Marc was a painter who, along with the better known Wassily Kandinsky, founded a modernist art movement called *Der Blaue Reiter.* Marc, like his Futurist contemporaries, thought of war as a natural cleansing process that would result in a better future. Face to face with the realities of mechanical warfare, Marc changed his mind, but was killed in battle before the age of forty.

Almost a generation later came Jan Tschichold, a graphic designer who was born in Germany, rose quickly to stardom by staunchly supporting modernism and its aesthetic application to design, and then, after taking refuge from Hitler in Switzerland, utterly renouncing this stance. Like Marc, Tschichold's responses to both modernism and the realities of war were extreme, but unlike Marc Tschichold survived and so provides us with an illustration of the kind of uneasy relationship to technology and industrialization that characterize this period in time.

INDUSTRIALIZED EDUCATION

The issues of industrialization and where it came from are important because the general purpose of this thesis and the body of work on which it rests is to question the assumptions of contemporary education by proposing alternatives. The industrial approach favored by most of the public schools in this country is the result of a series of historical events and choices, none of which are inevitable. By unpacking a bit of history in section II, I hope to highlight the advantages and dangers of the industrial approach and, ultimately, to propose this thesis as a fragment of an alternative.

The beginning of section three will return briefly to these themes with a look at Reggio Emilia, one of the most successful examples of constructionist approach to education in practice. Reggio was

founded in post war Italy, in the shadow of Fascism, in an overtly political move to ensure that future generations would never again support the rise of a Fascist power.

Section II From Fascism to Futurism and Back

PRELUDE TO SECTION II

Section two consists of a close examination of the cultural history of computers and computation, standardization and education. This history is examined in order to arrive at a framework for the design of expressive, computational spaces for children. It is possible to skip this section and proceed directly to section three. However, I would like to encourage the reader to continue on, as the historical perspective presented here will go a long way towards situating the work to follow.

Specifically, this section presents the development of computational devices as the expression of a longstanding desire to improve humanity's ability to analyze quantitatively. A link is made between quantitative analysis as a hallmark of industrial processes and the character of contemporary education, which has been informed in no small part by industrialized mass-production.

Chapter three will examine modernism as an embodiment of the optimism that arose from the general worldwide move towards industrialization and standardization. Considering WWII as a catalyzing force, this chapter takes a look at the advantages and disadvantages of applying industrial style quantitative analysis to human beings. Chapter four will look at how and why industrial standardization is applied in the realm of education, what this allows and the problems it creates. In order to better understand the intellectual challenges faced by adoption of computation for expression in a standardized educational environment, chapter five takes a look at the issues put forth by Sherry Turkle and Seymour Papert in their article *Epistemological Pluralism and the Revaluation of the Concrete.* Chapter five then looks at how hacker culture deals with the dominant top-down nature of computation. Chapter six completes the picture by looking at how the dominant form factor of computation is a result of its intended use as a single user machine for quantitative analysis, how this perception is changing and why.

3

MODERNISM FACISM AND STANDARDIZATION:

The Dominance of the Industrial Approach in Culture and Education

"The standardization and electro-mechanization of the things we use daily will be further increased. ...But electro-mechanization as an end in itself is nonsense: its true purpose, by satisfying basic needs through mass-produced objects of highest quality, is to make possible for the first time a true and unlimited awakening of all the creative powers of man. ...In some fields, standardization, rationalization and mechanization have made great progress, but in most others almost everything remains to be done."

Modernist designer Jan Tschichold, from his 1926 book
The New Typography: A Handbook for Modern Designers, 12.

MODERNISM IN THE 20th CENTURY

In 1923, a young graphic designer named Jan Tschichold (1902-1974) paid a visit to the first modern art exhibit sponsored by the Dessau Bauhaus. Held in Weimar, capital of the newly founded German democratic republic, the list of participating artists reads like a who's-who of the early modernist movement in Europe, including Herbert Bayer, Josef Albers, Paul Klee, László Moholy-Nagy, El Lissitzky, Kurt Schwitters, Man Ray and Piet Zwart.[13]

This exhibit may or may not have prompted Tschichold's actual writing of the seminal 1928 book *The New Typography: A Handbook for Modern Designers*, but there is no question as to the influence of the exhibited work. Tschichold takes great care in illustrating this influence in his book, including a number of examples of modernist work by Lissitzky, Zwart and Schwitters, as well as paintings by Picasso and the co-founders of *Der Blaue Reiter:* Franz Marc and the better known Wassily Kandinski.

Marc's work in particular displays the strong influence of the orphistic and cubist art movements. Led by Pablo Picasso (1881–1973), the latter found beauty in the scientific development of photography and it's ability to show multiple aspects of the same object at once through multiple exposures. This characteristic of time compressed into two dimensional space can be seen clearly in most all of Picasso's work.

Meanwhile the Italian Futurists, lead by the fiery magazine editor Filippo Marinetti (1876–1944), launched an entire art movement obsessed with the idea of mechanization, speed and power of machines. Joyfully drunk on the raw power of engines, racecars and industry, Marinetti sought to capture this emotion in his artwork and in typography. Marinetti supplemented his visual work with numerous manifestos and "Although the stated aim of Futurism was to free Italy from its past, the deeper purpose was to promote a new taste for heightened experience."[14] Tschichold found what he was looking for in Marinetti's distain for the past and love of the future and quoted him extensively, as in this extended passage:

> *"I am starting a typographic revolution, directed above all against the idiotic, sick-making conception of the old-fashioned Poetry Book, with its hand-made paper, its sixteenth-century style, decorated with... great initials, flourishes and mythological vegetables... The book must be the futuristic expression of our futuristic thought. Better: my revolution*

[13] Mclean, *Jan Tschichold: A Life in Typography*.
[14] Stokstad, 1056.

> *is against …the so-called typographic harmony of the page, which is in complete opposition to the flow of style which the page allows. We will, if need be, use 3 or 4 different colours and 20 different typefaces on the same page."*

Curiously, the optimism shared by Marc and his contemporaries put them in the position of favoring the first World War, which they perceived as a cleansing force. Marc writes: "I… see in this war the healing, if also gruesome, path to our goals; it will purify Europe, and make it ready… Europe is doing the same thing to her body France did to hers during the Revolution." Marc enrolled in the army at first opportunity, but had a complete change of heart once he saw battlefield carnage first hand. Marc wrote home:

> "The world is richer by the bloodiest year of its many-thousand-year history. It is terrible to think of; and all for *nothing*, for a misunderstanding, for want of being able to make ourselves tolerably *understood* by our neighbours! And that in Europe!! We must unlearn, rethink absolutely everything in order to come to terms with the monstrous psychology of this deed and not only to hate, revile, deride and bewail it, but to understand its origins and to form *counterthoughts."*[15]

In 1916, shortly before he was about to be sent home, Marc was hit by a shell and killed at the age of thirty six.

WWI was the first mechanized war and, in addition to Franz Marc, cost the world nearly twenty two million people (civilians and soldiers, from wounds as well as disease, starvation and massacres).[16] But 1928, the year in which twenty six year old Jan Tschichold published his *New Typography*, was twelve years after Marc's death. Tschichold was sixteen when the war ended, and his late teens and early twenties were a period during which a sense of optimism for the future of Germany prevailed.

To be sure there was much to mourn, and there was dissent, but for the first time in German history, there was no ruling monarch. The bloody war was over and a liberal democratic republic (called the Weimar Republic after the city which served as it's seat) ruled over a newly free people. Under Weimar, women were granted the vote and participated actively in the workforce. Sexuality, including homosexuality, was an open topic of discussion in both fine art and popular culture. The work of Sigmund Freud began to enjoy wide distribution and psychoanalysis became a popular concept.

[15] Partsch, 91.
[16] Encyclopedia Britannica.

Popular entertainment was enjoying massive success, including an active nightlife and the famous cabaret scene. The fairly recent invention of cinema was growing in popularity as was a relatively new type of journalism, photojournalism, which spawned a number of publications so popular that they were printed on a weekly basis, sometimes recycling the same photographs in different contexts.[17]

The reuse of materials, aesthetic or otherwise, was a luxury newly available courtesy of industrial standardization, a hallmark of the industrial approach to manufacturing. Standardization marked a significant movement away from a long tradition of European craftsmanship that valued ornamentation and careful customization. Many Modernists perceived this tradition as an anchor around the neck of progress and, in 1926, Tschichold was no different:

> *"Car Aeroplane Telephone Wireless Factory Neon-advertising New York! These objects, designed without reference to the aesthetics of the past, have been created by a new kind of man**: the engineer!**"*[18]

> *"Contemplative introversion has given way to new realities, the thrills of active modern life. The theatre, an illusion of life, has lost its drawing power, for life itself has become a play: city street, neon signs, stadium! ...the striving for purity of form is the common denominator of all endeavor that has set itself the aim of rebuilding our life and forms of expression. In every individual activity we recognize the single way, the goal: **Unity of Life!**"*[19]

This spirit Tschichold distilled into the pages of *The New Typography*, intended not merely as a style guide for modern designers, but rather *the* contemporary book on design, which sought to condense all of the spirit of the Moderns, as well as Tschichold's and the Weimar Republic's optimism for the future into a truly contemporary language to be used for creative production. The mechanism for achieving utopia was engineering, and the hallmark of engineering was standardization, form in the service of function.

In terms of graphic design, the formal expression of this desire was the sans-serif type face, the direct result of the evolution of printing technology. Sans-serif type face was free of ornamentation, stripped down to it's essential parts and capable of standardizing communication in a completely

[17] Kaes, 769.
[18] Tschichold, 11.
[19] Ibid., 13.

revolutionary way. In order to understand this last point, it is perhaps helpful to realize that well into the Nazi era, German printing made extensive and nearly exclusive use of blackletter typefaces, making their printed works extraordinarily non-standard and difficult to disseminate outside of Germany.

Figure 3.1 Blackletter, Serif and Sans-serif typefaces (top-bottom)

Of sans-serif type Tschichold wrote that: "...it must be laid down that the sans-serif is absolutely and always better."[20] Better not only because it was aesthetically modern but also because sans-serif was the ultimate expression of a *functional* typeface, beautiful not only for its aesthetic, but also for the way it improved communication. Tschichold wrote:

> *"As a rule most people see only the superficial forms of technology, they admire their 'beauty' - of aeroplanes, cars, or ships - instead of recognizing that their perfection of appearance is due to the precise and economic expression of their function."*[21]

But within ten years the almost endearing naiveté with which the young Tschichold embraced modernist ideals was to turn deadly. Tschichold's adopted voice was that of Marinetti, a committed Fascist. As early as 1909, with frightening foresight - and when Tschichold was only seven years old - Marinetti spoke through the Futurist Manifesto:

[20] Ibid., 74.
[21] Ibid., 65.

> *"We will glorify war - the only true hygiene of the world... the beautiful Ideas which kill, and the scorn of woman. ...We will destroy museums, libraries, and fight against moralism, feminism, and all utilitarian cowardice. ...Therefore welcome the kindly incendiarists with the carbon fingers! Here they are!... Away and set fire to the bookshelves!"*[22]

ENTER THE KINDLY INCENDIARISTS

In 1933 Hitler was elected Chancellor of the German government. In the same year a fire broke out in the Reichstag, the German parliamentary building. It is quite likely that the Reichstag fire was set by the Nazis, but Hitler and Goebbels wasted no time in blaming it on Communists, hanging their case on the "confession" of a half-blind mentally disturbed individual. As the result of the fire, Hitler was able to enact an official decree suspending free speech which was used to silence opposition parties. Shortly thereafter Hitler was elected dictator, the Dachau concentration camp was constructed and the Nazi's began systematic, government run persecution of its citizenry.[23] Not only did those who were "racially inferior" suffer, but also "intellectual degenerates" including modern artists, designers and typographers.

Jan Tschichold, then acting as director of the Munich Academy of Graphic Arts, was placed by the Nazis into "protective custody" along with his wife, Edith. After they were released during a "general amnesty," the Tschicholds decided to leave Germany while they could and moved with their young son to Switzerland. Jan sums up what must have been a traumatic event in his stylistically straightforward manner: "Since freedom of thought and work for me come before everything else, I left my homeland and moved to Basel."[24] Jan would never again return to Germany.

More interestingly, the exiled Tschichold turned his back utterly on the modernist movement. The man who had once declared that sans-serif type "is absolutely and always better" began using traditional serif typefaces. In addition, Tschichold's affinity for strong graphic elements gave way to woodcuts, his asymmetry to symmetrical arrangements of type: exactly the style and "ornamentalism" the younger Tschichold had so reviled. This response was deliberately reactionary. According to Tschichold, the *New Typography*'s "...impatient attitude conforms to the German bent

[22] Stokstad, 1056.
[23] Kaes.
[24] Mclean, *Jan Tschichold: Typographer*, 133.

for the absolute, and its military will to regulate and its claim to absolute power reflect those fearful components of the German character which set loose Hitler's power and the second World War."[25]

CAR AEROPLANE NEW YORK!

While the technocentric optimism of the modernist movement was lost to Tschichold, it was not lost to the world. *New York* as a place and a concept was a clear inspiration to the modernist movement in Europe, and after Hitler's takeover of Germany, the movement was figuratively and literally transplanted to the United States. Bauhaus founder Walter Gropius moved to Massachusetts to teach at the Harvard Graduate School of Design where, to the chagrin of his more traditional New England neighbors, he constructed a modernist dwelling in Concord. Bauhaus director László Moholy-Nagy established the "New Bauhaus" in downtown Chicago. Josef and Anni Albers, also of the Bauhaus, put the Black Mountain College (where, among others, John Cage and Buckminster Fuller studied) on the map. Perhaps the most singularly recognizable legacy of the modernist émigrés were the designs of architect Mies van der Rohe. Mies' characteristically minimalist glass curtain architecture is a style which is now nearly synonymous with our notion of office building. While the New Bauhaus never attained the glory of its German predecessor, modernism found fertile ground in its adopted homeland, becoming the aesthetic language of choice during America's optimistic post war years. [26]

SETTING THE STANDARD

In part, the United States' ready adoption of the European modernist aesthetic was made possible by the fact that the industrial production that fueled the modernist's ardor was equally strong, if not stronger, in the United States. The success of the industrial boom in both places was based largely on the widespread adoption of industrial standards, and US manufacturing companies were quick to understand the advantage to industrial standardization.

Although not nearly the first to build a commercial motorcar, Ford motor company was founded in 1903, and became the world's largest manufacturer of automobiles by virtue of the speed of the first working belt-based assembly line, introduced ca. 1913. [27] The Ford assembly line itself owed its

---

[25] Ibid., 133.

[26] Some would say modernism was killed off by the second world war: *"The First World War did not destroy Modernism, but World War II almost certainly did. The optimism, the hope, the promise, the celebratory joy of life exemplified by [Matisse] .... could not continue beyond the totalitarian nightmare of brutality and genocide."* I disagree. Modernism as a fashionable approach for painting may not have survived, but this is willful disregard of the commercial arts, where modernist aesthetic set the tone for the 1950s, and thrives today. [Quote taken from http://www.sparkonline.com/october99/misc/art/podstolski.htm.]

[27] Ford Motor company website, available at: http://www.fordmotorcompany.com/.

success to the technique, developed by Ransom Olds of Oldsmobile, of using standardized interchangeable parts for manufacture.[28]

The Ford assembly line was operating for six years when, in 1919, the American National Standards Institute (ANSI) was founded under the name American Engineering Standards Committee (AESC). In 1919 AESC was "an ambitious program and little else"[29] but by 1926, the year the first SAT was administered, the AESC hosted the conference that was responsible for the creation of the International Standards Association (ISA), an organization that until the onslaught of World War II focused on international governmental standards. When WWII began, the need for standards was even greater, and ANSI's role was secure. From the history of ANSI published on their website:

> *"When the United States went to war in 1941, the ASA [American Standards Association - yet another incarnation of ANSI] was prepared with a War Standards Procedure that it had adopted nearly a year earlier. It was used to accelerate development and approval of new and revised standards needed to increase industrial efficiency for war production. Under these procedures, 1,300 engineers worked on special committees to produce American War Standards for quality control, safety, photographic supplies and equipment components for military and civilian radio, fasteners and other products."*[30]

After the war, the momentum of the standardization effort continued. "In 1946, delegates from 25 countries met in London and decided to create a new international organization, of which the object would be 'to facilitate the international coordination and unification of industrial standards.' The new organization, ISO, officially began operations on 23 February 1947."[31] The ISO continues to operate to this day.

From an "ambitious program and not much else" to an international organization of great importance to the manufacturing world, the ISO's development was driven largely by the need for manufacturing standards. It was clearly military response to the growing threat of fascism that forced this issue, in particular with regards to human beings and the increasing need to interface them with machinery of all kinds.

---

[28] Lienhard.
[29] ANSI historical overview from ANSI website, available at: http://www.ansi.org/about_ansi/introduction/history.aspx?menuid=1.
[30] Ibid.
[31] ISO website, available at: http://www.iso.ch/iso/en/aboutiso/introduction/index.html.

STANDARD MAN : ERGONOMICS AND ANTHROPOMETRY

Anthropometry and its descendant, the study of ergonomics, has proven a boon for the industrialized world. Anthropometric standards set the height of light switches, the width of door frames, the size of toilets, the height of stairs and the most efficient layouts for kitchens and other workspaces. When implemented properly, such standards ensure the usability of the spaces in which we live. Anthropometry can even ensure the usability of a space shared by those with a range of physical needs (by virtue of disability, age or vocation) by setting the parameters that define accessibility in public spaces. The word *ergonomics* comes from the Greek *ergo* meaning "to work" and *nomos* meaning "natural law."

> *"The word was coined by the late Professor Hywell Murrell, as a result of a meeting of a working party… [in] room 1101 of the Admiralty building at Queen Anne's Mansions on 8 July 1948 – at which it was resolved to form a society for 'the study of human beings in their working environment.' The members of this working party came from backgrounds in engineering, medicine and the human sciences. During the course of the war, which had just ended, they had all been involved with research of one sort or another into the efficiency of the fighting man."*[32]

But while the term *ergonomics* was coined post war, "the first sustained, systematic attempt to delineate the field, to develop principles and accumulate data, and above all to apply them, took place during the second World War."[33] *Anthropometry*, literally the measuring of man, was quickly being developed as the best means to ensure that the average "fighting man" could operate a vast array of equipment from numbers of different manufacturers. Machine guns, airplanes, tanks, jeeps, clothing, stretchers, radio equipment, gasmasks, submarines and body bags all needed to be standardized so they could be put to work quickly and without customization.

In 1960, an industrial design consultancy named Henry Dreyfuss Associates published a handbook on anthropometry standards entitled *The Measure of Man* followed more than 30 years later by *The Measure of Man and Woman*. In the latter they discuss the fact that the term Human Engineering was coined and used by the US Army, "Whereas *human factors* is used by other branches of the military." [34]

[32] Pheasant, 4.
[33] Damon, preface.
[34] Tilley, 7.

The Measure of Man and Women outlines the history of anthropometry in this way:

> *World War II required new and complex war machines, and the concept that machines, not personnel, win wars gave way to efficient man-machine relationships. Before long, many scientific disciplines were being consulted – psychology, engineering, anthropology, and physiology."* [35]

EUGENICS : WHEN STANDARDIZATION FAILS

The effect of anthropometry is widespread and largely positive, but the application of standards to other aspects of human existence, such as the human mind, can have disastrous consequences. In particular, the pairing of the measure of man with physical anthropology and a desire to standardize intelligence gave rise to anthropometry's evil twin: the "science" of eugenics.

Commonly associated with the Nazis and the horror of the concentration camps, eugenics was not an original fascist invention. While the Nazis certainly made extensive use of eugenic rhetoric to lend "scientific credibility" to their actions, the movement began in England and gathered steam in the United States before the rise of the Nazi party. In fact, a 1934 German racial hygiene law was drafted in reliance on a model bill written by the American eugenicist, Harry Hamilton Laughlin.[36]

Well before the Laughlin bill there was the British "grandfather of eugenics," Francis Galton, a nobleman who drew from his knowledge of distinguished people of history to come to the conclusion (in 1865) that "a disproportionately large fraction of them were blood relatives" and that as families of reputation were more likely to produce able offspring (in terms of physical features and character) the unworthy should be placed in monasteries and convents where they would be unable to propagate.[37]

By way of understanding contemporary eugenic rhetoric, it is important to note that Galton's original eugenic theories *did* take into account social circumstance (IE barriers to achievement caused by class structure) by deliberately disregarding them. Galton theorized that "Talent was rarely impaired by social disadvantage; witness the men of achievement who came from humble families; indeed,

[35] Ibid., 9.
[36] Reilly and Wertz.
[37] Kevles, 4.

witness the effect of the removal of social disadvantage in the New World. Culture is far more widely spread in America than with us."[38]

While it would be nice to assume that eugenics fell out of favor with the collapse of the Third Reich, this is not the case. Following WWII, US-occupied Japan passed a bill called the Eugenic Protection Law that permitted the sterilization of persons who had even distant relatives with any one of more than a dozen presumably inherited conditions. [39]

In the United States between 1907 and 1960 "…at least 60,000 people were sterilized pursuant to U.S. state laws. During the 1930s, the heyday of these programs, there were about 5,000 sterilizations a year, the vast majority performed on young women for most of whom the evidence of mental retardation was poor or non-existent."[40]

The quest of "purity through breeding" persists today even in policy, as in the case of the current government of China which advocates a style of family planning that openly embraces its eugenic history. As late as 1994 "China introduced the controversial Maternal and Infant Health Care Law, which makes pre-marital check-ups compulsory and allows doctors to order abortions of fetuses…."[41]

While the events of WWII proved an embarrassment to the proponents of eugenics, there are many who continue to argue in favor of the practice by recasting eugenics as simply misunderstood. In a bid to stave off "insidious genetic deterioration"[42] by virtue of "the fascinating prospect of man's limitless self-modification"[43] the author of *Fabricated Man* (published in 1970 by Yale University Press) says the following:

> *"…It will be too late (and indeed it will be inhumane) if we do not adopt measures to counteract the genetic deterioration which modern civilization and humanitarianism foster, and if we simply wait for balancing selection to overtake us and pull out the plug our hospitals now place in the way of the extinction of genetic defects."*[44]

[38] Ibid., 4.
[39] Reilly and Wertz.
[40] Reilly and Wertz.
[41] BBC Online Network.
[42] Ramsey, 8.
[43] Ramsey, 105.
[44] Ramsey, 3.

In 1994, Harvard Professor of Psychology Dr. Richard Herrnstein and Harvard and MIT graduate of political science Charles Murray co-authored *The Bell Curve: Intelligence and Class Structure in American Life.* This book "demonstrate[s] the truth of another taboo fact: that intelligence levels differ among ethnic groups."[45] Declaring unreservedly that "As far as anyone has been able to determine, IQ scores on a properly administered test mean about the same thing for all ethnic groups. A substantial difference in cognitive ability distributions separates whites from blacks, and a smaller one separates East Asians from whites."[46]

There is nothing particularly new or interesting in the eugenic white supremacy warmed-over arguments put forth by *The Bell Curve.* However, two things in particular are interesting about this volume: 1. The book makes extensive use of statistics gleaned from standardized tests, in particular IQ and SAT test scores. 2. The book was co-authored by MIT and Harvard trained professors, published by Simon & Schuster, backed publicly by a number of professors from established universities and debated extensively in academic circles.

A final example from the history of computation: One of the co-inventors of the transistor, Nobel Prize winning William Bradford Shockley, could not resist the temptation of applying industrial standardization techniques to the solving of social problems. *Shockley on Eugenics and Race: The Application of Science to the Solution of Human Problems* [47] analyzed the problems of race mixing and purportedly floated the idea of offering cash bonuses to those of low IQ who would submit to sterilization. This book was originally published in the 1980s and destroyed Shockley's reputation, although it can be said that his early work on the transistor truly enabled the existence of digital machines.

ABORT, RETRY, FAIL : THE MILITARY AND COMPUTATION

Physicist Richard Feynman in *Feynman Lectures on Computation*, reminds us again of the difference between computers and computation, by pointing out that "...Computer science, in a sense, existed before the computer. It was a very big topic for logicians and mathematicians in the thirties. There was a lot of ferment at court in those days about this very question – what can be computed *in principle?*"[48]

[45] Herrnstein, jacket cover.
[46] Herrnstein, 315.
[47] Shockley.
[48] Feynman, 52.

But the objects we use to access certain ideas have histories of their own that, often as not, greatly influence the way they are used. As far back as 1890, when punch card sorting and tabulating equipment was built for the US Census,[49] computers began to take recognizable shape. At least two hundred years before that time there were the mechanical analytical engines of Charles Babbage. Indeed the desire to "number crunch" and calculate larger numbers faster than either could be done by hand fueled most research into the idea of building calculating machines. If you include the abacus in the genealogy of calculating machines, the idea of embodying computation in an object pre-dates Babbage by four hundred years, and it is thought by some that indigenous cultures had mechanical calculating schemes even before this.[50]

But the digital computer with which we interact daily is quite a different beast. In 1947 Kiyoshi Matsuzake of the Japanese Ministry of Communications clearly beat Private Tom Wood of the US Army in a speed and accuracy contest of calculation. Tom Wood was using the most modern electromechanical calculating device and Matsuzake was using an abacus.[51] While this says much about the state of computation in the 1947 (and perhaps more about the skill of both opponents) it is doubtful that Matsuzake could perform the same feat today against a Pentium class machine.

What happened between the beginning and end of WWII created an intense demand for machines capable of assisting in calculating bombing runs, targeting trajectories and, especially, code breaking. Mechanical means were used first. Indeed analog mechanical techniques served until well into the 1960s and 70s[52]

A predecessor to the corporate computing giant IBM, the International Time Recording Company, was founded in 1900. As part of a growing industry based on card punch technology, IBM continued to expand even through the Great Depression. According to IBM's own archives, when the second World War began "All IBM facilities were placed at the disposal of the U.S. government" while "The war years also marked IBM's first steps toward computing." [53] The war was kind to IBM: "...After the many technical developments of the 1930s, the war years saw relative little advance in punched-card machine design due to the supervention of other wartime research priorities. The manufacture

[49] Aspray, 125.
[50] Ibid., 15.
[51] Ibid., 7.
[52] Ibid., 197.
[53] IBM Corporation website, available at: http://ibm.com.

and use of the machines, however, increased considerably. IBM, for example, emerged from the war with nearly twice the employees it had when the war began."[54]

One interesting side note to this history: while the war certainly greatly contributed to the development of digital computers, this occurred in a non-obvious way. Far from seeing the potential of digital, the military was perfectly happy with analog machines for gunnery calculations, missile stabilization and the like. The rise of digital was "…stimulated in no small part by the scarcity of the skilled labor required to manufacture and maintain precision mechanical systems."[55]

Once it was clear (by roughly 1970) that all mechanical analog calculations could be calculated digitally and were less prone to mechanical failure, and that this was far more cost effective thanks to improvements in transistor technology, the primary goal of those involved in computation became how to go about calculating more, faster. This quest has served us to great effect to the present day, fulfilling Gorden Moore's prophecy (called *Moore's Law*) that "the logic density of silicon integrated circuits has closely followed the curve (bits per square inch) = 2^(t - 1962) where t is time in years; that is, the amount of information storable on a given amount of silicon has roughly doubled every year since the technology was invented."[56]

This military legacy of computation infuses the language we use to talk about computers and computation. We program, execute, abort, and command the computer. We don't write, work with, talk to or dialogue with them. In fact, the notion of having a dialogue with a machine sounds somewhat out of place, and yet this is exactly what is happening when one programs interactively.

TSCHICHOLD AND THE LESSON OF WWII

Standardization, the hallmark of the movement towards industrialization that the modernist movements in art, design and popular culture embraced, is a powerful idea that enables a great many things to happen. But when applied to humanity, this quest for perfection becomes dangerous, in particular when it is paired with a value judgment, or applied to the quantification of intellect (or, as in most cases when both occur simultaneously).

Jan Tschichold, the designer whose younger self so confidently declared the unlimited awakening of the powers of man by means of "standardization, rationalization and mechanization,"[57] had a

[54] Aspray, 149.
[55] Aspray, 182.
[56] From the Hacker's Jargon File v4.43, available at: http://jargon.org.
[57] Tschichold, 12.

complete change of heart after experiencing first hand the rise of the Nazi party in Germany. Tschichold escaped alive and spoke publicly about what he experienced, and so we have a record of his work, his words and key dates.

For those of us born well after the end of the war, WWII is linked inextricably with the Holocaust. This is probably as it should be given the enormity of the crime, but the space of history has a tendency to distort the passage of time, and so contemporary readers of Tschichold might be tempted to assume it was the Holocaust that prompted his change of heart. In fact, Tschichold's transition is reflected in his work several years before he fled Germany, and was made public in his written works as early as 1935. While there undoubtedly are examples of government sponsored hate crimes and blatant anti-Semitism before this time, it wasn't until 1938 that the Holocaust truly began.

It is also likely that even if the Nazis had not taken power, Tschichold would have abandoned his absolutist position as his common sense as a designer overrode a youthful fascination with the modernist aesthetic. Nevertheless, Tschichold himself felt the need to align modernism with fascism and so we are left with the question: What was Tschichold reacting to, exactly, when he spoke about the "fearsome qualities of the German character" that gave rise to the Third Reich? It is not possible to know for sure, but given the preponderance of circumstantial evidence, it seems that Tschichold was in fact talking about himself.

Tschichold was not a member of the Nazi party. If anything, his writings show a politically leftist tendency, but there is no question as to the hard line, uncompromising nature of the designer Tschichold. To Tschichold, the purity of the page was *everything* and his work was not simply arrangement but a nearly fanatical search for truth and beauty through purity of form. Within the bounds of typography, this drive produced excellent results, exquisite layouts that serve as masterworks to be studied by designers today. Applied to human beings, this exact same approach resulted in unimaginable horror.

What is perhaps most terrifying about the Holocaust beyond the number of dead was the incredibly modernist method of murder employed by the Nazis. The Jews and others who suffered in the camps were not executed so much as processed: not only buried but recycled into products, not only murdered but used in manufacturing. This, I believe, was the ultimate horror of the Nazi regime for Tschichold: the utter perversion of his naïve modernist spirit into a ruthlessly efficient industry of

death. Records were kept, methods were researched, committees were formed and those murdered during the Holocaust were killed in much the same way we might manufacture computers.
But while we need look no further than the second world war for an example of the terrible consequences of this approach, the desire, especially of industrialized nations, to quantify human beings seems irresistible.

4 **STANDARDIZATION** IN EDUCATION

MENTAL ANTHROPOMETRY AND THE MISMEASURE OF MIND

Perhaps, by arguing in favor of racially based intellectual superiority and by finding that the results of standardized testing *actually do support this hypothesis,* the authors of the *Bell Curve* have touched a nerve. Perhaps ultimately they have also done us a favor, by illustrating the fundamental uselessness of standardized testing as a method of evaluation. Standardized tests do measure something, and the results can be compared to one another. Often, the results indicate how well one performs in school. The question I wish to ask then supersedes the question about whether or not standardized testing "works."

A letter to the editor of the Wellesley Alumni magazine, written by a parent supporting her daughter's boycott of the Texas standardized test (TAKS), phrases it this way: "As an economist, reading the school district ratings based on test scores, it seemed that the entire battery of tests could be replaced by one question – What is the median family income of this district? – without losing any predictive value."[58]

I am inclined to agree with this parent, but regardless of how one feels about this particular conclusion, the question stands: What are we really measuring with standardized tests? And closely on the heels of this: Do we *want* to measure this? If these tests measure "something else" and can also be used to indicate with some degree of accuracy the success of one's trajectory through school: What qualities do schools foster and encourage? Finally: If success in school is an indicator of status or used as an indicator of suitability for position in society, what does this say about the paramount values of our society?

These questions are unquestionably complex but are not irrelevant. The belief that computation ought to be used for quantitative tasks, or that expression with computers is acceptable but quantitative computational tasks are somehow more "real," can be seen as another expression of the same problematic paradigm that calls for standardized testing. As any college bound student will tell you, at the end of the day grades matter more than teacher comments. At the end of the day, C++, Java or even Logo will likely be considered a "more pure" method of accessing computation than what I will present below, by virtue of their similarity to what we expect code to look like. At the very least, I would ask the reader to question the fundamental ways of thinking that lead to such conclusions.

[58] Marciniak, 3.

STANDARDS vs. STANDARDIZATION

One of the problems with the adoption of industrial practice for education is a fundamental and occasionally deliberate misunderstanding of the difference between *standards* and *a standard*.

The word *standards* is a synonym for principles, values and level of quality. It is not the same thing as the word *standard* which indicates a kind of external framework or set of guidelines against which you can evaluate a person or thing.

Although they may debate what constitutes "highest," no one would ever argue that a factory, school or individual should strive for anything except the highest standards. But this is hardly the same thing as the adoption of a standard as method of evaluation. The words are not necessarily mutually exclusive, but they are not synonymous either. Yet consider the following text taken from the Massachusetts Department of Education's informational website on the statewide compulsory Massachusetts Comprehensive Assessment System (MCAS) exams:

> *"The fundamental goal of Education Reform in Massachusetts is to improve students' academic performance. The development of the high academic standards described in the Curriculum Frameworks was a step toward accomplishing that goal; the implementation of MCAS represents the next important step in the process. The Massachusetts Curriculum Frameworks and MCAS together create a statewide system designed to support students, parents, teachers, and schools by uniformly promoting high academic standards for all public school students of the Commonwealth."*

Up to this point we are clearly discussing academic *standards*, how to improve and maintain them. Surely a noble goal, but a subtle switch is about to occur in this consecutive paragraph:

> *"Results from the initial administration of MCAS serve as a baseline against which students and schools can measure their progress in achieving the newly established state standards. At the time MCAS tests were administered, many schools were still in the process of aligning their curriculum with the state standards. As a result, some students may not have been exposed to all of the content covered by the MCAS tests.*

Also, since this was the first administration of tests based on standards described in the Curriculum Frameworks"[59]

Without so much as a warning we find the topic has switched away from "high academic standards" and on to "newly established state standards." Away from standards as in "principles of education" to the word *standards* as used to indicate "the plural of standard." State standards are then explained in industrial terms as a *description* to be used, along with a method of establishing a *baseline,* in order to *measure* and correct the *alignment* of a school's teaching methodology.

By making this explicit I hope to avoid a misunderstanding. The argument presented here is not opposed to achieving high standards (quality). The argument is about the merits of using standardization and industrial methodologies (quantification) as a means to establish and maintain these standards.

STRIVING TO MAINTAIN THE AVERAGE: THE BENEFITS OF STANDARDIZATION

In order to understand why many push for standardization of education, it may be helpful to examine closely some of the benefits that standardization affords us from both an industrial and educational point of view.

1. Standardized systems ensure some level of quality control.
2. Standardized systems allow for a black box approach to development.
3. Standardized systems are relatively easy to maintain.

1. Quality Control

Industrialization of manufacture provides us with the ability to mass produce large quantities of merchandise with identical or very similar characteristics. This allows one to make assumptions that save time and money. For example, a factory that wishes to make a product that will be assembled needs to engineer very little with regards to fasteners, provided they are content using one of the wide range of standard screws, nuts and bolts they have available to them.

Quality control standards can also ensure safety. It would obviously be both impractical and impossible to field test every helmet, bullet proof vest or safety harness produced by industry.

[59] MA Department of Education website on MCAS, available at: http://www.doe.mass.edu/mcas/1998/bg/default.html.

Instead, such devices are manufactured to exacting standards which help ensure their suitability to task.

Certain professions rely on standardization of curriculum to ensure their employees arrive at their jobs with a skill set that is essential for their function. In many ways this is the function of higher education: ensuring that employers can count on the fact that their new hires have at least a rudimentary understanding of a particular domain. This actually makes sense with regards to some professions such as medicine, where thorough understanding of a knowledge canon is clearly beneficial (to say the least) to the employer and society at large.

2. "Black Boxing" and the After Market

Standardization allows one to treat a complicated system as a black box. Given the inputs and the desired outputs, it is possible to create a tool, material or resource for a system without having to understand how the system truly works. In education, the assumption is made that all humans at a particular age have the same mental characteristics and therefore that educational approaches can be generalized. We will return to this notion of generalization in the next section.

In industry, standardization allows for the creation of an after market. The software industry, for example, is entirely dependent upon the existence of computer hardware to run their products. It is only through standardization of hardware that software manufacturing is a viable business, allowing independent industries to co-exist and compete without revealing their trade secrets.

In education, standardization functions in a similar way by allowing an educational institution to make assumptions about the type and quantity of knowledge that their students will have contained in their heads at any given moment. This allows for structured planning of curriculum and, of course, allows the creation of an after market of textbooks, teaching aids and software products. The process of creating these educational materials is often entirely hands-off. The principle of production is simple: given the list of facts that "any third grader should know" one can create materials that will effectively take a third grader to the next level. This can be done without ever needing access to an actual child.

This point is significant and not meant merely as a swipe at the textbook industry. Although it is certain that some curriculum development takes place locally, public school systems by and large simply cannot afford to develop their own materials. The best interests of commercial interests are

commercial. Ultimately the textbook industry exists to turn a profit. And so the development, use and reuse of curriculum materials, coupled with financial interests to the tune of several million dollars a year, form a self-reinforcing cycle that ensures the amplification of ideas and mistakes in the same way audio feedback amplifies small noises.

3. Ease of Maintenance

The assumption of the black-box approach to education is that, given any normally functioning human, a particular series of stimuli will result in a particular outcome. While this is clearly not the case, it is an approach often used because it results in a system that is relatively easy to maintain. This is true by virtue of the fact that the industrial approach allows the manipulation of large numbers of individuals by reducing them to aggregate quantities.

This is perhaps the most compelling reason for the application of industrialization to school systems. It is impractical, indeed impossible, for a school board to meet and discuss educational options with each student passing through a school. Standardization of curriculum allows the administration to assure itself that a job is being performed. Standardized testing allows for a clear cut measurement of this job.

The question of *quality* here is truly of secondary importance, as a mixture of both truth and rhetoric compels the supporters of standardization to believe education is "in crisis." The rhetoric of crisis sets the stage for a belief that the situation is so dire, improvement must be made at any cost. Damage that might arise out of the imposition of standards is worthwhile, as it is more than offset by the guarantee that the standard of education is at least *acceptable*.

In most cases, this is achieved by enforcing standardization in a "high stakes" manner that, perhaps unintentionally, suppresses innovation and exploration by both learners and teachers, because of the perceived unacceptable risk these types of activities have for failure. *Acceptable* is the goal this method of standardization wishes to achieve and acceptable is what it does achieve, with no real chance for improvement over the baseline.

THE INDUSTRIAL APPROACH TO UNEXPECTED RESULTS

Reading between the lines up to now, it should be clear that industrialization is not the best approach to education. But perhaps the most compelling reason to consider an alternate approach is found in the way that industrial systems deal with unexpected or undesirable results.

If a factory produces unacceptable results, the solution is to recalibrate or replace the factory's machines until the product is acceptable. If one accepts this industrial view of education and also accepts the view that our educational systems is producing faulty product, the solution is the same: Our measurements indicate substandard output, we must recalibrate the system. Favorable outcome will be indicated by the result of further testing.

But the question is, what of the product? If a factory produces a batch of faulty devices, the devices can easily be recalled and destroyed. While eugenics may offer us this alternative, it is not a course of action that most would consider appropriate for human beings.

A more humane possibility, employed by manufacturers whose product is too complicated, expensive or rare to destroy, is to remedy the faulty outcome by targeted replacement of the faulty part. This is akin to the manner in which an industrialized educational system treats those who fail to meet the standard.

The goal is always a move towards the norm. Nonstandard power connectors, for example, have adaptors that enable them to be interfaced with more standard connections. In education, those who fail to conform to school standards are placed in remedial courses in order to catch up with their standardized peers. The language here is telling: the purpose of these classes is to *remedy* a problem. To bring a learner "up to speed" and to re-integrate them into the standard stream. Despite the fact that use of the word *remedial* has largely fallen out of favor, the existence and function of such classes has not changed, nor has adherence to the industrialized method.

TEACHERS AS HACKERS

Aggregate quantities, and indeed standardization, work best when you are dealing with a large number of objects that have similar approaching identical characteristics. These methods work well for ball bearings, toilet paper, cars, furniture and tablecloths. These methods do not work well with human children, and two minutes in the presence of one will tell you why.

Children's insatiable desire and ability to learn will force its way around the boundaries of curriculum and standardized testing. But while any given school system may choose to consider its population an aggregate quantity to be dealt with, there are a group of individuals who, on a daily basis, deal with students as individuals: teachers.

Anyone who has ever worked with children understands the discrepancy between the desire to standardize education and the reality of teaching. Good teachers customize material all the time, mixing and matching from what they are given to both satisfy the curriculum requirements and succeed in educating their students.

Computers may play a unique role in this equation. While the impetus for the introduction of computational machines into the classroom is often highly technocentric, and while the expected use of computation is quantitative, there is little reason that this must remain the case.

Given access to hardware, teachers may use computers to educate in a way the education department's purchasing office never dreamed. Papert saw this possibility in *Mindstorms,* in an era where the idea of a personal computer was still closer to science fiction than fact. Today the rapidly decreasing cost and growing availability of computational resources has made this possibility more of a reality. Teachers are the key, as they will ultimately be the ones working closely with students and computation.

In order to see how it might be possible for teachers and computation to exist in a less industrial manner, even within a standardized system, it may be helpful to see the way in which computer hackers have existed and continue to exist alongside a system historically defined as top-down. In fact as we will see, not only has this group co-existed, it has in many cases driven the development of computational hardware and software.

5 **HACKING,** CYBERCRUD AND EPISTEMELOGICAL PLURALISM

EPISTEMOLOGICAL PLURALISM AND THE HISTORY OF COMPUTATION

In the 1990s, Seymour Papert and Sherry Turkle co-authored an article entitled *Epistemological Pluralism and the Revaluation of the Concrete.*[60] In this article, Papert and Turkle argue that "in both the popular and technical cultures there has been a systematic construction of the computer as the ultimate embodiment of the abstract and formal."[61]

The article aligns this with the work of feminist scholars, connecting the historical inequality of the traditional, hard, male approach with the soft, bricolage, female approach to the story of science and computation. While Turkle and Papert are quick to point out that "When we say that hard and soft approaches are ideal types, we signal that individuals will seldom conform to either exactly, and that some will be so far from both that it is impossible to assign a type,"[62] the instance on labeling these approaches male/female does less to support their work than is supposed.

Inevitably arguing from an (admittedly unfair) place of weakness, it will be extremely difficult to successfully argue a "separate but equal" respect for styles of approach. More problematic, aligning these issues with gender fails to answer the following question: Why does hacker culture, undoubtedly male dominated and openly sexist, rarely work in the "hard" mode? The answer that hackers rarely conform to the cultural male ideal is both true and insufficient. The same argument could be made to science in general and yet we clearly see discrepancy between science canon and practice.

What I would like to propose is a slight modification to Papert and Turkle's arguments: that what they describe is not a devaluation of an epistemology (a way of thinking) but rather an over-valuation of a particular mode of expression.

Phrased another way: Formal "hard" language is a tool for expression, a means of communication typically favored by science. Epistemology as a description of a way of knowing occurs *prior* to the act of expression. Formalism is, therefore, not directly comparable to bricolage as a method of practice.

In terms of practice, while Turkle and Papert admit that hard and soft are ideals and do not represent everyone at all times, I posit that as defined, the hard and soft approaches will never succeed in defining *any* individual's approach to problem solving. All human endeavors pull freely

[60] Papert and Turkle, *Epistemological Pluralism*.
[61] Papert and Turkle, *Epistemological Pluralism*, 1.
[62] Ibid., 4.

from both, though some may chose to communicate their experience in different ways and for different reasons.

If these approaches can be organized along gender lines this is likely symptomatic. Bricolage's association with the feminine is not due to the fact that females prefer bricolage, but due to prolonged socialization which devalues bricolage (and creative practice in general) by emasculating it.

This is an important distinction because 1. It supports the work of Piaget and descendants by privileging the search for epistemology above the debate over methodology. 2. It is far more likely that one will be able to convince the dominant paradigm that it is insufficient than it is that one will be able to join paradigms cast as fundamentally different, especially given the stated inequality between these paradigms.

By making a case for unified epistemology, we can demonstrate that "hard" language is highly deficient in describing *both* male and female experience with technology. Formal language in discourse is the dominant way of discussing knowledge, but it is simply incomplete.

In any case the result, a devaluation of the concrete, is very real. This thesis enthusiastically supports the call for a revaluation of the concrete and shares the view that: "...besides being a lens through which personal styles can be seen, [computation is] ...a privileged medium for the growth of alternative voices in dealing with the world of formal systems." And the conclusion that "...as a carrier for pluralistic ideas, the computer holds the promise of catalyzing change, not only within computation but in our culture at large."[63]

HACKERS, HEROS OF THE COMPUTER REVOLUTION

In his book that lends its title to this section, author Steven Levy traces the history of computation as a story of a group of misfits that loosely organized itself at MIT and then spread out, forming the base of the computer revolution. The most frequent criticism of Levy's work is that it is too narrowly focused on one group of people and one institution. This criticism is valid (if nothing else consider the fact that in Levy's pantheon of heroes there appear only two women, one who is superlatively described as "timid" and the other as "defiant"). The history of computation is rich and varied, and the interdependencies and motivations are complex. What is quite valuable in Levy's alternate

[63] Ibid., 2.

history, however, is the emphasis placed on hacker culture: a community of practice that places style, technique and getting things done above structure, rules, top-down organization and methodologies. Here is the hacker ethic as codified by Levy:[64]

1. Access to computers – and anything which might teach you something about the way the world works – should be unlimited and total. Always yield to the Hands-On Imperative!

2. All information should be free.

3. Mistrust Authority – Promote Decentralization.

4. Hackers should be judged by their hacking, not bogus criteria such as degrees, age, race, or position.

5. You can create art and beauty on a computer.

6. Computers can change your life for the better.

Whether or not you believe computation as we know it exists because of the actions of the few dozen people mentioned by Levy, it should not be hard to connect this group's frustration with that expressed in *Epistemological Pluralism*. The subsequent quest to use the computer as a means to encourage revaluation of the concrete, in particular when confronted with social inequity, are also shared by both. The following text introduces self described innovator Ted Nelson's text *Computer Lib:*

> *"THE PUBLIC DOES NOT HAVE TO TAKE WHAT IS DISHED OUT.*
> *... I have an axe to grind. I want to see computers useful to individuals, and the sooner the better, without necessary complication or human servility being required. Anyone who agrees with these principles is on my side. And anyone who does not, is not.*
> *THIS BOOK IS FOR PERSONAL FREEDOM.*
> *AND AGAINST RESTRICTION AND COERCION...*
> *A chant you can take to the streets:*

[64] Levy, assembled from chapter 2.

COMPUTER POWER TO THE PEOPLE!
DOWN WITH CYBERCRUD!"[65]

Less polished than the arguments put forth in *Epistemological Pluralism*, these words nevertheless express the same frustration: computers may be the children of quantitative analysis, but they have the potential for much more. Computers can be used as agents for social change. The top-down approach is often stifling, and it is time to do something about it!

MASS DISTRIBUTION OF WONDERMENT

The top-down approach favors black boxing equipment, allowing for users to tweak parameters but never change the rules. The idea is that things are best left as they are, that experts define the standard and the users operate within it. While parameter tweaking does play a valuable role in understanding any system, it does not in itself qualify as a truly Wonderful Idea. The users must be allowed to observe and, ideally play with the limits of operation. This sentiment is expressed nicely in the following quote from *Hackers*:

"The eventual goal would be a mass distribution of wonderment... An environment conducive to the Hands-On Imperative. As [Bay area hardware hacker] Lee [Felsenstein] told a conference of the Institute of Electrical and Electronic Engineers in 1975, 'The Industrial approach is grim and doesn't work: the design motto is "Design by Geniuses for Use by Idiots," and the watchword for dealing with the untrained and unwashed public is KEEP THEIR HANDS OFF! ....The convivial approach I suggest would rely on the user's ability to learn about and gain some control over the tool. The user will have to spend some amount of time probing around inside the equipment, and we will have to make this possible and not fatal to either the equipment or the person.'"[66]

What Felsenstein was calling for here was the ability for anyone, at any time, with or without authorization, to use experimental *scientific method* to explore and, if necessary, improve the inner workings of a system. For Felsenstein and the other hackers, the issue was never about hard and soft, the "wrong" or "right" approach to exploration. The fight was for the *ability* to explore. Undoubtedly the hackers would continue exploration with or without permission, but with permission this exploration could be turned into a force for social change – mass distribution of wonderment.

[65] Levy, 168.
[66] Levy, 234.

A final point to be taken from Felsenstein's quote is with regards to the role of the facilitator, the "we" whose job it is to make exploration possible and non-fatal. This definition is truly an excellent way to think of facilitating an educational experience by virtue of the level on which it suggests we operate. Facilitators in this role are absolutely necessary to ensure that the experience of exploration is not fatal (literally or intellectually), but in this role the facilitator relinquishes their role as the center of the experience. Essential but almost in the background, the facilitator enables learning but does not define it.

WARNING: THIS MODIFICATION WILL VOID THE WARRANTY ON YOUR APPLE

Many of the hackers and their friends as named by Levy (Marvin Minksy, Bill Gosper, Richard Stallman) continued their hacking careers in academia, expanding various branches of computer science. Still others (Bill Gates, Adam Osbourne, Stephen Wozniak, Ken and Roberta Williams) took their talents to the street, founding companies such as Microsoft, Apple and Sierra On-Line that would eventually bring in billions of dollars.

While commercial realities have led most of these companies to abandon the hacker ethic (at least in terms of their closed-system end products) this was not always the case. Vestiges of the hacker spirit, the hands-on imperative, can be seen in the text of the c.1980 Apple II Reference Manual, printed at roughly the same time Seymour Papert published *Mindstorms.*

Packaged with the computer itself, the Apple manual contains the expected setup guide, a rundown of features and a surprise to those accustomed to contemporary manuals: a complete schematic of the motherboard and a chapter that contains instructions, with close-up photographs, of how to modify your machine using a soldering iron.

The modification the manual describes is mundane (a conversion of the monitor frequency for those who wished to use their Apple overseas) but the fact that this act was openly condoned, even assisted, by the manufacturer is significant. At Apple in 1980, the sole concession to the closed, standardized approach was the following two lines of text printed on page 10 of the manual: "Warning: This modification will void the warranty on your Apple… This modification is not for beginners."

In light of this it is not difficult to understand the tremendous hope for the future that Papert saw when he talked about the merger of the new personal computer with dissatisfaction with school

systems.[67] Modifications are not for beginners, but the sole criteria for leaving this status is a willingness to dedicate oneself to exploration, and computers appeared to be just the tool to accomplish this.

APPROVAL OF THE ACADEMY

History has certainly privileged the formalized abstract mode of communication, giving value to only those things that can be formalized. This has often created situations where good ideas are denied a means of existence by virtue of the fact that they cannot successfully find a translator willing to put them into "hard" form. Such ideas have no voice in the academy.

As a result, the written history of computation follows a path of prizes and speeches, papers and conferences, military and industrial processes and their products. A look at practice tells us quite a different story. And a closer look at more recent developments in academic practice shows that cracks are forming.

If one seeks official approval that "soft" issues are worth considering, they need look no further than MIT, the place marked by Levy as the birthplace of hacker culture. That MIT is the *sole* environment that has historically fostered creative computational thoughts is debatable, but that MIT contributed in a major way is not. Hacker culture at MIT is a way of life, and to the extent that it considers practice and the social applications of science and technology, is something that infuses even the administration.

The existence at MIT of the Comparative Media Studies department, Science Technology and Self program, and the Media Lab, as well as initiatives such as the Center for Reflective Community Practice and the Initiative for Technology and Self are testament to this. MIT, as an organization, believes that it is important to consider all aspects of technology in the human experience, including those traditionally labeled soft. The respectability and status of these departments afforded by the university and the public at large is testament to the belief that a large portion of the population shares: these issues are truly important.

Unfortunately, many smaller and less endowed schools do not have these programs. In part because, like art in public schools, these issue are perceived to be "nice to consider if you have the money" but not strictly necessary to the education of students. It is true that MIT is well endowed,

[67] Papert, *Mindstorms*, 181.

and likely true that if this were not the case (or if some catastrophic reversal occurred) these programs would not likely exist. This consideration ought to be disturbing. While MIT should be praised for considering the social application of the technology it produces, the fact that other universities do not and that this is *not perceived as a fundamental lack* is highly problematic.

Traditional educational practice calls for a canon of knowledge, enforced by the work of canonical educators, whose function it is to ensure the standard is always applicable. As we have seen, the industrial approach to education fails to take into account the reality that every educational situation and every child is different. The hacker approach calls for infrastructure that does not insist on what should be done with it. In fact, a great deal of learning comes from using the infrastructure in unexpected ways (and a great deal of humor comes from the greatest possible discrepancy between expected and actual use). In terms of education, diversity of opinion and approach are essential. This thesis will present one possible approach, intended to add to and encourage a line of educational thought that provides an alternative to the status quo. The focus will be on a particular problem, because in the words of Jan Tschichold: “In some fields we have made great progress, but in most others almost everything remains to be done.”

6 **TIMESHARING** TRINITY

THE TIME SHARING TRINITY: SCREEN, KEYBOARD, MOUSE

In December of 1968, HCI pioneer Douglas Englebart presented the "mother of all demos" to the world: the NLS or oN-Line System, an early working prototype of a multimedia, multi-user time shared computer system. This system, the product of six years of work with the Augmentation Research Center at Stanford Research Institute, included videoconferencing, a very simple windowing system, hypertext and the never before seen computer "mouse."[68]

"NLS was designed around a Scientific Data Systems SDS 940 time-sharing computer with an approximately 96 MB storage disk. It could support up to 16 workstations, which were composed of a raster-scan monitor, a three-button mouse, and a device known as a chord keyset."[69] The system was then connected from the demo site to Menlo Park via a 50kbps data line.

At MIT in 1963 came the project MAC summer study, an attempt to determine the future of computation by bringing a number of researchers to MIT's campus to work with the Compatible Time Sharing System (CTSS).[70] A year later, in the Fall of 1964, the Dartmouth Time Sharing System became operational with BASIC as principle language for student program development.[71]

All of this work owes a historical debt of gratitude to the Video Display Terminal (VDT), whose development was spurred by the Bell Labs timesharing system Multics. The VDT was the first computer "screen," which also provided data entry by means of a teletype keyboard (a direct descendant of the venerable typewriter).[72]

Prior to the introduction of VDT and timesharing, the computer was essentially on par with an expensive machine-tool. A physically large device, a computer was costly to own and operate, required much training and a special environment in which to run. Typically, the care of computers was given over to designated operators. Most programming was done via punch cards, away from the cmputer itself, then submitted to an operator for processsing and the results returned (perhaps days) later. [73] (Incidentally, it is worth pointing out that this method of working does little to support one of the primary reasons that we use digital tools today: the ability to quickly iterate over a

---

[68] MouseSite.
[69] Wikipedia, *NLS*.
[70] Levy.
[71] Computer.org, *1964.*
[72] About.com, *Computer Keyboard.*
[73] Incidentally, this method of working does little to encourage the primary reason that we use digital tools today: the ability to quickly iterate over a problem – to attempt many different things because we can always "revert" to the original.

problem. The transition to an interactive form of computation changed the perceived application of the tool.)

Timesharing enabled a much larger user base. A single, expensive computer, could be shared by many users. Levy writes: "There were all sorts of justifications for this, not the least being the projected scale of economy. ...But there was something else involved... ...the belief that computing, in and of itself, was positive."[74] No one, besides a hacker, says Levy, would have believed in 1964 that it was worthwhile for everyone to have regular access to computation (considered equivalent to access to computers), but this was quickly to change. Ultimately time sharing was defeated by its own success as it became impractical to implement with increased user load. Growing demand and improved methods of production continued to drive down the cost of computation, and timesharing yielded first to the microcomputer and eventually to the personal computer. Thanks to this the unimaginable occurred: in the space of forty years we have gone from computation as a scarce resource to having more computational power per person than we know what to do with.

Despite the fact that this is true, it is telling that in all this time the basic computer interface remains largely untouched. Englebert's NLS workstation is an easily recognizable, if dated, version of the machine I am composing this text on.

THE SUCCESS OF "MEDIEVAL TORTURE DEVICES"

Many have decried the standard computer interface, declaring in articles with titles such as *The Tyranny of the Keyboard*[75] that the current interface is nothing short of a "medieval torture device." Nevertheless, the existence of the timesharing trinity has persisted, and for good reason: this arrangement actually does its job as a multipurpose device very well. More than forty years of application in every domain from mathematical modeling to fine arts has turned up few advancements in hardware, while the establisment of a solid billion dollar personal computing industry serves as testament to the utility of the screen-keyboard-mouse combination.

But while the general-purpose desktop interface has served us well and is likely to stick around for a long time to come, there is a growing sense that this isn't the only way that one can talk to a computer.

[74] Levy, 55.
[75] Hersh.

HUMAN COMPUTER INTERACTION

Human Computer Interaction, or HCI, as a cohesive body of ideas and research is a fairly new phenomenon. The objects that HCI studies (digital computers and their various input/output devices) were really not in existence until at least the 1960s, and the purpose of HCI (to improve human interaction with computers) was not particularly useful until commercial computing became a reality, nearly twenty years after that. [76]

Insofar as HCI can be said to be the study of interface design, this field can trace its roots directly to anthropometry and human/machine interactions. The difference, of course, is the application of these ideas to the digital machine, which is far more multi-functional and flexible than any mechanical device. This flexibility brings with it a whole host of new and interesting problems, not the least of which is the point and the method at which the designers of the computer choose to allow the digital machine to interact with the physical world.

GRAPHICAL USER INTERFACES

Graphical User Interfaces are a familiar part of the contemporary computer experience. The development of these interfaces has a rich history, but it is truly the Apple Macintosh interface, introduced by way of the Apple Lisa in 1982 and the actual Macintosh in 1984,[77] that define our GUI experience. Prior to this, computers used text based interfaces (when they weren't using punched cards) via a keyboard or modified teletype.

Multi window GUIs mark a fundamental shift in the understanding of how we can use computers. If timesharing can be seen as allowing a single computer to support multiple users, multi windowing can be seen as the logical next step: allowing a single user to simultaneously operate on multiple tasks.

One of the most powerful ideas at work here is the notion of manipulable objects and the notion of multiple, changing representations. Visually, one of the most significant design choices made by the designers of these representations is also one of the most subtle: the use of color and shading to mimic three dimensional objects on screen. Strictly two dimensional interfaces tend to be far more difficult to use than those that mimic three dimensional effects, such as shading and "buttons." These effects are not mere eye candy, they actually enable users to work more quickly by easing the

[76] Myers.
[77] Ibid.

ability to differentiate objects. This is the case because of the way our brains work in space, assigning levels of priority to objects based on how close they appear in our visual field.

In a sense this design decision, made possible by the availability of excess computation, represents one of the earliest examples of computer interface designers making clear concession to user comfort based on the way human beings operate.

TANGIBLE USER INTERFACES

Tangible User Interfaces (TUIs) are the next logical step in this process: not merely introducing mimicry of the analog world, but introducing the analog world itself into the computer system. Hiroshii Ishii, the director of the Tangible Media Group at the MIT Media Lab explains on his group's website: "The goal [of our research] is to change the 'painted bits' of GUIs (Graphical User Interfaces) to 'tangible bits,' taking advantage of the richness of multimodal human senses and skills developed through our lifetime of interaction with the physical world."[78]

In its early form, this quest can be seen as a reaction to the decidedly unfriendly approach computational machinery has historically taken towards interface. Humans live in the analog, physical world – intimately familiar with space, light, color and tactile feedback. The original computers paid little heed to the needs of their users, providing input and output nearly as an afterthought. GUIs, a vast improvement over the days of punch cards and monochrome text-only screens, did much to improve usability but leveraged only the *metaphor* of physical space, whereas TUIs seek to leverage the *reality* of it.

One characteristic of TUIs vs. GUIs can be seen in the specificity of task that often accompanies the use of a TUI. While GUIs (windowing systems) support applications as diverse as photo editing and text composition, it is unlikely that a TUI will ever be quite as flexible.

The characteristic should in no way be perceived as a failure. In fact, specificity of task is exactly what makes TUIs so intuitive. By leveraging physical context, TUIs can afford to be far more ambiguous than GUIs. This particular characteristic lends itself to the use of TUIs as ambient data devices rather than as tools requiring full user focus. [79]

[78] Ishii.
[79] Dahley.

MIDDLE GROUND: AUGMENTED REALITY

One of the fundamental principles of computer science is that digital computation as we know it does not change the type of problems we can solve, only their size and scope. In other words, given a calculator, or even a pencil and paper, one could solve any problem solvable using digital computation. The sole advantage to using computation, then, is the ability to solve bigger problems faster. This is no small feat. The order of magnitude between those problems one can solve by hand and what problems one can solve on a computer is so massive that it effectively (if not theoretically) means that computation allows us to solve otherwise unsolvable problems. Simply put: there are particular characteristics of computation that allow for tasks not practically possible in the physical world. This fact becomes even more apparent when one considers the realm of expression. Examples include the otherwise impossible musical compositions created by contemporary electronic musicians, or the building plans of architects such as Frank Gehry which rely on computation for their existence.

By reintroducing the physical world as a requirement for interaction with computation, we run the risk of losing the advantage of using computation in the first place. In a bid to combine the best of both, TUI research has recently focused its efforts on the creation of *augmented reality*. The idea here is to "…allow the combination of the advantages of physical interaction with the dynamic qualities of graphical displays."[80]

OF KEYBOARDS AND PUNCHCARDS

When punched cards were the preferred method of communicating with computational devices, keyboards seemed like an excellent idea. Now that keyboards have a long and established history, as do graphical user interfaces and even tangible user interfaces, we are able to ask more clearly: what is computation really for?

Computation had its birth in the desire and need to compute large numbers. This has and will continue to be accomplished to great effect, but the result of this work (a surfeit of readily available computation) has enabled us to apply computation to a great many other tasks.

With these new applications of computation have come a new set of questions occupying our quest for environments that will be used to introduce children to computational concepts: Is the dominant method of manipulating computation (text based programming) which was developed and used well

[80] Piper.

for the *traditional* modes of use (quantitative analysis) really appropriate to all uses of computation? Specifically, is this method appropriate in education?

Finally, what of the interface? Augmented reality leverages the familiarity of our analog world and our natural inclinations to hold, shape, poke and prod with our hands to the infinitely malleable target allowed by computation. 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 things move, bend and break. In designing the environment described in section IV I will combine elements of both TUIs and GUIs.

POSTLUDE TO SECTION II

We have just explored how the roots of computation come from the same place as the roots of the standardized approach to education. For many years computation has been considered capable of changing social structures, and in some cases much headway has been made. By keeping this history in mind and closely reconsidering the raw material (how computers function, the power of computational ideas apart from this embodiment and the social implications of technology) we have an opportunity to accelerate this process of change.

Specifically, this thesis proposes a new way of introducing young children to computation Ultimately this is only a small fragment of the wide range of possibilities. In any case the first step to providing a possibility is to establish a framework for design, a task we now turn to in section III.

Section III Towards a Framework for Design

PRELUDE TO SECTION III

The development of computational devices is the expression of a longstanding desire to improve humanity's ability to analyze quantitatively. Quantitative analysis is a hallmark of industrial processes which inform the character of contemporary education. In many cases the industrial approach to education is less than ideal, and the traditional means and methods of accessing computation do not encourage computational thought.

The act of education is overtly political. The first chapter of this section will briefly visit, by way of a particular school, the notion of education as an act of resistance: in this case constructivist education as resistance to fascism. This is done to help demonstrate how and why computation in an educational context might be used as a force of empowerment and as an antidote to industrialized education.

The latter half of this section will take into account the historical analysis presented above in order to arrive at a set of design guidelines for creating and analyzing spaces for computational expression. Finally, this section will examine existing work in light of the guidelines presented. This is done as an introduction to section four, where a new system for computational expression for children ages four to seven will be presented.

7 **DEMOCRACY** TECHNOLOGY AND GRASSROOTS INVENTION

CHILDREN AND POLITICS

There is a peculiar discomfort that arises when one discusses politics and children at the same time. There is a longstanding notion that children, in their innocence, should be sheltered as much as possible from the ugly adult world. And there is certainly a level on which this discomfort is founded: almost as many political causes have been forwarded "for the children" as have been forwarded in the name of "God and country." In his introduction to *The Children's Culture Reader*, Comparative Media Studies Professor Henry Jenkins takes a careful look at the myth of childhood innocence and the extent to which the emblem of child and childhood drives actual and social politics. Jenkins writes: "Childhood – a temporary state – becomes an emblem for our anxieties about the passing of time, the destruction of historical formations, or conversely, a vehicle for our hopes for the future. The innocent child is caught somewhere over the rainbow – between nostalgia and utopian optimism, between the past and the future."[81]

Childhood becomes a signifier to adults who have "ugly adult" aims and make particular choices to achieve these aims. Children, the embodiment of childhood, become the unwitting recipients of these choices. The act of education, of selecting bits of information and conveying them with authority to another, is an overtly political act, influenced by historical events and choices. These choices are sometimes political, sometimes not, sometimes in recent memory and sometimes part of a tradition that has forgotten why it behaves the way it does.

Constructivism has historically maintained ties to politics, especially to radical politics, Marxism and Communism. The Brazilian Paulo Freire (1921-1997) has as much (perhaps more) to say about class signifiers and status as he does about "school." Ivan Ilich, the author of *Deschooling Society*, spent most of his life working as a Roman Catholic priest, resigning his priesthood as a last resort when the Vatican ordered him to leave an organization he helped to form. Leontev, Vygotsky and others draw on Marxist ideology to inform their work, and all are considered required reading for understanding constructivism in practice.

In light of this, it is curious to me that much educational writing finds itself in the rarified space of academic neutrality – a space where the political and religious views of some of the most progressive educational thinkers are regarded as superfluous at best and embarrassing at worst. While it is undoubtedly true that, in the words of one review, "many are put off by Paulo Freire's

[81] Jenkins, 5.

language and his appeal to mystical concerns,"[82] it is not true that these concerns played no role in Freire's thinking. Far from it! Given the overt political stance Freire takes and the care taken in his writing and the fact that he held political office in Sao Paulo, one can safely say that these "off-putting concerns" played an extremely significant role in Freire's thinking. The same can be said of Illich. While Illich far from wholeheartedly embraced the Church's dogma, the fact that he spent nearly twenty years of his life as a priest, leaving only when forced out, speaks volumes about what drove him as a thinker.

I am not advocating a necessarily radical approach for all concerned with education. Nor am I suggesting that one ought to agree with any author's political or religious stance in order to draw from their writing. What I wish to say is only this: Education is an inherently political act. It is far better that we proceed by acknowledging this and then honestly stating our claim, rather than by taking the far more dangerous route of pretending that there is no motivation. Far from adding unnecessary complication, declaring ones political intentions may lend great strength to the cause. Consider the success of preschool education in Reggio Emilia, Italy.

REGGIO EMILIA

The Reggio Emilia preschools of northern Italy, arguably the most compelling and successful example of constructivism in action, were created in direct response to the experience of living in a fascist state. Benito Mussolini, a one time secondary schoolteacher, ruled Italy from the 1920s until his execution at the hands of "freedom fighters" towards the end of WWII.[83] Predictably, Mussolini's rule included a complete re-writing of schoolbooks and the demand that all teachers pledge their allegiance to the state.[84]

What was to become known as the Reggio approach was established shortly after the end of the war, when working class parents in a historically politically active area of Italy built new schools for their children, focusing their efforts on ensuring the schools would be places "where children could acquire skills of critical thinking and collaboration essential to rebuilding and ensuring a democratic society."[85]

[82] Infed.org, *Paulo Freire*.
[83] Biography.com.
[84] BBCi.
[85] New.

While it is thankfully true that the Reggio of today operates a generation removed from the shadow of Mussolini, the early survival of the schools and strong sense of community they stand for today are testament to the sense of political importance with which the schools were founded. It is also important to note that the hope for the future that characterized the founding of Reggio was of a fundamentally different character than that which had driven most of Europe up to that time. While the rise of Fascism may have been a catalyst for the formation of the Reggio schools, they owe their existence to the community and the latent politics of the area in which they formed. Overtly democratic thought and a desire to remain free of the influence of any top-down authority (including the Catholic Church) was a kind of "pre-existing condition" in the Reggio area that lent a great deal of momentum to the establishment of the Reggio schools.

Longevity, lack of compromise, level of community involvement and high quality of work are all characteristics of Reggio which allow it to stand out from other experiments in education. It is certainly true that many alternative schools have and continue to exist all over the world (in particular the Modern School Movement is worth a close examination[86]). By privileging Reggio here I do not wish to denigrate the excellent work done by many of these organizations, I only wish to emphasize the effect that an overtly political stance has in mobilizing community and ensuring support for education. Because the community shares a belief structure that is greater than "mere school," Reggio succeeds in going far beyond the "child storage plus socialization" approach that seems to define many preschools.

I hope this also explains why I have provided the historical review offered in section two. Unlike the industrialized approach which assumes standardization and the existence of a norm, or the Fascist approach which is violently anti historical, the constructionist approach recognizes the importance and the uniqueness of an individual's history and questions the assumptions that lead to normalization. For a constructionist approach to succeed in an institution it must, as with Reggio, recognize the individualized history of practice (rather than the canonical history of theory) and explicitly define the space in which it wishes to operate.

ASKING LOUD QUESTIONS

It is only fair, given what I said regarding the need to state one's political intentions, that I explain my own stance. As an undergraduate at the School of the Art Institute of Chicago, I was given a T-shirt on family weekend that read "SAIC – Asking Loud Questions." The T-shirts were the work of a marketing

[86] Avrich.

company, but the spirit of the phrase was quite wonderful. The goal of education should be empowerment – not the conveyance of an official canon, but the teaching of critical thought, the ability and the permission to ask questions loudly. The key to effective education is the ability of learners to reflect on their internal constructs in light of new information they encounter in their environment. No process, says Piaget, could be more natural, and so what better way to improve the quality of education than by encouraging students to become reflective practitioners? My goal here is to assist in the creation of a technology and methodology that empowers children as critical thinkers and also to help teachers inspire this kind of thinking in their students. This vision is shared by the Grassroots Invention Group at the MIT Media Lab of which I am a part. The Grassroots Invention Group seeks to empower communities by developing, with their participation, the technologies and ideas necessary to become true participants in a global community rather than recipients of technological solutions delivered from on high.

The remainder of this thesis will be dedicated to a particular task: developing an environment intended to introduce young children, ages four to seven, to the basic ideas of computation. While we will not explicitly return to the history or political aims outline above, based on this groundwork it is hoped that the reader will continually make their own connections between history, education, computation and social change. As a parting thought: consider this work as a fragment of an alternative thought, one small foray into the history of education as it might have been, and with persistence,
still might be.

8 RULES FOR INFINITE GAMES WITH CHILDREN AND COMPUTATION

"There are at least two kinds of games. One could be finite, the other infinite. A finite game is played for the purpose of winning, an infinite game for the purpose of continuing the play."

"The rules of the finite game may not change; the rules of an infinite game must change."

"Finite players play within boundaries, infinite players play with boundaries."

"Finite players are serious; infinite games are playful."

"A finite player plays to be powerful; an infinite player plays with strength."

"A finite player consumes time; an infinite player generates time."

From *Finite and Infinite Games : A Vision of Life as Play and Possibility*, James Carse

ONE INFINITE GAME: CHILDREN AND COMPUTATION

Taking into account the historical role of computers as a support for quantitative analysis and the industrialized nature of contemporary education, this chapter seeks to define a set of rules that can be used to develop and analyze systems[87] intended to introduce children to the powerful ideas inherent in computation.

In particular, the goal here is to use computation in the service of personal expression in a way that is not demeaning to either concept. The wording and intention of these guidelines is quite specific, as it is intended for young children (ages four to seven) but with slight adjustment these guidelines could be generalized to apply to many systems that wish to support constructionist learning with computation.

WHY CHILDREN AND COMPUTATION?

The question I have not specifically addressed until now is this: Why children? And riding on its heels: Aren't computational concepts awfully advanced for four year olds? I would like to answer these questions in reverse order, and both with quotes from Seymour Papert.

With regards to the advanced complex nature of computational concepts I would like to offer Papert's discussion on the possibility of introducing differential calculus to elementary school aged children:

> *"Of course the idea is absurd if your image of differential mathematics is something like 'D of x to the n is n times x to the n minus 1.' But what is really fundamental to 'the calculus' is a cluster of ideas that could be conceptualized in context of computer presence and computer fluency in forms that will be not only accessible, but also truly useful to very young children."*[88]

[87] **Definition of System** In the above text, for lack of a better term, I use the word *system* as the noun for what I am describing. By system I mean the inseparable combination of philosophical approach (constructionism), physical/virtual instantiation (microworlds, software tools for reflection) and related objects and thoughts that define an approach to education in a social context. At various points the wording may indicate I am talking more about the properties of a tool or more about the ideas that surround its use. I will do my best to make explicit to which I am referring, but in any case this lack of clear definition reflects the opinion that one cannot deliver one without the other. Tools must be used with a philosophy in mind, and this philosophy is both a result and an influence of the creation and use of tools.

[88] Papert, *The Turtle's Long Slow Trip*, 13.

The goal of this thesis is not to introduce children to computation in a way that would involve memorizing great numbers of obscure commands or ten different sorting algorithms. Rather, the goal is first to consider what is both fundamental to and truly useful about computation, and then to design a means to convey these concepts without relying on complex encoding methods. Inevitably this will mean that the environment presented below will not be "as powerful" as a high level programming language in terms of its general purpose ability to create a wide range of applications. Nevertheless, it will demonstrate an alternate representation of computation that will "be not only accessible, but also truly useful to very young children."

By way of understanding why this is a good idea (in answer to the "why children" question), I offer the following quote from *Mindstorms*, in which Papert discusses the reason it might be worthwhile to teach children theoretical physics:

> *"Our discussion... suggested that theoretical physics may be a good carrier for an important kind of meta-knowledge. If so, this would have important consequences for our cultural view of its role in the lives of children. We might come to see it as a subject suitable for early acquisition not simply because it explicates the world of things but because it does so in a way that places children in better command of their own learning process."*[89]

Computers, by virtue of their ability to model entire worlds, allow us to completely rethink what it is possible to teach and to learn, where and when. There are certain classes of ideas (Wonderful, Powerful ideas) that are worthwhile to learn because of the way in which they, when internalized, affect the learner's perception of the world. In most cases, powerful ideas are best taught in particular contexts where they appear naturally "close to the surface," although once they are learned they may be applied, often to great effect, outside of this context.

In the case of computation, it is my assumption that while computers provide ideal raw material for introducing children to computation, the way that computers have been used traditionally to this end is far from ideal. This work builds on the small but growing alternate tradition that seeks a better way to introduce children to computation. The hope is that not only will children perhaps be able to apply this knowledge to traditional computational activities (such as structured programming and

[89] Papert, *Mindstorms*.

quantitative analysis), but they will also be armed with a new kind of formalized thought process, allowing for reflection and encouraging them to take command over their own learning process.

NO THRESHOLD, NO CEILING

The goal in creating an environment to introduce children to any complex concept should be to achieve "low threshold, high ceiling." What this means is that an environment should allow for meaningful access at a novice level, but not prevent users from continuing to work as they develop.

In his 2003 article *Symbolic Programming vs. Software Engineering -- Fun vs. Professionalism – Are These the Same Question?*[90] Brian Harvey presents two points which relate directly to the approach taken by this thesis. The first point is with regards to diversity of approach and is what Harvey calls "No threshold, No ceiling." In the Logo community, Harvey argues, there are so many versions of Logo that any limitation caused by one version is removed by the ability to choose another version more appropriate to task. The second point that Harvey makes is with regards to usability and the effort that has gone into making Logo less opaque. While a great deal of effort went into word choice for Logo, "…a new generation of computer users accustomed to point-and-click interfaces would question the user-friendliness of any Logo environment. I think we have to admit that Logo syntax, although better than Pascal, is still far from transparent. Can we develop more compelling notations without sacrificing the essence of programming? I guess this is what Boxer [another programming language] is about, in part."[91]

This is important to this thesis because 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 "develop a more compelling notation," and is mitigated by the fact that CTRL_SPACE is presented as part of a family of solutions. If a user hits the ceiling, they ought to discard CTRL_SPACE and continue with Logo (or other language of their choice).

WHY RULES?

It is important to emphasize a subtle but distinct difference between classes of rules. Rather than rigid top-down regulations, I would like to offer a few rules in the same spirit of the rules of a game. Like those established by children before playing tag, these types of rules are intended to define the edges of the domain in order to *allow* for play, and to extend play as long as possible. The rules

[90] Harvey, 7.
[91] Ibid., 7.

presented here are offered by me as a *participant*. Finally, these rules are merely to "get the ball rolling." They can, should and must be changed and adopted to fit the physical constraints of the playing field, the characteristics of the players of the game and of course the development of new technologies.

In his book *Finite and Infinite Games : A Vision of Life as Play and Possibility*[92] NYU religion and philosophy professor James Carse does a wonderful job of describing these classes of rules and what they mean not only for the game, but for the players of the game. At the beginning of this chapter I offered a few definitions from his book which I have used as my own set of guidelines to create this set of rules. From here on in I will refer to my suggested rule set as a set of *guidelines*, in order to avoid confusing them with the more rigid top-down class of regulatory rules.

GUIDELINE 0: Guidelines are in the service of the participants and subject to change by them
This is the one and only guideline I will state with regards to the rule set itself. This guideline means that any established rules must remain in service of the system. If the rules force the definition of the system itself in a way that diminishes the experience, the rules *must* be changed. In his essay *The Heresy of Zone Defense*[93] art and cultural critic Dave Hickey explains the way that the rules of basketball have always been modified in the service of the game. The rules, Hickey says, make possible plays that the creators of the rules never imagined possible, and every rule change made since the invention of the game has been explicitly made to enhance the players' and spectators' enjoyment:

> "…amazingly from [the time the game was invented] …until this, all subsequent legislative changes to the game have been made in the interest of aesthetics – to alter those rules that no longer liberate its players, that have begun to govern the game through tedium and inequity. And all these changes probably would have come to pass more rapidly had Naismith codified his most profound insight into the game he invented: *It does not require a coach."* [94]

Naismith may not have codified this for basketball, but I would like to do so here: even though I am offering a rule set, these rules have evolved from my experience with the spaces I will discuss. I believe it is useful to state definitions and defining characteristics, but these definitions will

[92] Carse.
[93] Published in *Air Guitar : Essays on Art & Democracy.*
[94] Hickey, 159.

undoubtedly change for me, and I do not wish to suggest this chapter as the only way to teach computation to children.

If a reader finds a situation where the application of these rules do not ring true (and they inevitably will), the rules may be discarded as they will not be relevant in that case. What I hope, however, is that the reader will instead extend and modify the ideas set forth here to fit the situations they encounter, and will then publicly share this re-definition. In this way the game play can only improve.

GUIDELINE 1: The use of computation should serve a clear purpose

Technology should never be used to justify an activity. Often, computers are used to give credibility to what is otherwise seen as a frivolous pursuit. Educational systems that have cut their art programs, for example, might allow children access to drawing or media manipulation tools on the grounds that the ability to operate these tools is a "useful skill." While this may be true, the emphasis on the industrial utility of image manipulation is demeaning to the user and to the process of expression.

The "wow factor" of new technology is often used to disguise otherwise insufficient material. The most egregious examples of this can be seen in the "edutainment software" that flooded the early personal computer market. Of questionable educational value, these programs purported to be effective merely because they made use of cutting edge technology.

On the flip side, the introduction of computation must enable access to those things that make computation useful. For example, a drawing program that duplicates as close as possible the workings of pen and paper is of questionable usefulness: why not simply use pen and paper? The answer to this should be that the drawing program enables more, or at least allows for a different approach to drawing than is possible with paper alone. If a system cannot make a strong claim for why computation is present, it is likely that computation is not necessary!

What we are striving for is an environment that empowers personal expression with the possibilities of computation. This requires first a respect for the expressive nature of the task at hand and, secondly, the ability to correctly match desirable outcome with computational concepts.

GUIDELINE 2: Users must be able to play with underlying rule set, not only its parameters

TUIs and GUIs allow us to abstract away the quantitative nature of computation and file down the rough edges of an otherwise difficult to use device. While this can be used to great effect, the designer must be careful not to remove all access to computation.

There is a fine line between true interactivity, where the user actually has some effect on the system, and the relegation of the user to the status of "cue issuer," where the only effect one has is on the pacing of pre-scripted events.

In an abstract sense, programming can be said to be the ability to manipulate a logical rule set for interaction, while the running of the program enables others to tweak the parameters. As any programmer knows, the act of writing a program consists of a great deal of tweaking, but the programmer always has the option to change the underlying logical assumptions that define the behavior of the computational object.

In an environment that seeks to introduce very young children to computational ideas, it is not optimal to support every logic structure known to computation, nor is it practical to introduce the kind of syntax necessary to construct elaborate systems from basic computational primitives. At the same time, we do not wish to provide so many prefabricated modules that the use of "computation" becomes the mere stringing along of objects with no understanding of what it going on.

As a final point, it is not likely that this type of understanding will come from an interaction between a child and the system alone. This is one part of the system that clearly depends on the presence of a facilitator.

GUIDELINE 3: Avoid excessive error correction

Software environments can serve as guides, prompting the user to behave in particular ways to reach a desired outcome. Software can also correct a user's input directly, providing what the system believes the user wants as opposed to what they are actually requesting.

On one level the ability to second guess a user's expectation is the primary goal of interface design. This is a powerful property of computers and has inspired an entire body of research into software agents that are capable of interpreting ambiguous human commands to return usable results.

On the other hand, the ability to debug is an extremely important part of the learning process, especially with regards to understanding computation. An environment for introducing children to computation should not be frustrating, but it also should not deprive the user of the debugging experience. If an environment makes it very difficult or impossible to make a mistake, there is little chance the user will even be unable to understand what they are doing correctly, as the software will be doing all the work. I am searching here for a subtle balance between providing reliable feedback and instruction.

GUIDELINE 4: Ambiguity is a good thing

Ambiguity is one of the most powerful features of human communication. Far from being problematic, it is ambiguity that allows for the existence of humor, poetry and efficient context sensitive communication. Imagine if, like a computer, human beings demanded an entire rule set and strict syntax for communication. What would you do if such a being was crossing the road in the path of a truck? Write and debug a program?

Only recently has it become possible to endow computers with the processing power capable of dealing with human ambiguity. By providing computers with a predefined context, the onus of interpretation is no longer on the user. This is particularly useful when dealing with young children, whose sense of context is strong and ability to abstractly describe location is far less developed than their ability to point and say “there!”

Access to computation has too long required the user to formalize their desires in unfamiliar ways. While formalizing a thought is a fundamental part of thinking computationally, if this is done unnecessarily or in a manner which appears nonsensical, it causes nothing but pure frustration. If a system can provide a clear context and is capable of understanding ambiguous commands, it will free the user to think more about what they want to do with computation and less about how to shape their thinking to match the computer’s preferred model of the world.

At the same time, it is recognized that excessive ambiguity can quickly spiral out of control. An environment that provides no structure, vocabulary or system of organization would be nearly impossible to debug after a certain point, as it would rely entirely on the child’s memory of their intention to explain a particular step in time. Accordingly, a system should seek to balance support for ambiguity with structure in such a way that minimizes unnecessary frustration at both ends of the process.

GUIDELINE 5: Difficult doesn't mean better

In choosing the concepts to introduce, one should not succumb to the belief that concepts that are difficult to understand are somehow more powerful than simpler concepts. This is decidedly not true, especially in computation. While recursion is a very powerful tool, it is not used nearly as frequently in text based programming as the simple **IF** statement.

In his book *Maeda @ Media*, Media Lab Instructor John Maeda writes about how a brush with the simplest of computational concepts profoundly changed his understanding of how one could use computation: "My math teacher in high school… suggested that I take his programming class. 'Nonsense" I though to myself, "I have two whole years of practical knowledge behind me." One day I finally went to one of his classes, where he introduced a construct called "FOR…NEXT" …when I went home and discovered that I could replace ten thousand lines of program statements with about fifty lines, I was dumbfounded."[95]

GUIDELINE 6: System should allow connection to the familiar

Any system which introduces a new concept should do so by way of a connection to an object or concept that is already familiar. The reason for this should be self explanatory: it is far easier to internalize new information if it builds off the old. In fact constructionism offers the view that all knowledge is constructed, and so if this is going to happen anyway, we might as well assist the process.

This seems an almost trivial point, but anyone remotely involved with computation knows that with a whole new set of concepts comes a whole new set of vocabulary and, in the case of programming, an entirely new set of languages and syntaxes.

Any community of practice, be it bakers, taxi drivers, politicians or computer hackers, develop their own language, a lot of which serves a particular purpose. But if our focus is to remain on computation as a concept, it makes little sense to speak to young children about "object oriented code" and "algorithm design." These words have no analog in a child's, or indeed many adult's worlds, although we may find the ideas they describe do have familiar parallels.

[95] Maeda, 18.

GUIDELINE 7: System should support growth

In guideline four I discussed ambiguity and mentioned briefly that when introducing computational concepts, it may be useful to limit the user's initial experience to a smaller and more manageable subset of ideas, rather than attempt to confront them with all of the power of computation at once.

While this is true, it is also true that failing to keep pace with a user will simply generate boredom. A system must be prepared to follow the development of the user in order to grant them whatever level of control they desire without overwhelming them when they don't need it. This type of design is known as "low threshold, high ceiling" and in terms of software and hardware environments, one of the best ways to accomplish it is to keep a system as open as possible. By allowing the user to decide which specific programming language, communications protocol or vocabulary for discussion is best for them, the system can support multiple levels of interaction, neither overwhelming the novice nor frustrating the advanced user.

GUIDELINE 8: System should encourage reflective public interaction

Any system of learning should encourage reflection, and one of the best ways to facilitate reflection is to facilitate discussion. The act of expressing an idea forces one to formulate it. This process forces one to pay close attention to detail and clarifies the meaning of a concept for the speaker as well as the listener. A kind of personal formalization occurs which solidifies the idea but is fundamentally different than either the external hard or soft formalizations discussed earlier. This is true for adults as well as children. In fact for very young children the act of describing the world in words constitutes a crucial phase of development. Any system for learning should support and encourage public interaction, where *public* may be defined as any group of two or more individuals.

GUIDELINE 9: System should encourage the creation of an artifact

One of the best ways to facilitate reflective public interaction is to focus the discussion around an artifact. An artifact allows for the externalization of an internal construct. An artifact encourages participation in a system by providing a clear goal to accomplish. An artifact also allows a group of individuals a central point on which to organize their discussion, encouraging the sharing of multiple viewpoints.

An *artifact* is traditionally defined as a physical or virtually physical object. While participating in the creation of performance, the performer is engaged in a medium of exploration, but I would like to suggest that when a performer formally presents a performance, the performance becomes an

artifact for discussion. In this sense I would also like to include dances, performances, storytelling and puppet shows in the definition of artifact.

COROLLARIES: THE ROLE OF THE FACILITATOR

Out of this set of guidelines should emerge an understanding of what it means to be a facilitator. It should be clear that there is a valid and worthwhile place for facilitation, but that the role is not one of instructor, expert consultant, lecturer or test grader. Rather a system of the kind that satisfies these guidelines requires an active participant whose prior experience gives them particular insight into what makes a better experience, and whose adaptability ensures the material covered remains contextually relevant to the learners.

This definition should also make clear that the current criteria for selecting a teacher (the one with the higher paper credentials) rings false. The criteria is not book knowledge, hours logged, status awarded by an institution. Rather, anyone at any time assumes the role of teacher by participating actively in a community of learners and becoming genuinely engaged in what is to be learned. In a very real sense, this kind of system seeks to flatten the hierarchy that characterizes industrialized teaching methods. The teacher in this case is free to make mistakes, because the process is as important as the product. This should sound remarkably similar to the kind of meritocratic approach that characterizes the hacker ethic described in previous chapters: teachers are judged by their teaching, which equates to participation, rather than by some other “bogus criteria.”

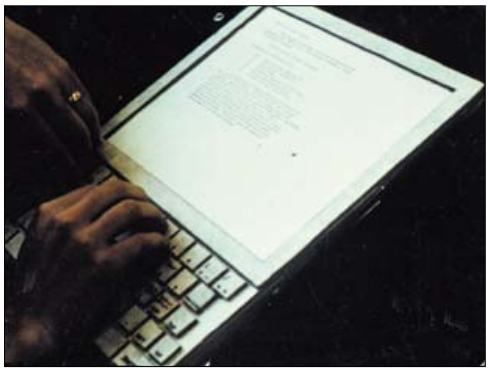
9 ON THE SHOULDERS OF GIANTS LAYING THE GROUNDWORK

A great deal of work has been done on and around the issues discussed in this thesis. A truly comprehensive examination of this work would fill a book, but I would like to briefly visit each of the projects which have served as inspiration or objects of contemplation in the development of CTRL_SPACE. In some cases, CTRL_SPACE can be seen as a continuation of the ideas that informed a particular project. In others, the work was undertaken for a completely different purpose, but offers an answer to a specific design problem I encountered. Taken together, these projects serve as the solid foundation, the giant's shoulders, on which CTRL_SPACE stands.

I. FIRST STEPS – FOUNDATIONAL WORK

Dynabook (1968)

Dynabook was an early conceptual prototype of a personal computation device that would be portable, wireless and allow for connectivity with other Dynabooks by virtue of access to one or more networks. This cardboard mockup (fig 9.1) resembles nothing so much as a laptop, but as it was constructed in the same year that Intel was founded and that Douglas Englebart gave his NLS demo, it represents a grand and futuristic vision. In 1968, mainframe computing still ruled the day, and the idea of such a small, intimate computing device was far from commonplace. In a sense, industry is still playing catch-up with Dynabook ideas: widespread wireless networking with ad-hoc support is only just becoming a reality.



▲**Figure 9.1** Dynabook Cardboard Prototype

This chapter begins with Dynabook because the existence of this thesis would not be possible without the pioneering work of the original GUI designers, of which Alan Kay is perhaps the most well known. Directly related to this thesis are Alan Kay's musings on the design of Dynabook:

> *"... we failed at designing for adults. And I remembered a wonderful phrase of Marshall McLuhan. He said, 'I don't know who discovered water, but it wasn't a fish.' The idea is if you are immersed in a context you can't even see it. So we decided to follow Seymour Papert's lead and instead of trying to design for adults we would try and see what this Dynabook of the future would be like for children and then maybe hope some of it would spill over into the adult world. So children were an absolutely critical factor here."* [96]

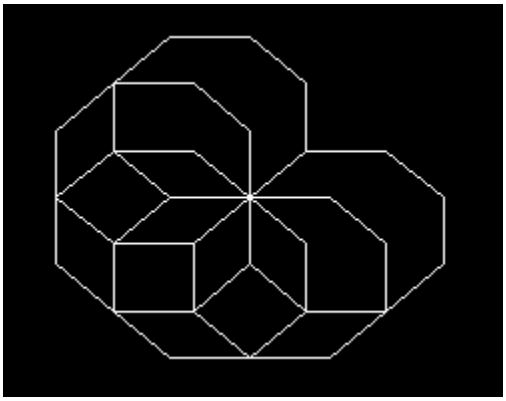
[96] Artmuseum.net.

The spirit of this story is the same as that which drives this thesis. For those of us who are involved daily with computation, it often seems that there is no fundamental problem. Immersed in a context, we see nothing wrong (or rather, we see everything wrong, but our observations remain narrowly focused. "I don't like this pointing device, programming language or software package, I prefer that one" rather than "Why do I need to use a pointing device at all?"). In section four I will discuss a few of the lessons learned when interacting with children and computation. The most striking feature about many of these lessons is that they are obvious in retrospect, but were not obvious before the interactions took place. Like others who work closely with computers, I often make assumptions about usage patterns and information conveyance that should not be made. Children's lack of familiarity with technology can highlight the problem areas that most of us take for granted.

Logo

The primary difference between the work presented in this thesis and Logo is formal, not philosophical. With CTRL_SPACE, I am attempting to create an environment in which the action of programming is entirely visual and physical, not textual at all.

The first point that should be made here falls into the category of "solutions commensurate with the problem domain." In no way do I intend to make the statement that visual programming is always the best solution. In fact this is *not* the case. Visual programming is only useful in very specific cases. Logo was intended by Papert to be used for mathematics and geometry. It makes no sense to completely eliminate textual representation in such a case. I would never suggest that CTRL_SPACE be used to teach or learn about geometry, although it is likely that one will find mathematical and geometric ideas at work. Similarly, it is not appropriate to advertise Logo as a drawing tool, although it is possible to draw with it.



▲**Figure 9.2** Logo

The second point I wish to make is more subtle and has to do with the way one chooses to use the options provided by a system. While building CTRL_SPACE in Director, I ran into the problem of how to save and record sensor data for later playback. Sequencing is done via a timeline and I wanted a

visual representation of sensor values, in short: a graph. I originally approached this problem by considering the graph as a display – a visual representation of an independent dataset. The solution I ultimately arrived at was to collapse the two, using the graph itself not only as a representation, but as the dataset itself. In this way, CTRL_SPACE uses pixel color of images to store and retrieve data.

When I explained to my advisor what I had done, he relayed to me an anecdote about Brian Silverman, one of the founders of LCSI. Brian has been known to solve problems with Logo using the drawing capabilities of the turtle, as well as physical placements of objects on screen. Logo supports this use of visual objects as more than representations, should the user decide to take advantage of it.

I also explained what I was doing to Ranjit Bhatnagar, a friend of mine who makes his living by coding in Director. His immediate response: "That's cheating!" Ranjit was joking, but *something* prompted this response, even in jest. Somehow my solution didn't "feel right." What was wrong? This is a complicated question that I believe has much to do with the same paradigm, discussed in section two, that privileges the abstract over the concrete.

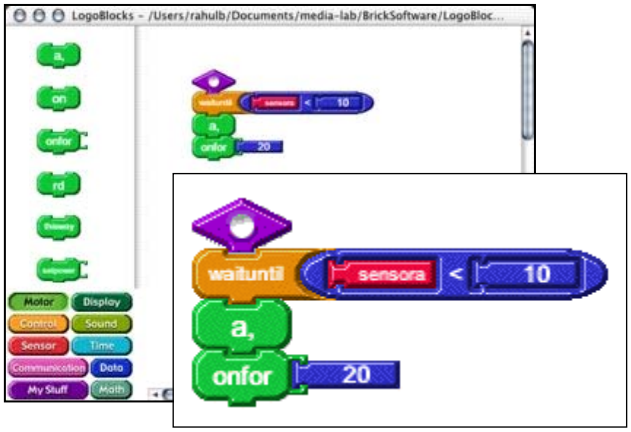
When I asked Ranjit later what he meant by "cheating" he gave the following answer: "um... cheating = a solution that's creative and successful but one that a more experienced practitioner probably would never have thought of." I appreciate this word as a way of describing what I am trying to accomplish. If the top-down approach is the "right" way to do things, I want to "cheat" (provided that we can demonstrate a method of cheating that avoids compromising the powerful ideas inherent in the domain).

II. AN ALTERNATE TRADITION: WHAT DOES IT MEAN TO PROGRAM?

LogoBlocks

LogoBlocks is a graphical programming language that was designed for use with the programmable brick (a Media Lab precursor to the Lego RCX or "yellow brick" microcontroller). 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, eliminating a blocking factor to programming.

This visual approach to programming is similar to that taken by the popular Microsoft Visual products (VisualBASIC, Visual C++, etc.) It is also similar to the notion of WYSIWYG (What You See Is What You Get) web page design, such as that supported by Macromedia Dreamweaver. The stance that this thesis takes on this approach is that it is a good start, but does not push the possibilities of visual programming far enough.



▲**Figure 9.3** LogoBlocks

By adding visual cues and by allowing spatial relationships to define computational relationships, LogoBlocks leverages our analog tendency to use vision and spatial association to categorize objects. This approach also leverages a kind of virtual physicality, by constructing a sense of physical "object-ness" around an otherwise ephemeral computational object. These are both extremely powerful concepts, but ultimately with LogoBlocks the user is still working with text. Why not eliminate the need for written language entirely? In the project proposal for LogoBlocks, author Andrew Begel examines some of the problems of adopting graphical programming languages, the foremost being the so-called Deutsch limit:

> *" The problem with visual programming is that you can't have more than 50 visual primitives on the screen at the same time. 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?"*[97]

The answer to this objection is simple: We aren't. Writing an operating system in a fully graphical programming environment might be a nifty hack, but it would not be straightforward, efficient or useful. The same can be said of introducing a four year old to assembly language. The purpose of this research 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 seamless transition to more "advanced" techniques when the time comes to write an OS, but also comes in recognizing that for many people and many cases, computation serves a particular task

[97] Begel, 10.

and is not needed for general purpose programming. I can think of no cases where it would be appropriate for a four year old to write an operating system. The same holds for many adults, although it is true that almost everyone can benefit from understanding the ideas involved.

Programming by Example: Repeat after me

In *Watch What I Do: Programming by Demonstration*[98] and *Your Wish is My Command*[99] Media Lab research scientist Henry Lieberman and others describe a method of programming by example. While the environments and the aims of the work are different from this thesis (in this work I am not interested in intelligent agents) the idea of imitation as a method of programming is shared. Imitation, especially with young children, is an excellent way to communicate information. Young children are very self-focused and are quite good at expressing what they want to do "like this." This characteristic is similar to what Seymour Papert leverages when he discusses *body syntonicity* and "playing turtle," although body syntonicity remains first person and "egocentric," while in this case I am asking the children to project themselves into another. In CTRL_SPACE, the use of the face robot (called ALF) as an analog to one's own face enables access to computational ideas in a familiar manner (more on this in chapter twelve). The act of imitation allows the child to teach ALF what to do, and the incorporation of sensors for input (more about this in chapter thirteen) allows us to literally program ALF by example.

▲**Figure 9.4** Dr. Legohead

Dr. Legohead and Topobo: Physical Programming Instantiated

Dr. Legohead, a product of Rick Borovoy's thesis *Genuine Object Oriented Programming*[100], resembles ALF by virtue of the fact that both are animatronic heads. The primary difference between this thesis and Dr. Legohead is the inclusion of an intermediate layer between the programmer and the programmable object.

Dr Legohead removed this "on-screen" layer in favor of attaching computation directly to physical objects. In this way, Dr. Legohead more closely resembles the recent Tangible Interface Topobo[101] project, a "constructive assembly system" which allows users to build

[98] Cypher.
[99] Ibid.
[100] Borovoy.
[101] Raffle.

an object and then program it physically (Topobo records and replays the physical motions made with it by the user).

Although I agree with all of the reasons Borovoy outlines regarding the importance of physicality, I maintain that some intermediate layer will prove useful in accessing certain concepts and will also ease transition to more abstract screen-based programming tasks should a child choose to do so.

▲**Figure 9.5** Topobo

The digital world can be used to model the analog, and theoretically (given enough time and space) the analog can produce the same results. But 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. My work is a logical next step – careful reintroduction of an abstract intermediate layer.

III. JALAPENOS AND FRENCH SAUCE? : EXPRESSION AND COMPUTATION

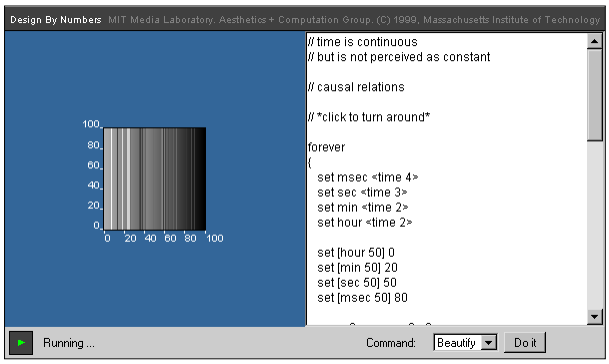
Design By Numbers

"A computer program is a utilitarian typographer's dream - a functioning machine composed completely of type"[102]

Design by Numbers is a project conceived by Media Lab faculty member John Maeda in the early 1990s. The impetus behind DBN is a powerful idea: What if one could access the code that underlies a drawing tool (MacPaint) and change the way the tools function? This is remarkably similar to the notion of access put forth by the hacker ethic and by constructionism. Maeda was also faced with the same concern that faces this thesis: that the tools that we currently use to access computation are simply too obscure, far from ideal for expression. Maeda writes that "...having seen java and c++ (languages that would easily discourage the most ardent of young futurists) take hold as the de facto

[102] Maeda, 118.

method for students to acquire computation skills, I chose to prescribe a minimal degree of knowledge in the ongoing Design By Numbers project."[103]



▲**Figure 9.6** Design by Numbers (DBN)

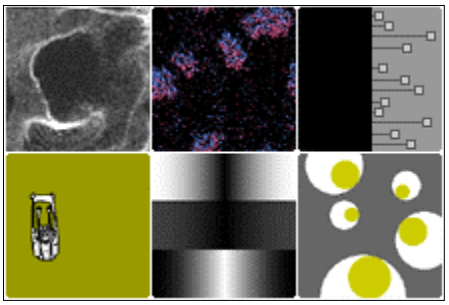
While the ideas regarding access to computation that drove the development of DBN are close to my work, the desire to "prescribe a minimal degree of knowledge" is not. While the point is certainly to demonstrate the usefulness of very simple structures, DBN's inability to support "use" very quickly becomes a limiting factor. In this sense, DBN is much like BASIC in the realm of programming: it flattens the learning curve but inspires patterns of use that become blocking factors as the user moves from "learner" to "user." This problem is considered acceptable because the canonical model of teaching tells us that learning and using are different activities, with learning consisting largely of participating in a simplified reality (practice for "real life"). In the DBN FAQ Maeda writes the following in response to a question about the small size of the drawing area frustrating a designer who wishes to use DBN: "DBN doesn't support high quality printing of this kind. The current version was written as a teaching tool rather than a tool for regular use."[104]

Instead of DBN, Maeda suggests using Proce55ing, "an advanced system for learning programming …which is a production-class environment for creating systems in JAVA with color, flexible screen-size, real-time three-dimensional graphics, and a host of features that approach the capabilities of Flash [a commercially available program for animating text and shapes]." While Proce55ing is a nice piece of software, it is interesting to note that we have come full circle. Java would "easily discourage" students, but it seems it is the only way to "use" computation for design, as anything simpler is suited only for those in the learning stage.

DBN is also problematic in that it allows one access code but prevents the use of "traditional" computer drawing tools (at least in the DBN workspace). While opening up the possibility of computational expression, this approach simultaneously limits the user to the aesthetic of code: large scale repetitions of multiples, sequential patterns, standard geometries and ultra clean dimensional projections. In this way the process of design is *not* augmented by the introduction of

[103] Maeda, 444.
[104] DBN FAQ available at: http://dbn.media.mit.edu/faq.html.

computation; it just appears formally different by virtue of the signature of the tool. Programs such as *Jet*, a vector graphics program that allows one to leverage Logo, provide better examples of the power of combining drawing with computation to blur the boundaries between the two.



▲**Figure 9.7** Proce55ing

None of this is surprising if one considers John Maeda as a designer working within the modernist tradition. Certainly defining his own trademark aesthetic, Maeda nonetheless remains strongly influenced by American modernist Paul Rand (1914- 1966), a staunch individualist whose formal style owes much to the European "avant-garde" design he studied at Pratt and Parsons in the 1930s. While Rand's work is uniquely identifiable, its formal vocabulary has been correctly described as Bauhaus.[105] Maeda writes that it was an encounter with Rand's work that "immediately inspired [me] to pursue the field of graphic design."[106] With regards to the focus on the aesthetic over the process, the answer can again be found in Rand as quoted in the opening pages of *Maeda @ Media*: "Art is primarily question of form, not of content."[107] Maeda was instrumental in bringing Rand to speak at the Media Lab and in arranging for him to receive an appointment to the lab's faculty. Rand accepted the position days before he passed away.

This raises an inevitable question with regards to CTRL_SPACE: By eliminating the ability to use text-based programming, aren't I encouraging the same problem? The answer is yes if one considers CTRL_SPACE in a vacuum. I present CTRL_SPACE as one of a family of approaches. While an effort has been made to make CTRL_SPACE as complete as possible, it is recognized that one cannot deliberately introduce limitations and expect the system to do everything. The question then becomes: Is the raw material provided by the microworld *enough* ? I believe this is the case with CTRL_SPACE, and by virtue of computational modularity the user may decide if and when they wish to transcend the boundaries by replacing the limiting component. Much effort has gone into ensuring that this is possible with ALF and CTRL_SPACE, in a way that minimally disrupts other elements of the system.

[105] WGBH Archives.
[106] Maeda, 306.
[107] Ibid.

Full-Contact Poetry

Full-Contact Poetry is an environment, designed and built in Squeak by Anindita Basu of the Media Lab's Future of Learning Group. Full-Contact Poetry is "...a software environment in which children can express their poetic thoughts, create their interpretations of writing by others and also share these expressions. The environment combines ideas from literary theory and analysis with constructionism to extend tools for poetic expression. Children can experience poetry by playing with words as objects, experimenting with typographic effects, moving words through space and navigating into and through the text, while also being able to incorporate and reconfigure sound and image."[108]

▲**Figure 9.8** Full-Contact Poetry

In relation to CTRL_SPACE, Full-Contact Poetry is significant in the way that computation and expression are given equal footing. This project is not a justification or modernization of poetry by virtue of the addition of high tech. Neither is it a softening of "hard" computer concepts by the addition of that which is more "soft." The project does not seek to compromise either, but blends the two to the advantage of both.

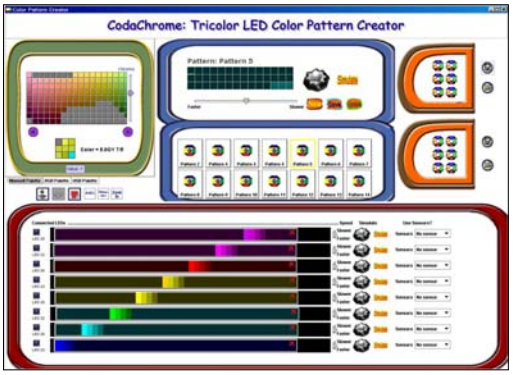
Key to accomplishing this is the lack of lofty claims. Full-Contact poetry does not seek to revolutionize computation or poetry, it simply (and powerfully) offers a new method of thinking through a domain. My thesis can be said to be the same in spirit. The work here does not purport to completely revolutionize animatronics, computation or puppetry, but rather to combine them in a way that is unique. In the process, Full-Contact Poetry and CTRL_SPACE both become objects to think with that open up doors to a host of powerful ideas.

CodaChrome

CodaChrome[109] is a software and hardware system designed for the thesis work of MIT Media Lab Grassroots Invention Group member Margarita Dekoli. The purpose of CodaChrome is to allow children to create objects which contain light modules capable of displaying millions of colors. In particular, this project is interesting in the consideration it has given to the representation of color through time. Much work has gone into developing a model which allows one to "program light" via a timeline of color choices.

[108] Basu.
[109] Dekoli.

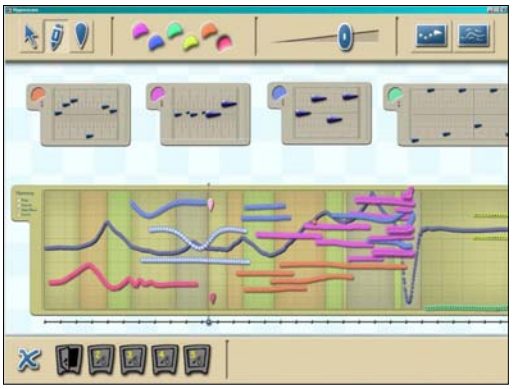
My work on CTRL_SPACE is very similar in that I am striving to develop the best way to represent an abstract concept (in this case motion rather than color) through time in order to enable children to easily manipulate an object in space. My decision to use a timeline in CTRL_SPACE owes a debt of gratitude to Margarita, for her generosity of time in discussing these ideas and also for allowing me to participate in her workshops with children.



▲**Figure 9.9** Codachrome

Hyperscore

Building on a tradition of alternate representations of music in software that begins with Metasynth and Simtunes, comes Hyperscore, a Media Lab project initiated by Mary Farbood[110]. Hyperscore provides an interface to powerful musical ideas and does so without using text. The success of Hyperscore serves as a "real world" example of non-text based music composition for children. Furthermore, the fact that Hyperscore can be used to great effect by adults as well as children is not an accident, but part of a deliberate design that does not consider "learning" to be a dumbed-down version of "use." Hyperscore, as Logo, is not "real work made simple" but is instead a complete alternate representation of a particular medium.



▲**Figure 9.10** Hyperscore

Perhaps the biggest drawback to Hyperscore is it's corrective property. Hyperscore is enormous fun to use, but it is nearly impossible to create a composition that sounds "bad." As a result, in order to truly understand what one is doing musically, Hyperscore requires the presence of an accomplished musician. Without narration, it is not clear what one is doing. It is critical in CTRL_SPACE that the children not only use computational ideas, but develop an understanding of them. I could make the claim that anyone manipulating ALF is playing with computation, and that would be true, but unless one understands the object-oriented nature of face making, the powerful idea is lost.

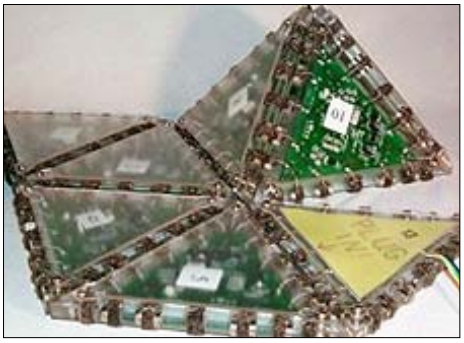
[110] Farbood.

IV. OTHER INFLUENCES

I have outlined the most significant influences to my work above. In addition, there have been a number of projects which embody one or more ideas that are also at work in the CTRL_SPACE environment.

The *Intel Playcam* software, a nonlinear editing suite for children distributed with the (now unfortunately out of production) Intel Playcam, served as an early visual and conceptual reference. The Playcam software provides fairly sophisticated access to a complex task (video editing) using minimal text and multiple screen modes.

The ideas best expressed by *Curlybot*[111] and *Triangles*[112], both examples of computational objects that are programmed physically, influenced the addition of CTRL_ARM to the project. Curlybot is a toy robot which will "play back" a motion by imitation. This project is significant for its focus on young children and the idea that introducing an intuitive idea of a complex concept is a worthwhile activity. Triangles is significant for the way in which it represents a complicated concept (topographical representation of information elements) with a simple and easy to manipulate physical object, and for its insistence that specificity of task is an important factor in increasing the usability of an interface.



▲**Figures 9.11, 9.12, 9.13**
Playcam Software, Curlybot, Triangles

EGGG (The Extensible Graphical Game Generator) is a system allowing for game creation by meta-programming. Although text based, EGGG leverages ambiguity by creating a strong context around the commands a user might provide. The EGGG tagline "Programming is hard. But programming for a particular domain need not be" is just as applicable to CTRL_SPACE as it is to game generation.

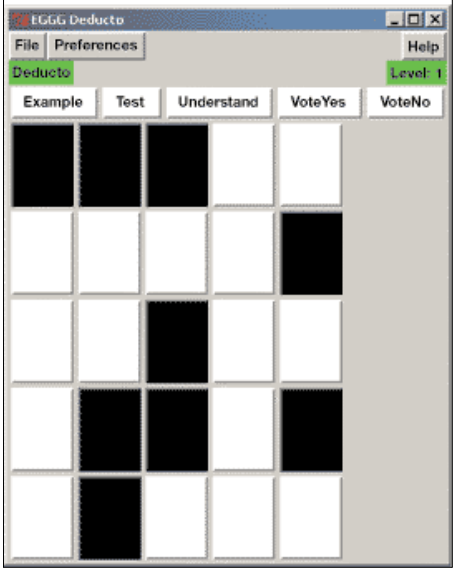
---

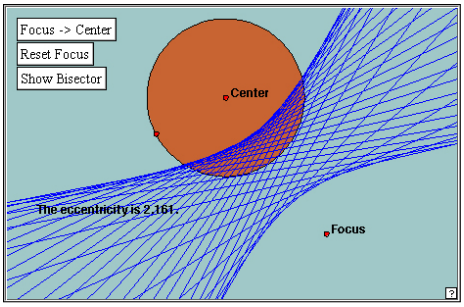
111 Frei.
112 Gorbet.

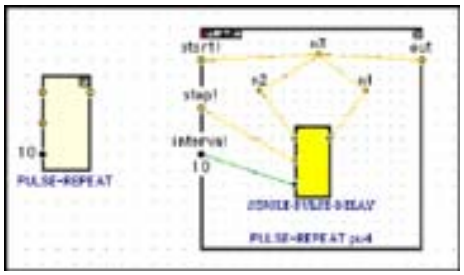
Programming Continuum, an unpublished project by the Media Lab Future of Learning group pulls a number of these ideas together by creating an environment that provides three levels of interchangeable representation of the same computational object. Children program a robot by moving it physically, but may also edit the graphical representation or raw Logo code on screen.

Geometers Sketchpad and the aforementioned *Jet* both blur the line between code and representation. Finally, the wonderful ability of Chris Hancock's *FLOGO*[113] to represent variables as their actual values in real time (of sensor inputs, for example) provides an inspirational example of a kind of inverted augmented reality. Augmented reality usually refers to the augmentation of the analog world with the power of the digital. In the case of FLOGO , this relationship is inverted by connecting abstract digital variables to time and space, endowing them with concrete meaning.

Figure 2 Deducto, a new game created with EGGG







▲Figures 9.14, 9.15, 9.16
EGGG, Geometers Sketchpad, Flogo

[113] Hancock.

Section IV Deep Dive: CTRL_SPACE and ALF

PRELUDE TO SECTION IV

In a sense, the previous three sections of this document have presented the conclusion of my research. The process began with the creation of a prototype environment, followed by a series of workshops that led to the development of the guidelines presented in chapter eight. After the guidelines were established, the problem was revisited and resulted in CTRL_SPACE, presented in the final two chapters. For the remainder of this document I will present my work in chronological order.

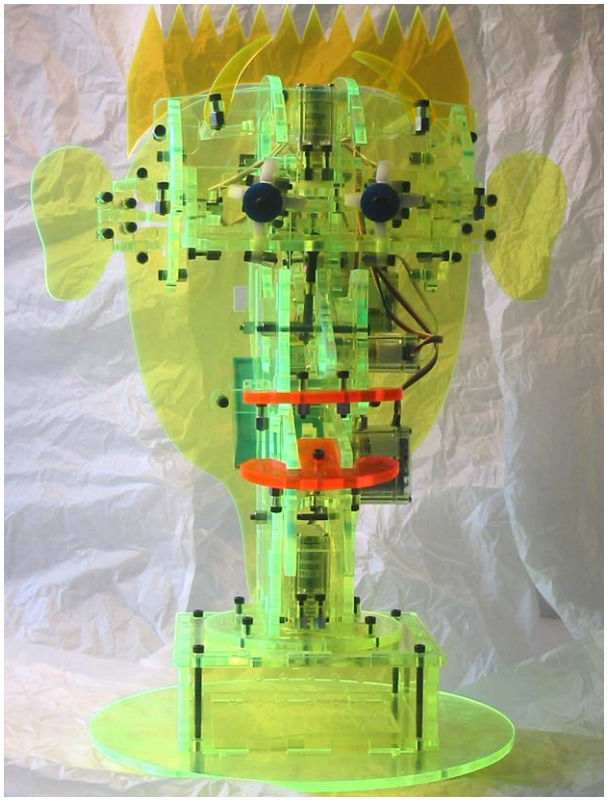
For this thesis I designed and built a software environment for introducing children ages 4-7 to computational concepts. I chose to build my environment around the Tower modular computing system and ALF: Acrylic Life Form, an animatronic head designed and built by Chris Lyon. Chapter ten is a description of what hardware was used and why. Chapter eleven describes my research methodology. Chapter twelve describes a series of workshops run with the ALF hardware, children and early prototypes of the software. Chapter thirteen presents CTRL_SPACE, a multi purpose environment for controlling animatronic objects that was the culmination of this research.

10 **ANIMATRONICS** AND COMPUTATION

INTRODUCTION TO ALF: ACRYLIC LIFE FORM

ALF: Acrylic Life Form is an animatronic head, designed and built by Chris Lyon, a member of the Media Lab's Grassroots Invention Group. ALF has six features operated by servo motors that are controlled by the Tower modular computer system.[114] By virtue of the flexibility of the Tower architecture, ALF may be addressed by software running on any number of devices (other Towers, desktop computers, palm devices) and can speak a number of standard communication protocols (serial, I2C, IrdA).

While ALF is a head-only robot with a limited number of movable parts, it has proven more than sufficient to address the issues at hand in the context of this thesis. Why this is the case will be discussed in depth below. For now, it is important to note that ALF is presented here not as the ideal object, but as a representative of a class of objects. The ideas presented in this thesis have broader application. Much can be gained by using other animatronic objects and ultimately, the CTRL_SPACE software presented later is intended to be multi-purpose.



▲**Figure 10.1** ALF: Acrylic Life Form

ALF's control structure is modular enough to 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. The task of building a controllable object opens up an entirely different and extremely interesting set of ideas concerning engineering, materials science, physics, electronics and control feedback which are unfortunately beyond the scope of this document. Before I present the software which was designed for this thesis, I would like to briefly discuss some of the reasons that the Tower and ALF were used in this project.

[114] Lyon.

WHY ANIMATRONICS?

The stated goal of this thesis is to introduce young children to computational concepts. Given this, what is the significance of a face making robot? 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 (in this case six) and easily sequenced to create actions. It is not difficult to discuss 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 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[115] and Tronick[116] highlight the specific importance of face making to early development. By the age of four, children are fully capable of understanding how to control their own faces and are intimately interested with the notion of representation on the face (i.e. what indicates sad, happy, angry). Therefore, the face provides us with an example 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.

In terms of the number of parameters provided (six in the case of ALF) nothing is significant about the specific number, but the order of magnitude is important. Technically, it is possible to control a nearly infinite number of motors by combining controller modules, each of which can support more than a hundred motors. The choice was made to limit the system to the number of parameters because of what I wanted to accomplish in the programming environment (the elimination of as much syntax as possible, the introduction of ambiguity) and also because it was important to be able to represent all states onscreen visually at once without overwhelming the user.

The limitations of the ALF / CTRL_SPACE combination are therefore deliberate design choices. It is understood that there are many cases where these limitations might become frustrations. Although

[115] Stern, *The First Relationship.*
[116] Tronick, *Maternal Depression and Infant Disturbance.*

this thesis does not present the use of more complicated objects, it was the intention to allow for this that led to the decision to use the Tower platform. Finally, animatronics are ideal for storytelling and puppetry.

STORYTELLING AND PUPPETRY

A great deal of research has been done regarding the powerful ideas surrounding children and storytelling. Of particular interest here is the work that discusses the construction of identity and self by way of avatars and/or storytelling technologies, In particular that of Justine Cassell (formerly the head of the MIT Media Lab's Gesture and Narrative Language Group)[117], Cati Vaucelle[118], Alison Druin (whose work focuses on robotic storytelling technologies)[119], Marina Umaschi Bers (whose work deals with children's construction of self in a virtual environment)[120] and Sherry Turkle (whose work on the construction and projection of self in digital space is well known).[121] My work seeks to leverage this research by taking advantage of the properties identified by the researchers listed above of storytelling and storytelling objects that allow for rich interaction.

In the course of working on another project, I had an opportunity to talk informally about my work with a counselor who works with troubled children at a local school. "This is great," he said "It's about control. I work with troubled children, the problem is often about control." I didn't build this system with counseling in mind although I am pleased if it ever proves useful for this. What I did have in mind was control: the ability of the child to act as puppeteer and to project parts of their self into animatronic objects which may then become actors in a story.

In terms of access to computational ideas, while it is the face robot that allows for object orientedness, it is the nature of storytelling which allows for the children to establish a rule set. The incorporation of multiple ALFs allow for multiple characters (or multiple representations of self). 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 discuss logic structures and conditionals.

The next two chapters will focus on workshops run with early prototypes of the software environment. The prototypes were limited and do not augment the true power of storytelling or puppetry, although

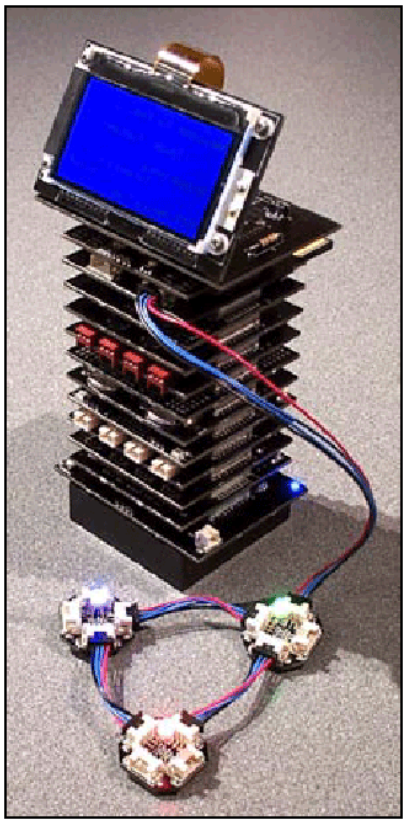
---

[117] Ryokai, Vaucelle and Cassell, *Virtual Peers as Partners in Storytelling and Literacy Learning*.
[118] Ryokai, Vaucelle, and Cassell, *Literacy Learning by Storytelling with a Virtual Peer*.
[119] Druin.
[120] Bers.
[121] Turkle, *Life on the Screen.*

they were a necessary part of the process. We will return to the possibilities of multiple ALFs, sensors and the manipulation of rule sets later, in the chapters on CTRL_SPACE and usage scenarios.

PHYSICAL MODULARITY

I have already mentioned the possibility of creating an entirely different animatronic object for use with CTRL_SPACE. In addition to this, there is in intermediate step that may be taken by significantly altering ALF itself. Chris designed ALF to be fully physically modular. All of his parts were cut on a laser cutter and assembled by hand with nuts and bolts that can be manipulated without even the need of a screwdriver. ALF is intended to be taken apart and modified by children (likely older than my target group), and the fabrication of new ALF parts can be seen as a way to introduce children to the use of machine tools and personal fabrication equipment. This introduction becomes possible through the efforts of groups such as the Learning Independence Network[122] and the use of tools such as the *Jet*, a logo enabled vector graphics program that was developed by the Grassroots Invention Group and Andrew Begel.



▲**Figure 10.2** Tower Modular Computing System

COMPUTATIONAL MODULARITY

The ability of the Tower to incorporate any number of layers, including those built by an end user, means that it is possible to include a wide range of sensing capabilities into the object being controlled and to add additional motors or other output devices. ALF is an animatronic head, but granted sufficient time and resources, it would not be technically difficult to create a full animatronic body, mythological creature or stage control system.

For this research I chose to connect ALF to a standard desktop computer via a serial cable. The software environment I constructed was built in Macromedia Director and operated by sending commands to a wrapper program running on the Tower. The ability of the Tower to receive communication over almost any protocol means that the control environment is extremely flexible. I chose to use Macromedia Director simply because my familiarity with the product enabled me to quickly and efficiently make changes to the interface. Director is by no means an ideal environment for everyone, but the nature

[122] More information available at: http://gig.media.mit.edu/projects/lin/.

of the Tower means that any environment capable of using the serial port can be used to control ALF without needing to make any modifications to either ALF or the Tower. Furthermore, the control environment can be bypassed entirely. Using the Tower in interactive mode, a user may operate ALF by creating and downloading Logo or even C or assembly code to the Tower foundation. Finally, advanced users may bypass the serial control software entirely, programming their own controller for ALF via IR, RF or even Ethernet.

11 METHODOLOGY WORKING WITH CHILDREN AND COMPUTATION

ORIGINS

This project began when I was asked to create a kiosk application for ALF, to be used as at the Science Museum of Minnesota. While the environment was being built, the issues that arose in the design stage began to raise questions, the most interesting being: could this be a good way to introduce young children to computation? The object itself certainly seemed ideal – visually interesting and fun to manipulate. Face making is an activity that is both enjoyable and developmentally significant for young children. The anthropomorphic nature of ALF makes it easy to create projections of self, and the storytelling nature of puppetry seemed to provide a metaphor for many computational concepts.

THE PROTOTYPE ENVIRONMENT: ALF SOFTWARE VERSION ONE

The functional portion of this environment consists of two screens, allowing the child to define various states of the ALF face and then to sequence them. Screenshots of this environment appear in chapter twelve.

The first screen, labeled *Make Faces* allows the child to adjust the six features of ALF's face by controlling onscreen sliders. The sliders are moved to position and then released, at which point ALF moves the particular feature to the indicated position. At any point, the user may save the current state of the servos by clicking on *save face.* The intent is to create a state, or face, and then label it for later use.

The second screen, labeled *program* allows the user to sequence the faces that were defined in the Make Faces screen by adding them to a list. Saved faces appear on this screen as a series of buttons, along with a small number of built in functions and special commands (such as *wait*). Users add steps to the program by clicking on the button. With each click, ALF performs the selected action and the step is added to the list at the right. The list is not random access – items appear on it in the order the user clicks and the order cannot be changed. At any point the user may click on *Run* which will cause ALF to set all motors to center position and then proceed through the list of actions.

This chapter describes the workshops I ran with this and one other prototype version of the software environment. As the reader will see, these early versions of the software fall short of the possibilities I have described as inherent in this domain. Nevertheless, without these workshops I never would have arrived at the guidelines outlined in chapter eight, nor would I have been able to develop with confidence the system described in chapter thirteen. That system, CTRL_SPACE, is a far more

complete environment that takes advantage of many more computational ideas than the system described in this chapter. We begin with the prototype software developed for the Science Museum.

After the development of this software, I ran a preliminary workshop with a four year old to see if some of these assumptions held true. The results of the first workshop were promising, but immediately revealed problems with the software design (specifics described below). Armed with this knowledge, I revised the software and ran another workshop. This process continued for a total of four workshops and two major software revisions, resulting in a third complete rebuild of the software called CTRL_SPACE and presented in chapter thirteen. Before we turn to the workshops themselves and the lessons learned from them, I would like to make a few points regarding the method of investigation.

ON THE USE OF COMPUTATIONAL VOCABULARY

In the previous chapter I made extensive use of terms that are traditionally used to describe computational concepts. *Containership, parameters, objects*, and even *programming* are good examples of this.

While this has been done to communicate with the reader, who is likely more familiar with these concepts than the average four year old, these terms were not frequently used by me when interacting with children, CTRL_SPACE and ALF. In fact these terms were avoided when possible and avoided entirely by the final version of the software, which eliminated text entirely.

The purpose of using particular terms that describe complicated concepts is to avoid the need to explain a concept every time we need to use it. This ability to represent “many things” with “one thing” is an extremely useful aspect of human communication (and indeed of computation). When the point of the process is to teach the described concepts, however, premature introduction of terms becomes highly problematic. Providing students with a list of terminology deprives the facilitator of a clear method of evaluating their own and their students’ progress. Any student who has ever had to take a written exam knows that one doesn’t need to actually understand material if one can parrot enough vocabulary words in proper sequence. This state of affairs does a grave injustice to the material, the student, and the facilitator, who is left with a false sense of security. After all, the students certainly *sound* like they know what they’re doing.

By initially eliminating the level of abstraction that is afforded by the use of such words, students and teachers must resort to actually conveying the concept they are discussing in familiar terms. This action solidifies meaning for all. This approach also provides a mechanism for evaluation. By paying careful attention to the words and analogies that students use to communicate to one another what they are doing, the facilitator is able to pinpoint confusions in understanding. In many cases this activity can turn into a pleasant surprise for the facilitator, who may find what they thought were familiar concepts described in a way that is both technically correct and completely unexpected. Vocabulary words can be introduced later, when their meanings can be attached to concepts which by then are familiar.

This approach finds support in the work of Piaget, as described by Eleanor Duckworth in *The Language and Thought of Piaget.* In a section entitled *Logic is Deeper Than Language*, Duckworth writes: "[Since] relationships are all built into language, one might think that children have only to pay attention when they are used in order to understand the relationships. But this is equivalent to assuming that, when a child hears 'doggie' as she looks out the window, she knows exactly what 'doggie' refers to. On the contrary, the evidence indicates that …children think their own thoughts and interpret the language in their own way, which does not necessarily correspond to what the speakers around them have in mind." [123] Earlier in the article, Duckworth discusses the fact that small children in particular tend to interpret meaning as that which matches whatever is occupying their attention at the moment. A young child's explanations are: "…full of 'You See?' and 'And then it comes here, see?' …But he talks as if what is obvious to him as he is now engaged in his explanation is – by that very fact - obvious to everyone else."[124] We will see this happen in some of the exchanges relayed below.

A final word from Duckworth of particular relevance to computation before we move on:
"The pedagogical implications here seem to be fairly clear-cut: Teaching linguistic formulas is not likely to lead to clear logical thinking; it is by thinking that people get better at thinking. If the logic is there, a person will be able to find words adequate to represent it. If it is not there, having the words will not help."[125]

[123] Duckworth, 23.
[124] Ibid.,22.
[125] Ibid., 25.

RESEARCH METHODOLOGY

A number of sessions were conducted during which I introduced children to ALF and a software control environment. I refer to these sessions as workshops. Although for many this word may imply a kind of structured activity, this was deliberately not the case. The approach taken does not resemble experimentation in the mode of scientific method. The purpose of this research was not to establish a list of conclusions but to develop a prototype of an environment to inspire further work. The purpose of working with children was not to conduct a user study but to engage in a participatory design experience. The workshops were not tests run after the fact but the driving force for the development of this project. This should explain the "informality" of the approach and also the relatively small number of children engaged in these workshops. The desire was not to develop a "sample population" but to make sure the prototypes and the ideas worked "in real life" early on, in the development stage, when changes are both more practical and more likely to take place.

It was important to me that the children not feel as if they were being tested. Besides the unnecessary stress placed on them by this possibility, there is a tendency among children of all ages to please authority. If I were to conduct these sessions as a "focus group," following up with an interview and holding them to a specific task, I very likely would have found resounding support for every point I wish to make in this thesis. I also would not, in good conscience, have been able to vouch for the veracity of such data. Anything resembling a test or a "school" situation was out of the question.

Instead, I presented the children with ALF in an informal setting and allowed them to determine the direction the session went. For the first ten minutes or so of each session, I gave no introduction whatsoever, allowing the children to experiment with the controls, push buttons and ask any question that came to mind (including a vast number of which had absolutely nothing to do with ALF, the software, my research, computers or computation). Allowing the children to continue exploring on their own, I observed what was going on. When a particular activity seemed to me to imply an understanding of a concept I would ask questions to try and determine (without leading) what the child was thinking at the time. If the child seemed frustrated, I would offer them help, trying as much as possible to ask them what they were trying to accomplish and what they thought "broke" rather than assuming anything at all. In some cases, when I felt the child had a fair grasp of the system and was ready for a challenge, I would ask them to complete a task designed to help me understand if they understood a particular idea. If a child "failed" in a task or stumbled at any point, I worked from

an assumption that these points were weaknesses in the system, and it was this that guided my design process for the next revision.

Knowing which questions to ask when in order to gain maximum insight into a child's thought process is much closer to an art than a science. I do not claim to have mastered this and can only offer that a great deal of patience is required, as well as a lot of trial and error. In general, this process is assisted greatly by the fact that for young children in particular, internal monologue is often externalized and one needs only pay attention to what the children are saying as they work. Recording the session for later review also helps a great deal.

A final note on process: In some cases, especially the final workshop session, I directly engaged the children in my task by explaining to them that I was making a system for other children to use and wanted their help to figure out what was wrong or what they wanted to see happen that they didn't see now. This line of questioning led to a great number of fantastic and often physically impossible suggestions, as well as raising quite a few points well worth considering.

WRITTEN NOTES

Alex : Feb 12th / 01:14
I'm really good. I'm really good. I'm really good at reading.

I have a tendency to take notes in situations where information is flowing very quickly and I have a feeling I might miss something. For the workshops, this seemed like a wonderful idea given the amount of raw data generated by a four year old in intellectual motion and also given the number of changes I expected to be making to the software interface. My first workshop, conducted as a one-on-one session with a single 4.5 year old boy, put this notion to rest almost immediately. After jotting a quick note to myself along the lines of "does not understand what function X is for" I put the notepad down and leaned towards the monitor to point out something. The child reached around my shoulder, slid the paper over and read what I'd written. "What don't I understand?" he asked.

Momentarily speechless, I managed a very poor explanation of what I was trying to do, hoping he would simply disregard the event and return to the "real" task. This was not the case, and for the remainder of the session I was repeatedly asked to "write this down. You should do this." After this session, I found that a quick free write following each workshop was most helpful. Also helpful in this

process was a "debriefing" session to discuss how the workshop went and what I thought should happen next. I am especially grateful to my advisor, Bakhtiar Mikhak, for assisting this process.

RECORDING

Whenever practically possible, I recorded the workshop sessions using a digital recorder and/or a video recorder. Recordings were enormously helpful as memory aids and also as a record of what I was not aware of either because my attention was on something else or, in workshops with more than one child, because I was physically outside of the workshop room, keeping an eye on an escaped research subject. One note that may be worth mentioning for future research: audio recordings served far better in this regard than video recording. Cameras have a tendency to focus attention and signify that an event is taking place. Especially with older children, the presence of a conspicuous camera may evoke either a spontaneous performance or performance anxiety. Of course no attempt was or should be made to record anything the children did or said without full disclosure to the children and their parents that recording was taking place (and no attempt was made to hide the recording equipment). Nevertheless, the children remained largely oblivious to microphones and audio recording devices, which also did not require clear line of sight in the way that video recording often does. Finally, I found that stereo microphones were far better than mono when months later I reviewed the recordings to try and determine which child said what. The sense of direction offered by the stereo recording plus my memory of who sat where helped in this task.

12 WORKSHOPS AND LESSONS LEARNED

The software prototype described in this chapter falls short of the ideal environment described by the guidelines offered in chapter eight. This is largely because the guidelines owe their existence to the lessons learned in workshops as described below. As realized in chapter thirteen, CTRL_SPACE pushes these rules much further.

FEB 12th: WORKSHOP WITH ALEX

With the first version of the software, I ran a preliminary one hour workshop with Alex[126], a 4.5 year old boy, to test my initial assumption that ALF might be useful for accessing computational concepts. Alex had seen ALF before, but had never worked with the software I created. What follows is a roughly chronological annotation of the transcript of this workshop. Transcribed text will appear in a different font, with my words in italic. Timestamps are in minutes and seconds and refer to the time the words appear on the audio recording.

This was perhaps the most structured of all the workshops. This happened both because it was a one-on-one session, and because Alex was very engaged and occasionally asked me what he should do next. Like most children, he was quite adept at problem solving, but what made it particularly fun to work with him was that Alex is very verbal and usually kept a clear and coherent running monologue of what he was thinking as he experimented with various options. This makes his sessions the most illustrative of any of the workshop transcripts.

TWO SECOND WAIT

In the following excerpt, Alex notices the *wait* functions in the programming mode. I have provided a quarter, half and full second options. I am hoping that Alex will understand that he can execute these no-op commands multiple times to achieve longer wait times. I ask him outright.

Feb 12th / 07:32

How do you think you would do that if you wanted to have more than one second?

```
      I don't know…
      Well, like… click it again? Like click, like click it again?
```

Try it and see

```
      Okay, lets see… program… okay let's see "Silly".
            *click*
      Okay. Wait one second.
            *click* *click*
      Mmm hmm! I think so!
```

So what will this do? Tell me before you click run.

[126] Names have been changed as my workshop participants were minors.

What??

What do you think this is going to do? right now?

Those two?

Uh huh

Uh I think it's going to do what I want it… what I think it would do. I think…

Which is?

I think it would wait two seconds.

OK

Let's see.
Okay I'm just going to do this. Run.

One… one two… Yup! Cool!

With very little prompting from me, Alex understands a basic notion of containership - that multiple instances of the same object can be created, and that they will exhibit the same properties as the original primitive. Two one second waits in succession is the same as one two second wait.

WIGGLE METHODS

In addition to the wait functions, I have created two "programs" (laugh and wiggle eyebrows) which appear as primitives in the same area as the wait functions. I am encouraged by Alex's success in understanding the wait function and decide to see if he can figure out how to build the "wiggle eyebrows" function by himself. This will involve creating two states, "eyebrows up" and "eyebrows down," and sequencing them in the program mode, perhaps with a wait between each state.

Feb 12th / 09:04

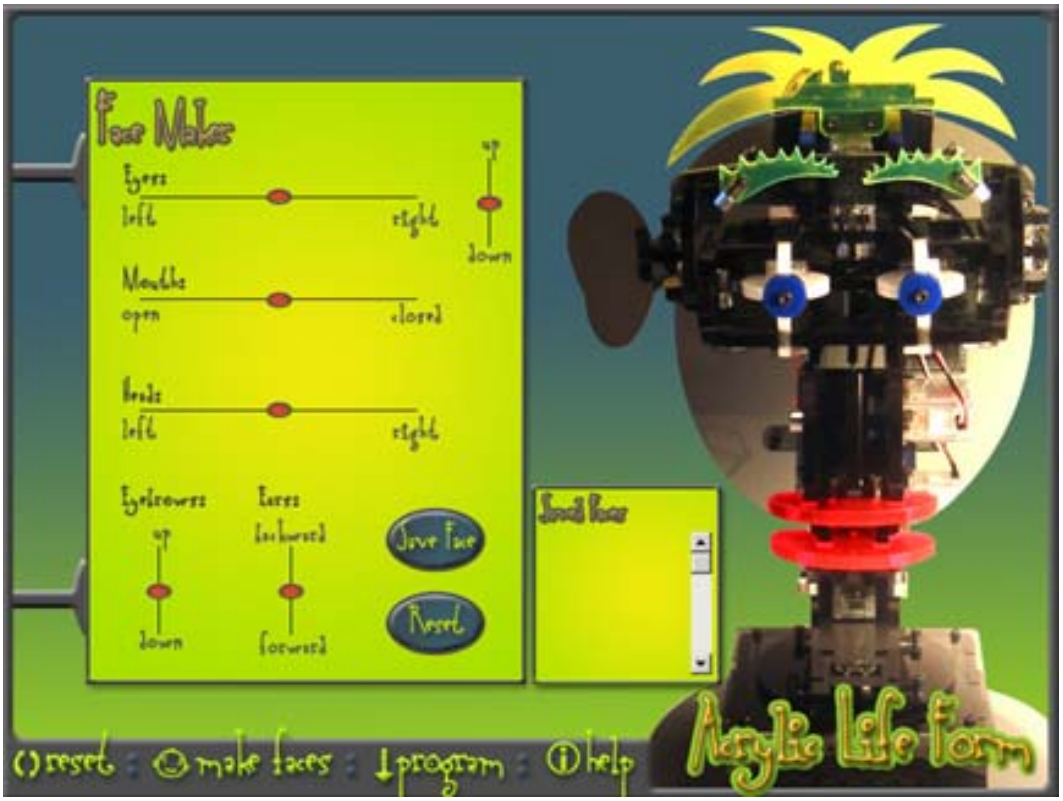
So how do you do think that I did the "wiggle eyebrows?"

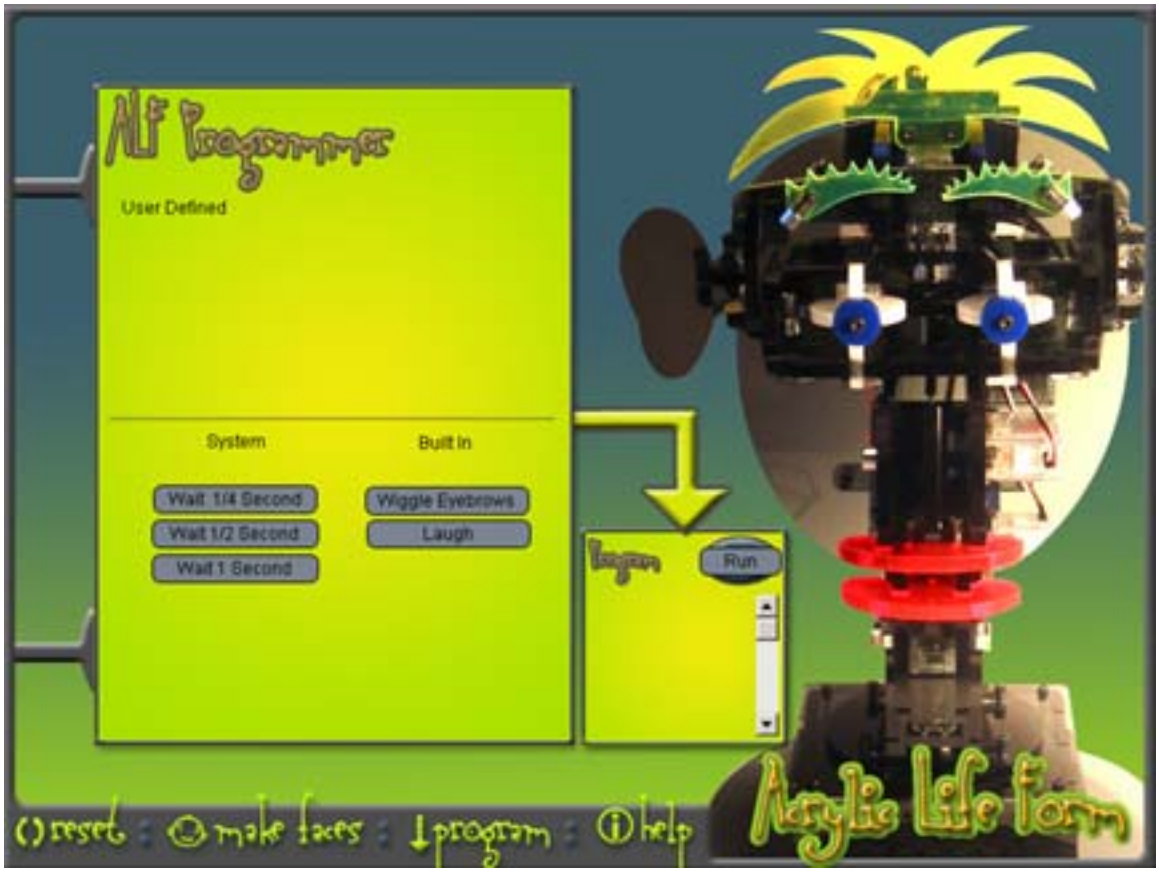
I think… okay.
(*click*)
I think you. I don't know!
Oh I! Wait…
I don't know how you do it.

So you did that (*click*) then you did that (*click*) Kind of?

▲**Figure 12.1** Prototype, Splash Screen

▼ **Figure 12.2** Prototype, Face Making Mode





▲**Figure 12.3** Prototype, Programming Mode

…Tell me exactly what ALF does when he's wiggling his eyebrows.

What?

What does ALF do when he's wiggling his eyebrows? What do you do when you wiggle your eyebrows?

(*Alex wiggles eyebrows*)

Do it one step at a time.

(*Alex raises his eyebrows (I do the same)*)

Up! And then what….

(*Alex lowers his eyebrows (I do the same)*)

And then what?

(*Alex raises his eyebrows*)

And then what?

(*This repeats several times until we're both laughing*)

Okay great

I can do it really fast. Cool.

Kind of.

(*pause*)

Is that what you did?
No… I don't think so.

(*pause*)

Well, tell me what…

I don't know!

Alex is puzzled. He has tried clicking on the built-in function, watches it make ALF move and then tries a few more buttons. He sees that the built-in works, and I can see he's trying to understand what is happening, but it's not clear. I decide to help him break the process down by using our own faces as the example. A mirror might have also been helpful here.

So can you do that with ALF right here?

What?

Can you make ALF do what you just did?

Okay. (*click*)(*click*)(*click*)(*click*)(*click*)

We are in Face Making mode now and Alex is moving the slider up and down. He stops and clicks on "save face." ALF is moving his eyebrows in real time, as long as Alex keeps moving the slider himself. He can manually control the face, but it's not the same as creating a primitive. Alex almost understands this, but can't quite articulate it to me or himself yet.

Hey… (*clicks on face he saved, eyebrows move once*) Cool.

Just how. I don't know… just a little bit… It's complicated!
I'm not too good at computers. I'm more good at math stuff.

We'll let's try it. So you were doing it with this control, right?

Uh huh

But how do you think we could make it do it automatically?

I know *this* doesn't work. (*clicks on face he saved*)
See it just kind of did… Does what you did first.

This is a big step. Alex has just explained to me that when you save a face in the face making mode, it is saving the last state, not recording your actions. The puzzle is about to collapse.

Okay. So that doesn't work, so that can't be what you did. So wait! Wait! I KNOW! When you save it, it puts it all together!

Okay, let's try it.

What make another one!?

Yeah.

Just like that?

Yeah

Oh God…

Let's see if we can do it, we'll do it together.

It is clear from Alex's excited tone of voice that he has had a breakthrough. He understands something about how the system works. I want to be completely sure that he really does understand it, so I want him to try and duplicate the built in function. He seems to think this is quite difficult but is willing to humor me. The challenge occupies us for most of the rest of the session. Finally, about 45 minutes later, Alex has succeeded in creating an “eyebrows up” state and an “eyebrows down” state and sequencing them. He has duplicated the function I provided. I ask him to explain to me what he has done.

Feb 12th / 54:19

Okay, so when you do [save face] and it gets into here, it doesn't put all those together... it doesn't put like … together... if you went like "ooh ooh ooh ooh" while doing it...

Uh-huh

It won't put that together.

Okay, why not?

Ah... it just doesn't!

Alex is again explaining to me that the face making mode and “save face” function does not record the users actions, it only stores the last state. During this explanation he is gesturing to the screen. The “ooh oohs” emphasize his moving the slider back and forth.

Okay, So if you wanted to put it together, what would you have to do?

You have to... okay... delete... delete... then go like, do this, what I did with the eyebrows.

Okay

But in program you can also just do this
(*points to "wiggle eyebrows" button*)

Right

Wiggle eyebrows. In program you can do that.

So far so good. One more check to see if he understands what's going on. Can he abstract this method of problem solving and apply it to something else? He can wiggle the eyebrows, could he do the same thing with another ALF feature?

Feb 12th / 55:20
What if you wanted to wiggle the eyes?

Wiggle the eyes? Well, this doesn't work [tries earlier face] that doesn't work... So if you did...

Feb 12th / 59:37

I mean like... you have to make an "eyebrows up" or... you have to make like a "whatever up" or a "whatever down" or... "whatever right" or "left," basically... right?

Yes, he not only understands that the method can be applied to other parts, but also that the motion makes no difference. You could make another up and down, or a left or right, "or whatever." Is this the "right" way to do things?

Feb 12th / 61:03
You can use that too, but if there isn't one... such as ears... wiggle ears... than in that you have to use that method.

Oh Okay.

But that's a good thing, right? It's okay.

To do what?

It's okay to have an "eyebrows up" and an "eyebrows down" and "wiggle eyebrows" in a built-in, right?

I think so, but why do you think it's okay?

It just is, you can choose. You can choose it... right?

You can choose?...

You can choose the built in or the user defined.

At this point Alex is reading "built-in" and "user-defined" off of the screen. He is doing a great job of explaining. I am thrilled. At the same time, his use of my vocabulary makes me aware of the fact that

whatever vocabulary I present, including the on screen labels I took for granted, will be picked up and used by the children. I make a note to carefully consider what language I use. This is one factor that contributed to the decision to completely eliminate text from the interface.

One more question. This interface does not allow one to store programs as primitives yet. I wonder if Alex really gets the equivalency - the fact that his program, which appears like the image on the left, is identical to the program shown on the right:

▲**Figure 12.4** Equivalent programs

I ask this as a value question, partially because earlier Alex seemed to be asking which way was "right."

Feb 12th / 61:45

Can you think of sometime when you might want to use this one instead of that one? You already told me one of them… like for eyes or something

```
Well you just choose. It's just like sometimes that and sometimes this
one.
```

Okay.

```
…they both kind of put it… They both put it together.
```

CONCLUSION OF PRELIMINARY WORKSHOP

I was quite happy with the results of this preliminary study. Not only did Alex seem to enjoy the experience, but with little prompting from me he proved that he understood some basic powerful ideas and demonstrated the usefulness of the face for understanding sequencing and containership. I enthusiastically put together a workshop involving Alex and two additional children.

FEB 26th: HOW MANY CHILDREN *WAS* THAT?

For this session, I invited Alex to return and work with two additional children: not-quite four year old Cindy, and five year old Jeff. After the success of the first workshop, I was ready for more and had in mind that Alex would teach what he had learned to the others. I made some minor revisions to the

software and brought in two more computers and ALF heads, so that each child could have their own setup. Despite my optimism, things did not go nearly as smoothly as the first time, although there were many lessons learned.

First, based on empirical evidence, the number of four year old children in one space seems exponentially greater than the number who are actually there. None of the children "behaved badly," but I was not prepared for the number of questions, the kinetic energy or the number of distractions the Media Lab offers children of this age. Although I later ran another workshop with this same group of children, I found that one-on-one worked much better than groups. (I am also very grateful my colleagues in the Grassroots Invention Group for helping to keep an eye on the group that seemed to be everywhere at once.)

I suspect that my preference for a one-on-one session would not hold with an ideal software environment, but as I was still hammering out the details of the software design, a degree of focus (on behalf of the children and myself) was required. This was difficult to accomplish with a 3:1 ratio.

THE TROUBLE WITH TEXT

Two minutes into the session I realized I had made a major oversight – the entire interface was text based! This was catastrophic, as only one of the children in the group could read. I had simply forgotten the fact that many four year olds cannot yet read, and my experience with Alex (self described as a "very good. very good. very good reader.") had done nothing to disabuse me of this notion.

Cindy and Jeff's mother was present during this session and helpfully explained the entire interface to her children. With her help the session was not an unmitigated disaster, but this was the final straw. My inclination to eliminate text from the environment, begun when Alex started using the phrase "user-defined," was set. The next version of the software would use minimal text, and ultimately I switched to a completely visual interface.

THE TROUBLE WITH MICE

A few minutes into the workshop, I began to realize what else I had taken for granted: a level of fluency with the dominant computer user interface. I had explained to the group that you could move each of ALF's features using the on-screen sliders:

Feb 26th / 01:38

Jeff: It's not sliding!

Let me try… this seems okay… Try clicking on the red part.

J: This one?

Right in the middle of the red part.

J: Right here?

And then you have to hold the mouse button down, and then slide it over…

J: Um…

Cindy: I was holding it down on it and sliding over and it didn't move!

J: And look at this…

I know I'm in trouble. There's no way I should need to explain how to operate my interface. This much instruction just to move a feature on ALF? The purpose of this project was not to get bogged down by the computer! The sliders still aren't moving. The children are getting frustrated, and I'm worried.

Feb 26th / 02:28

Hrm, let me try, maybe you're having trouble. Hold the mouse button down. So.. put the mouse over the red part, now click the mouse button.

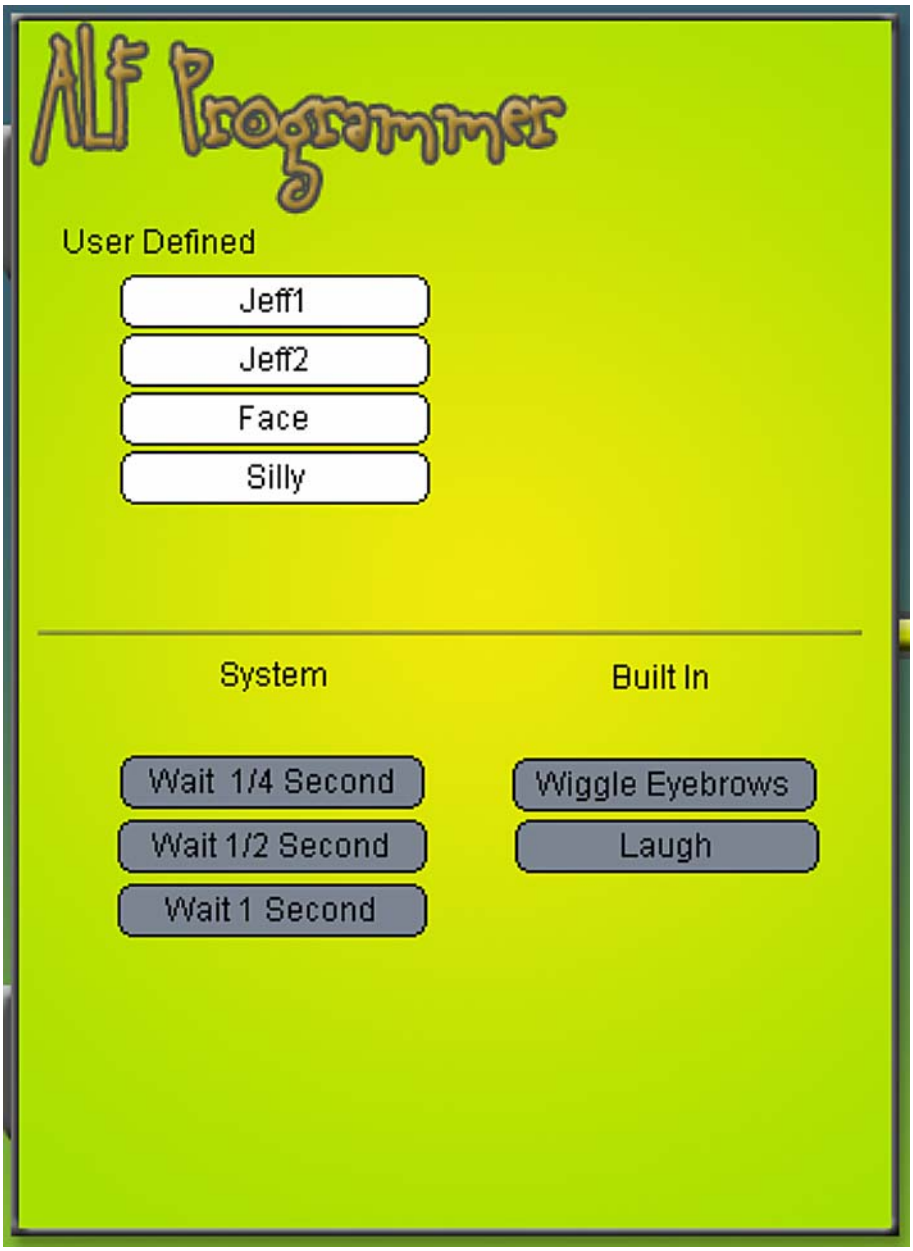
Oh! I See! I didn't tell you, it has to be this *mouse button!*

Lots of trouble! It takes two minutes of observation to realize the children are clicking on the right mouse button instead of the left. It doesn't take them long to switch, but this mistake is repeated at least once more this session. One more assumption I shouldn't have made: that four year olds have internalized the "left mouse button for primary function, right mouse button for additional functions" standard that I take for granted. Four year old hands barely fit around the mouse. This experience served as the catalyst for my consideration of input devices other than the standard keyboard and mouse, as well as suggested an immediate fix: eliminate the software distinction between mouse buttons.

WIGGLE METHOD REDUX

Despite the mouse button trouble and his limited reading ability, five year old Jeff seems to be making headway. I decide to ask him the same question I asked Alex in the first session, how he might duplicate the built-in functionality with his own program. This time, I ask him to duplicate

"laugh" which is essentially the same procedure as wiggle eyebrows, applied instead to the mouth. I am worried about the text getting in the way, but I have color coded the functions, so I start that way.



▲**Figure 12.5** Alex's programming screen

Feb 26th / 11:22
So I have a question for you. When you are in this part in the program.

Jeff: Okay program

So these buttons, the grey ones, what's the difference do you think between these and the ones that you made?

Jeff: These ones have different parts in them. These ones basically only have one part.

Cindy: These are a different color!!

Yes they are a different color

Jeff: How do you make them a different color?

I have already resolved to eliminate the use of text. This exchange contributes to my decision to use color as a method of distinguishing objects from one another in later versions of the software. More importantly, however, Jeff understands something about the different primitives – one is made up of multiple parts, the other is only one part. Further discussion makes it clear to me that Jeff is encountering the same bug that Alex encountered earlier – assuming that his state definitions are recordings of a series of actions (hence the "multiple parts"). I work with him a bit further and he *almost* gets it.

Feb 26th / 14:23

Jeff: Now how can I…
look at this.

Okay

It's not showing you what I did.
I…
click

Cindy: What is he clicking?

Jeff's trying to figure out…

J: Oh look! Oh I put the…

He's trying to figure out how to make the mouth laugh thing by himself.

J: What happened here?
And I… I made the mouth go open… uh… open shut open shut open shut!

Okay

J: And look!

It's not showing up there

J: I clicked on this. Okay. See what happens with the robot.

Okay. So what's he doing?

J: He's just moving this up and down, and I did not do that!
I made him laugh and then I made his eyes… His eyebrows up and down!

Jeff is seriously frustrated here. I can hear it in his voice. He's ready to give up. It's not doing what he wants it to. I am asked three questions at once and try to answer them while keeping Jeff from getting too exasperated. I try and see if I can help him to figure out what went wrong. ALF performed one movement and stayed that way.

Okay, is that the normal face that he's making? The "in the middle" face?

J: Uh-uh.. no…

What face is that?

J: Which face? The face that I made… is it?

Yeah.

J: So how can I… put it back?

More distractions here and Jeff is left to figure things out on his own. A few moments later I hear him exclaim. It seems he's figured something out.

J: Oh I know what I did! I clicked on this…

A misunderstanding. Jeff is clicking on the *wait* buttons. He seems sure that these are preventing his program from working properly. He asks me what they say and is less certain about his conclusion after I read them to him. Another distraction and I completely miss the following exchange, which I hear later on the recording. (Alex is sitting two seats down from Jeff. He apparently sees what is going on and wants to help, but he doesn't approach Jeff, he just starts talking and assumes he is heard. Jeff either ignores or doesn't hear him.)

Alex: Wait! Let me show you. Wait let me show you. Look.

J: It does the same, it doesn't work.

A: Look look look, if I do what? Hey… *click*

J: It doesn't work.

A: Hey!

J: I guess I should just go.

Reviewing the tapes I am painfully frustrated by Jeff's frustration. Had I not been distracted,it is likely that I would have been able to help him figure this out. He came very close on his own. I feel particularly bad about this because his main source of frustration comes from his inability to read and therefore understand what the interface is doing. At this point I notice Alex is trying to explain something.

Feb 26th / 17:32

So Alex, what are you showing us?

(more distractions)

Alex: Wait one second and then run.
See, it will wait one second if you put another one after the waitonesecond. Okay? That's what will… that's what happens.

Did you understand what Alex was saying?

I am hoping that Jeff will say no. Especially having missed the earlier exchange, it's not clear to me what Alex is telling us. I suspect afterwards that Jeff reads more into my question than I really meant. Perhaps he feels unsure of himself – a child who is a year younger than him can read and he can't. Alex has been making ALF go this whole time and Jeff is frustrated. I am asking if he understood what was going on. Does he think I'm disappointed in him? I'm not, but I am certainly not as focused as I should be.

J: Yeah.

Okay. So…

More distractions ensue! Things don't improve. Jeff has fun making ALF move, but he never does figure out the laugh function. Ultimately Jeff gets bored and wanders away from the computer to play on his own outside of the workshop room.

This workshop was disappointing, but by negative example taught me as much if not more than the first workshop. I was on the right track, but a lot of work needed to be done to make the interface support this kind of learning. On top of the list was the elimination of text from the interface.

TAKE THREE

Another workshop was run with the same group of children and a new version of the software. This time I was prepared for the chaos and recruited the help of my colleague Margarita Dekoli, who has both experience running workshops and also familiarity with the domain by virtue of her work on CodaChrome.[127] I also made significant changes to the software interface. Although it functioned in exactly the same way, text had been replaced with drawings and icons. This proved far more successful with the children, who suddenly understood what they were doing! I opened and closed this session by explaining briefly what I was trying to do and soliciting the children's opinions about

[127] Dekoli.

"what works, what you liked, what you didn't like." When I asked about the text versus pictographic representation, the children universally agreed that the non-text version was better and more fun, although towards the end Alex reversed his position because he'd been "using the one with words longer."

THE FLOGO GROUP

At the same time that I was running my workshops, Chris Hancock of the Lifelong Kindergarten Group was completing his work on Flogo[128] and was hosting a daily workshop with a group of middle school students from Milton Academy. Chris was kind enough to loan me his students for a short time so that I could watch them work with ALF and engage them in conversation afterwards. This group of children was considerably older than my target group. Because of their age and fluency with computation, they "hit the walls" of the environment far faster than my group of young children. This session, while short, provided me with invaluable insight.

Concurrency - The original ALF software as used in the workshops did not allow for any kind of concurrent movement. Chris' students found this frustrating, and several asked me how they could make ALF do two things at once. My original assumption was that it was not necessary to move more than one motor at once because ALF held its last position and could move fairly quickly (meaning that I expected the children to simply sequence all the moves they wanted to make a face). It became clear that this was not as it should be. Real faces do not move into position one feature at a time, nor does this kind of allow for the kind of containership that I wanted (which implies that any of six features can be in any of 100 positions at any given time).

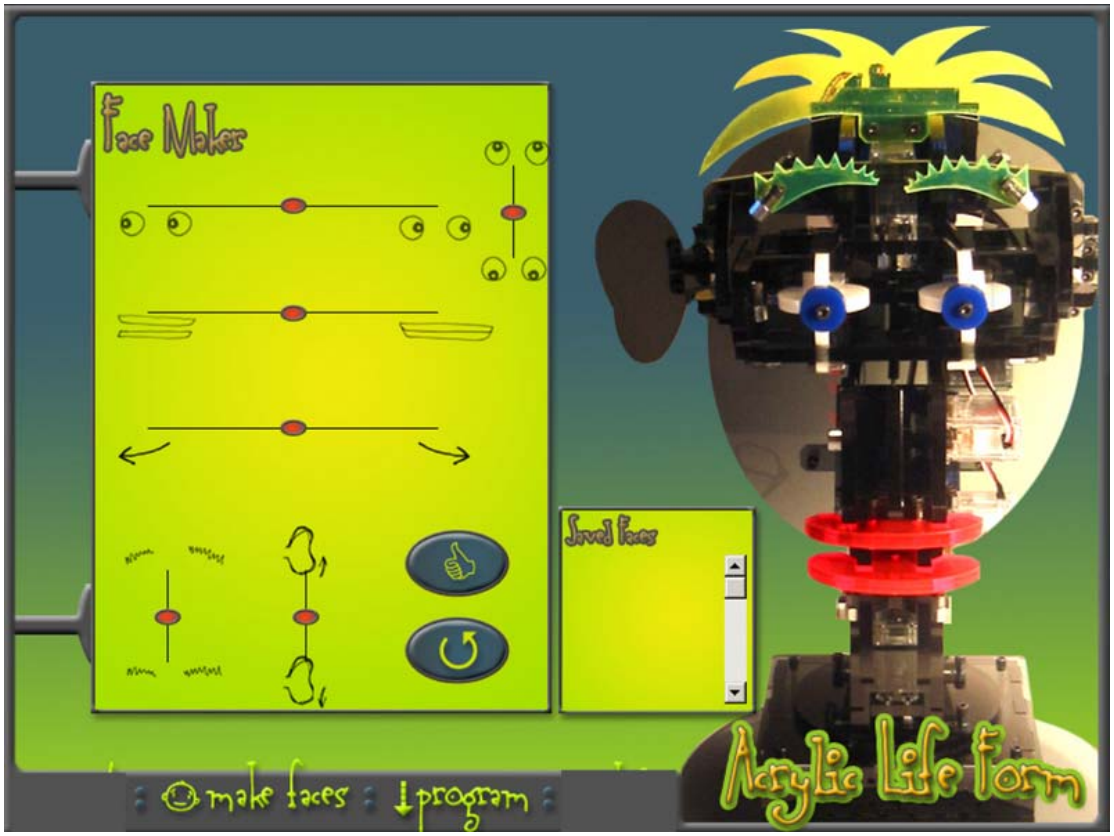
Ability to save and undo – The original ALF software began its life as a museum kiosk application and did not allow for the saving or loading of faces, programs or preferences. Working with four year olds, this was never a problem, as none of them ever had an inclination to save their work for later retrieval. Working with Chris' group, this became an instant problem given the large child to ALF ratio (ten or so children using three ALFs). Once brought to the forefront, it was apparent that this omission also prevented the reuse of code and the stated goal of using the system for storytelling, a process that would certainly require a number of versions of the same "software."

Length of programming space - The programming space provided by early versions of the ALF software was quite limited. Users were only allowed to designate eight primitives, and the programming list, while a scrolling text area and theoretically infinitely long, did not display many

[128] Hancock, *Real-time Programming and the Big Ideas of Computational Literacy.*.

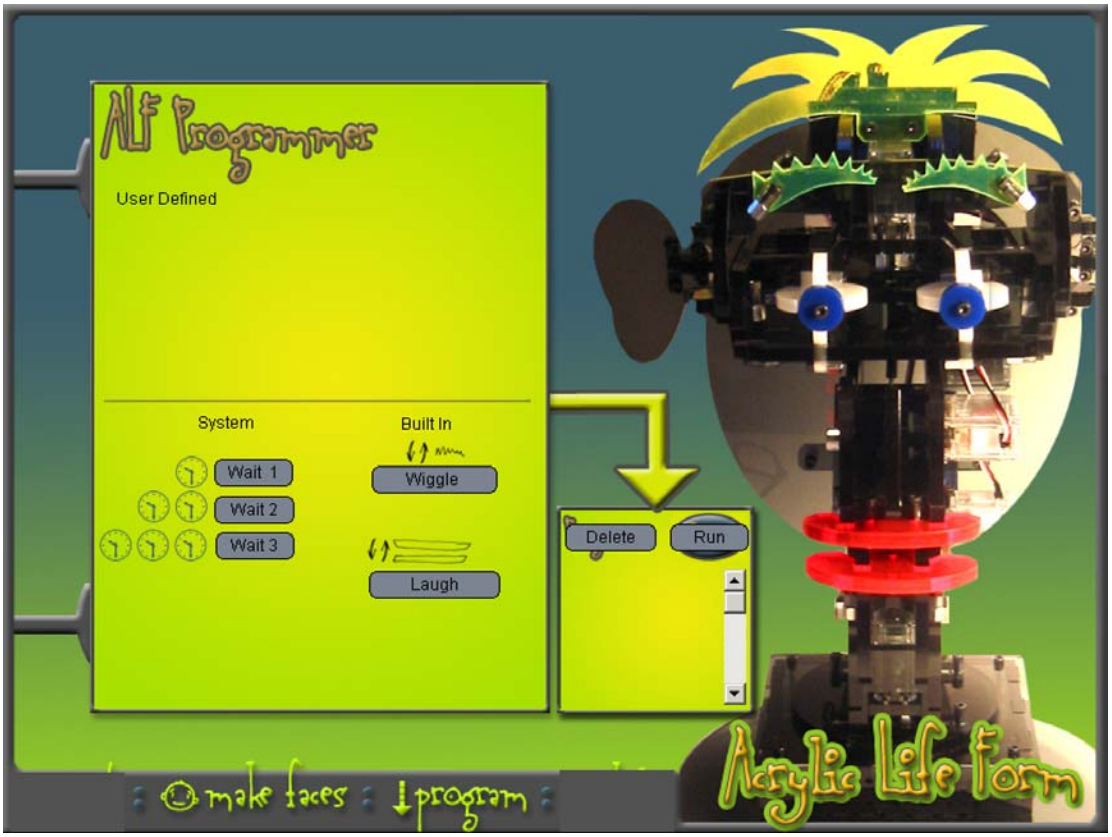
commands on screen at any given time. These limitations were quickly exceeded by Chris' students, whose initial interactions with ALF were far more complicated than my group's.

Random access, stepping and runtime indication – Familiar with programming, Chris' students wanted the ability to tweak their programs, to step through them and to insert and remove commands at will. The original software allowed for none of this, providing only a stack onto which children could push commands and, optionally, reset to an empty state. This was a known problem that did not trouble the younger children but quickly became a blocking factor with the older group.



▲**Figure 12.6** Prototype 2, Face Making Mode

Visual constancy and the placement of "holding spaces" – As you see in the screenshots, the size, shape and location of the *Saved Faces* and *Program* lists are identical. This proved confusing, as their visual similarity implied they were the same object, but their functions were completely different. Also, the decision to change the representation of the computational object "face" (from an item on a text list to a clickable button) was made deliberately in order to emphasize the fluidity of representation afforded by digital space. This idea is a powerful one, but after using the environment with children, it became apparent that my choice served only to confuse.



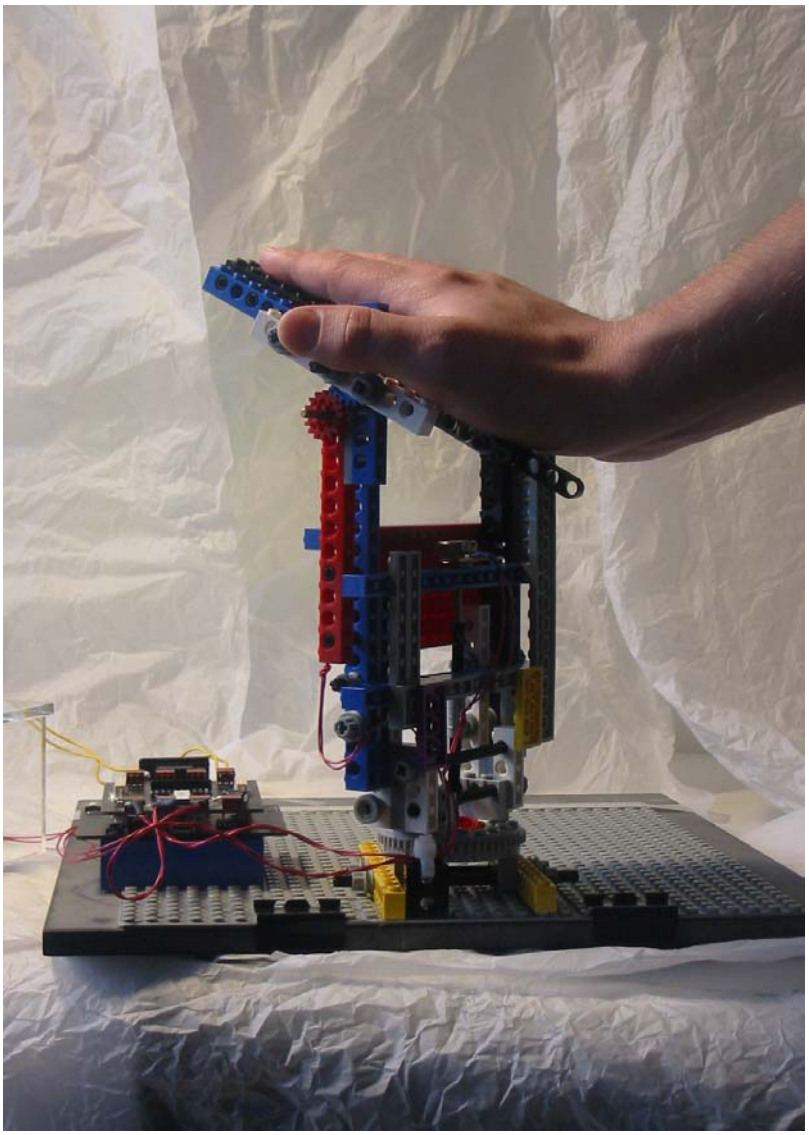
▲**Figure 12.7** Prototype 2, Programming Mode

13 CTRL_SPACE

It was unfortunately not possible to fully implement and test the CTRL_SPACE environment. The design presented below is complete. Proof of concept prototypes were constructed where the software was not completed. Given the design presented below and the promising nature of the initial workshops, I believe the stage has been fully set for future work.

CTRL_ARM

With the complete revision of the software environment and in light of the troubles encountered using the mouse and slider combination, I decided that it made sense to construct a tangible interface whose output could be fed through CTRL_SPACE to ALF.



▲**Figure 13.1** CTRL_ARM

Early on I imagined a device that might bring the act of issuing commands to ALF closer to the physical action of puppeteering. At the same time, I wanted to create an interface which would lend itself to a demonstration of the flexibility of computation and abstraction and not simply consist of a manipulable replica of ALF. To that end, I constructed a two axis armature, called CTRL_ARM, which uses analog potentiometers to measure hand movements (figure 13.1). CTRL_SPACE allows a child to map the 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. 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. The arm was constructed with an eye towards ergonomics

(as far as it is possible to do so with Lego) and anticipates multiple hand types. As shown, CTRL_ARM accommodates an adult hand (right or left). The platform covered by the hand in the photo is also split in two, with an open area about halfway up, intended to accommodate the curled fingers of a young child's hand. A handle is also provided (visible just below the wrist in the photo) for those who wish finer control, or else prefer to move CTRL_ARM like a lever.

In the process of constructing CTRL_ARM, I became aware of the vast number of possible interfaces that one could build, as well as the relative ease with which this is possible using the Tower system as a base. Not shown is a pressure sensor which I also connected to ALF. I imagined placing this inside a ball which could be squeezed to activate an action, or else inside the head of a cloth puppet which could be used in a manner similar to CTRL_ARM. Consequently, although I began constructing the CTRL_ARM hardware with the intent of presenting it as a finished object for children to use, I would now like to suggest that building an alternative input device is a rich computational activity and offer CTRL_ARM as a proof of concept. Although it is unlikely the youngest children could engage in this by themselves, they could certainly contribute to the design of an interface (as they have in this thesis). In practice, CTRL_SPACE may also be used with older children and adults, and the engineering challenges involved in constructing a tangible interface may be of particular interest to them.

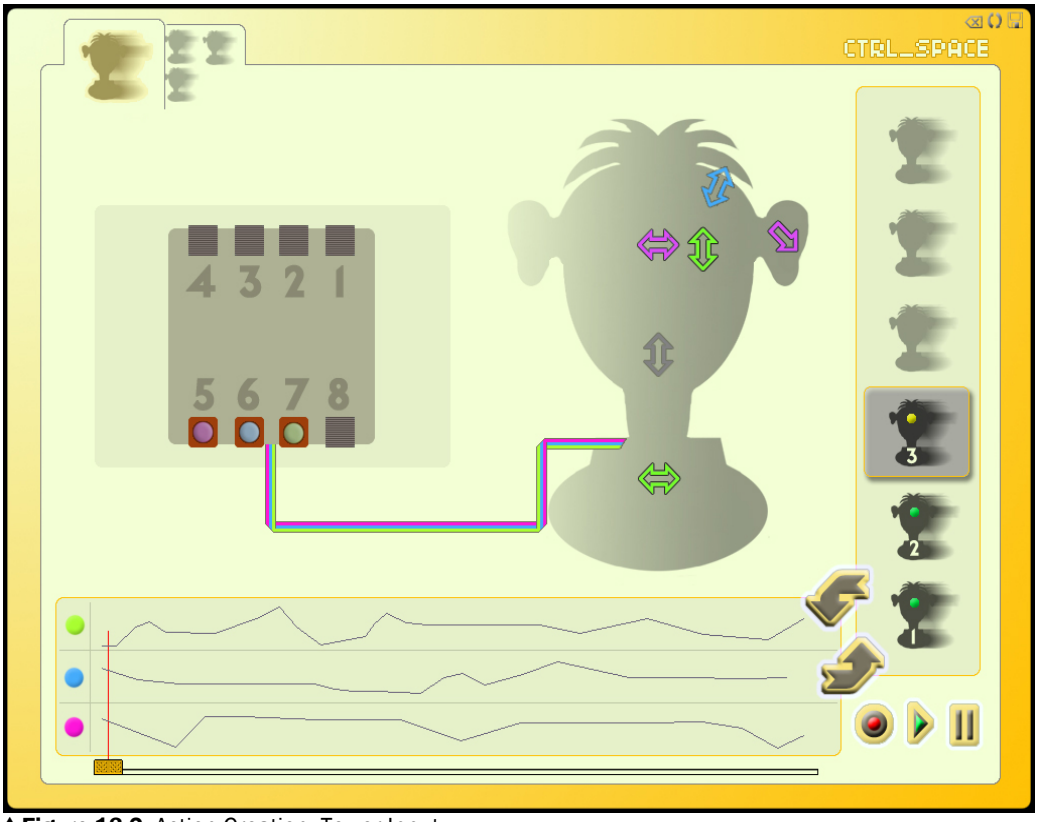
SOFTWARE

Building on the lessons learned from the workshops described in the previous chapter, the CTRL_SPACE software was completely redesigned. In addition to reworking the interface, the low level code running on the Tower was rewritten for speed. What follows is a description of CTRL_SPACE's key features. Chapter fifteen will describe possible scenarios for use.

THEORY OF OPERATION

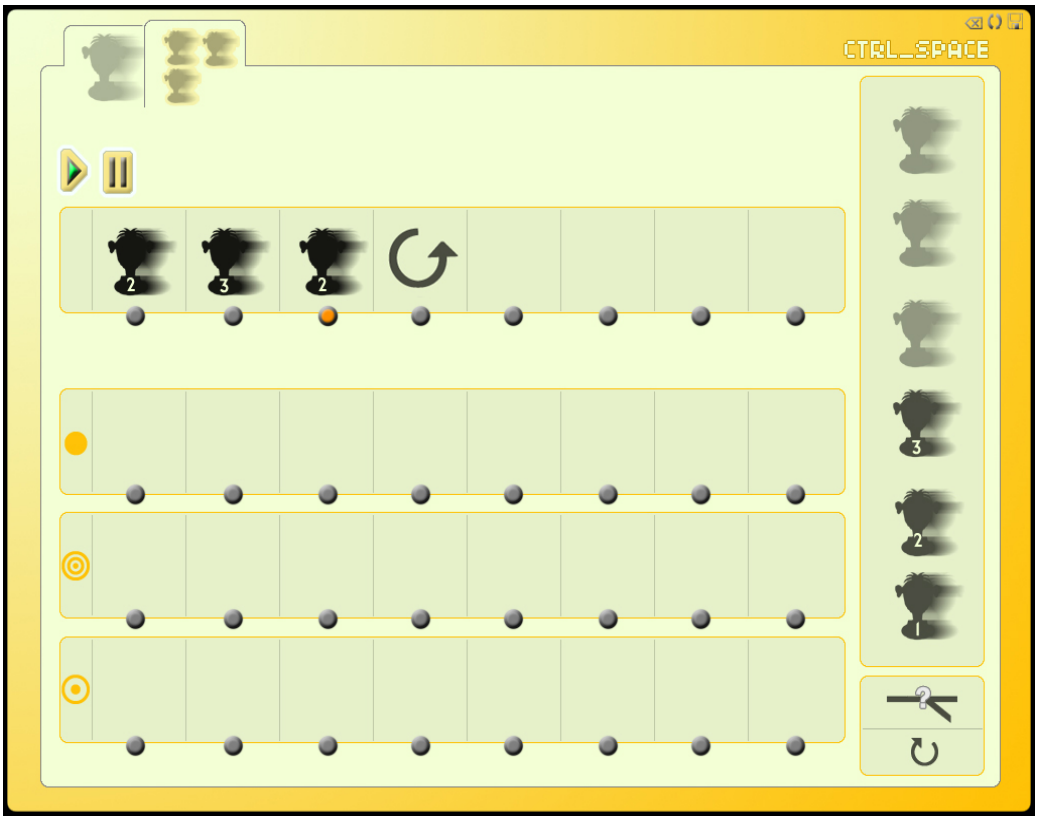
The CTRL_SPACE environment centers around the idea of *action.* The environment supports two modes of use: *action creation* and *action sequencing*. Figure 13.2 and figure 13.3 show screenshots from the action creation mode and the action sequencing mode respectively.

Actions are represented by two related fields: the timeline which shows a visual representation of change over time on a scale from zero to one hundred (the range of positions offered by ALF's

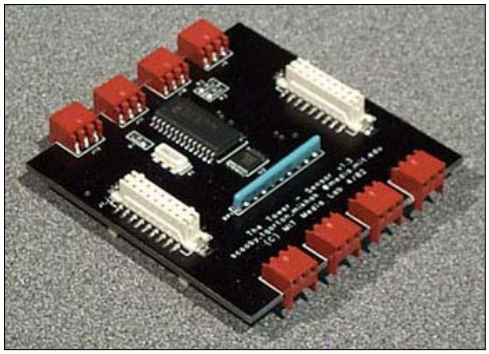


▲**Figure 13.2** Action Creation, Tower Input

▼**Figure 13.3** Action Sequencing

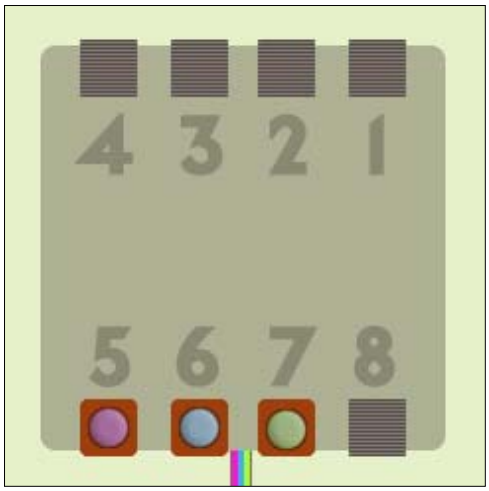


motors), and the mapping of these values to particular features of ALF, as shown by the color coding of the arrows on the ALF head.



▲**Figure 13.4** Tower Sensor Layer

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.



▲**Figure 13.5** Graphical Sensor Layer

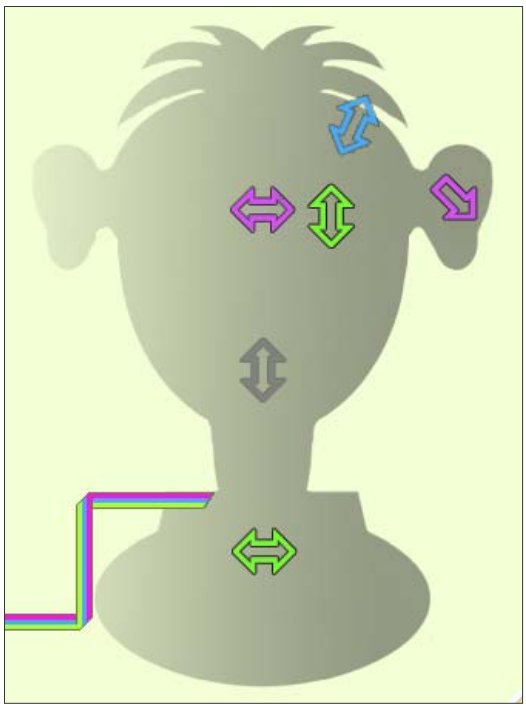
Once a child has built up a library of actions, they may use the action sequencing mode (figure 13.3) to define a "program" consisting of a sequence of a number of actions. Sequencing mode also introduces basic logic structure and branching on the basis of conditionals.

A NOTE ON TIME

While time and timing play an important part in the use of CTRL_SPACE, I felt that the introduction of time units (seconds, minutes, etc.) would only complicate matters. The width of the timeline represents the maximum length of an action. In practice this is about ten seconds. CTRL_SPACE records physical input in real time, an action as recorded lasts as long as a child wishes to continue moving, up to the maximum.

INPUT

Figure 13.5 is a graphical representation of the Tower sensor layer (figure 13.4) which is used for input. Initially, CTRL_SPACE included a graphical representation of the CTRL_ARM hardware, but as it became apparent that increasing the possible number of input devices cost very little and added much, I decided to use a Tower representation instead. The particular screenshot shown represents the Tower with three sensors connected. The numbers on screen correspond to the numbers screenprinted on the circuitboard of the sensor layer. The round shapes on the sensor ports indicate data flow on active sensor ports. As shown, three sensors are connected to the Tower, but all are grayed out (each of the colored circles is "dim." They appear brighter when data is being received). This indicates that no input is currently being provided by the sensors.



▲**Figure 13.7** Sensor Assignments

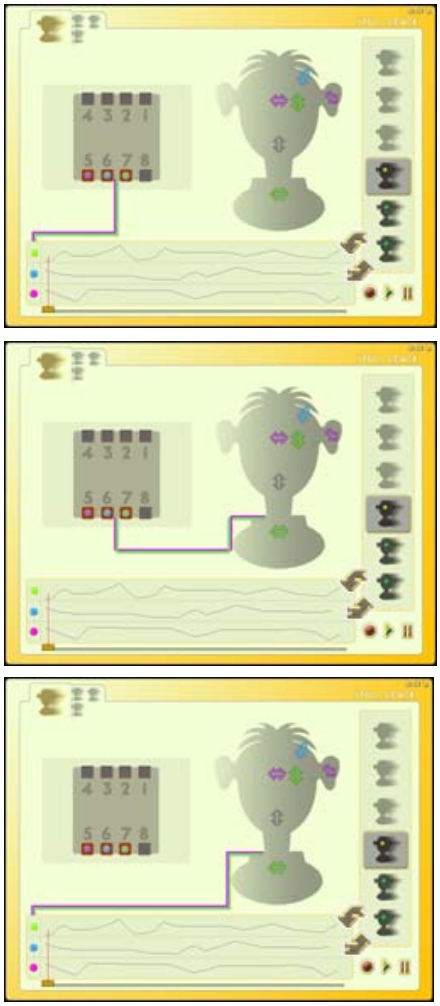
OUTPUT

Color is used to indicate which sensor provides which data. Each sensor is assigned a track in the action timeline at the bottom of the screen. The movement of ALF's features is controlled by assigning a particular color to each of ALF's six features. Grey indicates a null assignment. Arrows are gray by default. Colors are changed by clicking on each arrow. With each click the arrow changes color, cycling through all the options in order.

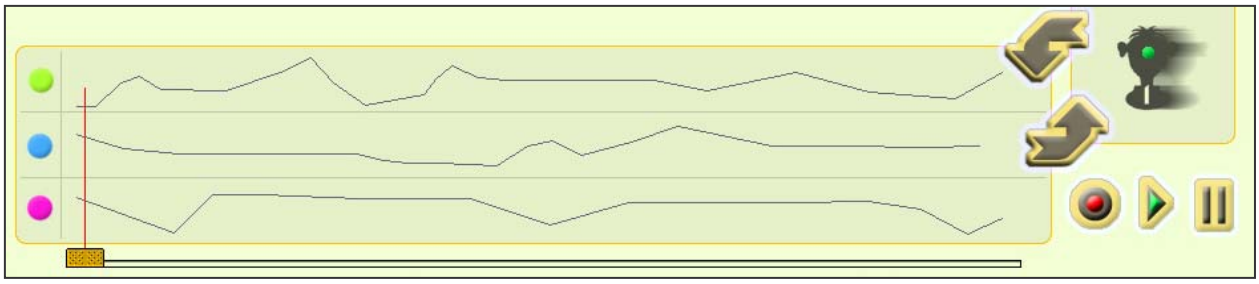
With the ALF settings shown in figure 13.7, ALF will use the data from sensor seven (green) to control the vertical movement of the eyes and the turning of the neck, the data from sensor five (pink) to control the ears and the horizontal movement of the eyes and the data from sensor six (blue) to raise and lower the eyebrows. The mouth, in this case, is unassigned and will remain stationary.

DATAFLOW

At the bottom of the input representation shown in figure 13.5, one can see three colored lines which correspond to the sensors of the same color. These lines indicate the path of the data collected by the computer. Depending on what the user is doing at a given time, these lines indicate where the sensor values are being sent to: from the Tower to the timeline for recording, from the Tower directly to ALF for preview or from the timeline to ALF for playback. The dataflow lines for each of these scenarios are shown in figure13.6.



▲**Figure 13.6** Dataflow



▲**Figure 13.8** Timeline

EDITING ACTIONS

In addition to controlling ALF by connecting features to the input of sensors, users can control each of ALF's features directly via on-screen sliders (figure 13.12). This is particularly useful for editing actions. After an action has been recorded, a point in time can be selected by moving the slider at the bottom of the timeline to the desired position. ALF hardware will reflect the position indicated by the timeline, as will the on screen sliders. The face can be edited by moving any of the sliders. Indicators next to each slider display the relationship between the slider, the timeline and a particular feature of ALF. If a feature is unassigned, one may still adjust the position via the sliders, but the unassigned feature will not appear in the timeline.

▲**Figure 13.9** Action Palette

SAVING AND RETRIEVING ACTIONS

An action is defined by a combination of the values specified in the timeline and the feature assignments specified by the color assignments. These assignments are saved along with the sensor timeline by clicking on the lower arrow to the right of the timeline as shown in figure 13.8. Actions may be loaded by first selecting them and then clicking on the upper arrow.

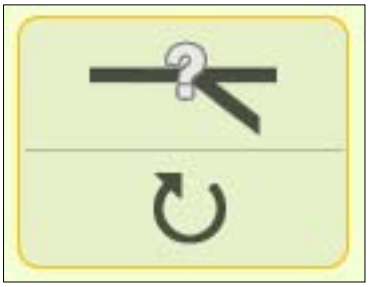
The action palette at the right of the screen (figure 13.9) provides a storage space for actions which have been saved for later use in an action sequence. The arrows may be used to save or load saved actions into the timeline. As shown, the system provides storage for six actions at a time.

Bank switching should be implemented to allow for more faces, although allowing for too many may introduce confusion.

The grey select box (figure 13.9) indicates the action currently being edited or modified. Saved actions are indicated with a green dot. Saved actions currently being edited (changed but not saved) are indicated by a yellow dot. Grayed out faces with no dots indicate available storage space for new actions. To create a new action, the user selects one of these "empty" actions.

ACTION SEQUENCING

Once a number of actions have been created, the user can sequence them in action sequencing mode. Action sequencing mode uses a metronome like representation of time rather than a timeline. The top row of frames indicates the main program while the three rows below indicate optional subroutines (figure 13.3). Actions occur in the order they are sequenced, from left to right. The action currently being executed is indicated by an orange dot appearing in the gray circle under each frame of the sequence. The duration of the entire program is equal to the duration of the sum of the sequenced actions. Instances of actions are created by dragging them from the action palette to the sequence grid. In addition to the action palette, the user is given a palette of special functions (figure 13.10) which are used to control program flow. The first option indicates a branching of program caused by a conditional statement, the second indicates a loop. There are several kinds of loops and branching functions which will be discussed below.

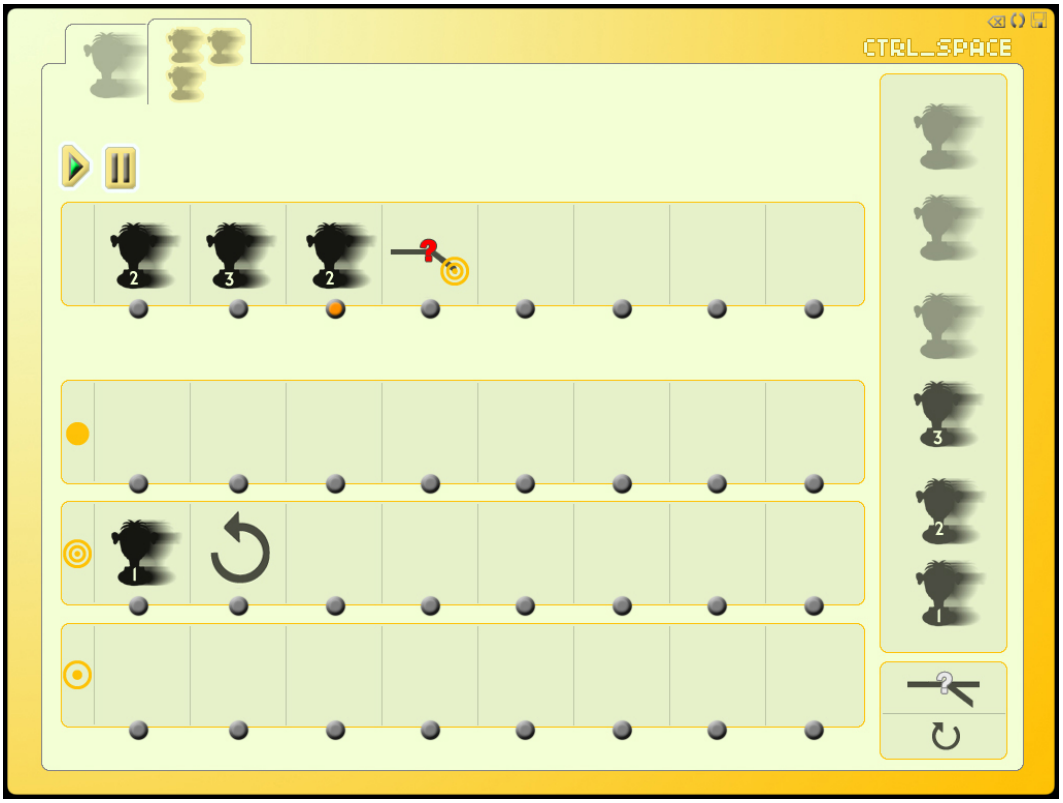


▲**Figure 13.10** Special Functions

SAMPLE PROGRAMS

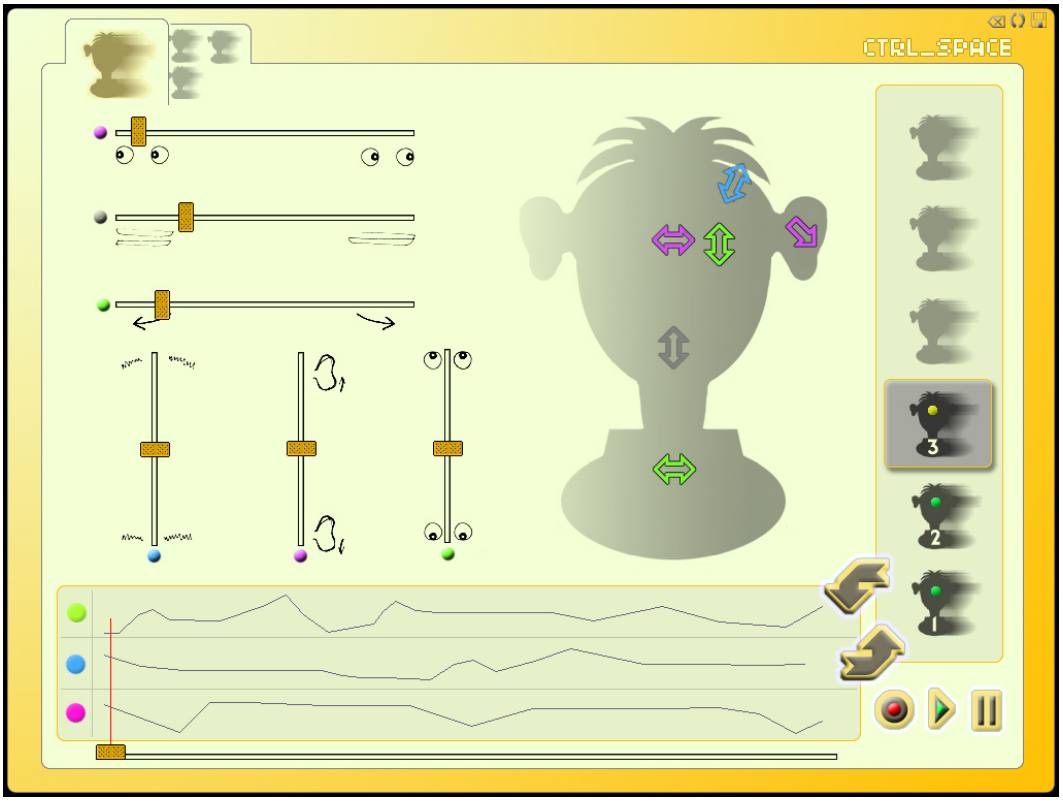
In figure 13.3 there is a simple program which, when run, performs action 2, action 3 and action 2 before performing a loop, which starts the sequence over again. Figure 13.11 shows a more complicated program. This sequence performs action 2, action 3, action 2, but following that performs a conditional test. The red color of the question mark indicates that this is a *wait until* style of conditional, which will pause program execution until the condition is true. If the condition is true, the program continues at subroutine double circle. This subroutine consists of action 1 on an infinite loop. In pseudo code, this might appear as follows:

```
action_2()
action_3()
action_2()
waituntil (CONDITIONAL)
loop(action_1)
```



▲**Figure 13.11** Action Sequencing

▼**Figure 13.12** Action Creation, Slider Input

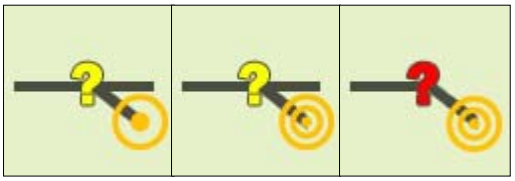


CONDITIONAL BRANCHES AND LOGIC STRUCTURES

When a user drags the conditional branch icon onto a frame (or clicks on a frame which contains a conditional) they are presented with a dialog box (figure 13.14) which 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 subroutine indicated. 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. Figure 13.13 indicates such a structure. The pseudo code representation is as follows:

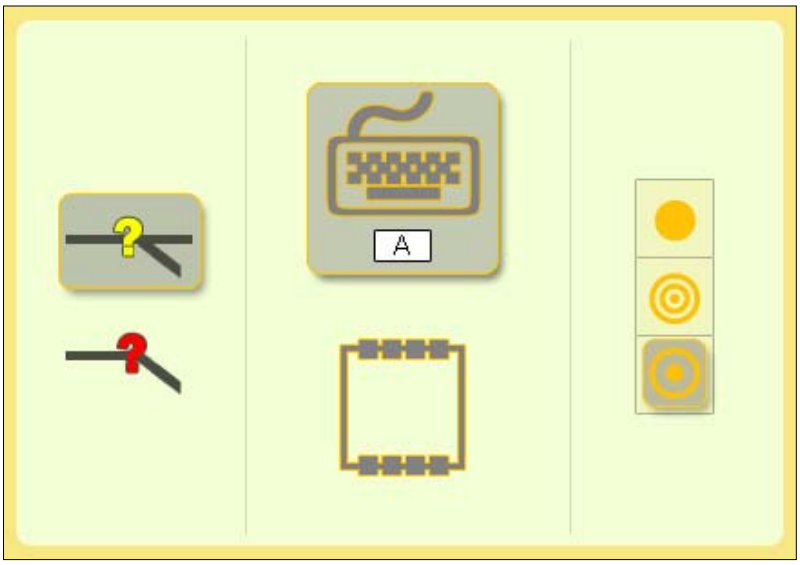
```
if CONDITION A
{
       subroutine ⊙

}

else
{
       if CONDITION B
       {
              subroutine ◎
       }

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



▲**Figure 13.13** Logic Structure

TYPES OF CONDITIONALS

CTRL_SPACE supports two kind of conditionals: keyboard and sensor input from the tower. Figure 13.14 shows the dialog box provided to the user for building a conditional. If the conditional is a keyboard conditional, the user presses the key they wish to use as a trigger. If it is a sensor conditional, they are given the option to select which sensor and threshold.



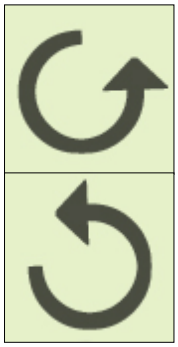
▲**Figure 13.14** Logic Dialog Box

TYPES OF LOOPS

Loops come in two flavors and can also be thought of as limited jumps. Loops cause the program execution to return either to the previous frame or to the first frame of the main sequence. This is specified via a dialog box when the user places a loop onto a frame, or edited when a frame containing a loop is clicked. Figure 13.15 top shows a “loop entire program” loop, bottom a “loop previous frame” loop.

ADDITIONAL FUNCTIONALITY

Additional functionality is provided by a series of icons in the upper right corner of the screen. These three icons provide a mechanism to reset the software, calibrate ALF and save and load workspace respectively. These functions are nearly hidden because they are administrative. None of them is likely to be of use while working with CTRL_SPACE, and it is more than likely they will only be accessed by a facilitator.



▲**Figure 13.15** Types of Loops

14 GUIDELINES APPLIED TO CTRL_SPACE

APPLICATION OF MY OWN GUIDELINES TO CTRL_SPACE

I hope that throughout this entire document the reader will be able to make connections between the guidelines outlined in chapter eight and the development of this project. By way of analysis, I will now present the aspects of the design that most explicitly demonstrate an attempt to fulfill each of these guidelines, where I feel I have succeeded and more importantly, where I feel the system falls short.

GUIDELINE 1: The use of computation should serve a clear purpose

The introduction of puppetry and computation into animatronics changes the nature of both. While it is the storytelling nature of face making that lends itself to an alternate representation of computational concepts, it also fundamentally changes the possibilities of storytelling. Although a single instance of CTRL_SPACE with one ALF is not sufficient to have this effect, multiple ALFs controlled by a single child would allow for the creation of multiple characters, which can interact with a child, group of children or each other by virtue of sensors and computation. CTRL_SPACE as presented supports a single output. For this project, four ALFs were constructed and each was used with its own computer. This could be applied to a group situation, but ideally the environment would be developed further so that a single computer could be used to control multiple ALFs. This allows one child to control multiple objects easily.

GUIDELINE 2: Users must be able to play with underlying rule set, not only parameters

As we have seen, the initial kiosk application created for ALF functioned mainly by allowing users to tweak the parameters of ALF's face. One computational concept was introduced in the ability to sequence these faces into actions, although the initial versions of the software allowed only one level of this. The CTRL_SPACE environment represents a concerted effort to provide deeper and more interesting access to computation by providing basic logic elements and multiple levels of containership.

GUIDELINE 3: Avoid excessive error correction

CTRL_SPACE and ALF work very well in this regard. By design, the system avoids syntax entirely and so does not allow one to make syntactical errors. This is extremely important, because the resolution of syntactical errors involve not an understanding of computation, but an understanding of the language being used to describe a computational event. By avoiding the need for syntax we may remain focused on the important issues of how computation itself works.

At the same time, this system does not wish to deprive users of the experience of debugging a problem. The physical nature of ALF makes this system ideal for debugging, as the result of errors in sequencing or logic are clearly visible. One does not need a chart of error codes to understand that there is a problem when ALF looks left instead of right. This makes the process of debugging a computational object quite literally tangible.

GUIDELINE 4: Ambiguity is a good thing

In language, syntax and grammar make strings of words less ambiguous. By embedding syntax in the visual environment, we no longer require the user to explicitly define the meaning of particular "words." This is made possible by solid visual interface design, which made possible by the existence of powerful and inexpensive computation.

Eliminating the need for syntax makes CTRL_SPACE easier to use and understand, but this is accomplished only by embedding the context in the interface itself. Despite making an effort to stick to well understood signifiers, such design choices are being made by one with a comparatively intimate understanding of computation. Inevitably, time will show that this interface has made use of cultural and contextual referents that are far from universal.

GUIDELINE 5: Difficult doesn't mean better

The computational concepts that are addressed by CTRL_SPACE are perhaps the simplest: basic logic, the notion of states, sequential events. Whenever possible, I have included the ability to use and talk about more complicated ideas (such as concurrency, made possible by use of a multi-track timeline). I believe the focus on these basic concepts improves the usability of the system, preventing the user from immediately encountering bewildering puzzles such as a two dimensional visual representation of recursion.

GUIDELINE 6: System should allow connection to the familiar

The familiarity of the face as a base unit of computation has been discussed at length. I believe CTRL_SPACE does well in this regard. In cases where it doesn't, I have made an effort to provide enough "hooks" that the system can be easily expanded to suit a particular case.

GUIDELINE 7: System should support growth

CTRL_SPACE deliberately limits the ability to access certain methods of computational expression. For example, much has been made about the elimination of text from the interface. In terms of

general purpose programming, this makes things very difficult. I discussed this problem and suggested a solution in chapter nine: CTRL_SPACE should be considered one of a family of solutions. Effort has been made to give CTRL_SPACE a high ceiling *and* a very low threshold. Any compromise of the latter by the former is mitigated by the ability to swap out the control software if it becomes a blocking factor.

GUIDELINE 8: System should encourage reflective public interaction

CTRL_SPACE by itself does not encourage reflective public interaction. Many digital environments including some of those mentioned in chapter nine (Zora, Full-Contact Poetry) do offer or define clearly a space where the user can post and discuss the work they have done.

Earlier I defined "system" as not only the digital and physical objects in use, but also as the philosophy and approach taken. This is where I wish to rely on my definition: CTRL_SPACE is intended to be used as a tool for performance. The facilitator of this performance is responsible for encouraging reflective practice and contemplation. Although it wouldn't hurt, my experience as an undergraduate at the Art Institute of Chicago taught me the value of face to face communication in a critique environment, and I don't believe a digital discussion space is the most affective way of reflecting on a performative object.

It has been suggested to me that young children "aren't ready" for a formal critique. While I agree that one must proceed cautiously, I believe this objection comes largely from a widespread misunderstanding of what a critique is and how it should function. A critique is primarily about trust and coming to a consensus as a community of practice. It can be as complicated as a two hour discussion or simple as a few comments. A critique, properly facilitated, is a classic example of constructionist practice – a method of grassroots evaluation whereby value judgments come from the community rather than top-down from an externally consulted expert. I feel strongly enough about this to say that if the person receiving a critique ever feels threatened or hurt, the critique should be considered a failure.

Perhaps some of the discomfort here comes from the fact that the word *critique* implies a formalism of approach, to the extent that it seems laughable when I suggest that a group of four year olds discussing what they like about ALF constitutes a critique. This, however, is exactly what I was inviting the children to do when I openly asked them how they felt about the environment. Provided that it is properly facilitated and the peers involved are of similar age and understanding, I see little

reason that a young child shouldn't learn how to participate in a critique. For a positive example of this approach in action, I recommend *Making Learning Visible : Children as Individual and Group Learners* a book that resulted from a collaboration between Reggio Emilia and Harvard's Project Zero.

GUIDELINE 9: System should encourage the creation of an artifact

ALF is an artifact, but as an artifact for discussion ALF is far more useful to its creator Chris Lyon, the Grassroots Invention Group and myself. For potential users of CTRL_SPACE, the artifacts for reflection fall into two categories: performance and animatronic object. A group of children who present a performance using CTRL_SPACE create an artifact for reflection by bounding the performance: "This is the performance, here it begins, now it ends." This can then be reflected upon as an artifact: a clearly defined occurrence in time.

I do not wish to imply that all performances (or even all performances that might be created with CTRL_SPACE) define such boundaries. Some of the most interesting performances owe their existence to "alternative" approaches which deliberately play with the conceptual and physical boundaries between performance space and audience space, process and product. CTRL_SPACE, however, suggests a bounded space by virtue of the fact that it is built around a linear timeline with a definite beginning and end. The user is most likely to begin in this mode, though it is up to them how and when they wish to use the environment.

I also suggested that the creation of an animatronic object for CTRL_SPACE is a rich activity. One can imagine presenting ALF as proof of concept and then asking children to build their own robot. This is perhaps better accomplished with children older than my target group, but it is a powerful activity and highly recommended. Similarly, the environment supports any manner of "home made" input devices by virtue of the use of the Tower. The design and creation of an input device is another area full of possibilities. Some of these will be explored in the following chapter on scenarios for use.

15 SCENARIOS FOR USE

Although it may be useful to explain my vision for how the system might be used, this chapter is not intended to serve as an activity guide for CTRL_SPACE. For the environment to be truly effective, it should be adapted to the situation in which it is used: to the number of children involved, the amount of time and resources available and, perhaps most importantly, to the children's interests. Experimentation is key, and I look forward to seeing what others make of the environment.

STORYTELLING AND PERFORMANCE

Storytelling is a word that brings to mind many images: story time at the library, a favorite book, an elderly relative in a rocking chair, a shaman around a fire. Performance seems to imply lights, a stage, a script. While these interpretations are legitimate, and while CTRL_SPACE may in fact be used as a support for these types of organized storytelling, I imagine that a more mundane version of storytelling is an equally rich domain for children to explore.

For adults, storytelling is an everyday occurrence. The stories we tell each other are perhaps unexceptional (perhaps not), but the performative nature is obvious when considered. Think of a group of good friends out at a bar, swapping stories. Someone tells a story that focuses everyone's attention, and they tell it well, eliciting emotional and physical responses, using props, lighting. The story sparks a memory and someone tells another story, and another, and the result is a kind of group performance that will entertain for hours.

For children, storytelling indicates the first steps towards understanding agency in themselves and in others. Stories are a way of explaining the world. For very young children, the line between signifier and signified, past and present, object and objectified is still being drawn and the most basic of stories can reveal a fascinating look at the child's mental construct.

What is most interesting to me is the possibility that the CTRL_SPACE environment might be used as a tool for reflection, an object to think with that allows a child to explore what it means to tell a story by analyzing and breaking it down computationally. I have no illusions that four year olds will become philosophers expounding insight on signifier and signified, but an intuitive understanding of concept is more than legitimate for a young child. For the facilitator of the activity, observing and helping a child construct a story can also be an insightful experience. One example from the workshops I ran: While working with Alex, I asked if ALF was making a "normal face." I often called the default face that ALF makes (all motors in the center position) the "normal" face. This particular time, Alex strongly objected. It wasn't a normal face, he explained, because *no one* makes that face normally.

He proceeded to show me (by example) how ridiculous the face would be if he wore that expression "normally." For Alex, this exchange was likely of little significance but for me it provided an insight into how differently the two of us perceived what ALF was. For me, ALF was a mechanical device made of motors first, and a schematic of a face second. Before this exchange with Alex, it had never occurred to me that the default, the "normal," should be anything other than "motors in the center." Alex's comment made me realize that, at least at that moment, he was seeing ALF as a "one," and seen from the perspective of an animate (or semi-animate) object[129], my "normal" face was quite bizarre!

ANIMATRONIC CHARACTER AS EXTENSION OF SELF

One of the most interesting things that one could do with ALF is to use ALF as an extension of oneself. When telling a story, children will often project their feelings onto others or even onto inanimate objects. Dolls, stuffed animals and pets play an important role in this kind of storytelling.

Children do not need to be prompted to do this, but by introducing an environment which can record actions, one creates the ability to formalize and reflect on this relationship. A child could tell a story about their day, assigning a different emotion or event to one of a number of ALFs, or to a number of actions which are later sequenced. The most interesting thing to watch for here would be the use of emotional vocabulary: "ALF is angry because I looked at him" rather than "ALF is making the 'angry face' action because I triggered this proximity sensor."

ANIMATRONIC CHARACTER AS THE OTHER

Sensors may be used as triggers to activate actions with ALF. It is easy to imagine that a child could create a dialogue with an ALF. The child speaks to ALF, triggers a sensor and ALF "speaks" back. This gives an opportunity for the child to reflect on relationships by actively constructing a performed relationship. Bugs the child might encounter and how they express them will be of particular interest here – determining if and when ALF acts inappropriately in a given situation, for example.

CTRL_SPACE AS AN ENVIRONMENT FOR SCRIPTING PERFORMANCES

CTRL_SPACE lends itself to scripted performance. The ability to break down a story into component parts and sequence it with external cues provided by sensors resembles a traditional stage play. It is easy to see how the environment might be used with any age group as the support technology for

[129] Sherry Turkle's recent work with robotic toys and children has identified a trend among many children to designate such intelligent objects as "sort of alive." Not alive like humans or dogs, but not inanimate like rocks or pieces of toast.

such a performance. Floor or flex sensors could replace CTRL_ARM, and lights or music could replace ALF. A good example of such a performance (conducted with Squeak, Logo and custom hardware) was the recent interactive dance performance *Roballet*, held at the MIT Media Lab. An environment like CTRL_SPACE is well suited for such a performance.

CREATION OF AN INPUT DEVICE

Creating an input device for CTRL_SPACE offers a rich space for exploration, in particular for older children. A number of engineering problems are involved in selecting the right sensors, determining their dynamic range and appropriate application. If this is done in the context of a performance, there are a number of aesthetic considerations which have an impact on design decisions: is the sensor bulky? Is it wireless or tethered? How does the introduction of a new sensor change the nature of the performance?

CREATION OF AN OUTPUT DEVICE

It is possible to modify ALF or to replace ALF completely with another motorized object. The design or redesign of an output device for CTRL_SPACE covers the same ground as the creation of an input device, with the added satisfaction of seeing one's creation "come alive."

LONG TERM SCENARIO - PULLING IT ALL TOGETHER

One possible scenario, best suited to a school or another environment where a large number of mixed-age children are available, is to use CTRL_SPACE with a Tower for a school play or performance. While the early introduction might consist of creating control routines for ALF as presented here, later the students might design their own ALFs, analyze the control software, build their own environment or combine all of the above. This idea could then be applied to a school-wide performance, such as a school play. The younger children might select a favorite story and write the script and design the sets. Older children could build the sets, animatronic characters and sensors. All children could be involved with the performance itself.

AESTHETICS OF PERFORMANCE

In all of these activities, the aesthetics of performance are extremely important. I would like to encourage anyone who uses this system to avoid considering the computational concepts inherent in CTRL_SPACE to be the "real material" while relegating the performance component to the status of the sugar in the medicine. Such an approach does little justice to either concept and, frankly, takes all the fun out of working with children and animatronics.

CONCLUDING THOUGHTS

I began this chapter by explaining that this is not an activity guide. Several times I have mentioned that any approach to introducing computational concepts to children (indeed any teaching activity at any age) requires a holistic approach including the environment, philosophy and activity. Much work has gone into the design and construction of CTRL_SPACE, and I have presented a theoretical method of using the environment in school, but computer software and hardware, no matter how developed, will remain insufficient by itself. CTRL_SPACE alone won't do anything. An engaged facilitator is absolutely necessary.

When I was an elementary school student in the 1980s, my school adopted Logo as part of its math curriculum. Although I enjoyed "playing with the computer" (an early introduction that very well may have contributed to my lifelong interest in technology), in retrospect my school's use of Logo was an educational disaster. After scheduling time in the computer lab, classes were read programs out of an activity book. Logo was presented as a drawing tool, and the teachers provided examples of complex drawings as the goal. The closest we ever came to generalizing the concepts came when we drew our initials on graph paper and translated them into code. Not long after this experience, I taught myself to program a Commodore VIC20 computer in BASIC. The idea that Logo was, in any way, related to this activity (or to math, for that matter) never occurred to me until after college, and I only understood what it was really intended for when I encountered Papert's writings in graduate school.

I believe my experience with Logo had a lot to do with my teacher's bewilderment when confronted with this strange new technology: a room full of computers that likely cost the school more than her yearly salary. The computer was a foreign object, unrelated to anything else, and it was treated as such. This is conjecture of course, but what I know for sure is that this is not likely the case today. The computer, for better or worse, is a familiar object, and I have great faith that contemporary teachers are better at filtering the technological wheat from the chaff than they were fifteen years ago. Even so, it seems important to say again: this environment is useless without the right approach.

I do sincerely hope that this work reaches educators, that it will continue, and that those who use it will share the stories of their successes and failures. With the right frame of mind and a healthy spirit of exploration, I believe that working with children and animatronics is an exciting and enriching form of "serious play" that is often a great deal of fun.

References

MouseSite, http://sloan.stanford.edu/mousesite/1968Demo.html.

(1998). Chinese scientists back eugenics. BBC Online Network. UK: Available at:http://news.bbc.co.uk/1/hi/world/asia-pacific/198555.stm.

(1999). World War One. Encyclopedia Britannica, Online edition, available at: http://www.britannica.com/bcom/eb/article/3/0,5716,118863+10+110198,00.html.

(2003). "Letter to the Editor." Wellesley(Spring 2003).

About.com (2003). Computer Keyboard, http://inventors.about.com/library/inventors/blcomputer_keyboard.htm.

Ackermann, E. (2003). "Constructivism: One or Many." Learning Environments(IOS Press, Amsterdam. Series 'The Future of Learning', vol 2.).

Allen Cypher, D. C. H., David Kurlander, Henry Lieberman, David Maulsby, Brad A. Myers, and Alan Turransky (1993). Watch What I Do: Programming by Demonstration. Cambridge, MIT Press.

Andrew Dahley, C. W., Hiroshi Ishii (1998). Water Lamp and Pinwheels: Ambient Projection of Digital Information into Architectural Space. CHI, Los Angeles, California USA, ACM.

Artmuseum.net (2003). Dynabook, http://www.artmuseum.net/w2vr/archives/Kay/01_Dynabook.html.

Aspray, W. e. (1990). Computing Before Computers. Ames, Iowa State University Press.

Avrich, P. (1980). The Modern School Movement: Anarchism and Education in the United States. New Jersy, Princeton University Press.

Basu, A. (2002). Full-Contact Poetry. MIT Media Lab. Cambridge, MIT.

BBCi (2003). Modern World History: Fascism in Italy, http://www.bbc.co.uk/education/modern/fascism/fascihtm.htm.

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

Ben Piper, C. R., Hiroshi Ishii (2002). Illuminating Clay: A 3-D Tangible Interface for Landscape Analysis. CHI, Minneapolis, Minnesota, ACM.

Bers, M. U. (1999). Building a Society of Self: Zora, a Narrative Graphical Multi-user Environment. Computer Support for Collaborative Learning, Palo Alto, CA, Stanford University [Available from Lawrence Erlbaum Associates, Mahwah, NJ].

Biography.com (2003). Mussolini, Benito (Amilcare Andrea), http://search.biography.com/print_record.pl?id=6179.

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

Carse, J. P. (1986). Finite and Infinite Games : A Vision of Life as Play and Possibility. New York, Ballantine Books.

Computer.org (2003). http://www.computer.org/history/development/1964.htm.

Damon, A., Howard W. Stoudt, Ross A. McFarland (1966). The Human Body in Equipment Design. Cambridge, Harvard University Press.

Dekoli, M. (2003). Coloring Time with Codachrome. MIT Media Lab. Cambridge, MIT.

Donoghue, D. (1994). The Old Moderns. New York, Alfred A. Knopf.

Druin, A., James Hendler, ed. (2000). Robots for Kids: Exploring New Technologies for Learning. San Francisco, Morgan Kaufmann Publishers.

Duckworth, E. (1996). "The Having of Wonderful Ideas" and Other Essays on Teaching and Learning - Second Edition. New York, Teachers College Press.

Farbood, M. (2001). Hyperscore: A New Approach to Interactive, Computer-Generated Music. MIT Media Lab, MIT.

Feynman, R. P. (1996). Feynman Lectures on Computation. Reading, MA, Addison-Wesley Publishing Company, Inc.

Hancock, C. (2001). Children's Understanding of Process in the Construction of Robot Behaviors. AERA, Seattle.

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

Hayes Raffle, A. P., Hiroshi Ilshi (2003). "Topobo." Unpublished, available at: http://tangible.media.mit.edu/projects/topobo/topobo.htm.

Herrnstein, R. J., Charles Murray (1994). The Bell Curve : Intelligence and Class Structure in American Life. New York, Free Press (Simon and Schuster).

Hersh, J. (1996). The Tyranny of the Keyboard. http://www.tifaq.org/articles/keyboard_tyranny-feb98-jay_hersh.html.

Hickey, D. (1997). Air Guitar : Essays on Art & Democracy. Los Angeles, Art Issues Press, Inc.

Hiroshi Ishii, B. U. (1997). Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. CHI, Atlanta, Georgia USA, ACM.

Jenkins, H., Ed. (1998). The Children's Culture Reader, NYU Press.

Kaes, A., Martin Jay, Edward Dimendberg, eds. (1994). The Weimar Republic Sourcebook. Berkeley, University of California Press.

Kevles, D. J. (1985). In The Name of Eugenics: Genetics and the Uses of Human Heredity. Cambridge, Harvard University Press.

Krausse, A. C. (1995). The Story of Painting from the Renaissance to the Present. Koeln, Koenemann Verlagsgesellschaft mbH.

Levy, S. (1984). Hackers: Heros of the Computer Revolution. Garden City, Anchor Press/Doubleday.

Lienhard, J. H. (1997). No. 207. Selden's Automobile, available at: http://www.uh.edu/admin/engines/epi207.htm.

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

Maeda, J. (2000). Maeda @ Media. New York, Rizzoli Publishing.

Marciniak, C. P. (2003). Scoring the Tests (Letter to the Editor). Wellesley. Spring 2003.

Marianetti, F. War, the World's Only Hygene, available at: http://www.unknown.nu/futurism/war.html.

Matthew Gorbet, M. O., Hiroshi Ishii (1998). Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography. CHI.

Mclean, R. (1975). Jan Tschichold: Typographer. London, Lord Humphries Publishers Ltd.

Mclean, R. (1997). Jan Tschichold: A Life in Typography. New Jersey, Princeton Architectural Press.

Myers, B. A. (1998). "A Brief History of Human Computer Interaction Technology." ACM interactions. Vol. 5(2): 44-54.

Negroponte, N. (1996). Being Digital, Vintage Books.

New, R. S. (2000). "Reggio Emilia: Catalyst for Change and Conversation." ERIC EECE http://ericeece.org/pubs/digests/2000/new00.html.

Nies, B. L. (2002). Eugenic Fantasies: Racial Ideology in the LIterature and Popular Culture of the 1920s. New York, Routledge.

Orwant, J. (2000). "EGGG: Automated Programming for Game Generation." IBM Systems Journal 39(3&4): 782-794.

Papert, S., Idit Harel (1991). Situating Constructionism. Constructionism, Ablex Publishing Corporation.

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

Papert, S. (2002). The Turtle's Long Slow Trip:Macro-Educological Perspectives on Microworlds, Baywood Publishing Co., Inc.

Partsch, S. (1991). Franz Marc 1880-1916. Cologne, Taschen Verlag.

Pheasant, S. (1996). Bodyspace : Anthropometry, Ergonomics and the Design of Work Second Edition. London, Taylor & Francis.

Phil Frei, V. S., Bakhtiar Mikhak, Hiroshi Ishii (2000). Curlybot: Designing a New Class of Computational Toys. CHI, ACM Press.

Phillip R. Reilly, D. W. (1999). "Eugenics: 1883-1970." GeneSage Geneletter Feb 1999: available at: http://www.genesage.com/professionals/geneletter/archives/eugenics18831970.html.

Piaget, J. (1977). The Essential Piaget. New York, Basic Books.

Ramsey, P. (1970). Fabricated Man: The Ethics of Genetic Control. New Haven, Yale University Press.

Reggio (2001). Making Learning Visible : Children as Individual and Group Learners. Reggio Emilia, Reggio Children, srl.

Ryokai, K., Vaucelle, C., Cassell, J. (2002, January 7-11). Literacy Learning by Storytelling with a Virtual Peer. Computer Support for Collaborative Learning, Boulder, CO.

Ryokai, K., Vaucelle, C., Cassell, J. (2003). "Virtual Peers as Partners in Storytelling and Literacy Learning." Journal of Computer Assisted Learning 19(2): 195-208.

Sammet, J. E. (1969). Programming Languages : History and Fundamentals. Englewood Cliffs, Prentice - Hall Inc.

Seymour Papert, S. T. (1992). "Epistemological Pluralism and the Revaluation of the Concrete." Journal of Mathematical Behavior Vol. 11, No.1(March): 3-33.

Shockley, W. (1992). Shockley on Eugenics and Race: The Application of Science to the Solution of Human Problems, Scott Townsend Publishers.

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

Stokstad, M. (1995). Art History. New York, Harry N. Abrams, Inc.

Strohecker, C. (1991). Why Knot? Media Arts and Sciences. Cambridge, Massachusetts Institute of Technology: 554.

Tilley, A. R. (1993). The Measure of Man and Woman. New York, Henry Dreyfuss Associates.

Tronick, E. Z. (1986). Maternal Depression and Infant Disturbance.

Tschichold, J. (1995). Die neue Typographie: Ein Handbuch fur Zeitgemaß Schaffende (The New Typography: A Handbook for Modern Designers). Berkeley, University of California Press.

Turkle, S. (1995). Life on the Screen: Identity in the Age of the Internet. New York, NY, Simon & Schuster.

WGBH Paul Rand Interview, http://main.wgbh.org/wgbh/NTW/FA/TITLES/Paul324.HTML.

Wikipedia NLS, http://www.wikipedia.org/wiki/On-Line_System.