ABSTRACT

Title of Dissertation: PHYSICAL PROGRAMMING: TOOLS FOR KINDERGARTEN CHILDREN TO AUTHOR PHYSICAL INTERACTIVE ENVIRONMENTS

Jaime Montemayor, Doctor of Philosophy, 2003

Dissertation directed by: Assistant Professor Allison Druin
Professor James A. Hendler
Department of Computer Science

StoryRooms is a child-centered ubiquitous computing environment (ubicomp) developed for young children to expresses stories. Physical programming is a set of tangible tools and user interaction metaphors for children to control the behaviors of embedded objects in StoryRooms. In this dissertation I describe StoryRooms and physical programming, along with the two studies which showed that kindergarten students had the capacity to understand and use the physical programming approach to control the specialized StoryRooms.

PHYSICAL PROGRAMMING: TOOLS FOR KINDERGARTEN CHILDREN TO AUTHOR PHYSICAL INTERACTIVE ENVIRONMENTS

by

Jaime Montemayor

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2003

Advisory Committee:

Assistant Professor Allison Druin, Chair
Professor James A. Hendler, Co-Chair
Assistant Professor Benjamin B. Bederson
Associate Professor Clyde Kruskal
Professor Emeritus Stanley Bennett

DEDICATION

To my wife Heather and my son Kyle

My best friends

ACKNOWLEDGEMENTS

I would like to thank my advisors, Allison Druin and James Hendler, for guiding me throughout these years at Maryland. One cannot hope for better mentors. Without the tremendous intellectual freedom that they gave me and the faith they have in my work, I would not be here today. I would also like to thank Ben Bederson, Clyde Kruskal, and Stan Bennett for being on my defense committee.

I have had the unbelievable good fortune of working with brilliant people at the University of Maryland. I would like to thank Sean Luke and Jeff Heflin from the old PLUS group for being sounding boards of my too often crazy ideas. I would like to thank Brian Postow for helping me juggle theoretical problems. Houman Alborzi had, from the very beginning, kept me honest in my thinking.

I must give credit to everyone at HCIL, in particular, those that worked closest with me at HCIL-2. My work would not exist were it not for their collaborations and contributions. I need to thank two of the lab's artists-in-residence, Allison Farber and Lisa Sherman, who transformed my ideas into beautiful objects to play with. Sante Simms, who never gave up on a problem, lent me engineering support. Gene Chipman, an

old salt at electrical engineering, developed the critical hardware for my projects. My discussions with Harry Hochheiser and Juan Pablo Hourcade helped center my work around human factors issues. Over these past six years, the dozens of bright young children that I have worked with inspired me and helped me to many fruitful insights.

Finally, I have to thank my family, the most important people of my life. It would be an understatement to say that my wife Heather and my son Kyle have had to make tremendous sacrifices to accomodate my dream of attaining a Ph. D. Despite many ups and downs in my personal life, they stood with me and guided me to this moment. For their love and compassion, I can only say, thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 The Rise of Ubiquitous Computing

Computers are everywhere. They are no longer confined to laboratories or business offices. In 1991, Mark Weiser described a future in which computers would not be confined to the desktop, limited to the interface metaphors of display, mouse, and keyboard [112]. Today, his vision, **ubiquitous computing** (ubicomp), has become a reality in many of our daily lives. Computers take on all shapes and sizes. Cellular telephones allow us to communicate with almost everyone on earth, from almost anywhere. GPS technology in your car provides driving directions and can even plan alternative routes. Thanks to sensors attached to windshields, as we drive along highways, tolls are automatically deducted from our accounts. Public restrooms automatically clean themselves (figure 1.2). Water fountains automatically squirt. Home appliances automatically perform their duties under the X10 protocol (figure 1.1). In business, information seamlessly follows its intended human recipient as she moves about the building [110, 108]. At the Baltimore/Washington International Airport, an experimental SmartPark system, equipped with simple sensors and overhead displays,

Figure 1.1: The X10 system allows the user to customize the behavior of home appliances.

directs drivers to available slots in the parking structure. Computers have moved beyond the business office. They are rapidly becoming embedded into our environment, intricately intertwined into our daily lives.

1.2 Don't Forget the Children

Despite these and many other successes in the development of ubiquitous computing, researchers haven't fared quite as well with respect to children. Take the automatic toilet (figure 1.2) as an example, while mistakes such as premature flushes might irritate adults, the surprise could be terrifying to children [25]. Automatic sinks can be puzzling as well! Where do we place our hands to activate the sensor?

As computers become pervasive in our surroundings, the user base of ubicomp environments naturally will expand to include young children. Systems that interact with this population need to be concerned with their special needs. For example, the very young do not have well developed motor skills in their hands [45], thus are unable to control machines that require finesse. Or, because of their height, children may not be able to reach sensors and activate embedded devices [64].

One way to alleviate the frustrations, uncertainties, and confusions that the ubiquity

Figure 1.2: Sensors trigger automatic toilet flushing.

of computers can trigger is to give the user control over environmental behaviors. But even in the rare instances when users can control an environment, for example, X10-savvy appliances, they are inevitably meant only for adults.

This omission is unnecessary. Children should and can be in control of their interactive environments. One can imagine that the sink sensor is a tangible device and that children can place it exactly where they know it will respond correctly. Here is a more fanciful scenario: A child wants to wake up to music at 8AM, so she takes out two blocks from her "bag-of-tricks." The faces on one block have numbers on them. She arranges it so that the number 8 faces toward her. The faces of another block depict various kinds of alarms: music, buzzer, light, etc. She places this block so that the music faces her. These two blocks are sufficient to empower the child to set her own wake up time. In both examples, the environment conforms to children's needs and allows them to control technology in their own ways.

1.3 Ubiquitous Computing, Children, and Control

From this interplay between technology, children, and control, comes a wealth of research questions. How does a child's physical size affect the accuracy of the system? How do imprecisions in sensors affect a child's expectations of technology? Can tools be created for children to customize the behaviors of their physical environments? And, what are design guidelines for pervasive computing environments, when children are intended (or more seriously, unintended) users?

Among these topics, I focus the work of my dissertation on the tools for children to control the ubicomp environment. This includes the following questions:

1. What kind of tools are needed?
2. What do the tools look like?
3. How are they used?
4. Do the tools require new interaction models?
5. Can children in fact use the tools?
6. Implicitly, the ability to control may require a programming model. What is that model?

Beyond tools, we also need to consider to what extent young children can understand the concept of a computationally enhanced interactive environment.

In order to understand the issues I just raised, I developed a research ubicomp environment, **StoryRooms** [2], from combining storytelling with ubiquitous computing technology.

Definition 1 *A StoryRoom is a room-sized ubiquitous environment that, through interactions between the computational devices and the people within the environment, expresses and provokes a storytelling experience.*

The choice of storytelling was deliberate: It is a compelling topic and enjoyable activity for children. And it was by studying the behaviors and interactions of children within the StoryRoom context that I was able to gain insights into children, control, and the ubicomp environment.

1.4 Stories and Children

Storytelling is pervasive in children's lives. From their earliest memories, they are listeners, writers, drawers, and performers. More than just being the recipient of stories, children have also been given the tools to author, or construct, their own. With crayon and paper, they can draw and write. They can wear costumes and act on the stage. They can even build cardboard fortresses and become knights in make-believe kingdoms. Storytelling can even involve non-traditional elements. For example, children used a physical robot (PETS [28, 66, 67]) to move around and express "emotions" as part of the storytelling experience.

1.5 Stories and Ubiquitous Computing

We can move beyond the single computational device (the robot) to express stories. Physical Interactive Environments [1] (PIE) with interactive objects can also convey

[1]Throughout this dissertation, I will be using four terms to describe physical environments with embedded computational devices. The term ubiquitous computing environment follows Mark Weiser's

stories (e.g., [80]). Although you many not have noticed them, PIEs are not new, nor are they uncommon. Over the past two decades there has been an explosion of new kinds of interactive experiences (in storytelling [80], education [92, 30], kids' play museum [82], and entertainment [90]). Below is a well known story converted into a suggested StoryRoom to give the reader a better understanding:

> You enter a room with two friends. Inside, you find three houses built with cardboard box, colored paper, and paper glue. One house is made to look like it is made out of straw; another, sticks; and the third, bricks. You also notice colorful and squeezable physical icons that look like hands, mouths, and sun rays connected to these houses.
>
> A loudspeaker, embedded inside a mouth icon, utters the voice of a wolf; "I am hungry! I am hungry!" Thinking that you might become the wolf's dinner, you each scamper into separate houses. By turn, from a speaker, just outside each house, you hear "Little pig, little pig, let me come in!" In turn, you say, "No, no, no. Not by the hair on my chiny chin chin!" From the loudspeaker: "Then I will huff, and I will puff, and I will blow your house down!"

You may recognize this as an adaptation of the classic story, *The Three Little Pigs.* Your parents may have read it to you; you may have read it to your children; you may

---

definition. A physical interactive environment can be a ubicomp environment. Or, it can be more conventional, where the devices are not embedded into the environment and still follow the forms and functionalities of the display, screen, and mouse. A story-room is any PIE that expresses a story. And a *StoryRoom* is a ubiquitous computing environment that I developed specifically to study the relationships between children and ubicomp.

have performed it on stage; you may have even seen it as a cartoon show. As this example demonstrates, stories can also be experienced through physical interactive environments.

Story-rooms provide a setting that can be educational, experimental, collaborative and fun; and, they offer a new medium for telling stories, in addition to the traditional expressive forms of writing, drawing, or discourse. StoryRooms encourage children to participate in physical interactive stories. Moreover, these special ubicomp environments encourage children to construct things, to turn abstract concepts into concrete objects, and to collaborate. This constructive process is how children make sense of and refine their mental models of the world [75]; it is one way children learn. With new tools such as sensors and effectors, child authors can add magic to their make-believe stories. It is as if next to the crayons and papers, they suddenly find a magic wand that really works.

Unfortunately, unlike the more traditional storytelling approaches (writing, drawing, acting), there are few, if any, constructive tools for children to create their own stories inside a PIE. Just as adults encourage children to write on paper, draw on canvas, and mold lumps of clay, we should also provide a setting for them to create their own interactive environments.

This presents an opportunity for us to design tools for children to control the interactive behaviors within StoryRooms as part of the storytelling/story-building process. Moreover, the successful tools for the StoryRoom may also lead us to insights into tools for children to control more generalized ubicomp environments.

1.6 Why Technology for Children

Children already create their make-believe worlds out of everyday things such as boxes, blocks, and stuffed animals. Why then should this creative process be interfered with technology? After all, compared to traditional manipulatives, technology can be expensive, fragile, difficult to use, and environmentally unfriendly. But, despite its detractors [3], computational technology does not have to be a detriment [45]. Its repeatability and shareability features imply that these ephemeral worlds can be saved and replayed; shared and constructed across geographically distant locations. Undeniably, any new technology introduced into children's world must not get in their way, must not harm them, or detract from their interactions with others. So computers, like wooden blocks and crayons, are all just tools that can support positive learning experiences. Seymour Papert, in a 2002 talk at the University of Maryland, offered this insight.

> "…well, they [non-technical objects] obviously work well, since we all use them even now. You can have entire projects, theories, models, etc. that can be spelled out on paper. But it seems to me that a dimension is lacking—That with technology, things can work, break, and can be fixed."

Papert was alluding to the idea that as children break and fix things, they become debuggers, problem-solvers, and understand more about the world around them. Let us return to the *Three Little Pigs* StoryRoom example: We can imagine that elementary school aged children could have created the props (the three houses); they could have recorded the sound effects and speeches; but how did they program the room to interact with the visitors? That is, what tools did they use, and what steps did they take, to

create the interaction rules for the story? It turns out that programming systems are excellent candidates for this task.

1.7 A Conceptual Programming Tool

I believe that a toolkit for children to construct a PIE would 1) be minimally abstract; 2) possibly non-textual; 3) operate within the constraints of young children's physical dimensions; and 4) address the technical challenges of ubiquitous computing environments, such as scale, context awareness, gesture recognition, networking, and location tracking [1]. Until recently, the few systems that generate interaction rules in physical interactive environments have been screen-based text or graphics [29, 66]. They were designed to be tools for adults and not for children.

Therefore, one of my research questions became: Can pre-literate children define states and transitions for computational objects in a ubicomp environment? And, perhaps even more appropriate for children, would the programming activities be more natural, concrete, and direct, if the interaction instructions were created from physical manipulation of real objects in the environment. In later chapters I will show that this is indeed possible with a programming-with-example approach [69]. Here is an example task: Every time I step on this rug in my bedroom, I want that desk lamp in the room to turn on. A possible sequence of physical activities might be 1) invoke a programming recorder, 2) step on the rug, 3) turn on the light, and 4) turn off the recorder. By touching objects in the room, I am creating an instruction that relates the rug to the state of the light. Furthermore, to find out if the instruction is correct, all that I need to do is to be inside the room and step on the rug. I call this technique of using physical gestures to indicate programming intentions **physical programming** [65]. This idea

will be developed more fully in Chapter 6. For now, I will use the following definition.

Working Definition 1 *Physical programming is the generation of computer programs by the physical manipulation of computationally augmented (or aware) objects in a ubiquitous computing environment.*

The introduction of the physical programming technique into the StoryRooms environment enables children to create their own interactive stories without any adult help.

1.8 Contributions

In this dissertation I describe a children-centered framework (StoryRooms) to study the relationships among children, ubicomp system, and user control. I further suggest that a well-designed programming metaphor could be a solution.

This dissertation presents the results of my research on providing tools for young children to control ubiquitous computing environments. My contributions can benefit ubiquitous computing, tangible interfaces, and programming systems for novice users. More importantly, through this work, I demonstrate that it is possible for ubiquitous computing environments to conform to children's needs and desires. In order to accomplish this, and with the assistance of an intergenerational design team at the Human-Computer Interaction Lab:

- I significantly fine-tuned the collaborative design practice with children, *cooperative inquiry*;
- I developed a ubiquitous computing framework, **StoryRooms**, to study children's interactions with interactive environments;

- I developed a set of tangible tools for children to control device interactions in ubicomp environments;
- I developed a new programming metaphor, **physical programming**, and demonstrated that this approach is simple for children to understand and to program StoryRooms.

1.9 Organization

I begin this dissertation with a survey of the four research areas that have given me the most insights: end-user programming, ubiquitous computing, participatory design practices, and technology for learners. Next, I devote a chapter on the *cooperative inquiry* design framework. I discuss the need for a children-inclusive methodology, and I also discuss my own contributions to the design process. In chapter four, I present the inspiration and precursor to my work on interactive storytelling environments, a physical and constructive storytelling robot called PETS. In chapter five, I describe the conception, evolution, and development of the StoryRoom concept. I will describe the early prototypes and the lessons I learned along the way. In chapter six, I describe a conceptual toolkit that is required to construct StoryRooms. Having identified the elements of the toolkit, in chapter seven, I describe and discuss the first usability study, with a semi-wizard-of-oz StoryRoom programming prototype, to observe kindergarten children within a StoryRoom environment. In chapter eight, I describe a second study, in which I observed that kindergarten aged children can, independent from adults, create their own fully interactive physical storytelling experiences. In chapter nine, I take a step back and consider the relationship between StoryRooms and automata. In chapter ten, I describe some user interface designs that, while not implemented as part

of my dissertation, reveal intriguing possibilities for the future. I conclude with some final words on my contributions, applications, and potential future directions for this research.

1.10 Definitions and Abbreviations

In this dissertation, I will use the terms and abbreviations listed below.

AT Adult Team. This group, of which I was the leader, was comprised of all the adult members of the intergenerational design team (see below).

ATM Adult Team Member(s).

ATT Adult Technical Team. This was the team of adult members with technical skills such as computer science and engineering. Again, I was the primary leader.

ATTM Adult Technical Team Member(s).

HCIL The Human-Computer Interaction Lab at University of Maryland. This is the research hub for the intergenerational design team.

IDT Intergenerational Design Team [28]. The research team of interdisciplinary adults and elementary school aged children in the HCIL. This team is directed by Allison Druin. I led research sessions related to PETS and the StoryRooms[2].

PETS Personal Electronic Teller of Stories [66].

[2]The composition of the research group varies widely depending on the tasks at hand. To be clear, I have tried to indicate the primary responsible group of researchers when I can. For instance, when I write IDT, I mean that the entire intergenerational team was involved. When I write AT, I mean that all the adults contributed.

Physical Programming The generation of computer programs by the physical manipulation of computationally augmented (or aware) objects in a ubiquitous computing environment [65].

PIE Physical Interactive Environment.

Entities Computational objects and human users within a ubiquitous computing system.

StoryKit A construction kit of low-tech and high-tech elements for children to build StoryRooms [2].

StoryRoom A room-sized ubiquitous environment that, through interactions between the computational devices and the people within the environment, expresses and provokes a storytelling experience to the user [2].

Ubicomp Ubiquitous Computing [112].

Chapter 2

Related Work

As I stated in the Introduction, my research goal is to develop a child usable programming tool to construct interaction rules in StoryRooms. Four research areas heavily influenced my work:

1. Technologies that interact with physical environments;
2. Programming environments for novice users;
3. Technology for learners;
4. Participatory design practices.

Each field is important relative to my work. I am developing a programming tool for young children (2) to control interactions in physical interactive environments (1) called StoryRooms (3), using participatory design techniques (4). Below, I will describe each area in more detail and discuss their relationships to my research.

2.1 Technologies that Interact with Physical Environments

Physical interactive environments (PIEs), enhanced with computational devices, are all around us. They can be museum installations, petting zoos, and amusement parks. From as early as the 1960s, institutions such as the Exploratorium in San Francisco have been exploring ways for visitors to learn about scientific and mathematical concepts through physically interactive experiences [92]. Many others enable children to explore such varied subjects such as music, at the Eloise W. Martin Center in Chicago, Illinois, and animals, at a working farm, the Macomber Farm in Framingham, Massachusetts [30]. Projects such as NYU's Immersive Environments [29], MIT's KidsRoom [13, 80], and University of Maryland's StoryRooms project [2, 65], explore the expressiveness of PIES for storytelling.

The enabling technology of many recent PIES have come from ubiquitous computing [112], augmented reality [56], tangible bits [50] and graspable user interfaces [33]. The development of direct interactions with real objects comes from a shared belief among these researchers that people are more adept at, and comfortable with, manipulating everyday objects in their natural settings. These technologies also share difficult technical challenges, such as scale, context awareness, gesture recognition, networking, and location tracking, and software infrastructure [87, 1].

Until only a few years ago, little research has focused on user interfaces to control PIEs. The field has a promising future. My work on StoryRooms and physical programming [2, 65] directly addresses this area, by enabling novices users an approach to physically and directly manipulate objects to create their personal settings. In addition, Phidgets [42] and iStuff [7] are both physical interface constructors; currently

these two systems still require the user to revert to the computer screen for programming activities. XWand [114] is functionally similar to the magic wand in the physical programming user interface. An X10 [118] enabled system allows the user to control home appliances. A user can directly control appliances by manipulating dials on custom X10 devices attached to the appliances. For more complex tasks, the user often must refer to computer-based programs.

2.1.1 Ubiquitous Computing

Until the late 1980's, human-computer interaction (HCI) researchers have been predominantly concerned with issues surrounding the desktop computer. The possibility that computers would eventually become embedded into our physical surroundings and support our activities was first outlined by Mark Weiser. In ubiquitous computing environments, computers surround, but do not intrude on us. Seamlessly integrated into our lives, they become *effectively* invisible:

> "...such a disappearance is a fundamental consequence not of technology, but of human psychology. Whenever people learn something sufficiently well, they cease to be aware of it." [112]

Ubiquitous computing systems share at least three attributes: a) a set of computing devices (possibly heterogenous), b) a set of supported tasks, and c) an infrastructure such as network and location service [87, 118]. They also share two fundamental technological issues that remain difficult to solve: scale and location. Computational devices in ubicomp systems can number in the hundreds or even thousands, and can vary in size from as small as a post-it note to as large as a large wall-sized display. The problem of scale requires infrastructure such as networking protocols to manage the large

numbers of wireless and mobile devices as well as software to support new interaction models [1, 8, 109, 112, 113]. Examples of software needs include self-describing data structures and robust behavior under questionable connectivity conditions. The location problem is due primarily to the device and user movements. To date, but for a few demonstration systems (e. g., [108, 111]), there are still no commercially available local area position tracking systems that can track entities at a resolution on the order of a few centimeters.

A large subfield, context-aware computing, addresses problems related to the frequent contextual changes in a highly mobile and unpredictable environment [1, 91, 110]; these include error-prone recognition, context fusion (how to decipher context events such as who,what,when, where, and why) [1].

Contextual problems have since been broadened to include social aspects of human-computer interaction [9]. Ubiquitous computing environments' inherent physicality, sensor and actuator imprecisions, bring forth new sets of problems that are different from the traditional (and highly controlled) desktop computing environment. Some questions concerning these issues include [9]:

1. How does a system know when I am addressing it?
2. How do I know a system is doing what I commanded it to do?
3. How does a system know the parameters of my command?
4. How do I know the system correctly understands my command, and is correctly executing it?
5. How do I recover from mistakes?

Shafer et al. further identified several important distinctions between desktop and ubiquitous computing systems [93]:

1. Multimodal interaction;
2. Physically embodied interaction;
3. Dynamic set of devices;
4. Lack of a single focal point;
5. Multiple simultaneous users.

While these issues have been well studied (or are not relevant, for example, physical embodied interaction) in the GUI environment, research targeted to ubicomp is just emerging. An example is input devices designed for physical spaces. The Pebbles project [70] shows how multiple users employing multiple interaction modes, can control multiple available devices. iStuff [7] allows users to easily connect physical user interface input/output elements to applications in the environment. Phidgets [42], like iStuff, is a physical interface construction kit. X10 [118] is a home automation protocol for the user to control appliances, where signals travel through the AC power lines or RF channels. As far as I know, all of the above currently rely on the WIMP interface to establish sensor/actuator relationships, and have not expanded to using the physical objects themselves to help create the relationships.

Because StoryRooms is a ubiquitous computing system, it faces the same general technical issues as other ubicomp environments. But some problems directly impact the StoryRoom functionality. For example, although this was not an issue for our usability studies with young children, the lack of a high-precision local positioning system prevents the StoryRoom from allowing more sophisticated physical gestures, which

might be useful in enhancing the physical programming language. On the other hand, the advent of many new physical input devices, when combined with careful children-centered design, could further enable them to control the environments. Another limit in the current StoryRoom is the dimensions of physical icons, which are currently still too large. On-going efforts in this field to miniaturize complex sensors and actuators will directly benefit my work.

2.1.2 Augmented Reality, Tangible and Graspable User Interfaces

Researchers of augmented reality append digital and communication abilities onto everyday objects. (A common approach is visual overlay of digital information onto real-world objects.) Users take advantage of their familiarity with the natural affordances of physical things, such as paper and eraser, to accomplish everyday tasks as well as digital operations. There are three ways to augment real objects: 1) augment the users (wearable computing), 2) augment the physical object, and 3) augment the environment surrounding the user and the objects. Many augmented reality and ubicomp systems employ a combination of these three approaches [57].

Augment the Users

The user wears or carries devices to sense virtual information about artificial or real objects. For example, when a user reads a MagicBook through special stereoscopic glasses, drawings on the pages rise into 3-dimensional shapes and invite further exploration [11]. Because of the large physical space, many children, and unpredictable movements within a StoryRoom, this approach would not be appropriate. The eyewear would be difficult for children to wear and to traverse through the room.

On the other hand, a hand-held “magical lens,” made from a small LCD screen and a camera, could reveal the identity of an embedded device, whether for debugging purposes during programming, or for revealing mystery notes during play mode. Some museum self-guided devices show this to be a promising approach (e. g. [115]).

Augment the Physical Object

In this approach, physical objects are modified by the embedding of input, output and computational devices on or within it. In StoryRooms [2], children append sensors and actuators (embedded within physical icons) onto low-tech props. The icons concretely indicate the props’ augmented features, and allow children to quickly create interactive environments by combining simple materials with high-tech devices. The DataTiles system [84] uses physical see-through tiles, embedded with RFID tags, to store virtual data such as baseball card collection and weather maps. This is an excellent interface, but unrealistic for the StoryRoom application, as each physical object must contain a display screen. The Listen Reader [6] augments a real children’s storybook with ambient music and sound effects, so that children can enjoy the physicality of the book as well as manipulate music by moving their hands over the surface of the pages. StoryRoom can be thought of as a room-sized version of the Listen Reader, but with the additional power of allowing the user to create new “storybooks.” TICLE, a vision based system encourages learning about geometry from playing with physical Tangram puzzles [89]. Because it requires computer vision and a stationary desktop, its interfaces are inappropriate for StoryRooms.

Augment the Environment

In the final approach, neither the user nor the object is affected directly. Instead, independent devices provide and collect information from the surrounding environment, displaying information onto objects and capturing information about the user's interactions with them. The KidsRoom [13] relies on a vision-based tracking system to monitor children's locations and their body gestures, while sensory outputs, in the forms of sounds and projected images, provide feedback. Hand gestures remain a popular input approach (e. g., [72, 86]).

This is a difficult model for the StoryRoom. First, it is difficult to interpret imprecise sensor data and correctly infer user intentions. Second, the underlying technology can be expensive, difficult to set up, or sensitive to environmental conditions (such as light, line-of-sight, sound).

Tangible and Graspable Interfaces

Physical objects can be closely coupled with digital data structures. This connection is either called tangible bits [50] or graspable user interfaces [33]; and it extends direct manipulation [94] to real objects, such that operations on physical things modify digital data. Environments that support tangible user interfaces generally require three components: 1) interactive surfaces, 2) coupling of virtual data bits with graspable physical objects, and 3) background awareness [50].

The usefulness of this interface technique suggests the viability of a physical approach to programming. For example, The AutoHan project and its MediaCubes tangible programming approach demonstrate an intriguing system for controlling the home environment [12]. Although not strictly a ubicomp system, TellTale, a phys-

ical worm whose body segments can record and play back children's oral stories [5], demonstrates a physical approach to storytelling. Physically connectable blocks are popular interfaces for systems that teach programming concepts to children (e.g., [103, 116, 61]). But while the physical connections represent programs, these systems generally support scientific discovery, and not storytelling. A useful tool to program StoryRooms would be a combination of the programming strengths of the blocks with the storytelling focus of the worm.

2.2 Programming Systems for Novice Users

Programming is an act of communication. When we express ourselves, whether to a machine or to a human, our language can range from the concrete, such as pictorial representations of real objects, to the abstract, such as the English language. The language may even involve gesture, such as in American Sign Language. It should be apparent that not one single communication channel is appropriate for everyone and for all occasions. Because I am interested in a language that is suitable for young children (early elementary school and kindergarten students) to control devices within ubicomp environments, the communication model needs to be non-textual, minimally abstract, and needs to support physical activity. The language needs to be non-textual because many kindergarteners are pre-literate. It should not require abstract ideas, since that ability appears to arrive later in a child's cognitive development [78]. Finally, the language should support physical activity due to young children's need for physical movement.

2.2.1 Non-textual language

> Visual communication is basic.
> However, without well-defined interpretive procedures,
> they are usually ambiguous [105].

Pictures (without text) can serve in a wide variety of applications. Symbols on highway signs, Olympic sport figurines and more are socially accepted and can invoke universally agreed upon interpretations in the reader. For example, icons in the WIMP user interface signify the underlying data structures stored inside a computer. An icon of a folder on the computer screen could evoke the functionality of a real folder inside your office drawer: to hold things.

Pictures can be easy to understand, but a language of pictures[1] is limited. First, these languages do not scale up, because they do not include syntactic rules. Second, images cannot represent everything. Some ideas are inherently abstract, and it is difficult to (without meta-reasoning rules) convey them adequately. Here is an example: What is a picture that communicates "yesterday?"

By combining pictures and rules for drawing them, a pictorial language becomes far more expressive. The Elephant's Memory [49] is one notable example. It has only a small set of about one-hundred combinable "signs, or logograms." Despite the small number of primitives, it can generate more than just concrete ideas. By various combinations and relative positions of the symbols, the resultant pictures can represent and convey highly abstract ideas.

[1] Here I mean the pictures and images such as those in Modley's Handbook of Pictorial Symbols [63]. I do not include "squiggles" that are strictly included as syntax to affect the interpretations of the pictures.

Language does not have to involve imprints on two dimensional surfaces. Physical gestures have been a form of human communication in many cultures (for example, American Sign Language [ASL]). Sign language can be as rich as any written natural languages. Signing is fascinating to watch. At once, you might observe a gesture that immediately reminds you of a notion (for the idea "monster," you raise your arms and act like a monster); then you are seeing the "spelling" of a word (if the language has an underlying textual language); or you see a motion that is purely symbolic (e.g., "family," "father," and "grandfather" in ASL). Physical gestures can be highly expressive, visually understandable, and easy to perform.

These observations suggest the following. First, a small set of pictures, combined with learnable syntactic rules, can be highly expressive. Second, the pictures can be physical, as in physical icons. Third, physical gestures can be the syntactic operators to the physical icons. Interestingly, the physical programming approach in StoryRooms [65], a result of several years of development with young children, share many of these qualities.

2.2.2 Visual Programming Models

I have just suggested that physical icons and physical syntax can be a language. Now let me take a step back and look at the relationship between pictures and programming. This is generally referred to as visual programming.

No uniform definition exists for the term "Visual Programming." Myers describes it as

> "…any system that allows the user to specify a program in a two- (or more) -dimensional fashion…conventional textual languages are not considered two-dimensional since the compilers or interpreters process them

as long, one-dimensional streams." [69]

Shu defines visual programming languages (VPL) as languages that use

> "some visual representations (in addition to or in place of words and numbers) to accomplish what would otherwise have to be written in a traditional one-dimensional programming language... the language itself must employ some meaningful... visual expressions as a means of programming." [95]

Burnett and others wrote,

> "Visual programming languages let the programmer sketch, point at, or demonstrate data relationships or transformations, rather than translate them into sequences of commands, pointers, and abstract symbols." [19]

These definitions have two ideas in common: 1) the syntax of a VPL should contain elements that can only be expressed through multiple dimensions, and 2) the program is expressed in a visual way, and not only as text.

Similarly, a physical programming language should contain elements that can only be expressed through gestures (analogous to actions such as sketching and pointing on a desktop computer) in the physical space, and that the program can be created and experienced in a physical way.

Green's classification [38] of visual programming languages is succinct and often cited[2]. They are: 1) flowcharts, 2) data-flow, 3) visual production, 4) logic-based,

[2]There are many taxonomies for visual programming languages and environments. A recent survey on programming environments and languages for novice programmers is by Kelleher and Pausch [52].

and 5) spreadsheet. In the next section is a summary of the different programming models.

Control-flow, or Flowcharts

Visual programming began with attempts to make flowcharts executable, this was led by the belief that flowcharts are useful teaching tools for training novice programmers [38]. But, the introduction of new classes of users, such as workers, tinkers and programmers [71], requires different levels of programming skills, and flowcharting, a conceptual abstraction became inappropriate for some. Furthermore, this model declined in popularity, especially when a controlled experiment showed that graphical representations were not better than text [20]. One of the earliest and most enduring criticism of visual programming languages, which stems from the flowchart based systems, is the scaling-up problem [19]. That is, for small problems, flow charts adequately represented programs, but they quickly become a jumbled mess with an increase in the program size. An interesting example is the "static representation" problem [19]. The extra dimensions in visual languages can support dynamic activities (e. g., programming-by-demonstration). But as the visual program grows, the activities could overlap and cause confusions.

Data-flow

In this model, data travels from input nodes, to operators, and leave from output nodes. An operator executes as soon as all its input nodes have been filled. Graphical representation of flow of control, such as iteration, is difficult. Commercial products such as LabVIEW and Prograph both offer different graphical syntaxes to address this is-

sue. Some research have shown that data-flow based languages are better for novice programmers [4, 44], But others doubt this claim [73].

Visual Production Systems

These are similar to textual production systems. The productions are rules with a left side "picture" and a right side "picture." When a situation occurs in the visual world, and the situation matches the left side, then the rule fires and the world is redrawn and transforms into the scenario dictated by the right side. KidSim[3] is one example [21].

Constraint, or Logic-Based

One common constraint operation in text-based editors is search-and-replace. That is, find all occurrences of "bat" and replace them with "cat." CHIMERA [53], a 2D object-based illustration system, shows how this feature can be implemented in a higher dimension environment. So, one example may be, "find all squares that are blue, and replace their color to bright red." The query constraints can be on a similarity metric based on location, shape, and graphical properties such as line width, and color. ToonTalk [51], shows a solution to the related problem of generalization. In this system, generalization is accomplished by the removal of constraints in default computations.

Spreadsheet

The common spreadsheet supports many of the qualities of an ideal visual programming language. For example, activities within its cells are based on data-flow, and, the

[3]This product is now called StageCast Creator.

worksheet supports direct manipulation.

2.2.3 Novice User Programming Systems

Most prior work on novice user programming systems have been focused on the traditional desktop computer model (see [52] for a survey of novice programming environments). Papert's mechanical turtle [75], the Curlybot [35], AlgoBlock [103], Electronic Blocks [117], and Tim McNerney's Tangible Computation Bricks [61] , are few rare examples of programming systems that incorporate tangible manipulation of real physical objects. Despite the difference (2 dimensional screen versus 3 dimensional physical space), research in visual programming languages, in particular, programming-with-example systems, can offer useful insights into physical programming issues.

If traditional text based programming languages have been powerful and useful for creating technology, why tinker with a good thing, and devise new programming metaphors? The motivations for the research behind VPLs is the belief that "a picture is worth a thousand words," or that extra dimensions can express more clearly, concisely, and easily the semantics of a program [19]. Data can be represented in two ways, analogical and Fregean (symbolic). For example, a picture of a bicycle is an analogical representation of the bicycle object. Whereas the word *bicycle* is its Fregean representation [96]. Concepts such as yesterday, or hungry, have no analogical representation, and must be represented by symbols.

Researchers have also known for a long time that programming is not an intuitive skill, and that a good visual programming language can be an effective pedagogical tool, so that computer science students can learn the art of computing more easily. A good

VPL should be easy to write, easy to understand, easy to debug, easy to learn, and easy to maintain. Programming languages can be difficult also because of the *blank-canvas syndrome*. Textual programming languages are abstract, non-interactive, and Fregean. "What is needed is a lightweight, non-threatening medium like the back of a napkin, wherein one can sketch and play with ideas" [97].

These are ambitious goals. Indeed, VPL is not without its detractors (e.g., [15]). Moreover, even within the VPL community, there is as yet no definitive empirical research that shows that VPL is better than text based programming languages. (A programming tool that is more closely constructed to solve problems within a domain space is just more likely to perform better than that tool which is not [39]). However, it is clear that programming systems for pre-literate children: 1) require non-text based interactions, and 2) need to minimize abstractions by providing concrete and direct manipulation of program elements. Physical programming is not necessarily better than other approaches, but it may be more appropriate for the younger population.

2.2.4 Programming with Example and Programming by Example

The concept of example based programming, or programming by example (PBE), was first introduced in Pygmalion [97]. A subset of visual languages, these systems have the clearly defined goal of providing end user programming. Allen Cypher explains, "...these techniques need not be programming per se: rather they need to achieve effects that can currently only be achieved through programming." In his view, end user programming can be: preference, scripting languages, macro recorder, and programming by demonstration. Myers makes a distinction between *programming-by-example* (PBE) and *programming-with-example* (PWE) [69]. In the former case, the challenge

is for the system to infer the user's intent, by observing her activities. For example, the programmer might want to demonstrate a concrete example in order for the system to create abstractions [53]. This is called the generalization problem. In the second case, the programmer specifically dictates to the system her intents for future reuse. This may also be thought of as programming within the user interface.

Two well known PWE systems for children are KidSim/StageCast Creator and ToonTalk. KidSim allows the programmer to define visual production rules, through comic strip like picture frames [21]. In ToonTalk [51], computational abstractions are replaced by concrete and familiar objects. A ToonTalk program is a *city* that contains *houses*. *Birds* fly between houses to transport messages. Houses contain *robots* that can be trained to accomplish small tasks. To program a robot, the programmer enters into its thought bubble to show it what to do. These two languages fill a void in the programming languages spectrum. They offer children ways to explore in microworlds.

The StoryRoom language is a programming-with-example system. There is clear advantage to defining interaction rules of objects in a 3-dimensional space while being inside the same dimension. For example, it is clearer that physical actions in the room can have direct physical implications. Contrast this with keyboard actions having a symbolic link into 3-dimensional objects residing in 3-dimensional space. This abstraction may be difficult for children.

2.2.5 Programming by Tangible Interactions

Little prior work exists on physical programming. Papert's mechanical turtle [75] helped children learn programming in LOGO. More recently, Curlybot [35] is another robot that encourages learning mathematical concepts from physical play. In a

modified version of PETS [28], children with physical disabilities generate physical movements for the robot to remember and reenact [81]. Physical blocks are popular tangible interface elements. McNerney's Tangible Computation Bricks [61] allow programmers to manipulate and connect physical action blocks that can react to sensor inputs. AlgoBlock [103], an educational tool for older elementary to junior high school students, is a collection of physical blocks, each of which represents a command in a LOGO-like language. The output of the program is still revealed on a display screen. Eletronic Blocks' [117] are special purpose sensor, action, and logic stackable blocks for preschool children. These systems control the behavior of the tangible devices in their environment. A shortcoming of these blocks-based systems is that they are all about the blocks themselves. In contrast, StoryRoom objects bridge the interactions between the human and the physical environment [2, 65].

2.2.6 The Scaling Up Problem

Visual languages are not immune to the scaling up problem [19]. This discipline suffers in two ways. Ideal visual languages allow programmers to point at, sketch, or demonstrate data relationships or transformations, rather than translate them into sequences of commands. These different ways of expressing program syntax and semantics lie at the heart of the fundamental problem of visual programming languages: attempts to make them usable for large scale problems often require the reintroduction of the complexities that they were supposed to simplify. This is the scaling up problem. The second scaling-up problem is the limited domain in which current visual languages have been successfully applied. Researchers are looking for ways to make VPL general purpose (e.g., [51]).

At this time, physical programming appears to suffer from the scaling up problem as well. As I will show in 6.1, physical programming is used to create transition rules of a state machine. The complexity of the state machine grows rapidly with the size of its alphabet and set of states. This means that the number of transition rules cam grow rapidly too.

2.3 Technology for Learners

I hear, and I forget.
I see, and I remember.
I do, and I understand.

—Confucius

Children learn by playing with blocks, drawing on paper, and building make-believe worlds. When Friedrich Froebel developed the kindergarten in the 1830s, he began a tradition of teaching that encourages self-learning, discovery, and personal expression [16]. Froebel's teaching material (*gifts*), objects such as wooden blocks of crystalline structures, balls, strings, and sticks, were play things for children to explore shapes, symmetries, and other mathematical concepts. This learning philosophy is also supported by the constructivist theories of Jean Piaget and Papert's theory of "Constructionism." Papert asserts that learning is an active process, in which people actively construct knowledge from their experiences in the world [76]. This process of constructing one's personal mental structure is called "Piagetian learning" [75]. People don't get ideas; they make them. Furthermore, the most effective learning occurs when they construct objects that help make sense of their internal mental models of the world. Jerome Bruner offers a similar perspective, that people think in three ways: 1) enactive, doing things to think; 2) iconic, thinking with pictures; and 3) symbolic, thinking with abstract symbols [18].

With few exceptions, computational technology introduced into classrooms, particularly in kindergartens and elementary schools, have not been completely successful in encouraging this type of active exploration, self-learning, and collaboration [106].

In large part, due to the constraints of their mice and keyboards, when coupled with poorly designed software, can actually inhibit Piagetian learning.

Recognizing both children's innate abilities and the potential afforded by new technology, researchers began looking for ways to develop new computationally enhanced environments to encourage self-learning. In particular, Seymour Papert and Mitchell Resnick, at MIT, influenced by Jean Piaget, became proponents of "computational objects to think with" [75, 85]. Learners rely on personally identifiable objects to generates ideas, and make sense of them, in their minds. A system that can be individually molded into meaningful objects enables children to learn from the construction of their personal *objects-to-think-with*. Some of the more successful systems include the Logo programming language [75] and LEGO's Mindstorms Robotic Invention System [59]. These are programming systems that encourage children to learn scientific concepts.

Next to these systems that support scientific inquiry are storytelling technology. Stories preserve our cultures and histories, enable us to communicate our ideas and feelings, and educate learners of all ages [17, 37, 74]. With this understanding, many researchers have been developing systems for children to explore novel storytelling approaches. These include SAGE (Storyteller Agent Generation Environment [107], PETS (Personal Electronic Teller of Stories) [28], and Microsoft's Actimate Barney [100]. At the University of Maryland, I developed the StoryRoom and physical programming systems for children to construct room-sized physical storytelling environments [2, 65]. Unlike other storytelling systems, StoryRoom explicitly encourages children to construct their personal physical objects, using common materials such as paper, crayon, box, and tape, as part of the storytelling experience.

Most construction kits are virtual programming environments for children to construct microworlds [21, 75]. These virtual worlds have also been used as testbeds for re-

searchers to probe the extent to which six to eight year old children understand programming and rules (e.g., [48]). PETS [28], StoryRooms [2], and TellTale [5] are a few systems that encourage children to manipulate or use physical objects as part of the storytelling experience.

2.4 Participatory Design, Methods and Processes

The design process is inherently fluid and dynamic, and since no formal methods exist that guarantee insights or breakthroughs, a practice that encourages collaboration and the spontaneous outbursts of creativity can affect the quality of the work place. This design practice is called *cooperative design* in Scandinavia (e.g., [101]), *participatory design* in the United States (e.g., [40]), and *consensus participation* in England [68]. They share the common belief that user participation is necessary to create technology that attends to the idiosyncrasies of different work environments. Furthermore, whereas in the past there was a gulf between the "know" and the "know-nots," or, technologists versus users, these researchers believed that users needed to fully and actively engaged in the process, and not be regarded as token participants [41].

Research on the relationship between children and technology had been sporadic and appeared in related but distinct disciplines. Educators and child psychologists discuss the learning impacts from interactions between children and technology (e.g., [75, 102]). In the HCI community, the first publication related to children's issues was Tom Malone's study of games for children [58]. Children and technology became a significant research topic in the early 1990s (e.g., [77, 99]). At the same time, children's roles in the design process were identified as user, informant [88], and design partner [23, 24]. Elementary-school-aged students have been the subject of many

of these studies. Researchers at the University of Maryland have also worked with kindergarteners as designers and as informants [32, 64]. In the chapter that follows I will discuss these methods and their context.

Chapter 3

Cooperative Inquiry: A Participatory Design Framework for Collaborating with Children

When people discuss the design processes, they often refer solely to the end product of that process, the technology. For me, my design goal was a new kind of educational technology, one that incorporated many different constructive and collaborative learning experiences. I am also interested in the development process and the learning experience that came from building and studying technology. Many researchers have referred to this type of learning as the outcomes of *cooperative* or *participatory design* process [31, 40, 68]. Educators [55] have also call it a community of practice. Druin described this as "... a community of people with different skills that learn as they work toward shared goals [23]." At the University of Maryland, we developed a methodology that embraces this close collaboration between children and adults, ***co-operative inquiry*** [23, 24]. To best understand how my dissertation research evolved, one must understand my experiences in the design team of children and adults.

I worked with two groups of children: elementary school aged students from seven to eleven years old, and kindergarten students (4-6 years old). Although they were

close in age, my approach to working with the two groups was quite different. There were many differences between the two age groups. Some were obvious, such as the fact that most kindergarten children were pre-literate. Other differences were more subtle. For example, if a kindergartener proposes a good idea, then magically every student also would have the same one too. This was not true for the older children, who took pride in creating novel ideas. In the sections that follow, I will describe the composition of the intergenerational team I worked with. Then, I will describe my roles in the group. I will also present some common questions about the IDT and its design approach. Then I will describe specific activities that I have found to be effective for working with the primary school and the kindergarten children.

3.1 The Collaboration Between Children and Researchers: An Intergenerational Design Team

Our research team, the intergenerational design team (IDT), has always had at least twelve members. Between six to eight members were between seven and eleven years of age, and came from local (public and private) elementary schools. These children stayed with the team for an extended term, for an average of 2 years to as much as 5 years. The adults were undergraduate students, graduate students, and faculty, from diverse disciplines such as art, education, engineering, and computer science. We shared a common goal: to understand why children were interested in and wished to play with new and existing technologies. This investigation led us to develop a variety of prototypes (e.g., [28, 2, 46, 47]) and to the development of the principles behind

cooperative inquiry.

3.2 My Role in the Intergenerational Design Team

I was a member of the intergenerational design team ever since its inception at the Human-Computer Interaction Lab, in 1998. My role of being a **design partner** has been the one consistent activity throughout my projects: PETS, StoryRooms, and physical programming. I have taken on other roles too. I evolved from being a student of the *cooperative inquiry* design framework to one of its main contributors. Throughout the many design sessions, I observed and analyzed the many specific and practical activities. These became a foundation set of standard operating procedures for our research sessions (section 3.5).

As a computer scientist, I have always contributed my technical abilities to the team. In particular, my strength in rapidly creating working prototypes for the team allowed us to have concrete objects-to-talk-with.

I was also a mentor to many undergraduate students who came through the IDT program. Whether they came from mechanical engineering, computer science, or children's technology, I challenged them to perform to the best of their abilities and at the same time learn the nuances of having children as partners.

In the beginning, because PETS was primarily an experimental project to understand and solidify the *cooperative inquiry* methodology, I found myself in several roles at once. I was a student of the technique. I learned to be with children. I facilitated design sessions. I learned to ask the right questions. And, I designed the enabling technology for the robot. Having many roles at once has been the normal mode throughout my six

year tenure. The first year was my transformation from a computer science specialist to a human-centered generalist.

When I began my dissertation work on StoryRooms and physical programming, I was already much more comfortable as an adult design partner. So I devoted much more time to observing the sessions, thinking about what happened, what went wrong, and what was brilliant. It was during this time that Druin and I discovered that one of the strengths of our methodology was that it created an atmosphere where the **elaboration** of brainstorming ideas occur with remarkable frequency [24].

I think it is safe to say that I have made many contributions to the IDT over the years. With certainty, I know this would not have happened had I not learned from the individuals in the group as well.

3.3 Working with Children

It is clear to many people that a successful design team should include people with many different specialties. For example, a mechanical engineer could design and build the physical structure of a tangible plaything. Of course, given enough time, anyone might be able to do this. And he (the non-engineer) would surely gain a valuable learning experience. While this may be an enlightening learning process, it would probably not be a very efficient way to develop products. Similarly, we can see how other experts would be integral to the design team. The educator would guide us and shows us ways to collaborate with children. The participatory design practitioner would offer a design framework. And, the artist would transform “hard,” “cold,” and “ugly” machines into beautiful functional friendly objects. It might even be acceptable to include children in the limited capacities of testers or users.

But it was not obvious to observers of the IDT research why children had "so much control" on our team. After all, many would suggest that it is difficult enough to have a good working team of adults; there would be no advantages to making the development process more complicated by including young people who (presumably) don't know as much as grown-ups. Visitors to HCIL have often asked the following questions: "How do you work with children?" "How do you select the 'right' child for the group?" "Won't they slow down the development process?" "Why do we need them? After all, we were all young once, so we should know what and how children think." The common underlying thread seemed to be this: Can children be effective partners, and, are they qualified? Having spent these past six years with the young designers, I believe the answer is decidedly affirmative.

3.4 How to Work with Children

Without a doubt, it is difficult to work with children to design technology. There are many reasons why this is the case. For instance, young children have more difficulties verbalizing their thoughts than adults. This is particularly true when they want to convey abstract ideas [78, 79]. So unless adults acquire the skills to properly communicate with them, it would be difficult to involve them in development efforts. In addition, while there has been immense amount of research into communication among adults of varying skills, it has only been in the past decade that researchers have come to understand how to work with children. Also, traditional types of relationships between adult and child, such as speaker-listener, parent-child, or instructor-follower, are not always useful in collaborative settings. Finally, misconceptions about the proper roles of adults and children can often lead to frustration. For example, children are not "just

short adults," implying they are equally capable of any task that an adult can handle. A seven-year old child should never operate heavy machinery. Alternatively, just because a child utters a statement does not necessarily mean that it must be followed. The child may be an expert on being a child, but she would not always know what is in the best interest of young people in certain situations. In short, children are an entirely different user population with their own culture, norms, and complexities [10], and they should be treated as people with special knowledge about the subject of "being child;" just as engineers are people who know a great deal about building things.

3.4.1 Won't They Slow Down the Design Process?

Yes, children can slow down the design process. Products can take longer to build. This is especially true for a new team. My experience has been that it takes about six months before an intergenerational team becomes productive [2]. Also, since there are different forms (i. e. verbal, drawing, writing, building) of communication preferred by each child, the adults need to recognize that the same conversation might be repeated several times, using these different channels. This extra effort translates to more time. Alternatively, the design process can slow down just because children can offer insights for better technology, resulting in new features and ideas that were not part of the original design goals.

But consider what might happen if adults try to build technology without incorporating children's insights. Here is a hypothetical scenario:

1. An adult has a "great" idea for new technology for children.

2. The adult builds it.

3. The adult shows the gadget to kids and asks, “What do you think?” And, “What should I do to make it better for you?”

4. The kids look at the adult and mumbles “I dunno,” or “I don’t understand,” or “I don’t like it.”

5. The adult tries to rephrase “What do you think?”

6. Kids mumble and walk away.

Does this sound familiar? What happened? I believe this gulf in communication is the **phenomenon of two monologues**. What appears to be an interaction, or dialog, is really just the two sides trying to guess at what the other is really saying. There is no transfer, or flow, of information. Let us look at steps 3) through 6) again in table 3.1.

Table 3.1: Phenomenon of two monologues

Adult’s Perspective	Child’s Perspective
Adult shows the gadget to kids and asks, “What do you think?” And, “What should I do to make it better for you?”	Information overload! What’s all that stuff? What’s going on? Why are these things purple? Why is that thing round?
Kids look at grown-up and mumbles “I dunno,” or “I don’t understand,” or “I don’t like it.”	Just say anything.
Adult tries to rephrase “What do you think?”	Child thinks, “Hmm. I don’t like purple. Therefore, I don’t like this thing.” Child says “I dunno?”
Kids mumble and walk away.	Who was that?

The problem is that children have no context in which to understand the object presented before them, and, grown-ups have no context in understanding how children comprehend their environment. Consequently, when all a child sees is the technology,

she cannot understand the rationale behind its features. Now, the adult might try to decipher the ever so deep "I don't like it." But, he may very easily misinterpret the child, because that is just not enough information to work with. And if the adult does misinterpret his observations, and if he then tries to create a new iteration based on this false interpretation, then the technology may stray from its intended objective.

3.5 Working with Children: Standard Operating Procedures

Druin describes *cooperative inquiry* as a "philosophy and approach to research that can be used to gather data, develop prototypes, and forge new research directions [24]." It is composed of three types of activities [24]:

Contextual Inquiry Observe what children do with what technologies they currently have.

Participatory Design Hear what children have to say directly by collaborating on the development of "low tech" prototypes.

Technology Immersion Observe what children do with extraordinary amounts of technology (similar to what they might have in the future).

Within this framework are three iterative events: 1) setting expectations, 2) brainstorming, and 3) reflecting on the session (figure 3.1). In this section I present a classification of the types of activities that I have found to be effective for many frequently occurring situations in the IDT research lab. Consider this the adult team members' list of standard operating procedures. Different activity patterns dominate each event phase.

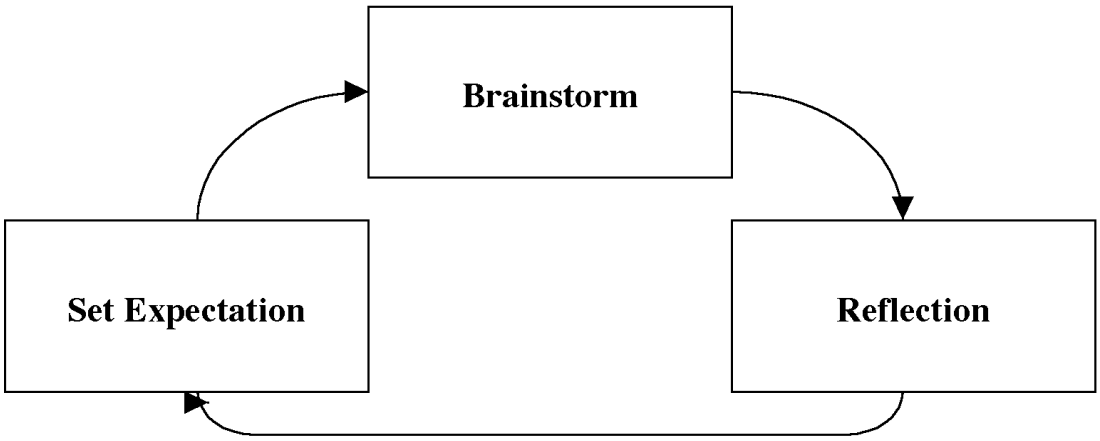

Figure 3.1: The three iterative activities of the *cooperative inquiry* design methodology.

The expectations and reflection phases are unlike the brainstorming phase in that the research team adheres to a standard set of questions and activities for every session. On the other hand, the brainstorming phase is intrinsically very different because the dynamics of each session can vary so much that there is really no fixed recipe. Instead, I offer a "trouble-shooting" chart that contains a list of common issues and some exercises to address these situations. Since the primary brainstorming goal of *cooperative inquiry* is to foster **elaboration** [2], many of these exercises can make elaboration happen more frequently, easier, and faster.

3.5.1 Activities for the Setting Expectations Phase

Adult team members set expectations during two distinct activities: adult debriefing and snack time. This order is important. Adult debriefing occurs after every research session, and includes only adults. They review and analyze the session, and agree on a goal for the next session. Then, during the snack time of the new session, the adults introduce the day's design goal to the children (figure 3.2), whether it be brainstorming

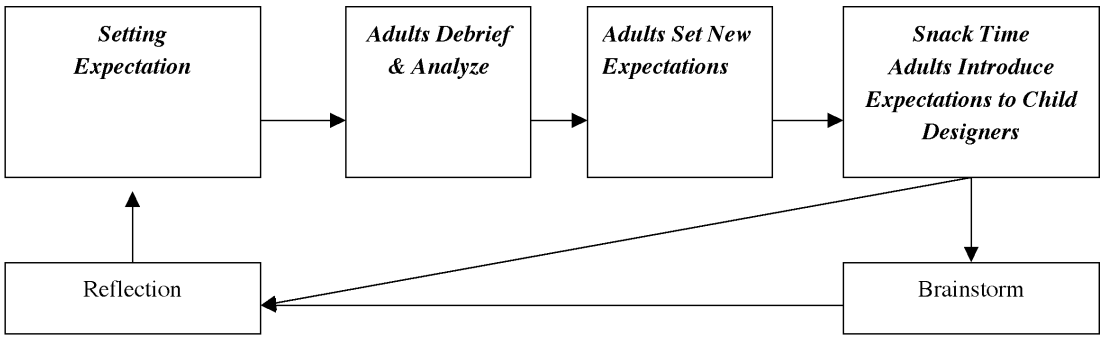

Figure 3.2: Adult debriefing and snack time are critical moments for setting expectations. A typical chain of events is 1) adults reflect and evaluate on the current session, 2) adults set new goals for future sessions, and 3) adults present the goals to children during snack time.

or reflection events. It is important to note that by snack time the goal has already been altered from the technical and abstract into a form that children can comprehend.

For brainstorming events, although adults define a list of expectations, they do not enforce the outcome of the session. Indeed, they may find that children take the design team in completely different directions, or that their ideas undergo an extensive elaboration process. An expectation is just a working topic and not set in stone.

Adult debriefing and analysis

The ATM debrief after every research session. More than just a way to set future goals, these review sessions also help the adults refine the *cooperative inquiry* methodology. Perhaps the most important lesson that I have learned is that debriefing can help a design group identify and tailor the specific techniques that work best for its members. Table 3.2 contains the standard questions for debriefing.

Table 3.2: Standard questions during adult debriefing sessions.

Question	Reason

Table 3.2: Standard questions during adult debriefing sessions, continued

Question	Reason
Were the ATM's expectations met?	This is always the first question. It sets the tone for the review session.
What techniques did not work? Why? Did the ATM make any obvious mistakes?	This helps the adults distinguish mistakes that were due to deficiencies in the design process or from lapses in adhering to the *cooperative inquiry* framework.
Were there any elaborations? What were they? What were the sequence of events?	Since elaboration is the main brainstorming goal, the ATM analyze the events that led to the elaborations and identify their triggers.
Did any critical ideas occur?	Conceptual breakthroughs are often important cornerstones in the resulting technology.
What did the children and adults write in their journals[1]?	During brainstorming, it is not always possible to hear every person's contribution. Journals offer another democratic way for all members to voice their opinions.
What did the children videotape, write, or draw?	It is very easy for adults to focus on topics that they think are important for everyone. By reviewing children's reflections of brainstorming sessions, adults can uncover issues that are important from the children's perspectives.
Are more sessions needed to achieve the current goal? Should the goal be refined or changed to new set of expectations	The ATM may have underestimated the effort required to attain a goal. Or, the IDT may have made a breakthrough that enables the team to move to the next problem.

[1]With the start of every year, the adults and children received a journal to enter their observations, ideas, and reflections. For pre-literate children, it was common for them to express their ideas through drawing. Often, the children would also verbalize their thoughts for the adults to transcribe into the journal.

Setting New Expectations

Adult debriefing gives the ATM information needed for the analysis phase, where the adults refine or re-define an expectation, choose supporting design activities, allocate human resources, prepare prototyping materials, and develop one simple expectation statement so that both adults and children can understand the goal. Table 3.3 contains a pre-session checklist and questionnaire the ATM use before each design team meeting.

Table 3.3: Pre-design session checklist and questions.

Question and Checklist	Reason
What is the type of the next session (contextual inquiry, low-tech prototyping, sticky session, etc.)?	This is a useful first question since contextual inquiry, participatory design, exhibitions, etc., require different preparations.
Can the goal be reasonably attained within the session?	The ATM define goals that are small enough so that the IDT can accomplish something for each meeting.
Choose appropriate brainstorming activities.	For example, if the ATM plan a contextual inquiry session, then the adults need to define the context and prepare the lab for this activity. Or, if the plan is to sketch new design ideas, the IDT would use low-fidelity prototyping methods.
Allocate team members to work on appropriate activities.	Just as adults have specialties, children also have strengths and weaknesses as designers. After the ATM have decided on the brainstorming activities, the adults select the best people to handle the subtasks within those activities.

Table 3.3: Pre-design session checklist and questions, continued.

Question and Checklist	Reason
Prepare, collect, or purchase prototyping material.	Different kinds of sketches (prototype) require different material. For example, a storyboard session may only need markers, crayons, and lots of paper. But a project such as a robot might require clay, popsicle sticks, glue, scissors, LEGO pieces, motors, etc.
Develop a simple statement of the new session goal.	All members should easily understand the expectation. Thus, "We are going to design an ad-hoc distributed networking protocol." would not be appropriate, since children may not have the background to understand this statement. Alternatively, "Let us invent toy animals that play with each other when you put them all in a room." might be a more concrete goal statement.

Setting Expectations During Snack Time

Because the IDT uses snack time as a transition for both adults and children to become design partners, the ATM present the expectation for the session in incremental steps, from general bantering about fun topics to more closely related ideas and finally to the explicit expectation statement that the adults prepared during adult debriefing. Table (table 3.4) contains the list of discussion topics.

Table 3.4: Snack time discussion topics the adults use to transition from general banter into presentation of session goal.

Things to Say	Reason
Share some fun stories and jokes.	To prepare the group for a brainstorming session, adults and children share fun stories to break down the age and authority barriers between them. The topics are often silly, such as knock-knock jokes, favorite junk-foods, and most disgusting ice cream flavors.
Ask a round-table question of general interest that is related to the research agenda of the day	This is a transitional statement. These questions move the children from their regular role as student and child to the role of designer. For example, if the ATM goal for the day were about building an interface to query a digital library for 5 year old children, then the adults may ask, “Do you ever look things up on the web?” Or, “Have you gone to the library to look for books about vegetables?”

Table 3.4: Snack time discussion topics, continued.

Things to Say	Reason
Ask a question that ties the previous brainstorming session to the day's goal	The child designers move closer to understanding the context of the day's session. Here is an example, taken from a design session on StoryRooms: "Do you remember last time when we went around the table making up a story? Well, today, we are going to do that again, just to warm up. After that, we are going to think about what kind of computer we can make that lets us turn this room into that story."
State the expectation for a contextual inquiry session	Begin the session by talking about related subjects that the children are already familiar with. Then, narrow to the specific subject. For example, when the IDT began the StoryRooms project, the ATM asked the children to think about how many ways they know to tell a story (i.e., reading, movie, music) and what elements good stories share. The adults asked such questions as, "What was your favorite movie of this month?" Then, after everyone has given his opinion, the adults follow up with, "Today, we are going to figure out why you like these movies."
State the expectation for a contextual inquiry field trip	The ATM explain why the group is going, and what the team should think about while there. For example, the IDT visited Port Discovery in Baltimore to learn about story spaces. The ATM might say, "We are going to Port Discovery today because they have a mystery story-room. Think about what you like and do not like about the way they built it. Also think about whether it is exciting, boring, fun, or frustrating."

Figure 3.3: Adults and children looking for patterns during a stickies session.

Table 3.4: Snack time discussion topics, continued.

Things to Say	Reason
State the expectation for analyzing a problem	Usually after a contextual inquiry session, the IDT has identified issues to investigate. But we don't necessarily understand why these problems exist. To prepare the IDT for sessions to understand the nature of a problem, the ATM would say, "Last time, we looked at Gadget X. Today, we are going to start with a stickies session[2] and figure out what we liked and what needs work." (figure 3.3)

3.5.2 Brainstorming

Five issues occur frequently during brainstorming sessions: 1) understanding technology, 2) evaluating technology, 3) designing technology, 4) stagnating design session, and 5) uncooperative children. The following lists these situations (figure 3.4) and describes some practical responses (table 3.5).

[2]A stickies session is an analytic activity. Both adults and children contribute several sticky notes that contain likes, dislikes, and suggestions for improvements to the technology under review. All the notes are posted on the wall for the team to offer comments.

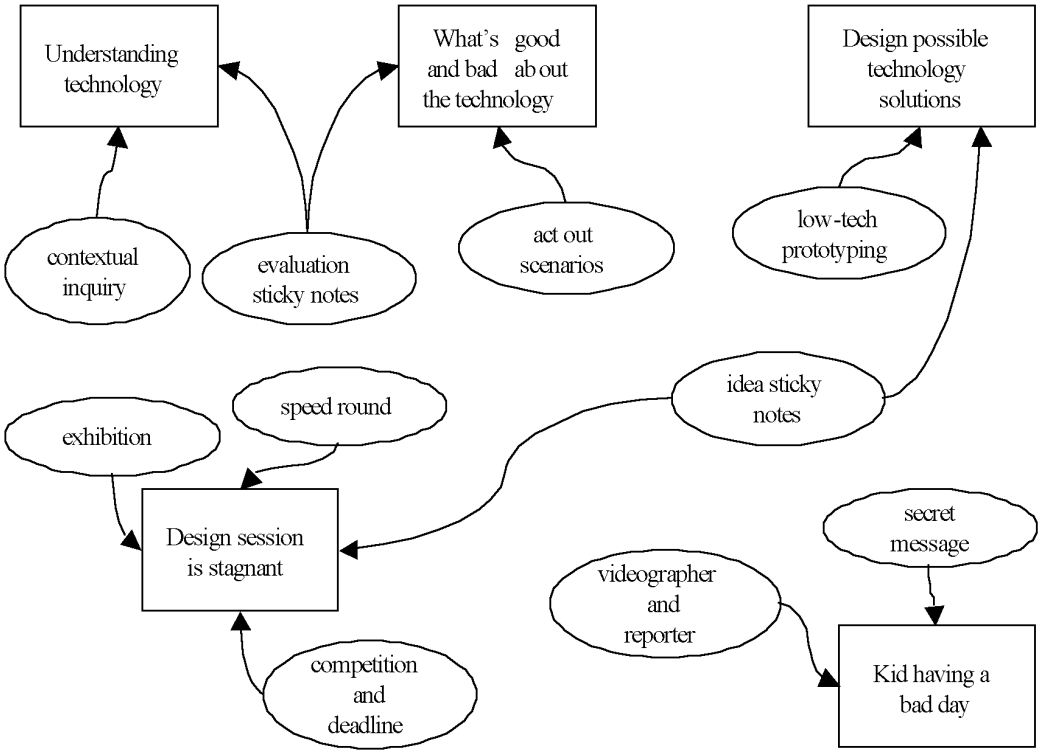

Figure 3.4: Issues that arise during brainstorm sessions and how the IDT solves them. In the rectangles are issues such as understanding technology to overcoming a stagnating moment. The bubbles are the solutions that have been consistently useful.

Table 3.5: Common brainstorming events and useful responses.

Situation	Exercise	Explanation
Understanding existing technology	Contextual Inquiry	A great way to understand technology is to use it. Every new project begins with contextual inquiry sessions of relevant technologies.
Understanding existing technology	Evaluation Sticky Notes	The IDT follows every contextual inquiry session with an evaluation stickies exercise. Evaluation sticky notes offer a democratic way for design team members to voice their opinions about a technology. Each member writes on sticky notes 3 things they “like” and 3 things that they“don’t like.” These comments are posted on a white board for all to evaluate. Then, as a group, we identify major classes of positive and negative features, the most important issues (by frequency), and good ideas, from all the categories, that deserve further investigation.

Table 3.5: Common brainstorming events and responses, continued.

Situation	Exercise	Explanation
What is good and bad about the technology?	Act Out Scenarios	Stickies sessions identify the problems that the design team wants to solve. The next step is to dissect the problem, find out both positive and negative characteristics. One very effective method to analyze problems is to act out scenarios. That is, the team members become the various parts of the technology and act out what the technology might do in a situation. This gives both the adults and children a concrete[3] understanding of what is happening. It makes abstract notions concrete and physical and visual. This works well because there is no need to build prototypes, which takes time and effort.
Sketch new technology solutions	Low-tech prototyping	Low-tech material such as paper, glue, and cardboard boxes enable both adults and children to visualize their ideas for new technology. It is inexpensive, quick, and fun.
Sketch new technology solutions	Idea Stickies	There are times when even low-tech prototypes requires more effort than it is worth. Use idea stickies instead. Idea stickies are just sticky notes of different colors. Sketch pictures, icons, words, on these papers and shuffle them around to act out scenarios.

[3]Recall that young children think best in physical ways.

Table 3.5: Common brainstorming events and responses, continued.

Situation	Exercise	Explanation
Stagnant period, kickstart the elaboration process	Speed round	The design group sits in a circle and a leader begin by stating a goal. Then, each person around the circle contributes an idea. This person only gets one second to think about it. If he cannot contribute an idea, then you say, "Too late!" and move to the next person. This works wonders when the group is not moving forward because it removes inhibitions and encourages everyone to just blurt out something. As soon as a truly great idea comes out, the design team can pounce on it and elaborate. When a person is not given enough time to think, he can be silly with his idea. The children seem to understand this and instead of worrying about whether they appear foolish to others, they instead enjoy trying to outdo each other.

Table 3.5: Common brainstorming events and responses, continued.

Situation	Exercise	Explanation
Stagnant period, kickstart the elaboration process	Idea Stickies	When a discussion is becoming too abstract, children may lose focus. When this happens, make the ideas concrete by putting them down onto idea sticky notes. Because some children cannot read as well as others, use pictures instead of words. Idea stickies regenerate the brainstorming process because the children can see and touch them. So even though they contains the same elements as verbal discussions, their physical presence remind and focus the kids on the problem.
Stagnant period, kickstart the elaboration process	Competition and deadline	If the children become bored, split the team into groups and make the low-tech prototyping session a competition. Not only do they enjoy trying to outperform each other, but the result of these competitions can often become the first prototypes of new technology.
Stagnant period, kickstart the elaboration process	Exhibition	Tell your IDT team that you will show off the new technology and give them a deadline. An exhibition sets a tone for the design sessions and lets both the adults and children know that what they are making is truly important.

Table 3.5: Common brainstorming events and responses, continued.

Situation	Exercise	Explanation
Kid having a bad day		The ATM constantly remind ourselves that half the IDT team is 11 years old or younger. When even just a couple of them are having a bad day, the meeting could easily become unproductive. As adults, we accept that some sessions are just not going to be productive.
Kid having a bad day	Videographer and reporter	Offer the child who is having a bad day to be a reporter and give her the video camera. Ask her to film the session. This removes her from direct interactions with the team, but she is still contributing by recording the meeting. If a video camera is not available, make her the journalist and ask her to write down, or sketch, what she observes, to put into her journal.
Kid having a bad day	Secret message	Each child team member reacts to bad days in different ways. It can be helpful for the adult team leader to develop special gestures for every child, so that when she is uncooperative, she can be told about it without being publicly embarrassed.

3.5.3 Reflections

Four exercises capture events in our lab: 1) adult debriefing, 2) journals, 3) videotape, and 4) team presentations. By analyzing this wealth of data, the ATM have been able to

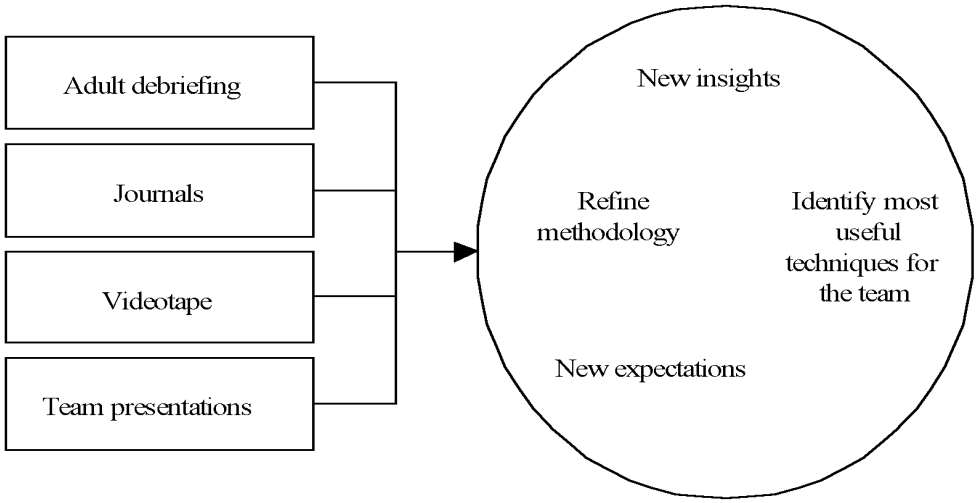

Figure 3.5: Artifacts from the IDT research sessions help define future goals.

refine *cooperative inquiry* as well as create interesting new technologies for children. In Table 3.6, I describe these four activities and the type of information they capture (figure 3.5).

Table 3.6: Reflection activities and the type of information they capture.

Class	Explanation	Exercise
Adult debriefing	This was discussed in the expectations section above (3.5.1).	Refer to page 46.

Table 3.6: Activities during reflection, continued.

Class	Explanation	Exercise
Journals	Journals are more than just archives. They are also records of changes and refinements in the *cooperative inquiry* practices. For the children team members, it is a way for them to remember their contributions as designers in the team. It is useful to set aside 10 minutes at the end of the day, and write down reflections, contributions, and observations.	It is useful to throw out a few questions for the children to answer. This helps them write. I have found these to be effective questions: "My best idea today was…" "Today I learned…" "My contribution to the group was…" "Today I liked…" "Today I did not like…" They also enjoy sketching any prototypes that they built into their journals.
Videotape	Footage captured by children can reveal their perspectives of the design process and what they perceive to be important issues.	Both adult and children members videotape design sessions.
Team presentation	These presentations bring closure to the day's accomplishments. They also allow the teams to review and comment about the work of other groups.	If the session involved competitions or members worked in sub-groups, the group always reserved 10 minutes at the end of the session for the teams to discuss what they had done and what was difficult.

3.6 Working with Kindergarten Aged Children

When the IDT expanded its research methods to include kindergarteners as designers, I did not quite know what to expect. Were they mature enough to be productive members of a design team? What were the best roles for them? What changes to our methodology were needed to best accommodate their needs? During its pilot year of study, the design team discovered many changes were needed to work with kindergarten students [32].

One modification that I learned was that word choice was incredibly important. It could either make the abstract idea seem easy to understand, or the simplest concept impossible to grasp.

3.6.1 Use Their Words

I found that using words within the kindergarten children's vocabular allowed the adults to communicate our ideas more clearly to them. By close observation and review of videotapes, I picked up on the types of words, or even exact phrases, kids used. For example, while the adult IDT members were presenting the StoryRooms concept, a child, trying to make sense of his observations, made the comment that there were "invisible wires" in the room. This was a profound moment, as the adults had been up to that moment stumped on finding a way to explain the most abstract part of the technology, which was the wireless interactions between the physical icons. Now if one child was able to describe what was going on, it was probably safe to use those same words on others too. Use "kid-friendly," or "kid-originated" language with them. They will understand concepts at their level.

3.6.2 Level of Concreteness

Ask simple, specific, and pointed questions to kindergarteners. At this age, they may be less able than the older, elementary school students to produce good design ideas in a completely open-ended forum. Here is an example from the physical programming project [65]. Suppose I want to say, “I am connecting these two things.” I might instead say, “I am making an invisible wire between these two things.” The word “connect” is difficult for the child because there is no physical manifestation of connectedness. But a wire is real. It can transfer things from one place to another. To the kindergartener, it makes sense that a wire, even if it were invisible, can be laid between two objects.

Chapter 4

PETS: My First Physical Interactive Storytelling Construction Kit

The seed for a physical interactive storytelling space came from my early work on a special robot called PETS (a Personal Electronic Teller of Stories). The goal of this project was not so much the creation of a new technology, but rather, it was a way for the IDT to develop a participatory design framework that could include children as designer partners [23, 24, 26, 27]. After the robot was completed, I not only came to understand more deeply the roles of children in a design team, I also came to understand that children were interested in tools for them to create their own interactive stories.

By itself, PETS was a successful demonstration of what a physical storytelling kit might contain. But more importantly, it was my experience working on this project that sparked the far more ambitious project of creating construction tools for interactive spaces such as StoryRooms. Despite their apparent differences (one being a robot and the other an environment), conceptually both systems are quite similar. First, the story construction process requires many physical components and an element of

"programming" or "scripting." Second, the child user **owns** the entire process. An adult does not have to interject high-tech mumbo jumbo on behalf of the children for the system to work. Finally, both systems gave children a way to explore new kinds of storytelling experience.

This chapter on PETS is an entry into physical storytelling systems. It offers a glimpse into the motivations behind creating a kit for children that is constructive, physical, and that can generate stories.

4.1 PETS Tells a Story

> "There once was a robot named Michelle. She was new in the neighborhood. She was HAPPY when she first came, thinking she would make friends. But it was the opposite. Other robots threw rocks and sticks. She was SAD. Now no one liked her. One day she was walking down a street, a huge busy one, when another robot named Rob came up and ask [sic] if she wanted to have a friend. She was SCARED at first but then realized that she was HAPPY. The other robots were ANGRY but knew that they had learned their lesson. Michelle and Rob lived HAPPILY ever after. No one noticed the dents from rocks that stayed on Michelle." [22]

This was just one of many stories that children wrote with the help of PETS [28], my first physical and interactive storytelling construction kit.

Because storytelling is inherently constructive, the resulting products from the IDT have been kits that enable children to create their own stories. My goals too had evolved from the PETS storytelling robot [28] to a kit that enabled children to build physical and interactive story environments [66, 2]. By giving children the tools to build their own interactive physical environments, they could begin to experience a level of creative autonomy that was previously limited to adults.

4.2 A Description of PETS

PETS, first developed in 1998, was a robotic story telling environment for elementary school age children [28]. The PETS construction kit contained a box of fuzzy stuffed animal parts and an authoring application on a personal computer. Children could build a robotic animal, or pet, by connecting animal parts such as torso, head, paws, ears, and wings. Next, they wrote and told stories using the *MyPETS* software. Just as the robotic animal was made from discrete components, *MyPETS* was also constructive. This application enabled children to create emotions, to name their robotic companion, and to compile a library of stories and story starters (figure 4.1).

Each emotion that the robot performs was represented by a sequence of physical movements that conveyed a specific feeling to the audience. The child designers helped the team to define six basic emotions and the movements that accompany them: happy, sad, lonely, loving, scared, and angry. They were chosen because of their significance to children in their everyday lives and because these actions represented emotions that were sufficiently different from each other that the audience would not confuse one from another. For example, to express loneliness, the robot drooped its arms down and looked left and right, as if it were looking for a friend. To show happiness, PETS

Figure 4.1: PETS. On the left is a computer displaying the *MyPETS* software. In the middle is the PETS robot decorated with paws, a pig's snout, horns, and flappy ears. To the right is a flying saucer for PETS to ride on. On top of the flying saucer are an optional pair of wings.

waved its arms really fast, turned its head left and right, and spun the spaceship it rode on. And, when the robot was "sad," it droops its head and arms, and moved forward at a slow, deliberate pace.

PETS encouraged creativity through interactive and iterative play. Children were constantly writing and rewriting their stories (figure 4.2), and in the process, developed their own writing styles. They also enjoyed building different kinds of animals. Whenever they wanted, these children could command *MyPETS* (figure 4.7) to tell their story and watched their animal act out emotions. As the robot encountered each "emotional" word in the story, it performed that emotion by moving its body in the sequence specified by its creators.

A critical feature of PETS was that the child user was always in control. Unlike products such as the Actimates Barney, where the robot directed the flow of action, and the child followed its instructions, children can decide their own activity patterns.

Three versions of PETS were designed. They were named in the order of their creation, $PETS_0$, $PETS_1$, and $PETS_2$. Each successive version was a more refined "sketch" of the IDT's collective vision of the robotic storytelling environment.

$PETS_0$ was a prototype that I created in early 1998 to understand the technical issues related to interactive robots. It was in fact the result of my own technology immersion process. The knowledge gained from building this robot became a rough technology roadmap for future PETS.

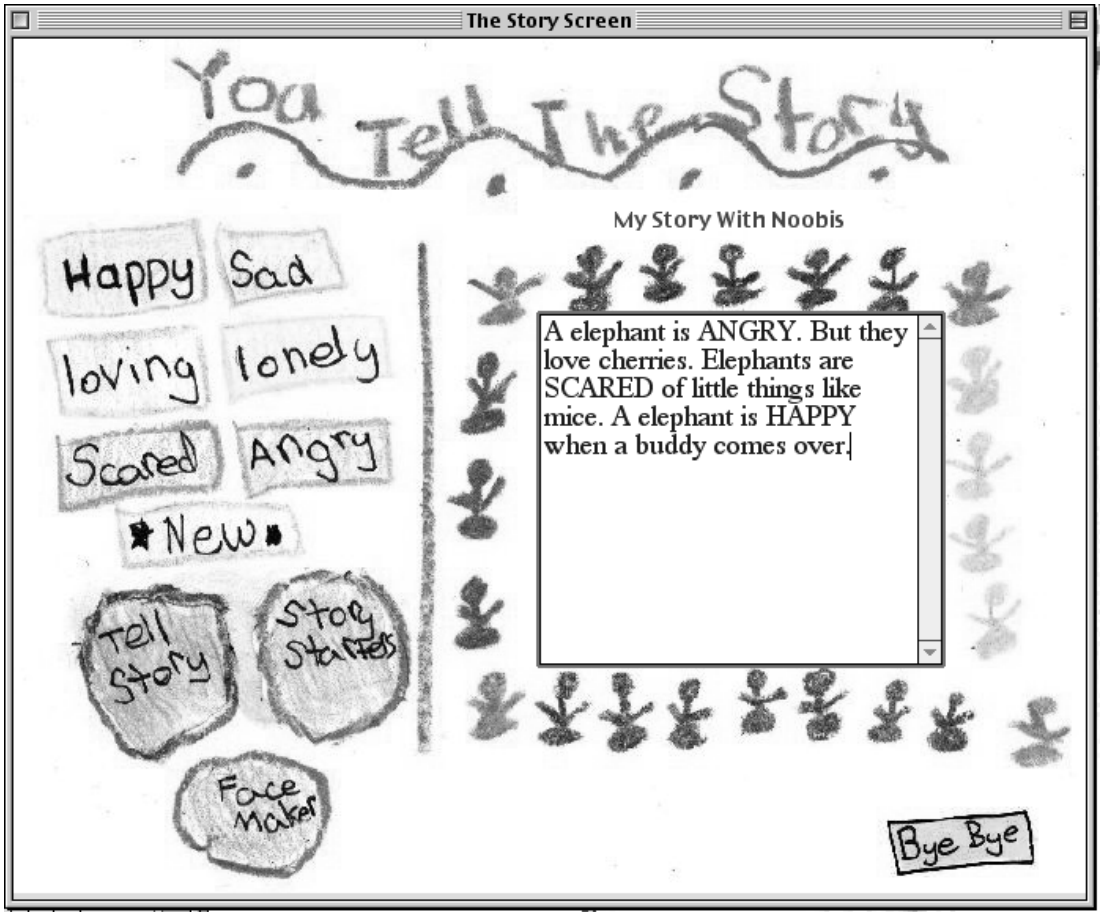

Figure 4.2: Children typed their stories and insert “emotions” with The Story Screen. A child provided the sentences in the above story.

4.3 PETS$_1$

PETS$_1$ (figure 4.3), built during the summer of 1998, was the first "full-featured" and demonstrable robot from the IDT. This machine had primitive reactive behaviors. For instance, its head followed or turned away from a beam of light, depending on its "mood." Also, it would move its paws toward you if it was "happy," or pull away if it was "scared." Most of its skeletal structure was constructed of LEGO blocks, and the limbs were simple plastic boxes. Various fabric materials covered the robot to hide the mechanical components. For instance, the head was covered with a fabric with cow hide prints; some limbs were covered in feather; and a furry skirt was draped over the entire robot to create a rounded and soft shape. A Handy Board [60] micro-controller in the torso controlled embedded motors and servos. In addition to on-board programs for the robot's primitive behaviors, the controller also received commands from *MyPETS* through a connecting wire. Sensors (e. g., light and touch) throughout the body informed the robot about its environment.

Conceptually, PETS$_1$ contained three major components: 1) the skeleton, 2) the skin, and 3) the software.

4.3.1 The Robot Skeleton

The PETS$_1$ skeleton was primarily made from LEGO blocks since it was an easy prototyping tool for both adults and children. It had a modular design. For example, the eyes (light sensors) were detachable and could be placed in different places of the robot's body. The limbs, made from plastic boxes and embedded with servo motors, attached to the torso by LEGO pegs. The wheelbase was separable from the rest of the robot (figure 4.4).

Figure 4.3: PETS$_1$, with a furry body, dog paw, duck foot, and cow face.

The most difficult problem of the skeleton involved the joints. For example, the neck, a 2-degrees-of-freedom joint, had to support a large head, and was the weakest point of an inverted pendulum. Also, the limbs fell off easily, since they were connected to the body with only several short pegs. Wiring was another issue. Although individual body parts were detachable from the skeleton, they still had to be tethered by telephone wires to the body. Eyes and limbs that were truly physically independent modules would have been much nicer.

4.3.2 The Robot Skin

Typically, the design of a robot stops at the "skeleton" phase. But since one goal of this project was to create a "fluffy" and "huggable" pet, I was concerned with the appearance of the robot as well. The IDT created shapes by padding the skeleton with

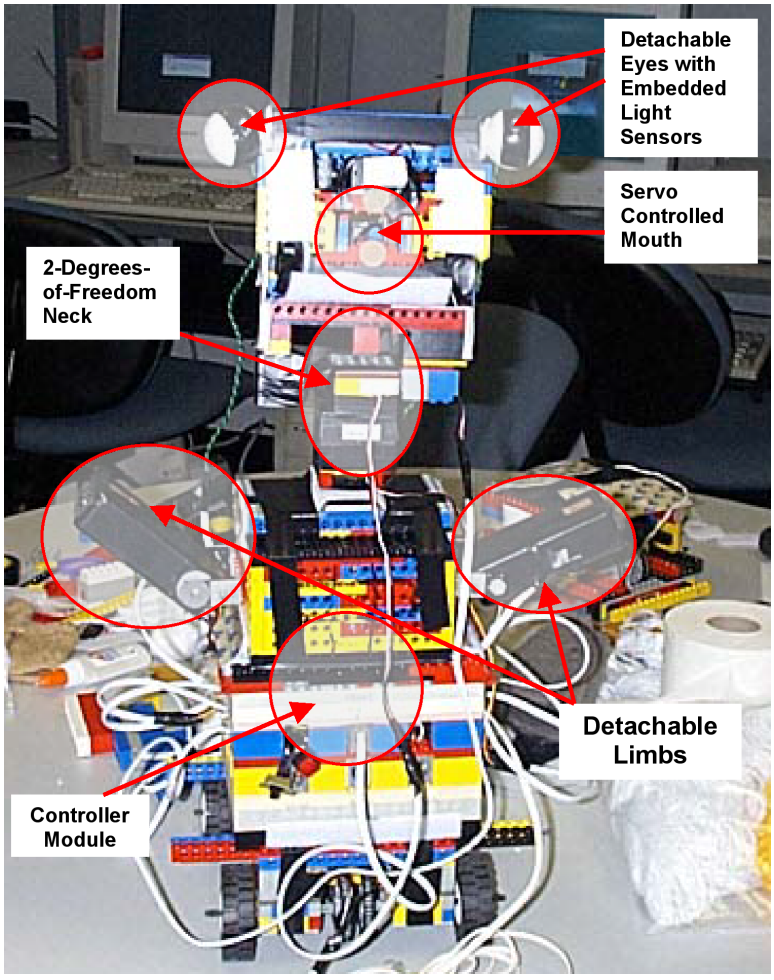

Figure 4.4: The $PETS_1$ skeleton.

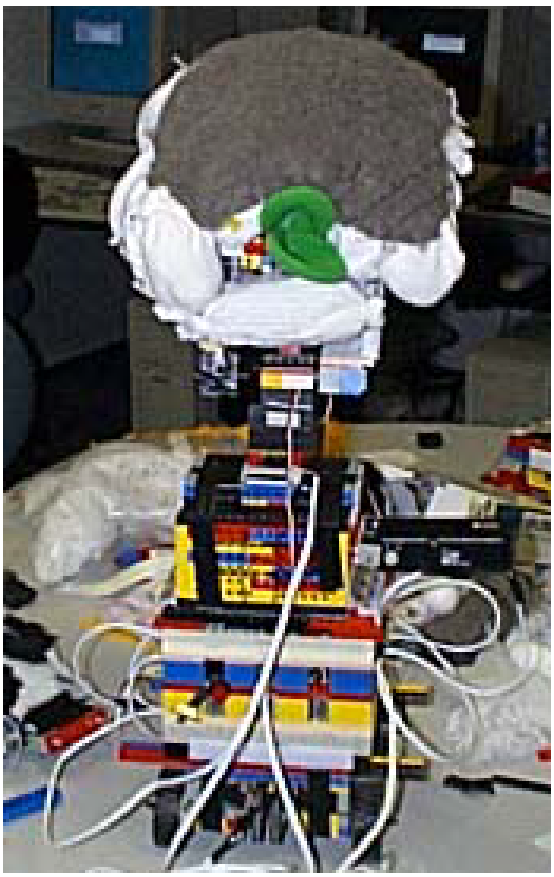

Figure 4.5: The $PETS_1$ skeletal head with padding.

socks and fabric sheets (figure 4.5). In addition, a skirt was draped around the body and covered the skeleton (figure 4.3).

4.3.3 The Software

$PETS_1$ was designed to operate in two modes: 1) autonomous, and 2) remote control by *My PETS*. In the autonomous mode, $PETS_1$ was a reactive robot with the following *abilities.*

See The light sensor eyes.

Listen The microphone ears (hardware was not implemented at the time).

Speak The loudspeaker.

Run The wheelbase.

Arms The appendages

Feel The touch sensors on the Arms. One is in front of the arm, the other is in the back side.

Remote control *My PETS*.

Each ability is a separate process. Sensor processes update data onto a global blackboard, while actuator processes refer to the blackboard and affect devices accordingly. These abilities supported *personalities*. $PETS_1$ supported SHY and CURIOUS. The following example illustrates the relationship between personality and abilities. If the robot were set (by software) to be SHY, then when the Left Feel is triggered, the Left Arm would pull back in the opposite direction of the touching. If instead the robot were set to CURIOUS, then the *Arm* would move into the direction of the touching. Another words, suppose I am shy and a person touches my arm, I would pull my arm away from the person. But if I were a curious person, then I would push my arm into the person's hand.

When $PETS_1$ was in the *Remote Control* mode, it was not reactive and was a slave to the commands issued by *My PETS*. In this mode, *My PETS* sent streams of commands to the robot that activated various movements.

Figure 4.6: The *MyPETS* software, the transmitter box, and the skeletal components of $PETS_2$.

4.4 $PETS_2$

After the IDT completed $PETS_1$, three major goals for the next version were set: improved aesthetics, durability, and wireless communication. The next prototype, $PETS_2$, built between September 1998 and April 1999, included many refinements over its predecessors. For example, $PETS_2$ had a foam outer-shell covered with felt (figure 4.6). It had a more graceful shape and vibrant colors. The foam shell not only provided the shape of the robot; it was also a buffer against abusive play from children. In addition, it had a much sturdier skeletal structure built from metal, plastic, and polycarbonate materials. The robot also communicated with the *My PETS* software via wireless radio frequency channels.

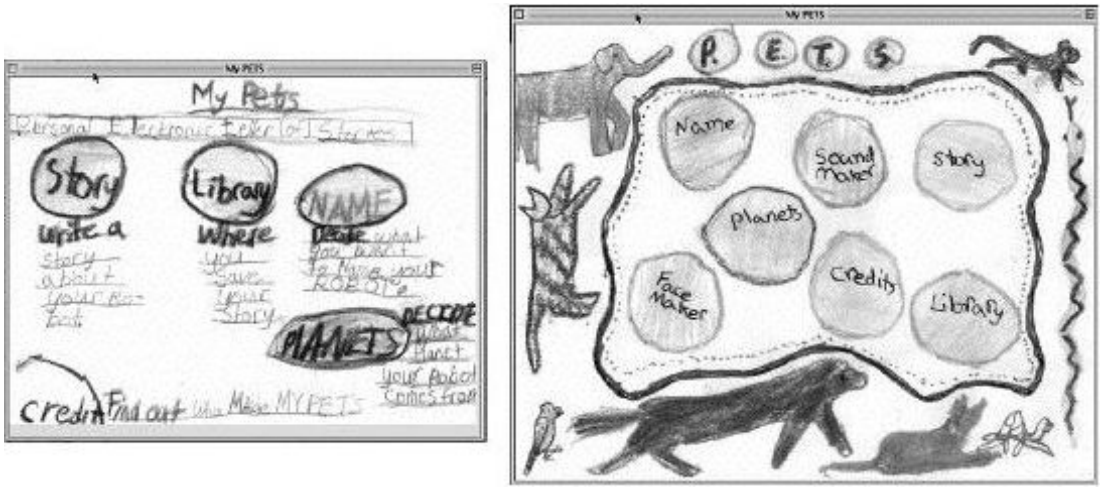

Figure 4.7: Main screens of the *MyPETS* application. The left image is from $PETS_1$, and the right image is from $PETS_2$.

4.4.1 Limitations

$PETS_2$ had several significant differences from $PETS_1$. First, unlike $PETS_1$'s LEGO blocks, the $PETS_2$ skeleton was made from much sturdier polycarbonate sheets and steel posts. Its "skin" was a single felt-covered foamy shell, and did not suffer from problems such as the body skirt coming off $PETS_1$. But, this new design still did not adequately address the weak neck and the method for attaching limbs onto the body. The $PETS_2$ project did not suffer too much from this, as the children were focused on the storytelling and observing the robot's performances. These mechanical problems could be solved by collaborating with mechanical engineers.

Unlike the $PETS_1$ software architecture, which had a reactive layer (eg. [14]), an autonomous behavior was not built into the $PETS_2$ software, because I focused on the storytelling and the sequencing of actions in *My PETS*. But this lack of a primitive behavior meant that $PETS_2$ could not protect itself from obstacles, such as a wall.

4.5 Robots, Children, and Learning About the Design Process

Children are drawn to physical play things. They love robots! I saw this repeatedly during the year I worked on PETS. When I demonstrated PETS to young visitors to HCIL, inevitably, they were drawn to the robots in the lab, even though it was filled with lots of other toys for children. I believed this was because robots are inherently intriguing and highly interactive objects. Indeed, studies have shown that in settings where there are both things to observe and things to play with, young children are usually attracted to activities where they can become interactive participants. For example, in zoos, they prefer to interact with pigeons and squirrels than the more exotic animals behind bars [43].

Accordingly, a robot as a design goal was useful for two reasons. The first reason was that it was much easier to get the children to be excited about inventing something they like, rather than to create the next generation toaster oven, for example. The other was that the process of building robots is inherently collaborative and physical. Since the IDT was interested in both the process and the product, having a project that required collaboration and lots of construction was helpful in studying and testing *cooperative inquiry* (3) in action.

But, building these interactive, robust, and child-friendly robots was extremely difficult. I believed that there were two main reasons: First, a project such as PETS required an interdisciplinary effort, thus, a team with diverse talents. Putting together such a group was not easy. Second, interactions between a robot and its environment were often unpredictable. This uncertainty presented many technological and engineering challenges.

4.6 Lessons Learned from PETS

PETS provided many insights into physical interactive storytelling. In particular, the affinity that children have toward real objects and their strong desire to write stories and share them with others. It also showed children's desire for "fuzzy" and "cuddly" robotic objects. But this project also revealed some shortcomings of storytelling using robots. They can be fine actors, but it can be awkward to use them to express physical storytelling experiences such as *the wind blows through the plains*, which can be simply implemented using a contact sensor, a fan, and a projected image of a desolate plain. These limits led me to my next project: StoryRooms.

Chapter 5

Stories within a Physical Interactive Environment

The transition from storytelling robots to storytelling environments was natural. Although a physical robot can be an actor, some story elements are either inconceivable or awkward to express through a robot. Children can easily use the robot to "express" sadness or happiness, but might have difficulty making it project: *it was a dark and stormy night*.

In the summer of 1999, I began work on a technology that would lead me to my dissertation research: technology for children to construct their own storytelling physical interactive environments. Lessons I learned from PETS, such as sequencing physical events to express abstract ideas, and sensor-effector interactions, formed the foundation of this new endeavor. I believed, along with my team, that with the right set of tools, children could construct their own StoryRooms. And, through interactions in this environment, children can enjoy a new storytelling experience [2].

Designing StoryRooms with children proved to be extremely difficult. Although children are natural storytellers, and although they have encountered many forms of storytelling, that a physical space can be expressive was initially too abstract for some

of the child designers[1]. To understand the relationships among storytelling, interactive technology, and physical environments, three StoryRoom prototypes were built with increased interactive levels: *The Red Balloon*, *Hickory Dickory Dock*, and *The Sneetches*. *The Red Balloon* and *Hickory Dickory Dock* were mock-up environments, in which the children pretended to be the proximity sensors and the sound and light effectors that were needed to emulate the interactivity in the physical story. The third prototype, *The Sneetches* was the first environment that contained real computational abilities; a software application monitored contact sensor inputs and selectively activated sounds, lights, and images within the room. From these low-tech prototypes I learned about the structures of physical stories and their technological requirements.

5.1 The Red Balloon

The Red Balloon is a classic movie about a boy who finds a friendly red balloon that follows him wherever he goes. Some mean-spirited kids want to take it away from him. When the red balloon keeps floating away from the bad kids, they pelt it with rocks, until it bursts and falls onto the ground [54]. As the movie ends hundreds upon thousands of balloons begin to rise from all over the village.

In the HCIL, A red lamp becomes the red balloon. One child controls it by flickering it on and off and saying, “I’m the red balloon.” Whenever the “good boy” walks by her, she turns the light on and utters the sentence. If the “bad kids” behave menacingly toward her, she turns off the light and crumbles towards the ground.

In this exercise, a child takes the role of the computer that controls the interactions between physical objects and people. These interactions added a new kind of expe-

[1]Truth be told, the StoryRoom concept was difficult for some adults too.

rience to the original medium of the *Red Balloon*. The IDT began to understand that physicality can afford a rich experience. Also, by being directly responsible for the interactions, the child designers learned about the role of technology in these storytelling environments.

5.2 Hickory Dickory Dock

Hickory dickory dock.
The mouse ran up the clock.
The clock struck one.
The mouse fell down.
Hickory dickory dock.

The second low-tech StoryRoom was an adaptation of the classic children's rhyme, *Hickory Dickory Dock*. Once again, some children were asked to be the technology, while others were visitors to the story.

5.2.1 Setup

Paper labels representing special effects, such as *Touch*, *Sound*, and *Light*, and objects, such as *Telephone* (figure 5.1), *Computer*, and *Chair* were placed next to actual objects in our lab. Three paper plates were decorated as props. On each plate was one of the the words *Hickory*, *Dickory*, and *Dock*. Digitized sound effects were stored on a computer. One child was responsible for sound effects. Another was given a flash light and was responsible for light effects. Finally, three children were asked to be different kinds of

Figure 5.1: The phone object in Hickory Dickory Dock. The paper light and sound "buttons" represent features of the "phone" object.

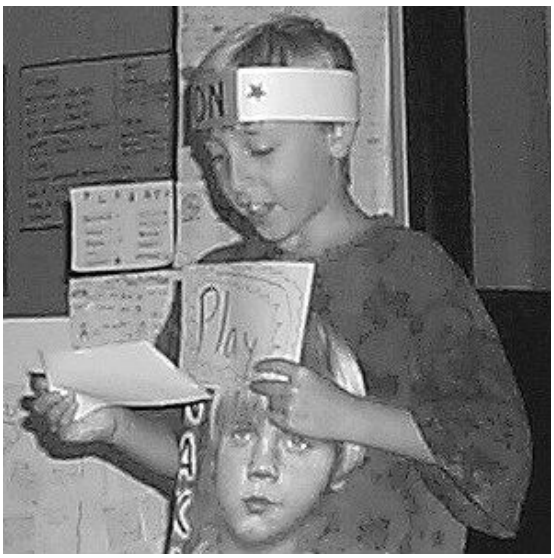

Figure 5.2: A child as a low-tech wizard-of-oz in the Hickory Dickory Dock rhyme.

narrators. Each successive speaker offered an increasing level of interactivity with the story room visitor.

5.2.2 Direct Recitation

A visitor entered the room and selected a narrator by placing a crown on top of that person. She then pushed a paper labelled *Start* on the speaker's chest to begin the story. When the narrator (figure 5.2) uttered the words, "hickory, dickory, dock," the light-effects person would shine a beam of light on the paper plate with the corresponding word. Then, when the narrator said, "the mouse ran up the clock," the light-effects person aimed the light at the real clock on the wall.

5.2.3 Recitation with Choices

In this variation, the visitor chose the object that the mouse would climb. So, the narrator said, "Hickory, dickory, dock. The mouse ran up the…," and paused for

the child to push the *Touch* label next to its physical counterpart. If she pushed the *Telephone*, the narrator would say, “telephone,” and the sound-effects person would activate the ringing phone sound. Finally, the narrator completed the last two lines of the story.

5.2.4 Full Interactivity

The visitor explored the various sensors (the paper labels) and heard the speaker utter the corresponding words. When she was ready to experience the rhyme, she asked the narrator to recite her version.

Dickory hickory dock dock dock.

The mouse ran up the chair.

The clock struck one.

The mouse fell down.

Dickory hickory dock dock dock.

5.2.5 Lessons Learned from Low-Tech Scenarios

I learned that the same story can offer very different experiences, based on the level of interactivity afforded by the physical environment. Also, given the same setting, children had a choice of storytelling experience they want, from simply listening to a story, to creating a new story by rearranging the elements. Furthermore, I observed that simple effects, such as light and sound, can elicit highly entertaining atmosphere. This was a critical finding. I knew then that I was offering children the ability to add “magical” effects onto their stories. For them, because of the sensors and actuators, their stories really could come alive in the StoryRoom

5.3 The Sneetches

After the previous low-tech stories, a fully interactive, semi-autonomous StoryRoom, based on the classical Dr. Seuss story "The Sneetches," was created [36]. This story was chosen[2] by the child members of the IDT.

5.3.1 The Story

The Sneetches live on a beach. Some have stars on their bellies, while others do not. The star-bellied Sneetches think they are better than the plain-bellied ones. So those with stars alone could have fun, and they always looked down on the plain-bellied Sneetches. One day, Mr. Sylvester McMonkey McBean shows up with his strange contraptions, and he advertises that his machine will put on a star on any plain bellies for just three dollars a piece. Of course the plain-bellied Sneetches jump on this opportunity. But the original "better" Sneetches become upset because there is now no way to tell them apart! Coincidentally, Mr. McBean has another machine that will take stars off for ten dollars. The original star-bellied Sneetches all have theirs stars taken off. This causes a cycling of Sneetches going into one machine and directly into another, one group wanting to be different, the other wanting to be the same. When the Sneetches spent all their money, there remains the original two groups. Mr. McBean leaves the island, laughing about how the Sneetches would never learn. But miraculously, the Sneetches do learn a great lesson, that it does not matter how they look on the outside, all Sneetches can have fun together.

[2]Each child was asked to write down a list (at least ten) of his or her *all time* favorite books. The lists were dominated by Dr. Seuss titles.

Figure 5.3: Our child designers try out the Sneetches StoryRoom. The box on the left is the *Star-Off* tunnel. The box in the middle is the *Toy*. And the boxes on the children's stomachs are the *Stars* on the Sneetches bellies.

5.3.2 The Interactive Version

In this prototype StoryRoom [66], children became the Sneetches by wearing a special box (a Handyboard [60] and light bulb embedded within a cardboard box) on their bellies. Inside the Sneetches StoryRoom were the *Star-On* box, *Star-Off* box, *Narrator*, *Mr. McMonkey McBean*, and *Money* props (figure 5.3). The *Star-On* and *Star-Off* props were cardboard boxes with colored paper glued over it. Attached to each box were a light bulb and a contact sensor. The *Narrator* and *Mr. McMonkey McBean*, applications running on separate Macintosh computers, uttered digitally recorded passages from the book. The computer running *Money* was connected to an LCD projector, and projected an image of a pile of money, with the Sneetches on one side, and Mr. McBean on the other side (figure 5.4). Finally, the special boxes on the children's bellies were the *Stars* that could be visible or not.

Because StoryRooms are interactive, one of the adaptations was the addition of a *Toy* prop. The *Toy* helped convince the kids with stars on their bellies to believe that they were different from those without, and that they could change their bellies by going through Mr. McBean's machines. In effect, interactions with the *Toy* made the children feel as if they were physically on the island and that they were the Sneetches (figure

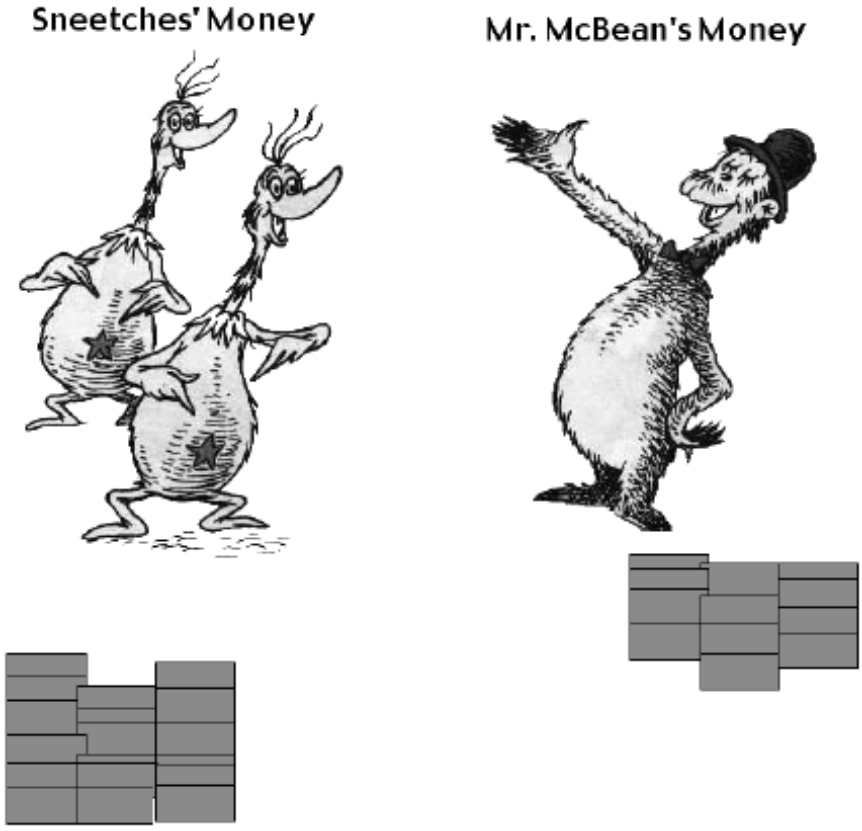

Figure 5.4: The projected image of Mr. McBean, the Sneetches, and a pile of money. Whenever a child crawls through a star on/off box, some money visually gets moved from the Sneetches' side over to Mr. McBean's side.

Figure 5.5: A child with a *Star* on his belly “plays” with the toy.

5.5).

When children initially entered the Sneetches StoryRoom, some of them started with a *Star* on their bellies (i. e. the lightbulb on their bellies lit up), while others do not. Next, the *Narrator* program introduces the story. These children explored the room and discover the *Toy*. They also noticed that the *Toy* lit up only for those who have stars on their bellies, but not for those who do not.

Soon, *Mr. McMonkey McBean* introduced himself (via the wizard), and told the children about the *Star-On* machine. When a child, who had no star on her belly, crawled through this machine, her belly lit up with a star; she heard Mr. McBean thank her for the three dollars she “paid” him; she also heard the “ka-chink” of a cash register; she sensed the *Star-On* box lit up as she crawled it; and finally, she saw that some of the Sneetches’ money had gone from their pile over to Mr. McBean’s pile (figure

5.4). And, when she went to the toy, it lit up for her (wizard)! This story continued, as children roam through the various props, until all the money had been spent, and concluded with some speeches from both *Mr. McMonkey McBean* and the *Narrator*.

5.3.3 The Technology

The Sneetches StoryRoom was made from low-tech and high-tech components. The *Star-On* and *Star-Off* tunnels were made from cardboard boxes, and decorated with colored papers and ink drawings. Taped to each tunnel were a contact sensor, embedded inside a thumb-shaped foam, and a light effector, embedded within a semispherical foam. These devices were connected by telephone wire to an Environment Interface (a Handyboard microcontroller). The Interface delivered the world data to the Monitor, via a serial port, and triggered an input event. The monitor consulted a list of sensor-effector trigger rules, and, if necessary, sent output signals to the Interface, which actuated the appropriate effector (figure 5.6).

The *Sound Effects*, *Narrator*, *Mr. McMonkey McBean*, and *Money* were applications running on different Macintosh computers on a network. They communicated with a Monitor program, using a simple file based messaging protocol. When a StoryRoom savvy application started, it registered its identity and list of services to the Monitor. The (adult) programmer of the storyroom then used this information to create trigger rules using the Monitor application. For example: if contact sensor A was on, trigger

1. *Sound Effect* to play the clashing-coin sound;
2. *Mr. McMonkey McBean* to play the digitized sound segment “three dollars, thank you”;

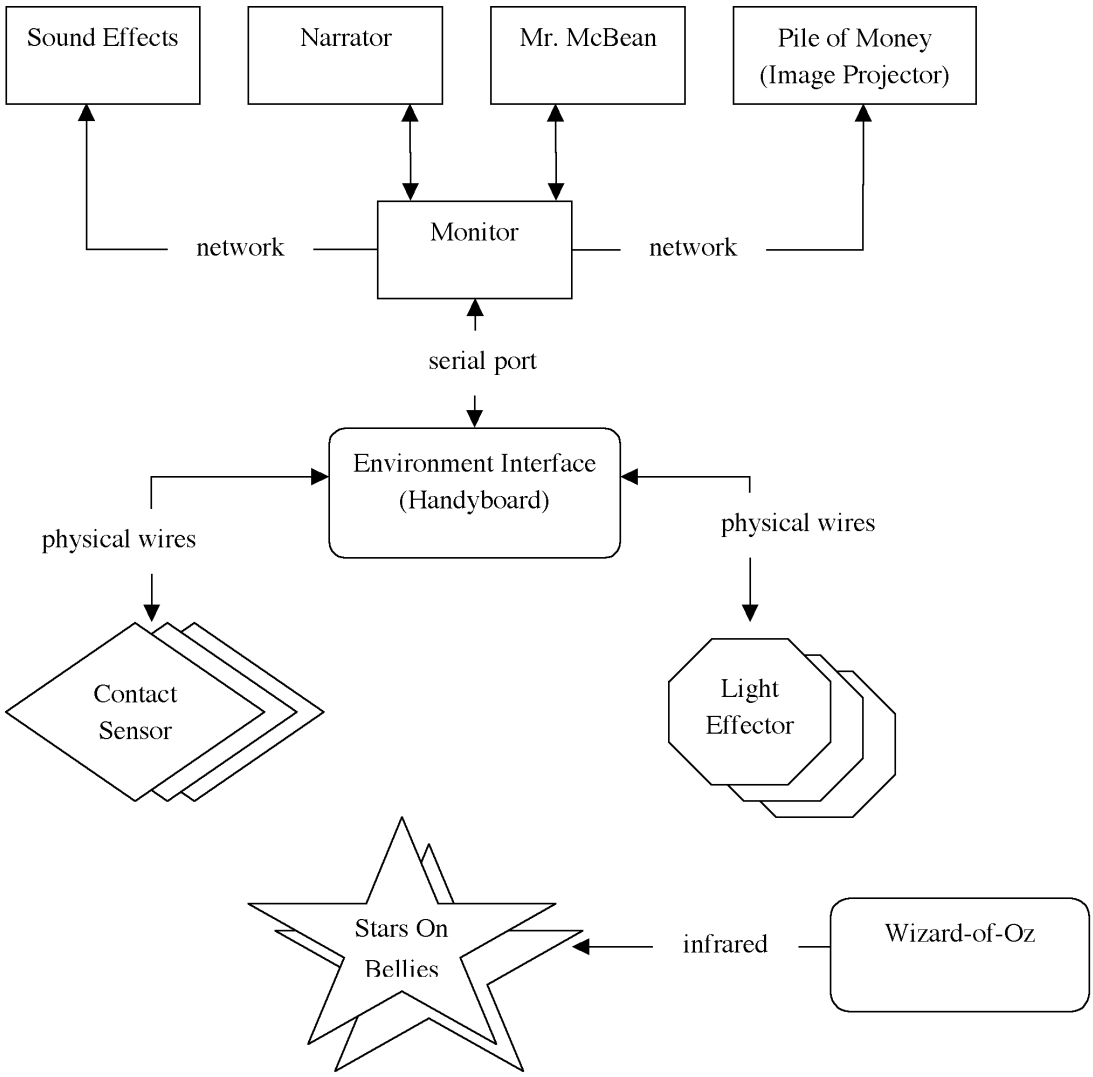

Figure 5.6: A diagrammatic overview of the technology underlying the Sneetches StoryRoom. The various sound generating applications reside on different Macintosh computers in our lab and communicate via a file-based messaging protocol with the Monitor. The *Money* is attached to a presentation projector. The monitor is an application that receives inputs from sensors and issues commands to actuators and the registered applications. The *Stars* are turned on/off by a wizard-of-oz, using an infrared control.

3. *Money* to update the projected image by moving an image of a coin from the Sneetches side over to Mr. McBean's side;

4. Environment Interface to blink the light effector A.

The *Star* bellies were controlled via infrared signal by a person (Wizard-of-Oz), who signaled the stars to appear or disappear as children exit the Star-On and Star-Off tunnels.

5.3.4 Lessons Learned from the Sneetches StoryRoom

The Sneetches StoryRoom helped identify three necessary elements of the conceptual storytelling construction kit: props, low-tech material, and physical symbols (or icons). A story in an interactive environment needs physical props to represent key elements of the story. These props are necessary to concretize the story. Children experience the enriched story from these physical interactions. For example, as children crawl through the *Star-On* box and watch the stars on their bellies glow, they begin to imagine that they were transformed into the Sneetches (figure 5.7).

One idea that surprised the IDT was that children derived at least as much fun from building the props as they experienced the storyroom (figure 5.8). In part, this may have been because they were already expert builders of low tech material, such as cardboard boxes, glue, and crayons. These activities were necessary steps toward their creative product, the StoryRoom.

Because embedding high tech devices within objects can be difficult for young children, and because not all objects can be modified to hold the devices, what was needed was an alternative to augment (figure 5.9) any physical objects with computing abilities. I learned that for children between seven and eleven years old, placing attractive

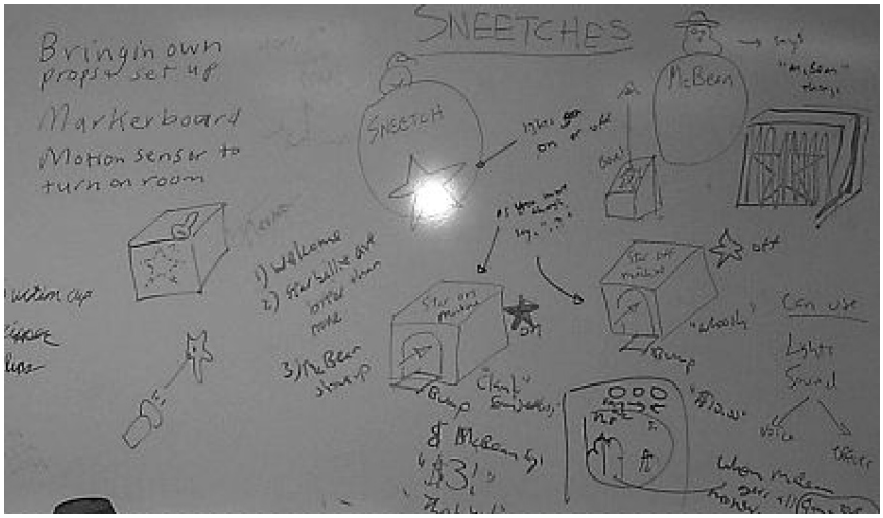

Figure 5.7: A wall sketch result of a Sneetches Room design session.

Figure 5.8: Two child designers working on the *Star-On* box.

Figure 5.9: Some early sketches of sensors, with shapes such as thumb, finger, and hand.

physical icons on a physical object was conceptually the same as augmenting it with computational abilities. So, when a child touched a hand (or thumb) physical icon attached to a prop, she could make-believe that she was touching the prop. And, during construction, when a child connected an icon to a prop, she was *casting* a spell, magically imbuing the prop with the computational feature represented by the icon.

5.4 StoryKit

These StoryRoom experiences helped to identify some, but not all, of the necessary components of the conceptual construction kit (**StoryKit**) [2]. After the Sneetches StoryRoom, I began the next phase of my research, that of designing an authoring system for physical environments. From many design sessions, I discovered that children had difficulty creating stories from nothing. But when presented with a few simple ideas, they were able to quickly weave intricate stories around them. Furthermore, story quality seemed to correlate directly with the level of intrigue of the seed ideas. Here were some example ideas: half a pencil; a torn piece of paper; a keyboard miss-

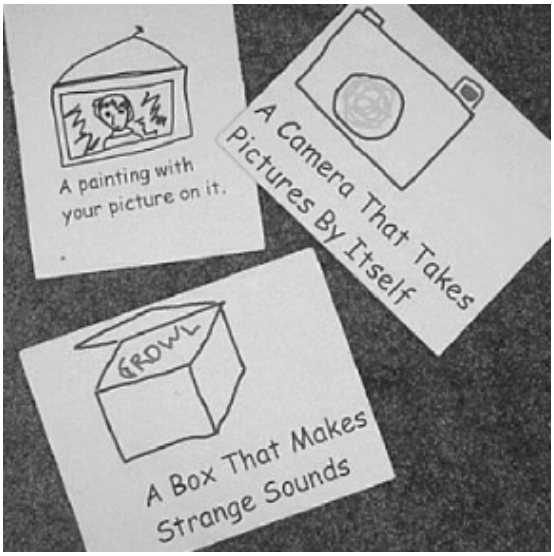

Figure 5.10: Idea Cards help propel children into developing structured stories.

ing the keys T, O, and M; and a bowl of blue tomatoes. All these ideas are unusual. Each was a mystery unto itself. What happened to the other half of the pencil? Why was the paper torn in half? Who took out the missing keys? And how did the tomatoes turn blue?

Interestingly, it seemed that some ideas needed to be "plain," such as a simple box, a tennis ball. These simple ideas served to support the mysterious qualities without adding to the complexity of the story. These seeds of idea were called, **Idea Cards**. They were the next component of StoryKit.

The StoryKit was missing just one more element, a programming system to define the interactions among the computational devices.

Now, the completed StoryKit would have the following elements:

1. Mysterious and simple Idea Cards to help children create stories (figure 5.10),

2. Materials that children can use to construct props from these ideas (figure 5.11),

Figure 5.11: Low tech materials are easy for children to construct play things.

Figure 5.12: High-tech devices embedded within physical icons project magical qualities to a child.

3. Physical icons that represent computational abilities to augment the props (figure 5.12),

4. Programming system to define the interactions among the devices (figure 5.13).

Here is how children might create a StoryRoom. They open the StoryKit and select some Idea Cards; weave a story around the ideas; create physical props, using low-tech material in the kit, to concretize the concepts; decide the interactions that should occur; augment the props with interactive computational abilities by connecting the physical

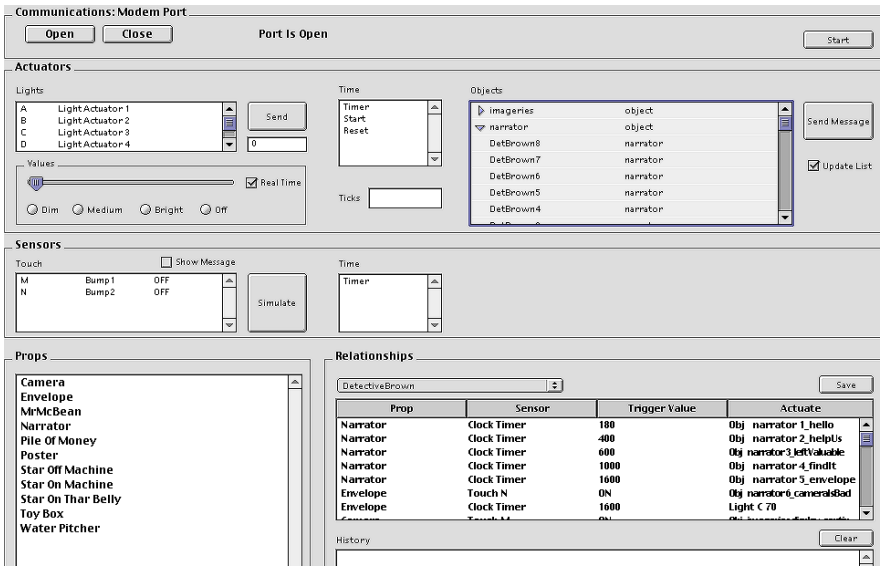

Figure 5.13: An example of a child-unfriendly control panel to define the interaction rules for StoryRoom.

icons onto them, and program the interaction rules.

5.4.1 An Early Design for Defining Device Interactions

My earliest design of a system to define interactions was written using RealBasic. It had a simple text-based control panel interface, called the Monitor. The control panel

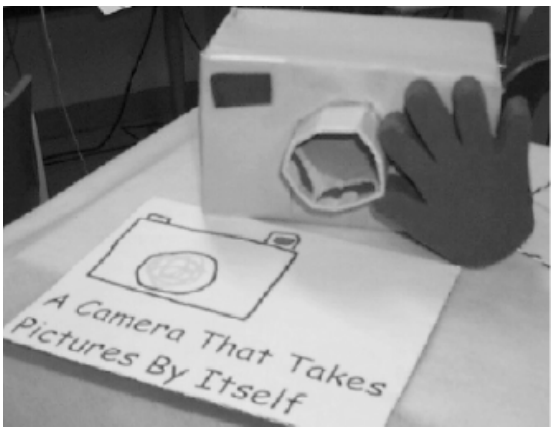

Figure 5.14: An idea card and prop.

included:

1. List of props.
2. List of actuators.
3. List of sensors.
4. List of objects.
5. List of relationships.

The actuators and sensors were hard-coded. The objects list was automatically generated, in real-time, and contained all the registered StoryRoom savvy objects in the environment. A StoryRoom savvy object registered with the Monitor its identity (e.g. , Imageries, Narrator) and its services. For example, a Narrator object might contain the sound clips Detective Brown narration one, Detective Brown narration two, etc. Another example is the *Money* object in the *Sneetches* StoryRoom (5.3.2), which responded to two commands: spend and reset. The spend command triggered the *Money* object to update the amount of money transferred to *Mr. McBean*'s side and redraw the projected image. The reset command tells *Money* to return all the money back to the *Sneetches* update the image.

The way to create relationships between sensors and actuators was by a drag-and-drop interaction.

1. Create a new relationship by dragging a prop onto the Relationships table.
2. Drag a sensor from the sensor list over to the new relationship.
3. Type into the Trigger Value cell a value for the sensor to activate an event.

4. Select, from either the actuators list or the objects list, a desired activity and drag into the new relationship.

5.5 It Is Not Always About Sophisticated Technology

From creating PETS and the various StoryRooms, these experiences suggested that perception was more important than technology to create a convincing story environment. It was not necessarily the level of sophistication in technology that fostered the creative and imaginative thinking in our children. Instead, a technology could create attractive and entertaining storytelling environments when it has the following:

1. Tools for children to be creative.
2. The ability for children to affect and control their space.
3. Simple interactions.
4. Ways to help children begin stories.
5. Hints to help children understand the story.
6. Technology that is physically attractive to children.

In PETS, the robot's physical appearance drew children near it; they controlled its emotions; and the spoken words helped guide them through the story. In the Sneetches room, children actively changed their own appearance and their surroundings by using the props; the Narrator and Mr. McBean offered both hints and explained to them how they were involved in the story; the contact sensors were simple to detect and easy to change; and since children built the props, they were attractive to other children.

5.6 A Programming System for Physical Interactive Environments

Throughout my research, I used crude user interfaces to create the device interaction rules for the physical environment. It was the weakest component of the StoryKit. Not only was the application textual, it also required heavy manual editing. Clearly the programming interfaces were inappropriate for young children. In fact, whenever the children finished a StoryRoom prototype, they needed my help to manually create the interaction rules for the environment. Since my goal was to provide a kit for children to be completely autonomous in their creative activities, I began work on a new programming approach for children to control StoryRoom interactions. The IDT's brainstorming activities led to three conceptual user interfaces: 1) arrow notes (figure 5.15), 2) comic strip (figure 5.16), and 3) time line. Because arrow notes was similar to time line, I will describe just the *Arrow-Note* and the *Comic-Strip* interfaces.

5.6.1 Arrow-Notes

The arrow notes user interface used colored boxes. The boxes represented objects, events, and branching tests. Overlapping boxes created relationships. For example, in figure 5.15, the overlapping "door," "if", and "button pushed" boxes mean "if the button that is on the door is pushed." The arrows were used to indicate concurrent events.

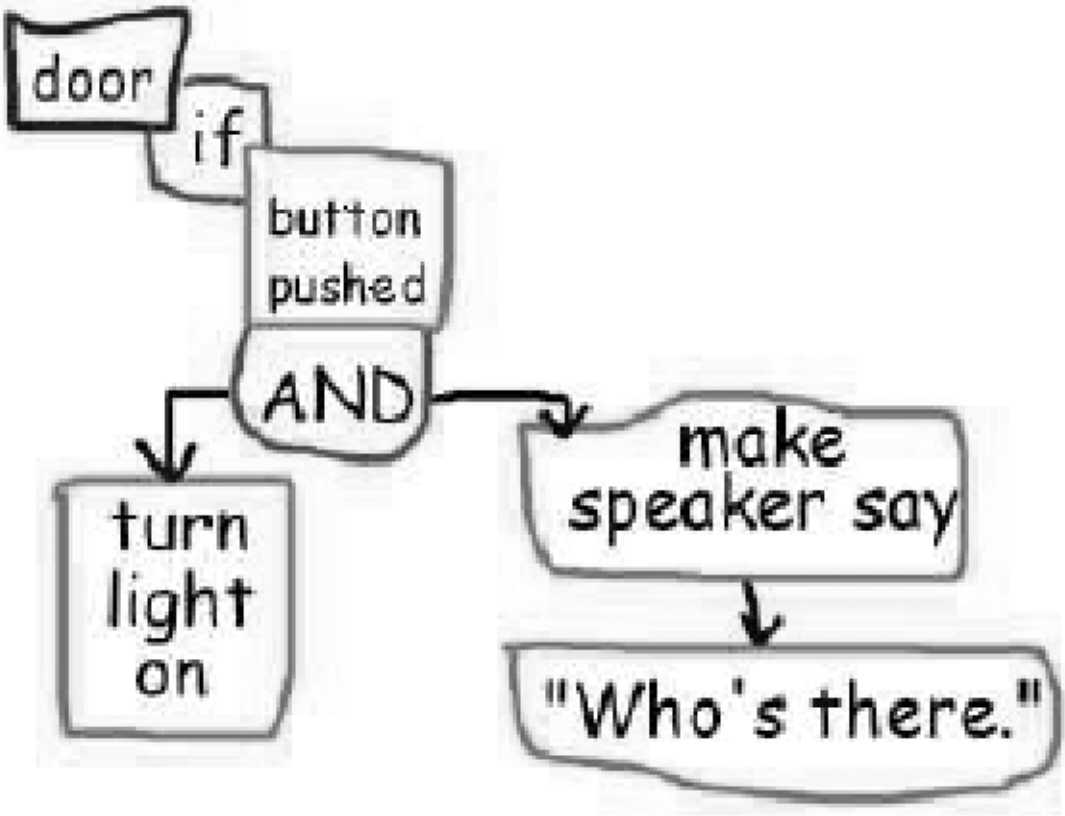

Figure 5.15: An example *Arrow-Note* styled program. These notes represent a portion of the program, and it is interpreted as: If the button on the door is pushed, then a) turn the light on, and b) make the speaker say “Who’s there?”

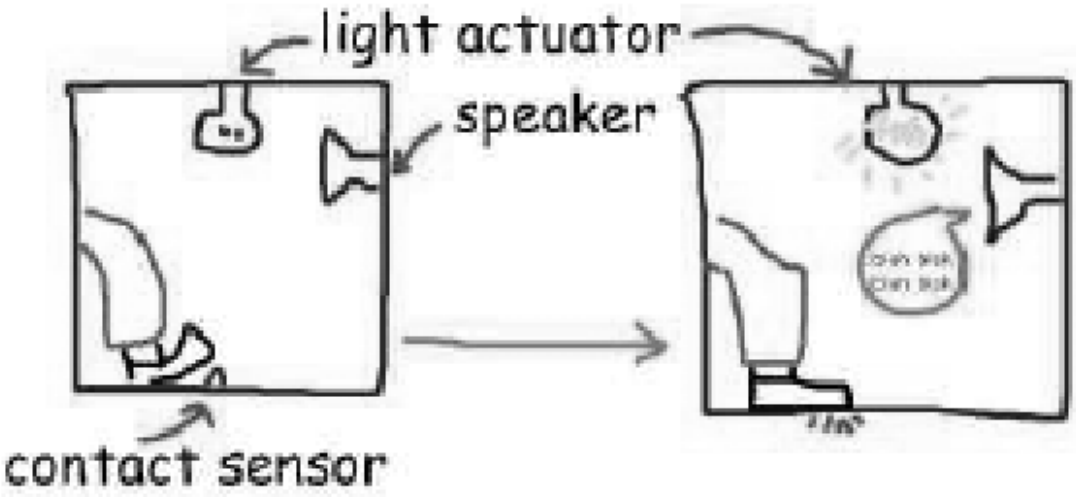

Figure 5.16: An example *Comic-Strip* styled program. This strip represents a typical programming "statement." The *before* frame shows a floor based contact sensor, a light actuator, and a speaker, all in the off position. In addition, the leg indicates a triggering condition. The *after* frame shows what happens to the room when the trigger occurs. The light comes on and the speaker makes sounds.

5.6.2 Comic-Strips

The *Comic-Strip* followed a programming-with-demonstration approach. Similar to KidSim [21], it used *before* and *after* frames to indicate activation rules. In figure 5.16, the frame pair means, "When I step on the sensor that is on the floor, then the light should come on and the loudspeaker should make a sound." Notice that it was possible to create complicated rules involving multiple sensors and multiple effectors.

5.6.3 Take Away the Screen

These designs represent two visual programming categories. The arrow-note is a flowchart, and the comic-strip is a visual production system. The fact that they were the design results of an intergenerational design team suggested that both adults and chil-

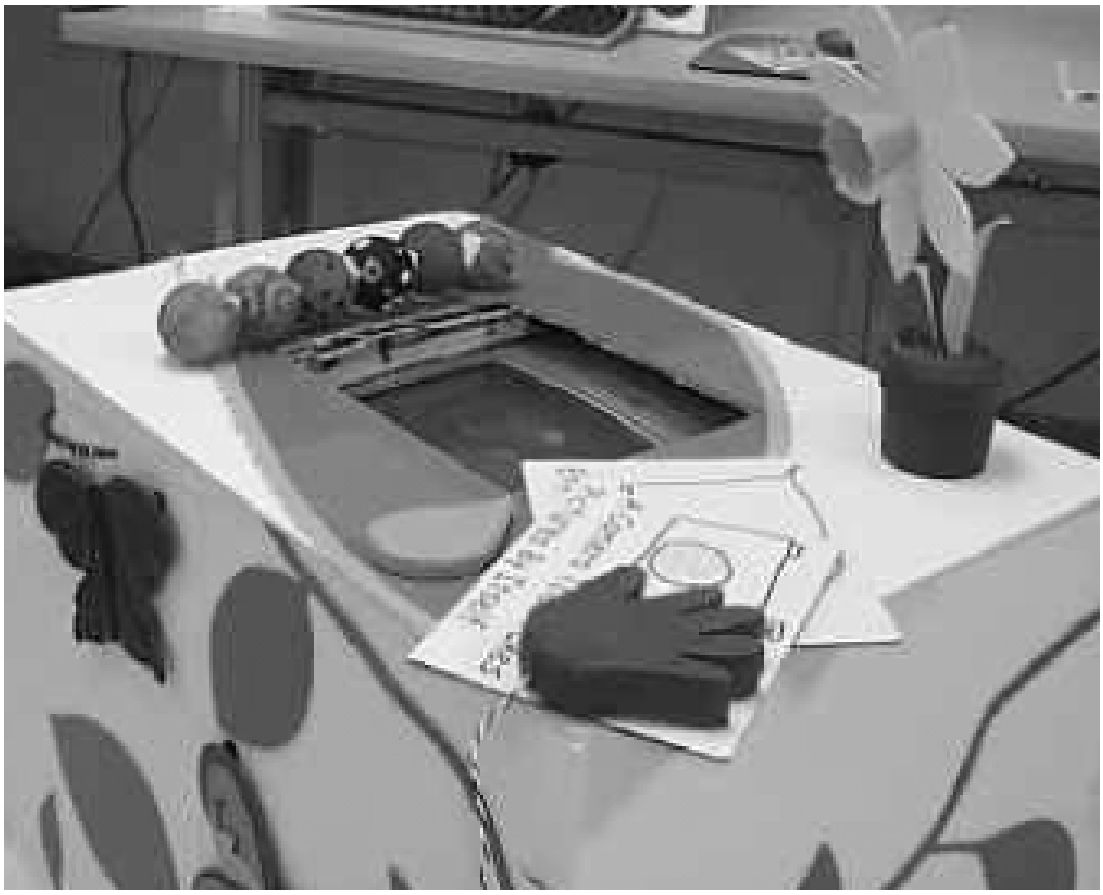

Figure 5.17: A conceptual Storybox. To the left was a StoryWorm made from individual worm segments. The head of a worm would "remember" a story. Each segment "remembered" a line (or interaction rule) in the story. The flower was a microphone for children to record sounds. The monitor in the center would display the different lines of a story. Each line corresponded to a worm segment next to it.

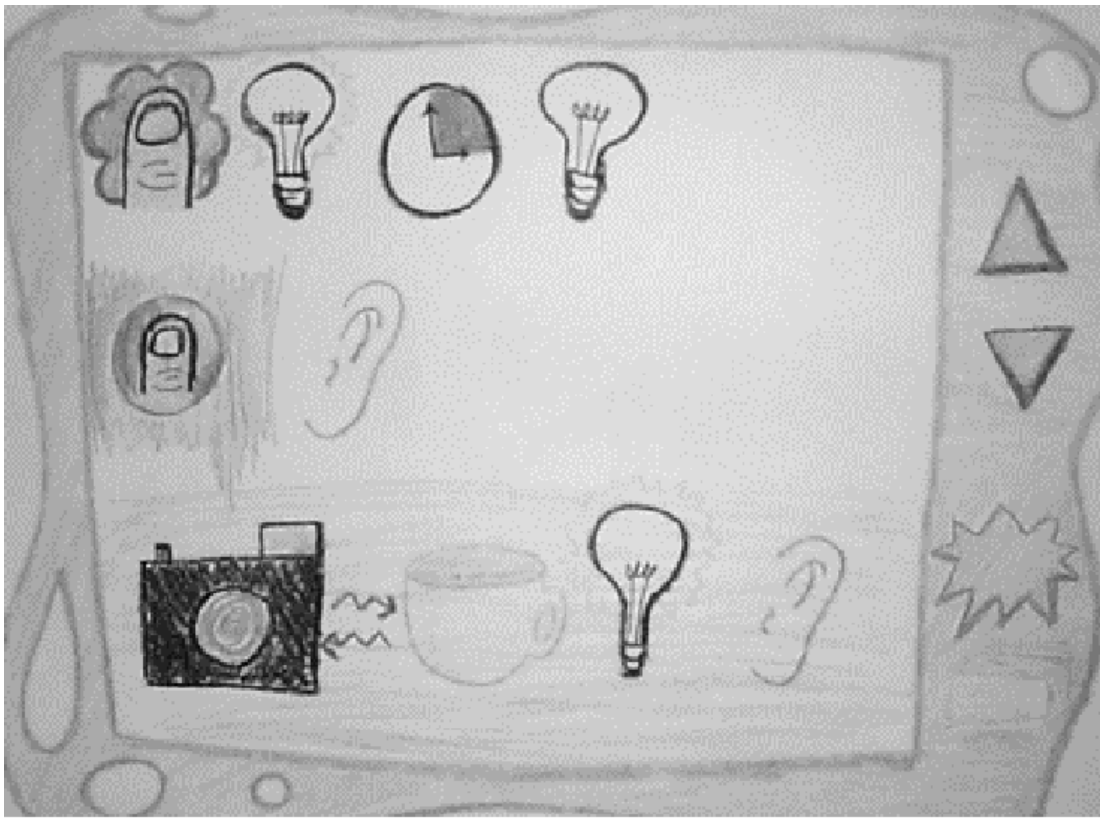

Figure 5.18: Some conceptual iconic sentences. The top line means “press the flower and the light stays on for 15 seconds.” The third line means “when the camera and the cup are near each other, the light comes on and the ear will listen.”

dren were able to understand these two visual languages. Given these two promising concepts for a graphics-based programming-with-example system, the IDT developed some scenario walkthroughs to gain insights into the programming activities within a physical environment. Some questions still remained: Should children use the graphical systems as the primary input model for programming, and then debug by interacting with the physical icons? Or, should children interact directly with the physical icons to demonstrate programming intentions, and then use the graphics system for review and editing?

An interesting thing occurred during these brainstorming sessions. Either the children

weren't using the screen at all, because they preferred the tangible elements of the programming system. Or, they kept dividing their attention between the physical interactions and viewing the computer screen to see whether what they were doing was being correctly monitored. In the first case, the screen became superfluous. While in the second, the act of always going back to the visual display made for an awkward programming process. I relied heavily on pictorial representation of programming because it was conceptually easier to understand. The surprising observation was that children apparently needed to make mental connection between the symbols on the screen with their physical counterparts in their environment, and not all the children we worked with had this ability yet.

So, could it be that for children, a programming system that relies on direct physical manipulation of concrete objects can be more natural? Educators have for a long time postulated that concrete learning is innate in young children, and effective even as they progress into abstract reasoners (e.g., [85]). At this point, I suggested that young children can and should use direct physical actions to author StoryRooms, from constructing props to programming interaction rules.

It was time to take away the artifacts of conventional desktop computing interfaces. In the next chapter, I will describe a completely physical programming approach that I developed. After that, I will describe the two studies that helped convince me that 1) a concrete and physical programming metaphor can be easy to understand, 2) kindergarten students can become programmers using this approach, and 3) they can create customized device behaviors in their physical environments.

Chapter 6

Physical Programming

In the Introduction, I presented the idea of **physical programming** by way of this working definition.

> Physical programming is the generation of computer programs by the physical manipulation of computationally augmented (or aware) objects in a ubiquitous computing environment.

I also described in Chapter 5 the motivation for developing this programming approach. But I have not specified the elements of the language, such as grammar and alphabet. In this chapter, I will:

1. Discuss the relationship of physical interactive environment to theoretical machines.
2. Refine the definition of physical programming.
3. Describe an implementation of the physical programming language within the StoryRoom context.

4. Describe the enabling technology behind StoryRoom and physical programming.

5. Discuss the limitation of the implemented language.

6.1 Physical Interaction Environments and Automata

Conceptually, a **physical interactive environment** (PIE) is a physical environment that contains devices and a set of instructions. The devices are sensors and actuators. The instructions dictate the behavior of the devices. Ubiquitous computing environment is an instance of physical interactive environment. Since StoryRoom is a ubicomp environment, it is also a PIE.

A physical interactive environment can be composed of a) your home, b) a central air conditioner, and c) a thermostat. The environment is the home. The thermostat contains a temperature sensor, an actuating mechanism, and a temperature setting interface. The temperature that you set is an instruction. The interaction may work this way:

1. You set a temperature *A* for your home.

2. If the ambient temperature, detected by the thermostat, is above *A*, actuate the air conditioner.

3. If the ambient temperature is at or below *A*, turn off the air conditioner.

The *Sneetches* StoryRoom (5.3) can also be described as a PIE. The physical icons such as thumbs and lights are devices. The instructions reside as a software program in the Monitor application. And the room is the physical environment.

6.1.1 Deterministic Finite Machine

The simplest automata is the deterministic finite machine. Below I will show that a PIE and its components make a finite state machine. Then, I will show how a deterministic finite machine can be converted to a PIE.

Elements of a physical interactive environment can be mapped to a finite state machine. The physical elements of the home example, the air conditioner, the thermostat, the temperature setting, and the house, make up a physical state machine. Let me now present this more formally.

Define augmented deterministic finite state machine
PIE = (S, σ, A, α, I, E, Q, Σ, δ, q_0, F) as

S is the set of sensors in the environment.

σ is the number of sensors.

A is the set of actuators in the environment.

α is the number of actuators.

I is a set of ordered pairs. For each ordered pair, the first element is from S, and the second element is a finite set of integers representing the sensor‘s possible values.

E is a set of ordered pairs that represents the finite set of possible values for each actuator.

Q is the finite set of states, each state $q \in Q$ is a set of ordered pairs. In each pair, the first element is an actuator from A and the second element is a valid value (defined by E) of that actuator. Each state

contains one possible instantaneous concurrent values of all devices in A.

Σ is the alphabet. Each alphabet character is a σ-tuple. Each element of the σ-tuple is an ordered pair, in which the first element of the pair is a sensor $s \in S$ and the second element a valid value (defined by I) for the sensor s. Thus, a σ-tuple represents one unique set of concurrent values detected by all the sensors in the physical environment.

δ is the total transition function.

q_0 is the start state.

F is the set of accepting states.

Notice that an input *string* for *PIE* is simply a temporal sequence of σ-tuples. The start state q_0 is an arbitrary moment in time, immediately before all the sensors and actuators are turned on. In my home example, the input would simply be a string of temperature readings.

For StoryRooms, a special purpose *PIE*, the semantics of F is determined by the "story" crafted by the storyteller. If a story is never ending, then, F is empty. If a story is a mystery with a solution ending, then that end can be an accepting state. In short, the status of a state as accepting or not is at the semantic level of the story but is not restricted by the machine. There is another way to look at this. Consider a very simple setup. You have a keyboard, a monitor, and a basic text editor program. The keyboard is a sensor; the monitor is an output; and the text editor is a finite state machine. Suppose that every time you press a key, the text editor outputs the corresponding alphabet onto the monitor. Here, a natural accepting state would be some meta-key combination which saves the string to disk.

The transition function describes the relationships between S and A. In the *PIE*, sensors (or timers) are needed to activate a transition between two states. A side effect of this transition is that the settings of the actuators (i.e., sound, light, wind, or springs) change. Note that the alphabet Σ and Q can be extremely large. While this is not a theoretical concern, it is certainly an important practical issue for any implementations. Take StoryRoom and its children users as an example. A light sensor may trigger any integer values between 0 and 100, where 0 is complete darkness and 100 is full brightness. Suppose children only cared about dark and light, and they don't care about the gradations in-between. Or, they just wanted to turn a light effector on and off. In both cases, only 2 out of 101 possible input and output values were "practical." How to reduce the sizes of I and E, and correspondingly, of Σ and Q, then becomes a human-computer interface question for the design team to answer, to filter out what is meaningful, what is necessary, what is frivolous, and what is redundant.

As an example, the air-conditioned home can be represented this way.

AC Home = (S, σ, A, α, I, E, Q, Σ, δ, q_0, F)

S = $\{\text{Therm}\}$.

σ = 1.

A = $\{A/C\}$.

α = 1.

I = $\{(Therm, \{1, 2, 3\})\}$. For simplicity, I am using a thermometer that can report only three possible temperatures.

E = $\{(A/C, \{ON, OFF\})\}$.

Q = $\{\{(A/C, ON)\}, \{(A/C, OFF)\}\}$.

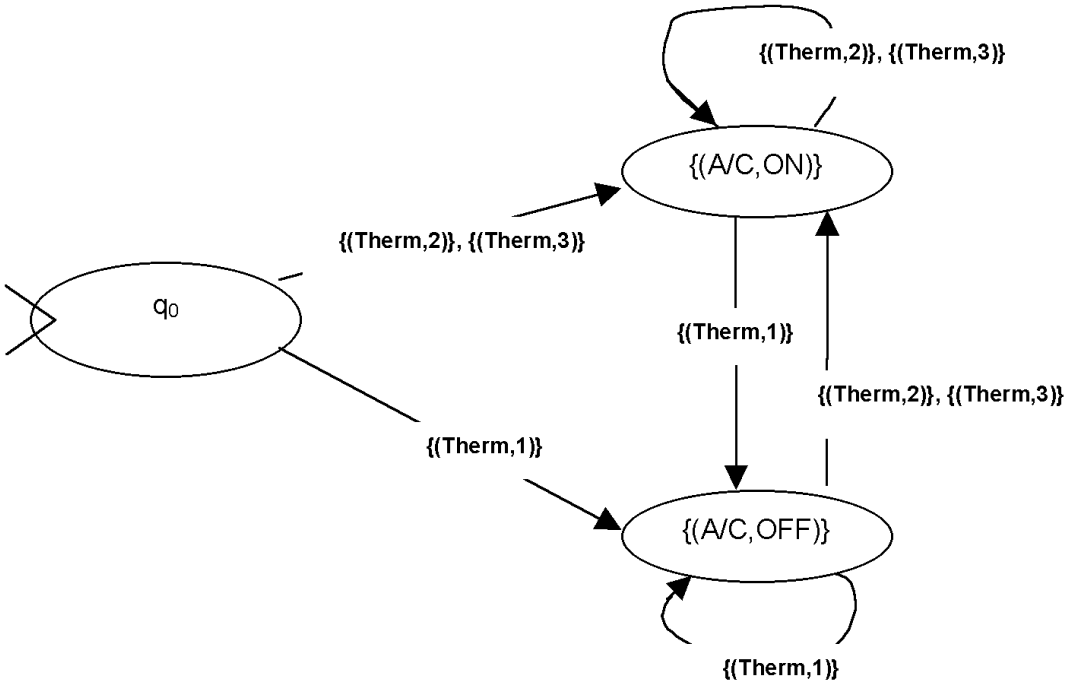

Figure 6.1: The state diagram for the home example.

$\Sigma = \{((Therm, 1)), ((Therm, 2)), ((Therm, 3))\}$.

δ is the total transition function (figure 6.1).

q_0 is the start state.

F $= \{(A/C, ON), (A/C, OFF)\}$.

The state diagram is an encoding of the instructions of a PIE. It tells us that given an input character (thermometer reading) and a source state, move to a new state and activate the actuators of this new state. As a short hand, I will refer to a physical interactive environment instruction as a **Physical Interaction Instruction** with the following definition.

Definition 2 *A Physical interaction instruction is one state transition of the deterministic finite machine encoding of a physical interactive environment.*

Thus, a PIE's set of physical interaction instructions (PII) describes the environment's interactive behavior. Using the state diagram of the home example, we can see that one PII is: if the thermometer is reading a temperature of 3 and the air conditioner is off, then transition to the "air conditioner is on" state and activate the air conditioner.

6.1.2 Transformation of DFA to a PIE

The task of transforming of a deterministic finite automaton to a Physical Interactive Environment is to convert *DFA* = (Q_{DFA}, Σ_{DFA}, δ_{DFA}, q_{0DFA}, F_{DFA}) into *PIE* = (S, σ, A, α, I, E, Q, Σ, δ, q_0, F).

1. Assemble $|\Sigma_{DFA}|$ binary valued sensor components, such as push-button. Let this set of sensors be S, and $\sigma = |S|$. I = S $\times$ $\{\{1, 0\}\}$.

2. Assemble an actuator component, containing two sub-assemblies: 1) a binary valued actuator, such as a fan, and 2) a light bulb. Let the sub-assembly 2 light bulb be the *acceptor light*. Use this actuator component as the start state q_0. If the start state $q_{0\,DFA}$ is an accepting state, turn on the *acceptor light* on q_0.

3. Assemble $|Q|$ binary valued actuator components, as in step 2. Let this set of actuators be A, and let $\alpha = |A|$. Turn on the *acceptor light* for any actuator whose corresponding state in the DFA is accepting. F is this set of actuators with the acceptor light on. E = A $\times$ $\{\{1, 0\}\}$

4. Q is a set with α sets. Each $q_i \in$ Q is a set containing α couples. Construct q_i this way:

   for each $q_i \in Q_{DFA}$

for j = 1 to α

assign q_i to the first element of the couple.

if i = j then assign 1 to the 2nd element of the couple, else assign 0.

next

next

5. To construct Σ, first enumerate the 2^σ possible combination of concurrent values of the sensors in S as an σ-bit binary string. Then continue with:

   for each σ-bit binary string

   create a σ-tuple containing σ-couples

   for i = 1 to σ

   assign $s_i \in$ S to the first element of the couple

   if $bit_i = 1$ then set 1 to the 2nd element of the couple, else set 0.

   next

   next

6. To convert the transition table T_{DFA} for δ_{DFA} into a table T for δ, change each row in T_{DFA}

   from

   $(in_i, st_i) \rightarrow (st_k)$

   to

   $(((in_1, 0), (in_2, 0), \ldots, (in_i, 0/1), \ldots, (in_\sigma, 0)),$

   $\{(st_1, 0), (st_2, 0), \ldots, (st_i, 1), \ldots, (st_\alpha, 0)\})$

$\rightarrow$

$\{(st_1, 0), (st_2, 0), \ldots, (st_k, 1), \ldots, (st_\alpha, 0)\}$.

6.1.3 A More General Definition of the PIE Deterministic Machine

The *PIE* in 6.1.1 represents a machine that only inspects the sensors in the environment to determine transitions. A more general machine would be able to decide a transition based on the current values of both the sensors AND the actuators. For example, a physical interaction instruction might say, "if contact sensor A is on and light actuator X is off then turn light actuator X on."

This more general machine can be described as an augmented deterministic finite state machine PIE_{SA} = (S, σ, A, α, τ, I, E, Q, Σ, δ, q_0, F)

S is the set of sensors in the environment.

σ is the number of sensors.

A is the set of actuators in the environment.

α is the number of actuators.

τ = σ + α.

I is a set of ordered pairs. For each ordered pair, the first element is from S, and the second element is a finite set of integers representing the sensor's possible values.

E is a set of ordered pairs that represents the finite set of possible values for each actuator.

Q is the finite set of states, each state $q \in Q$ is a set of ordered pairs. In each pair, the first element is an actuator from A and the second

element is a valid value (defined by E) of that actuator. Each state contains one possible instantaneous concurrent values of all devices in A.

Σ is the alphabet. It contains both sensor and actuators. Each alphabet is a τ-tuple of ordered pairs. In the first σ ordered pairs, the first element is a sensor from S and the second element a valid value (defined by I) of the sensor. In the remaining α pairs, the first element is an actuator from A and the second element a valid value (defined by E) of the actuator.

δ is the total transition function. the transition function of *PIE*.

q_0 is the start state.

F is the set of accepting states.

6.1.4 Automata With Memory

I have just shown that a PIE is a deterministic finite machine. Can the PIE be more powerful? The difference between a finite machine and all other more powerful machines is access to memory. Just beyond the finite machine is the one-counter machine. It is a finite machine that can read and increment/decrement integer values to a single memory slot. This may not appear exciting, until you realize that if a PIE has this memory and its corresponding operations, it can have a timer/counter/clock.

The next machine is the push down automata (PDA), with access to a stack. But its power derives from the unattainable physical attribute of an infinite stack memory. This appears to be a lost cause. But, we are saved by a two-counter machine. This is similar to the one-counter, except it has two independent memory slots to hold integers.

Turns out the two-counter machine, with just 2 slots, can emulate the PDA. More incredibly, it can emulate a RAM machine [1].

This is truly amazing. If a PIE can emulate the two-counter machine, then it is as powerful as any others. But what good is memory? What else can we do in addition to having a timer?

The following example will perhaps reveal a very nice property of "memory." Suppose we have a PIE enabled house with n rooms. Inside each room is 1 toggle light switch. Now, suppose we want this PIE to trace our activities and execute according to this rule: In whatever order I turn the light on, when I exit the house, the lights should be turned off in the reverse order.

Of course we can encode this problem as a finite state machine. But, it can be quite large, to the order of $O(n!)$. On the other hand, a PDA can offer a more compact solution by pushing the identities of the switches onto the stack as they are turned on, and turning the lights off as their identities pop off the stack.

The counting, stack, and RAM machines are more powerful than the FSA because they remember, and they can make decisions about the future by revisiting the past. So it would be in my interest to add memory to the PIE. I can even get around the problem of infinite memory by using the two-counter emulator. But Emulating PDA and RAM requires many steps in the two-counter machine, and since each of these steps may correspond to some physical programming activities in the PIE, a PIE equivalent of the two-counter machine may be unwieldy. For now, let me just take a look at a PIE with a finite tape. (Unlike theoretical machines, the physical environment is a finite place.)

[1] The various machines are very nicely described in Floyd and Beigel's text [34].

This PIE with a finite tape can make decision about a state transition based on the current of sensors, actuators, AND a recorded history! With this new capability, we can encode statements such as: If sensors A and B have never been triggered, activate actuator X. There are some more powerful implications. First, we can supply a **looping** construct. Instead of flattening out a repeated sequence of commands, we can collect the sequence and impose a counting modifier. In addition, the tape supports the **variable** construct.

But now comes the difficult part. For children, what are the physical symbols of memory? What kind of tools, metaphors, and interaction rules do we need to support the manipulation of both reading and writing memory? This was the final part of my work and it became immediately clear to me that much more time was needed for both the adults and the child designers to work on this problem. There were two outstanding issues. First, the adults were unable to clearly communicate the ideas of memory to children. And second, prior IDT work that were related to this problem resulted in complex and convoluted user interactions. Therefore, at this moment, because of time constraints, I cannot currently offer any definitive insights.

6.2 A Refined Physical Programming Definition

The working definition of physical programming included a rather vague term: computer programs. Based on my discussion in 6.1.1, I can replace *computer programs* with **physical interaction instruction**. Now a refined definition is:

Definition 3 *Physical programming is the generation of physical interaction instructions by the physical manipulation of computationally augmented (or aware) objects in a ubiquitous computing environment.*

The definition of finite machine only describes the content of the transition table, but not how the table's content is "written" or created. Similarly, the definition of a physical interactive environment also does not say anything about creating physical interaction instructions 6.1.1. This is where *physical manipulation* and *generation* come in. Rather than using the traditional model of a user typing or drawing program code, I want to explore new ways for the user to create program code (physical interaction instruction) by way of syntactically meaningful physical gestures.

Definition 4 *Physical syntax is a set of physical gestures that are used to interact with computational objects in order to create physical interaction instructions.*

Similar to the text editors used to create conventional program code (Java, C, etc.), Physical programming also requires tools for the user to create the physical instructions. In the next section, I describe one implementation of physical programming for StoryRooms and its suite of programming tools and physical syntax.

6.3 Implementation

Because the StoryRoom was designed for young children, its programming tools were tailored to their abilities. One metaphor that worked well with the IDT child partners was **magic**. That is, as a child was creating physical interaction instructions, she was casting magic **spells**. Furthermore, she could create magic when she was a **wizard**, but not at other times.

The programming tools to support this approach was comprised of a set of tangible *tools* and *icons*. The *tools* included a magic wand, a wizard's hat, and a once-upon-a-time lever. When a child wore the hat, she became a wizard. When she took off

the hat, she became normal again. The once-upon-a-time lever was a large switch for turning on/off the StoryRoom machine. The magic wand was used to generate physical instructions. The *icons* were the sensors and actuators, such as hand, light, fan, foot, and blinker. The shape of the icons imply functionality that could be used to augment interactivity onto props. For example, a hand implies touchable, a blinker implies “look here.”

The primitive physical gestures for StoryRooms were:

1. Wear the hat. Enter the programming mode of StoryRoom.
2. Put the hat back. Exit the programming mode of StoryRoom.
3. Pull the once-upon-a-time lever down. Activate the StoryRoom machine.
4. Pull the once-upon-a-time level up. Turn off the StoryRoom machine.
5. Press the *new-spell* button located on the magic wand. Start a new physical interaction instruction.
6. Wave magic wand over an icon. Include this icon into the current physical interaction instruction.

The physical syntax for creating a physical interaction instruction was:

Press the *new-spell* button .
For each sensor/actuator that I want to include into the instruction
    Wave the magic wand over the object.
Next

The StoryRoom had two distinct modes: authoring and playback. In the authoring mode, the programming system captured activities and saved interaction instructions into a database. In the playback mode, the system monitored sensor events and referred to this database to trigger actuators. A child initiated the authoring mode by becoming a wizard. When she wore the wizard's hat from a "magic table" and took the magic wand, she became a wizard and could cast spells onto the physical icons in the StoryRoom. By returning the hat and the wand to the magic table, she turned off the authoring mode (figure 6.2).

To create relationships among the physical icons, the child wizard waved the magic wand over any icons that she wanted to be within an instruction. For example, if the wizard wanted a blue light to turn on when a red hand was pressed, she first presses a *new-spell* button on the wand. Then, she waved the wand over both the blue light and the red hand. To the child wizard, she had just created "invisible wires" between these icons so that the red hand had control over the blue light (figure 6.3). A generated physical interaction rule is:

$$
\begin{aligned}
&(((sensor_1, 0), (sensor_2, 0), \ldots, (redhand, 1), \ldots, (sensor_\sigma, 0)), \\
&\qquad \{(state_1, 0), (state_2, 0), \ldots, (currentstate, 1), \ldots, (state_\alpha, 0)\}) \\
&\quad \rightarrow \\
&\qquad \{(state_1, 0), (state_2, 0), \ldots, (bluelight, ON), \ldots, (state_\alpha, 0)\}.
\end{aligned}
$$

The *new-spell* button was the programming language equivalent of the semicolon. Each press of the button 1) closed the currently recorded set of sensors and actuators, and 2) began another physical interaction instruction. Multiple sensors or actuators within a rule were treated to have *AND* relationships. When several rules shared the same sensor, they were interpreted to have *OR* relationships. For example, given sensors A, B, C, D, and actuators X and Y, if I want X to be actuated when A, B, and C

Figure 6.2: A child creating interaction rules. By wearing the wizard's hat, she knows that she can create magic. The magic wand gives her the power to create "invisible" wires to connect different icons. Here, she is waving the wand over a physical hand icon.

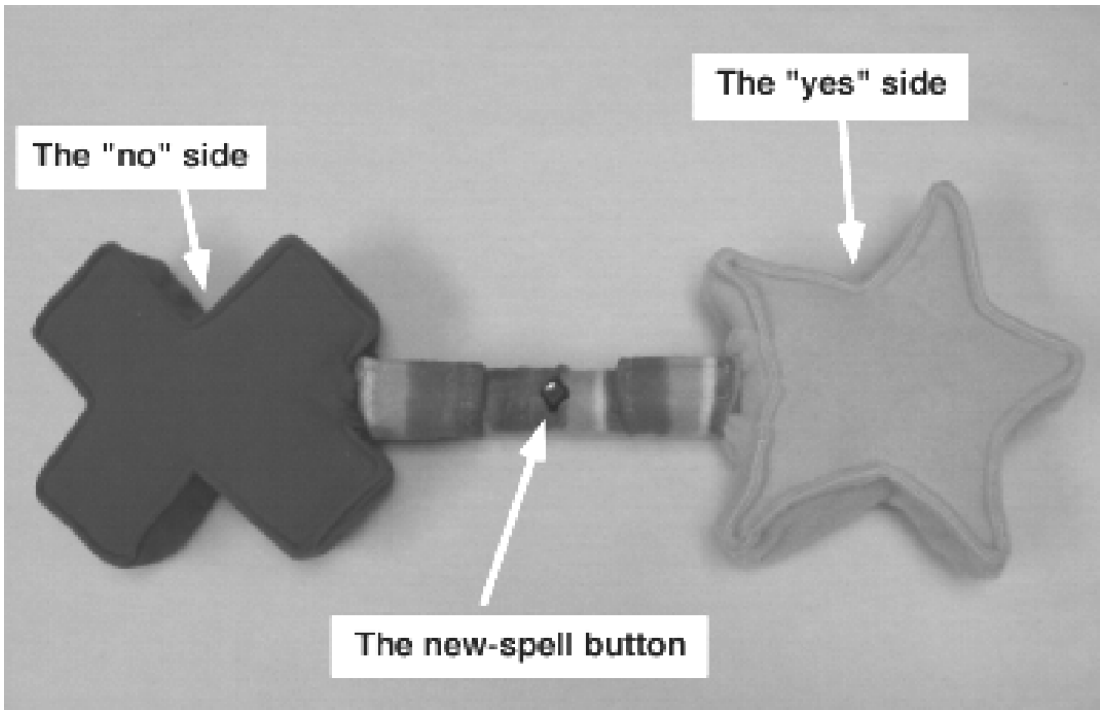

Figure 6.3: The *new-spell* button on the magic wand lets children create multiple independent physical interaction instructions. The *yes* and *no* sides were modifiers to the selection action of the wand. *Yes* meant include the positive action of an icon into a rule. *No* meant include the negative action of an icon. (If an icon was not selected by the wand, it was considered a don't care.)

Figure 6.4: A physical programming example. Given sensors A, B, C, D, and actuators X and Y, actuate X when A, B, and C are triggered simultaneously. The first step is to press the *new-spell* button .

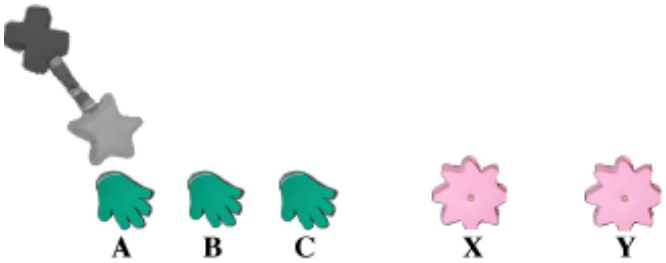

Figure 6.5: Step two: wave the star side of the wand over sensor A.

are triggered simultaneously, then I would press the *new-spell* button , wave the wand over A, B, C, and X (figures 6.4, 6.5, 6.6, 6.7, and 6.8).

The corresponding physical interaction instructions are:

1. $(((A,1),(B,1),(C,1),(D,0)),\{(X,0),(Y,0)\}) \rightarrow \{(X,1),(Y,0)\}.$
2. $(((A,1),(B,1),(C,1),(D,1)),\{(X,0),(Y,0)\}) \rightarrow \{(X,1),(Y,0)\}.$
3. $(((A,1),(B,1),(C,1),(D,0)),\{(X,0),(Y,1)\}) \rightarrow \{(X,1),(Y,0)\}.$
4. $(((A,1),(B,1),(C,1),(D,1)),\{(X,0),(Y,1)\}) \rightarrow \{(X,1),(Y,0)\}.$

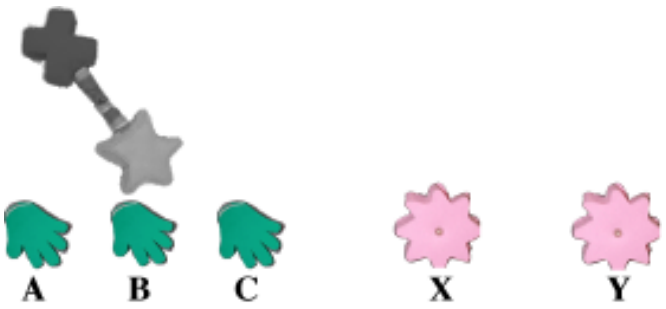

Figure 6.6: Step three: wave the star side of the wand over sensor B.

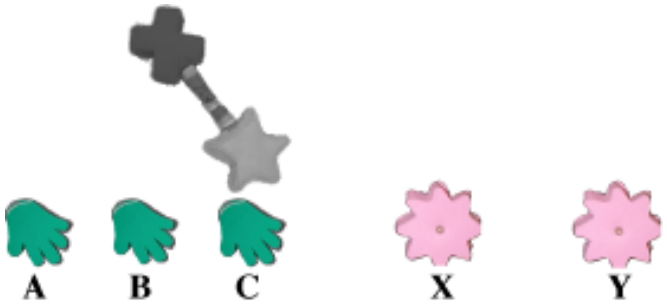

Figure 6.7: Step four: wave the star side of the wand over sensor C.

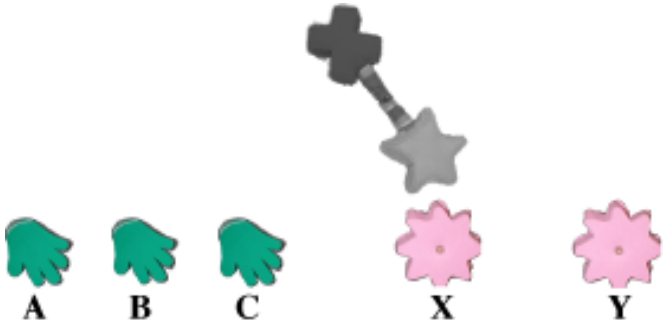

Figure 6.8: Step five: wave the star side of the wand over actuator X.

5. $(((A,1),(B,1),(C,1),(D,0)),\{(X,1),(Y,0)\}) \rightarrow \{(X,1),(Y,0)\}$.

6. $(((A,1),(B,1),(C,1),(D,1)),\{(X,1),(Y,0)\}) \rightarrow \{(X,1),(Y,0)\}$.

7. $(((A,1),(B,1),(C,1),(D,1)),\{(X,1),(Y,1)\}) \rightarrow \{(X,1),(Y,0)\}$.

8. $(((A,1),(B,1),(C,1),(D,1)),\{(X,1),(Y,1)\}) \rightarrow \{(X,1),(Y,0)\}$.

Notice that with the physical approach, five simple gestures—one button **press** and four **waves**—created the eight physical interaction instructions.

If I want X to be actuated when either A or B are triggered, then I would press the *new-spell* button , wave the wand over A and X; followed by *new-spell* button , and wave wand over B and X (figures 6.9, 6.10, 6.11, 6.12, 6.13, 6.14). These six gestures generate 2^6 physical interaction instructions.

Figure 6.9: Another physical programming example. Given sensors A, B, C, D, and actuators X and Y, actuate X when either of A and B are triggered. Again, the first step is to press the *new-spell* button .

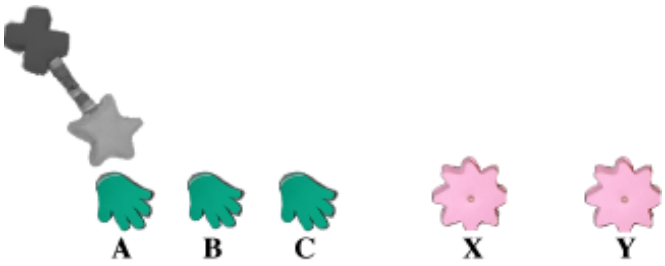

Figure 6.10: Step two: wave the Star Side of the Wand over Sensor A.

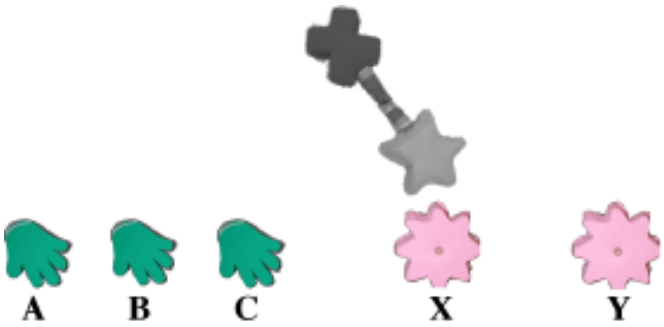

Figure 6.11: Step three: wave the star side of the wand over actuator X.

Figure 6.12: Step four: push the *new-spell* button for a new rule.

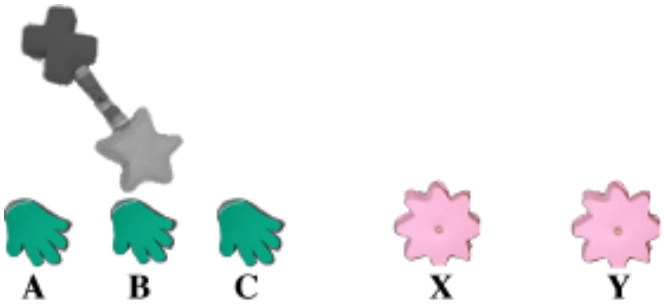

Figure 6.13: Step five: wave the star side of the wand over sensor B.

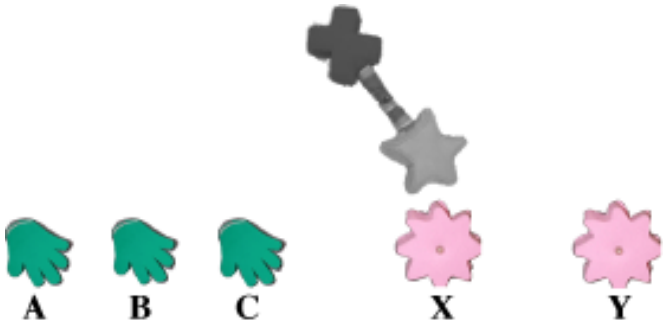

Figure 6.14: Step six: the star side of the wand over actuator X.

The magic wand had two terminals: the star side (Yes) and the X side (No). The star side imposed a positive modifier to the attribute of the physical icon, whereas the X side imposed a negative modifier. For example, given sensors A, B, C and actuators D, E, if I want E to activate when A is triggered AND B is not triggered, I would wave the star side over A and the X side over B. A device that has not been selected is considered a *don't care*. Note: The X side was only implemented in a wired version of the physical programming tools.

6.4 The Enabling Technology to Support Physical Programming

Physical interactions are fundamental in the StoryRoom, whether they 1) occur among the physical icons and children, 2) among the physical icons themselves, 3) between props and the children, or 4) among children themselves. I designed a system to support the first two cases, which required embedded devices (within physical icons) and a communication protocol to control them. In doing so, I came to understand that these devices had to be rugged, durable, and predictable in behavior.

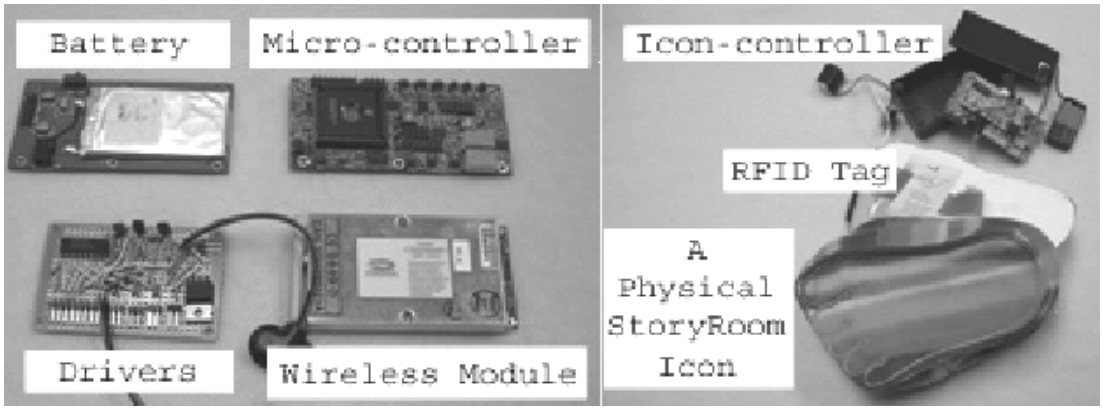

Figure 6.15: The components to a StoryRoom icon controller (left) and the components of a StoryRoom icon (right).

6.4.1 Embedded Devices

The embedded devices, or "icon controllers," consisted of several components (figure 6.15):

- A printed circuit board (PCB) with micro-controller and various general circuits for communications and sensors.
- A battery circuit board.
- A wireless communications module.
- A driver circuit board with custom circuits for controlling the sensors and actuators of a specific type of StoryRoom icon.

The multi-layer PCB micro-controller and battery circuit boards were designed by Eugene Chipman (a graduate student in the department of computer science), and professionally manufactured. The polymer rechargeable battery, with a packaged protection circuit, provided a minimum of 4 hours operation without recharging. The driver circuit boards were built in our lab from basic electronic components, and used external

batteries to drive the higher power consuming actuators such as lights and motors. These four components stack vertically into a single package less than 1” in height and were enclosed within a 2” $\times$ 4” plastic box and embedded into a foamy iconic shell.

A sophisticated wireless modem, the WIT2410 Wireless Module from Cirronet, Inc. [www.cirronet.com] was chosen, primarily because it had extremely low latency. It also had good bandwidth, reasonable size and power consumption, a package that eliminated most of the RF design challenges, and on-board management of the wireless protocol.

There was an unexpected design challenge: the placement of the power socket and the on-off switch of the embedded device. In order for the physical icon to not reveal the technology, the foam interior could not have an opening for charging the battery or one for turning the device on/off. This meant that we had to constantly remove the devices from its outer shell just to perform simple tasks.

6.4.2 Communication Protocol

A StoryRoom application ran on a single computer and monitored the activities of the environment and controlled the states of the icon controllers. Communication between the StoryRoom application and the physical icons follow a three-layered protocol (figure 6.16) similar to the TCP/IP Network model [104]. These include: 1) the wireless layer, similar to the link and IP network layers in the TCP/IP model, 2) the network layer, similar to the TCP network and transport layers, and 3) the application layer. The WIT2410 modules provided the wireless layer. Network layer software, running on both the icon-controllers and the computer with the StoryRoom application, pro-

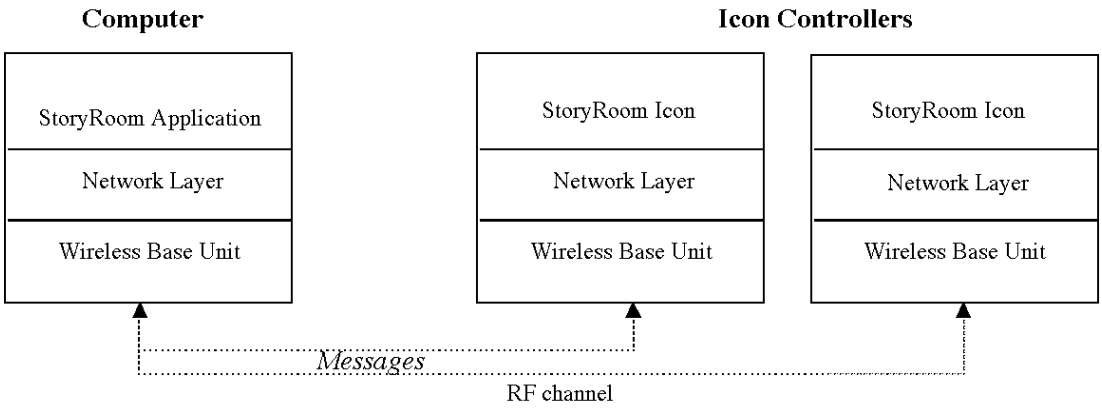

Figure 6.16: The StoryRoom network model.

vided delivery of application layer messages. Application layer software in the icon controllers executed incoming application messages and generated outgoing ones as needed.

6.4.2.1 The Wireless Layer

The WIT2410 was configured to operate in a point-to-multipoint mode where the base unit (attached to the computer running the StoryRoom application) could send and receive data from each remote (attached to an icon controller). Remotes send and receive data only with the base. The base transmitted a broadcast where every remote unit received the data. Units shared the RF channel using time division multiple access (TDMA).

6.4.2.2 The Network Layer

The network layer managed delivery of messages regardless of whether they originated from the application or an icon; however, a different packet structure was used for each. Packets originating at the application could carry multiple messages (fig-

special start byte	message ID	data length	instruction code	service code	value	unused	sequence number	checksum
0	1	2	3	4	5	6 8	9	10 11

Figure 6.17: Application originated packet format. Each packet was 12 bytes long.

special start byte	message ID	ack/seq	instruction code	service code	value	checksum
0	1	2	3	4	5	6 7

Figure 6.18: Icon originated packet format. Each packet was 8 bytes long. The ack/seq field indicated whether the message was an acknowledgement of an application message or was an icon originated message.

ure 6.17). These packets could be destined for a single icon or could be broadcast to groups of icons (or all icons). Packets originating at an icon carried only a single message to the application (figure 6.18). In both packet types there was a sequence number that represented the number of packets sent by a specific origin. The sequence number could be used for checking the order of arriving messages and was necessary for a future implementation of the network layer that will provide guaranteed delivery.

6.4.2.3 The Application Layer

The application layer provided a message format for the application to configure and control icons and for icons to provide both polled and event-driven information to the application. Each application inbound/outbound message was handled by individual threads. The read or update operations on the application's device database were controlled by semaphores. Messages could contain instruction codes, service codes and data. The instruction code was used to determine the function of the message. Examples include discovery, inbound data, and outbound set data. The application could generate instructions for setting or requesting the status of service parameters, setting

default service values for an icon or issuing a reset command to an icon. Icons could generate instructions for registration with the application and for reporting application-requested or icon-generated information. Service codes were used to identify icon specific functions. Some example services include touch (hand icon), intensity (light icon), and selected (icons during the programming mode).

The most common instruction codes were:

discovery An embedded device sends this icon-originated message to the application to register itself.

inboundData Am embedded device sends this icon-originated message to the application.

outboundSetData An application outbound message for embedded devices to update their internal values and actuate physical changes if necessary.

outboundRequestData An application outbound message to request the current state data from an embedded device.

inboundReplyWithRequestedData The complement to outboundResetData, an embedded device returns the requested data.

outboundResetData An application outbound message to reset the target device(s) internal values and actuate physical changes if necessary.

Each device could support one or more services. The most common services were:

intensity Primarily supported by actuators.

touched Primarily an attribute of contact sensors.

selected Supported by all devices. Used during the recording phase.

includeThisDevice Used during the recording phase, generated by a proximity sensing device (magic wand).

beginProgramStatement Used during the recording phase.

broadcast service ack All devices support this low level synchronization service.

single device ack All devices support this low level synchronization service.

discovery synchronization All devices support this low level synchronization service.

deviceType All devices support this low level synchronization service.

deviceColor All devices support this low level synchronization service.

proximity Supported by proximity sensing devices, such as the magic wand.

6.4.3 Icon Controller Hardware and Software

The 17C756A micro-controller from Microchip, Inc. [62] was used on the icon controller board. In addition to the micro-controller, the icon controller board had several support circuits, including a RS232 driver and two 7-segment displays for debugging purposes. One of the micro-controller's serial ports was dedicated to the WIT2410 wireless module. The other serial port was used for communication with the application computer, in the case of the base unit, or for control of sensor devices as needed in StoryRoom icons.

The digital input/output of the micro-controller was used for sensors and actuators. The driver circuit board had custom circuits depending on the device to be controlled.

A common sensor used in StoryRooms was a simple switch that ran through a simple circuit to provide a latch on the switch, which then drove one of the micro-controller's digital inputs. When an input change was detected, the appropriate message was sent to the application and the latch cleared to be ready for the next event. Actuators could be simple lights, which the controller drove through a transistor circuit that provided external battery power to the light. Other driver circuits included a motor driver and a circuit to drive glow fiber.

In order for StoryRooms to be a physically programmable environment, it was necessary to have physical tools that acted over physical icons. The most basic tool was the "magic wand," which allowed children to create icon *groups* (figure 6.19). The term *group* is identical to *physical interaction instruction* and is used when I explain the process to the children. The wand must be able to detect the proximity of other icons and identify them. The ATT used a radio frequency identification (RFID) system from SkyeTek, Inc. This system detected and identified RFID tags, which are inexpensive, passive, credit card sized pieces of paper and wire that were inserted into StoryRoom icons. Control of the system was through the micro-controller serial port, and RFID reader data was translated into a message for the StoryRoom application. The magic wand was able to detect and identify other StoryRoom icons consistently from a range of about 4".

I considered two possible methods for tracking objects in the StoryRoom environment: 1) proximity among objects, and 2) object tracking. Conceptually, a proximity enabling technology contains multiple transponders with unique identifiers (tags), at least one of which is a "reader," and a transfer of data from the reader to a data-processing computer. When tags come within the range of the reader, the reader is able to read their identities and then send the data to the data-processor. RFID technology is one

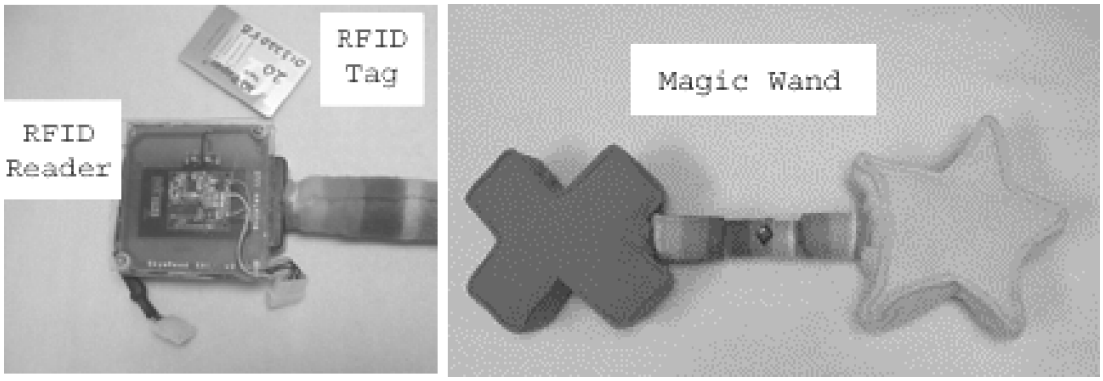

Figure 6.19: The Magic Wand and an underlying RFID reader.

popular implementation.

Conceptually, a way to track object locations in indoor environments is to strategically place beacons around the environment and to embed transceivers into mobile objects (in my case, this would be the physical icons). The beacons send continuous and uniquely identified signals. Each mobile object can calculate its location by decoding and calculating the differences in the arrival times of the signals. These location data can then be sent to a central data-processing computer.

Although the second choice would offer more interaction possibilities, I was unable to find any viable demonstration or commercial systems that could be as reliable as the simpler proximity sensing abilities of the RFID technology. With regard to the StoryRoom, this was not a detriment, since in physical stories interactions usually, if not always, took place when objects interact with each other or with people. That is, story interactions were built upon the proximity of objects to each other.

6.5 Limitations of the Implemented Language

The StoryRooms and its physical programming tool was limited by two main factors: 1) the underlying machine model, and 2) the choice of user interactions (i. e., physical syntax). Since StoryRooms is a physical interactive environment, it is a state machine. As such, it lacks memory and operations on memory. The physical syntax was incredibly simple. It only involved five kinds of physical actions: press, wave, pick up, put back, and pull.

In 6.1.4 I suggested that adding memory to *PIE* would introduce desirable features to the programming efforts of physical interactive environments. For example, with only a counter variable, the environment would be able to keep track of time, allow a user to specify when an action should occur, or count the number of occurrences of an event. More memory could provide more powerful programming constructs such as loop and sequence into the physical syntax. But this extra programming power comes at a cost of more complexity in physical syntax. For instance, I do not believe that a physical syntax for looping can be as easy as the *press* and *wave* combinations of the finite state PIE. Even if the primitive physical gestures were the same, a more complex combination of the gesture would probably be needed.

The current implementation of StoryRoom is a *PIE*. That is, the physical environment only inspects the current sensor values to transition into new states. In order for StoryRoom to support the more general and useful PIE_{SA}, I would need to modify the *press*-and-*wave* syntax. This is because the *PIE* version implicitly groups all the sensors into the pre-condition and all the actuators into the post-condition of a physical interaction instruction. But in a PIE_{TA}, since the actuators could appear in both pre-condition as well as the post-condition of the instruction, there needs to be a physical

gesture to precisely indicate where the actuators should be assigned. This is actually a difficult issue. Of the many physical syntax designed during the IDT sessions, all were more complicated (in terms of gesture count and gesture type) then the simple *press* and *wave* of the *PIE*. Again, adding more flexibility in the machine requires more complicated gestures.

Physical programming is not just for children to create StoryRooms. Prudent choice of physical gesture primitives can offer adults simple ways to control everyday devices. For example, let me extend the alarm blocks idea from the Introduction to a hypothetical set of blocks to control both a VCR and a microwave oven. This set of blocks contains: 1) hour block, 2) minute block, 3) second block, 4) play block, and 5) record block. Now, suppose I want to heat up my dinner for three minutes, I would place the minute block on top the microwave and turn it so that I see the number 3. Then I would place the play block adjacent to the minute block. This activates the microwave and I am happy to have my dinner. Using the same blocks, I now place the hour block on top of the VCR and turn it to see the number 8. Then I place the minute block and turn it to see the number 30. Finally I place the record block adjacent to these two blocks. I have just set the VCR to record at 8:30. Of course these two examples do not address important interface issues such as, how do I represent 60 numbers on a block. But the point I want to make is that while these blocks are not nearly as powerful as a remote control, they can perform very simple tasks simply.

Chapter 7

An Exploratory Study of a New Programming Approach for Kindergarten Children: Physical Programming

After the early StoryRooms—*The Red Balloon, The Sneetches*, and others—I still did not have a clear vision of a children-usable programming system to define sensor-actuator interactions. I was also surprised to find that the picture based prototypes did not give children a clear way to program. Fortunately, as the IDT's elementary school aged design partners demonstrated, direct manipulation of physical objects was a promising direction. This (programming) was the last unsolved element of the StoryKit and became my new goal: to develop a usable system and to find out to what extent kindergarten students could understand the interactive nature of StoryRooms and whether they could create custom interactions in their own stories.

This chapter covers the exploratory study I led on physical programming [67], a physical user interface metaphor for users to generate interaction rules by the direct and physical manipulation of physical icons. It was conducted both in HCIL, with IDT members, and at the Center for Young Children (CYC), a preschool on the University

of Maryland campus, which fosters and supports research and development activities.

In this early stage, I was less concerned with the potential educational benefits of the technology. I was instead interested in whether the technology was usable by these very young children. To understand the relationship between interactive environments and the young children, I explored three basic questions:

1. Can young children (4-6 years old) comprehend what a story is about in a physically interactive environment such as a StoryRoom?
2. Can they use or participate in an already created story in a StoryRoom?
3. Can they use physical programming to create a StoryRoom?

With the IDT, I used descriptive and qualitative research methods (further described later in this chapter) to answer these questions.

7.1 Participants

This study included three participant groups: 1) adult researcher, 2) elementary school aged researcher, and 3) preschool user. The adults were regular members of the IDT. At the research sessions, the adult team was composed of five people: two adults who facilitated the storytelling with the children; one videographer in the room; one researcher situated behind a one-way observation window using the computer to react to what the children did; and one assistant, who helped interpret the children's activities when they became difficult to see or understand. The elementary school students were also regular members of the IDT. Normally, they think about technology for their peers. But this project placed them in a more significant role— designers of technology for

people younger than they were—and to also consider whether the age difference would have serious usability impacts. This study involved two different preschool subgroups. The first were invited to HCIL for informal sessions. The next were students at the kindergarten.

7.2 Session Structures

After I identified the goal of a physical programming approach, I and the IDT team devoted numerous brainstorming sessions and developed the first semi-wizard-of-oz system, including physical artifacts such as icons and a magic wand (figure 7.2). Because the team did not have extensive working knowledge of kindergarteners before the study, the ATM invited two separate groups of two kindergarten aged children (ages 5-6) to our lab (figure 7.1) for informal observations. The adults explained how the programming tools worked, and observed their explorations with tools. The time they spent using the tools was unstructured. We wanted to see where they led us. One adult facilitated each session, four adults took notes, seven other children (regular design partners of the IDT) were also note-takers and periodically asked questions, and one child design partner videotaped the experience.

With these initial observations, the adult members quickly realized that the children's exploration of the prototype must be structured. The idea of interactive stories needed to be presented with increasingly more abstractions, in order for us to understand to what extent children understood the technology. One child design partner (age 11) wrote, "I don't think they got it when we started. When I showed them something it made sense then. I think it was good when they did it with me. Then they had some good ideas to show us." The notion of a physical interactive environment was

Figure 7.1: Prior to visits to the local kindergarten, the ATM fine-tuned possible interaction techniques with two pairs of children in the same age range.

conceptually difficult to understand and still somewhat uncommon, so to start off with the idea of programming one was difficult to grasp for children (and many adults). Therefore the sessions at the kindergarten that followed these initial sessions contained three parts, which conveniently correlated to the three research questions of the study:

Children as Audience An adult told an example story with a StoryRoom.

Children Joined Adults as Storytellers The children retold the story, so that they get to play with the props and squeeze the physical icons.

Children as Physical Programmers Children were shown how to program with the physical icons and were asked to make up a story.

The adult members conducted four subsequent sessions with the structures described above, at the kindergarten. In total, this study included 11 kindergarteners (ages 4-6) (table 7.1). Seven were boys, four were girls and each group included one girl and at

Figure 7.2: Early Low-tech design session on physical programming tools.

least one boy. The first three groups had three children participating and the last group had two children. The first three groups worked with researchers an average of 13 minutes/session, and the last group worked for 50 minutes to see if there were obvious differences in children when they had longer exposure to the tools and props.

Table 7.1: Group composition of the study.

Group	Number of Children	Gender
1	3	2 Boys, 1 Girl
2	3	2 Boys, 1 Girl
3	3	2 Boys, 1 Girl
4	2	1 Boy, 1 Girl

7.3 Wizard-of-Oz Prototype

In order to understand if young children who had not helped design my programming tools could use its physical metaphor, I, with the ATT, developed a mid-tech or wizard-of-oz prototype (figure 7.3) for this formative study. I thought it was important to have the flexibility to experiment with different technology behaviors depending on the user interaction. But I also learned from many low-tech design sessions that often the "wizard" (person) could not track the many concurrent activities in the environment and react appropriately. Therefore, I developed a software application, written in RealBasic [83] on the Macintosh computer, that allowed the wizard to define and group action-reaction rules on-the-fly as the children were using the technology. The wizard software broadcasted serial data packets via a 433 MHz RF Transceiver connected to the serial port on a Macintosh laptop. These signals were then received by RF transceivers embedded in the physical icons and interpreted by BASIC Stamp Micro-

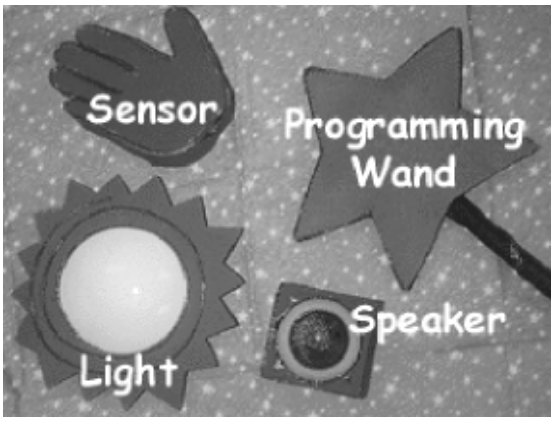

Figure 7.3: Physical Icons for programming a StoryRoom. From left to right: A "hand" to make an object touch-sensitive; a "light" to make an object "light up"; a "sound box" to attach a sound to an object; a "magic wand" to signal the authoring mode.

controllers [98]. Based on the data content, the microcontroller then could turn on and off activators such as lights, sounds, and buzzers. This implementation supported one-way communication, so children pressing the sensors, or tapping the icons with the wand did not actually activate anything. Through a one-way mirror adult researchers observed the actions of a child, and sent the appropriate response from the computer. For example if a child pressed the hand and expected a light to come on, it would.

7.4 Story for the Research Sessions

Based on preliminary meetings with the invited young children, The IDT designed a story that was understandable and involved the interactive technology.

Here is *The Irene Story*:

> Narrator: "One day, Irene was hiking in the woods behind her house, and she went farther than ever before. She became lost. Irene saw a cottage

just up ahead. She walked up to the cottage and saw a strange purple hand and pressed it." [Narrator presses the purple hand physical icon. A purple light placed next to a furry mouse lights up.]

Narrator: "She walks up to the purple light, and sees a mouse. She said, 'Mr. Mouse, do you know a way back to my house?' Mr. Mouse replied, 'I do not know where your house is. Maybe you should ask Mr. Koala.' Irene finds and goes up to Mr. Koala. She sees a green hand next to it. So she squeezes it and asks, 'Mr. Koala, do you know the way to my home?" [Narrator presses the green hand physical icon. A green light placed next to a snake lights up.]

Narrator: "Mr. Koala said, 'I do not know where your house is. Maybe you should ask Mr. Snake.' Irene follows the green light and sees Mr. Snake. She asks the same question. Finally, Mr. Snake says, 'Sure, I know just the way. Come, follow me back to your home."

Using this story as the anchor, two adult interactors led the kindergarteners through the three session segments. In the *Children as Audience* part, while the children sat and watched, an adult was the narrator of the story, pressing on the hand physical icons and pointing out the resulting sound effects and actuated lights in the room. In the *Children Join Adults* part, the adults encouraged the children to retell the story as they had just seen. In the *Children as Physical Programmers* section, the adults showed the children how the magic wand worked. The children were then free to use any of the props, existing objects in the room, and any number of physical icons, to tell their very own stories.

7.5 Default Interaction Rules

Because a standard did not yet exist for physical programming, I defined a set of interaction rules for this study.

- The magic wand was only used for programming activities.
- The glow-fiber and buzzer of the icons indicated the *selected* state of the icon, and were used during the programming mode. For example, when the wand touched a light icon, the wand's glow-fiber would blink. In addition, the icon would make a buzzing sound, its glow fiber would blink, and its light would turn on.
- To create a condition-action rule: "if red hand is touched, then turn blue light on," a child would take the magic wand, and tap the red hand and the blue light to create a group.
- To start a story, put away the magic wand.

These rules came from observations during IDT design sessions. The magic metaphor was chosen because children liked the idea that they could make things magical. Magic was also a reasonable explanation for the more abstract behaviors of technology. The ATT members used both sound and light to indicate an icon's selected state, because it was the easiest feedback method. Tapping was designed to be a deliberate and unambiguous action, so that both the system and the children would share the same expectation of behaviors. Finally, a simple cue of removing the wand away from the storytelling area was chosen to distinguish the play from the programming mode.

7.6 Data

This was a qualitative study. One video camera, located in the classroom, about fifteen feet away from the story area, captured the activities and dialogue of all children. Back in the HCIL lab, I reviewed the tapes and created a contextual inquiry chart [24]. From this chart, I noted the time, verbal discussion, and activities (table 7.2).

Table 7.2: Sample data from the contextual inquiry chart.

Time	**Quote**	**Activities**
32:23	F: can you tell a story with these things?	W: yeah
32:44	W: I want to be mouse, B: I want to be koala	W grabs mouse, B grabs koala, G grabs snake.
32:57	W: the mouse went...	W grabs purple set and moves to the cottage
33:07		W positions the purple hand and light by the cottage. B holds on to the green hand.
33:13	W: the mouse went to sleep one night	W: touches the purple hand, the light came on
33:15		B: squeezes the green hand
33:23	W: who's on my door	
33:55		B: squeezes the green hand. Green light came on.

7.7 Analysis

After a review of the dialogue and activities, three members of the team together analyzed the data charts and developed codes for "roles" (who a child was during a specific action (e.g., experimenter, story participant, etc.) and "activity patterns" (e.g., story-

telling, playing, etc.). Once the team agreed on these initial codes, then all the charts were coded. In figures 7.4 & 7.5, the frequency of these roles and activity patterns were summarized for the last third of each session. It was decided by the AT members that during the third part of the session was really when the children were most in control and had the most freedom to explore. During the first two parts of the session, they were learning primarily about the technologies employed. In the following sections, I will discuss my analysis of the four sessions.

7.7.1 Children as Audience

In this initial part of the session that lasted on average less than 2 minutes, children were shown the "Irene story" and across the four sessions, children were quite attentive. They were fascinated by the use of the physical icons to create a physical interactive experience. At no time did any children look bored; many of the children could not wait to use the physical icons themselves to try out the story experience.

7.7.2 Children Join Adults as Storytellers

During this section of the session, most of the children (10 out of 11) were readily able to recall and reenact elements of the story. They actively participated in the StoryRoom experiences of Irene. Many of them (9 out of 11) also seemed to understand how to use the physical icons to participate in the story. Interestingly, one child began to experiment with the physical icons' behavior during this part of the session. She kept pressing on the hand to see if it would repeatedly turn on a light.

7.7.3 Children as Physical Programmers

During this third and final part of the session, the children were shown how to physically program and they explored the use of these technologies for storytelling. My analysis of the roles and activity patterns revealed that the children spent most of their time experimenting with the tools (see Charts 7.4 & 7.5). They were not afraid to try out different combinations of taps with the magic wand, and frequently pressed the hand to explore the possibilities of what it affected. There were times when a technical glitch occurred(e.g., the researcher at the computer sent the wrong command to the physical icons, or was delayed in responding). This also prompted the children to continue to experiment with the physical icons. Interestingly, some of the children either waved the wand several times, or tapped repeatedly, until they saw the feedback they expected. In each session at least one child was able to form a definite idea about how to physically program with the tools.

Where the children seemed to have the most challenges with physical programming was in understanding the difference between the programming mode and the participation/use mode. The children understood that the wand helped them "make things magic." But they had difficulty understanding that it was a tool, and not part of the story. This confusion may partially come from the feedback of light and sound when the children were in programming mode. As the children touched the physical icons with the wand, a sound would occur and a glow light on the icon would turn on. Many children were quite excited by this and thought this "was the story". Perhaps by reducing the "excitement" of the feedback, that they may be more likely to see this as one step in the storytelling process.

In regards to storytelling, I found that the children told stories in three ways: (1) com-

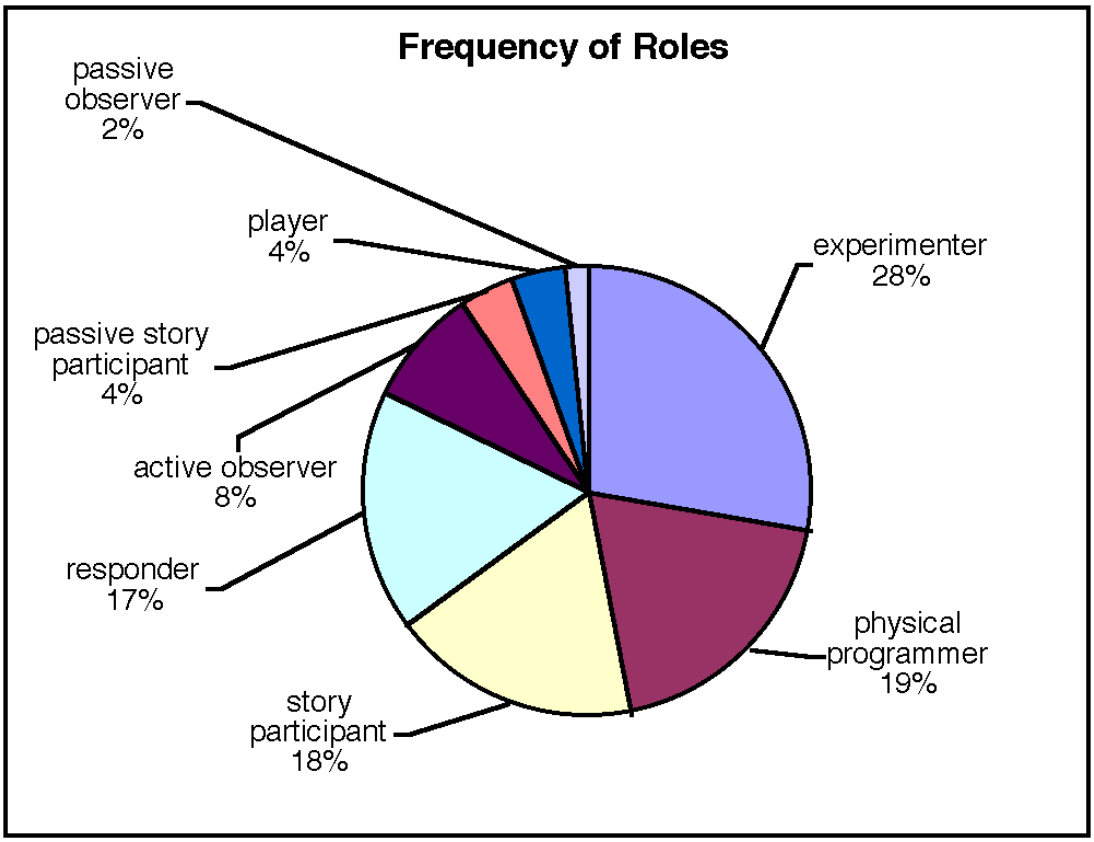

Figure 7.4: Frequency of children's roles for the last part of the session. There is a small but significant percentage of children who showed potential as "programmers."

pletely verbal with the use of no props or physical icons; (2) with the use of some props such as stuffed animals and verbal descriptions; (3) with the use of physical icons and props and verbal description. As Chart 7.5 summarizes, when the children were asked to tell a story, they most frequently just verbally told a story[1]. The children fell back into what they knew best. However, once the researcher asked if they would like to use the things in the room to tell a story, they most frequently used both the physical icons and the props to physically program. Surprisingly, it was far less frequent for the children just to use the props.

[1]Experimentation was not included since it was not considered a storytelling activity.

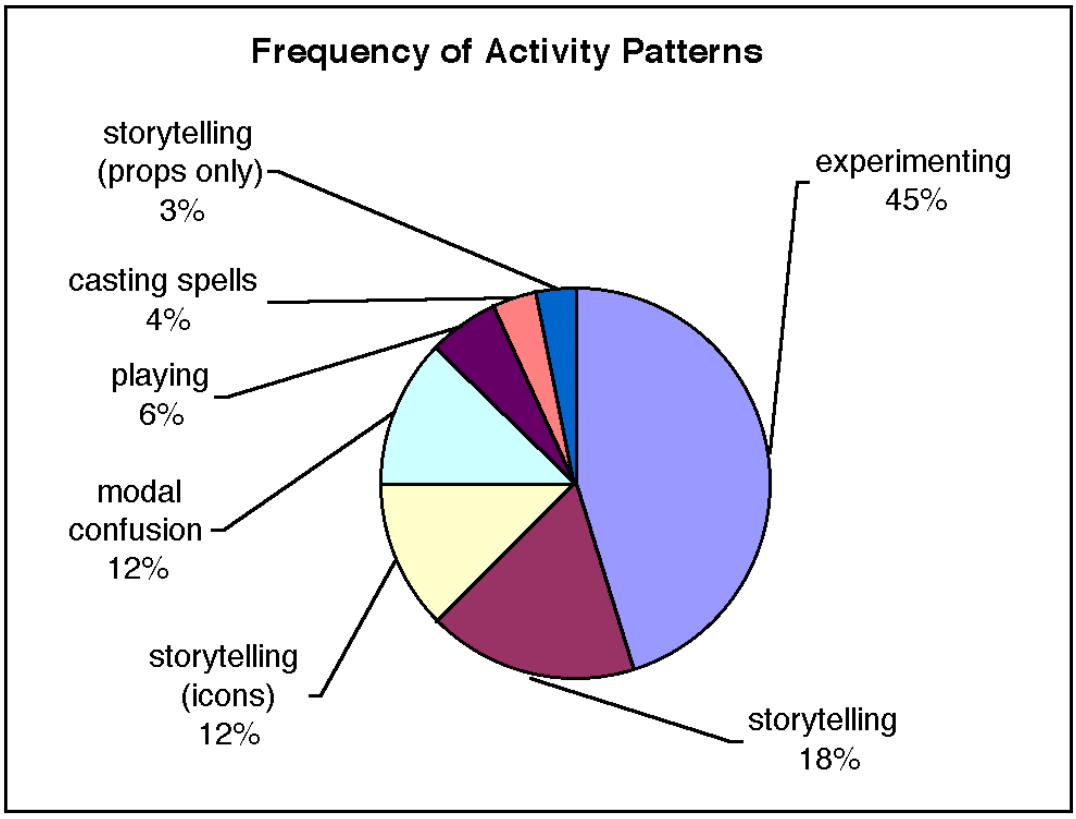

Figure 7.5: Frequency of children's activity patterns for the final part of the session. A large percentage of the activities (30%) appear to be storytelling.

The kinds of stories the children told were very similar to the Irene story they heard. In many cases only one or two elements were changed to make it their own. However, there were interesting additions to the stories they told. For example, one child incorporated the physical icon lights as decorations on a cottage prop. In her story she had the characters ask, "Who is there? Would you please turn off the lights (one of the actuators)? I need to sleep." Perhaps, had there been additional props (outside of the ones used for the Irene story) and more time to explore, more original stories might have emerged.

7.8 Lessons Learned from this Exploratory Study

In understanding what I have learned with children, let me refer back to the three initial questions: (1) Can young children comprehend what a story is about in a physically interactive environment such as a StoryRoom? (2) Can they use or participate in an already created story in a StoryRoom? (3) Can they use physical programming to create a StoryRoom?

With regards to the first question, I saw without a doubt that children ages 4-6, who had no experience in designing my technology, could easily comprehend what the story is about. I also saw with regards to the second question, that all of the children could also use or participate in an already created story. Once shown how to interact with the physical icons, they had no trouble interacting with the StoryRoom experience. I was also pleased to note that the introduction of technology did not get in the way of the storytelling experience.

As for the third question concerning physical programming, the answers are less clear cut. I did see in each session one or more children able to physically program. They

understood that placing the physical icons on a prop around the room either offered some input or output. They also understood that the physical icons had relationships to each other based on how they were programmed. In fact, out of the 11 children the ATM worked with only 3 children could not comprehend any aspect of this approach. Thanks to a longer session with the last group, I believe that if we had spent more time with each group, more children might have been able to accomplish higher levels of physical programming. But considering the short period of time the adult team spent with the children, they were able to accomplish much more than I initially expected. It was not surprising that their main difficulty was in understanding the difference between programming and participation in an already created story. At this young age, children's most common form of storytelling is improvisational storytelling (many times referred to as "play") where children freely move in and out of storytelling and "storylistening" [2]. This may be the biggest challenge in supporting children with physical programming. Is there a way to naturally move between programming and participating? The magic wand shows a promising direction.

With regards to lessons learned about the *cooperative inquiry* methods, I believed that the mid-tech or wizard-of-oz prototype served the team well. It went a long way in simulating the full experience of physical programming. It offered a flexible way of exploring my ideas with children, without having to spend many more months fully developing the technologies. But, even though it was just a prototype, it had to be extremely rugged. On numerous occasions, during the research sessions at HCIL and during the study, when the devices failed to work as the children expected, they stopped being users and became debuggers of the technology.

Chapter 8

A Usability Study of Physical Programming and Kindergarten Students

Encouraged by the results from the exploratory study, I and the Adult Technical Team refined the StoryRoom and physical programming prototypes, and conducted a longer term study. As in the exploratory study, I focused on the following three questions:

1. Can young children (4-6 years old) comprehend what a story is about in a physically interactive environment such as a StoryRoom?
2. Can they use or participate in an already created story in a StoryRoom?
3. Can they use physical programming to create a StoryRoom?

8.1 The Study Setting

Over a one-month period in the fall of 2002, a new group of 18 children (ages 5-6) used the StoryRoom and physical programming technologies in an initial empirical study. The children who participated in this study were racially and ethnically diverse,

varied widely in their academic ability, and were in the kindergarten program at the CYC. Children worked with the StoryRoom technology in the Great Room (a large open space in the middle of the CYC) with a team of four adults for sessions that lasted approximately 20 minutes. The pairs were diverse in gender, race, and ethnicity, and remained the same throughout the research (table 8.1).

Table 8.1: Group composition of the usability study.

Group	Child A	Child B
1	Girl	Girl
2	Boy	Boy
3	Boy	Boy
4	Boy	Girl
5	Girl	Boy
6	Girl	Boy
7	Girl	Girl
8	Boy	Boy
9	Boy	Girl

8.1.1 The Irene Story

Because the kindergarten children in the exploratory study understood the *Irene Story*, I used this story as the basis for teaching this new group of children the concept of StoryRooms. I changed the light in the first study to a blinking arrow because the I found that the arrow was better at getting children's attention and was therefore better able to direct the flow of the story. I also added the Wind physical icon to the final part of the story, because the IDT's child designers thought its ability to make them feel wind was important in storymaking.

The slightly modified "Irene Story" contained a cottage built from cardboard sheets and swatches of felt fabric; a stuffed mouse; a stuffed koala bear; a cave that is a

Figure 8.1: The completed setup for the story. The props include the cabin, the mouse, the koala bear, and the snake inside the cave. The foot icon is a contact sensor that was programmed to trigger the blinking arrow by the mouse. The hand icon was programmed to trigger the sun icon (light) and the wind icon (fan).

cardboard box with a hole in it; and a snake made out of foam.

A foot icon (touch sensor) was placed next to the cottage. A blinking arrow (actuator) was placed next to the mouse. A hand icon (touch sensor) was placed next to the koala bear prop and a wind actuator and light actuator were placed next to the cave. To support the story, the foot icon was pre-programmed[1] to trigger the blinking arrow, and the hand to trigger both the wind and the lights (figure 8.1).

[1]This program was created by an adult using physical programming.

8.1.2 Session Structure

This study shared the same session structure as in the exploratory study. The only two differences were that 1) the children had significantly more time to learn physical programming; and 2) a select group of the children had the opportunity to create a StoryRoom, from story inception, to prop construction, to physical programming, and finally to share the story with their peers.

Here is a quick reminder of the session activities:

Children as Audience An adult tells and performs the Irene story to the children.

Children Join Adults as Storytellers The children retell the story, so that they get to play with the props and squeeze the physical icons.

Children as Physical Programmers Children are shown how to program with the physical icons and are asked to make up a story.

Children as Audience

When children enter the Irene StoryRoom, they saw the icons and props set up in a semi-circle that follow the chronological order of the story. A researcher was the narrator and she helped them through the environment. First, she turned on the story by flipping the "once-upon-a-time lever." She then led the children to the cottage, next to which was the foot icon. She began, "*This story is about Irene, a little girl who is lost in the woods and cannot find her house. Irene asks the people in the cottage if they know where her house is, but they do not. Irene sees a strange foot and pushes on it.*" The researcher asked the children to press the foot. This activated the blinking purple arrow light next to a stuffed mouse. The children then saw a blinking arrow pointing

to the mouse. The researcher continued, "*Irene asks Mr. Mouse if he knows where her house is. Mr. Mouse says no, but that she should ask Mr. Koala*." The children ran to Mr. Koala, who has the hand icon near him. The researcher says, "*Irene then asks Mr. Koala if he knows where her house is. Mr. Koala says no, but that she should ask Mr. Snake in the cave*." The children pressed on the hand icon, which activated the fan and light placed near a snake prop in a cave. The children ran over to the cave and were told by the adult, "*Irene asks Mr. Snake if he knows where her house is. Mr. Snake says yes, just turn around and go ten feet and there it is*."

Children Join Adults as Storytellers

A researcher asked the children to retell the story that they had just experienced. The adult refrained from helping the children unless it was obvious that they either 1) forgot a part of the story, 2) retold the story incorrectly, 3) forgot to use an icon, or 4) were confused.

Children as Physical Programmers

An adult researcher demonstrated and asked the children to repeat the process of programming the interaction rules for a StoryRoom. This process consisted of five steps.

1. Put on the magic hat to enter the program mode.
2. Use the wand to create one rule that included at least one actuator and one sensor.
3. Take off the magic hat to exit the program mode.
4. Turn on the "once-upon-a-time lever" to enter the play mode and to review the program.

5. Turn off the “once-upon-a-time lever” to end play mode.

Again, the adult did not make any suggestions to the children unless it was clear that they were confused or could not continue without help.

8.2 Data

Data from these sessions were collected by videotaping and taking observational notes. To analyze the data, the ATM developed a coding scheme based on the data that emerged from the raw artifacts. Two members of ATM initially coded the tapes, after which the codes were refined. Inter-rater reliability was established by having two of the adult team members code 33% of the data. There was one discrepancy in coding which was resolved, and one team member continued to code the rest of the data.

A scoring system was designed to indicate the level of understanding that the children exhibited during the study (table 8.2).

Table 8.2: The scoring system for the usability study.

Score	Explanation
2	A child performed or remembered an activity independent from adult assistance.
1	A child (*A*) is given this score under 3 conditions: 1) *A* correctly performed or remembered an activity with help from adult or the child’s partner (*B*), 2) *B* intervened and completed the task before *A* could attempt the task, or 3) an adult intervened and completed the task before *A* could attempt the task.
0	A child could not complete the activity.

8.2.1 Can Children Participate in an Already Created StoryRoom?

The activity designed to answer this question provided the children with their first exposure to StoryRooms. An adult told children a story using the *Irene* StoryRoom and then invited the children to retell the story.

In analyzing the tapes, the ATM first determined if children were engaged – that is, if they listened to and/or observed the storytelling researcher for a majority (more than 50%) of the time while being told the Irene story. The adults also determined if the children reacted when something "magical" (e.g., the fan and light turning on when the hand was pushed) happened.

This analysis showed that 100% of the children were able to participate fully in this previously created StoryRoom. This laid the groundwork for the next part of the activity, in which the children were asked to retell the Irene story to a new researcher.

During retelling, the ATM looked for the children's ability to recall and retell the main events of the story and to use the StoryRoom elements in order to do so. Because the children were functioning as a pair in this activity, points were given when at least one of the children did a task. During this section, data from one pair of children had to be eliminated due to poor video quality.

The scoring of the retelling phase was based on eight activities. A child could score a maximum of 2 points towards each activity, for a maximum of 16 points. The activities were separated into 2 categories. The first involved telling the story using the props (house, mouse, koala bear, cave). The second category involved retelling the story with the physical icons and tools ("once-upon-a-time" lever on, foot, hand, "once-upon-a-time" lever off).

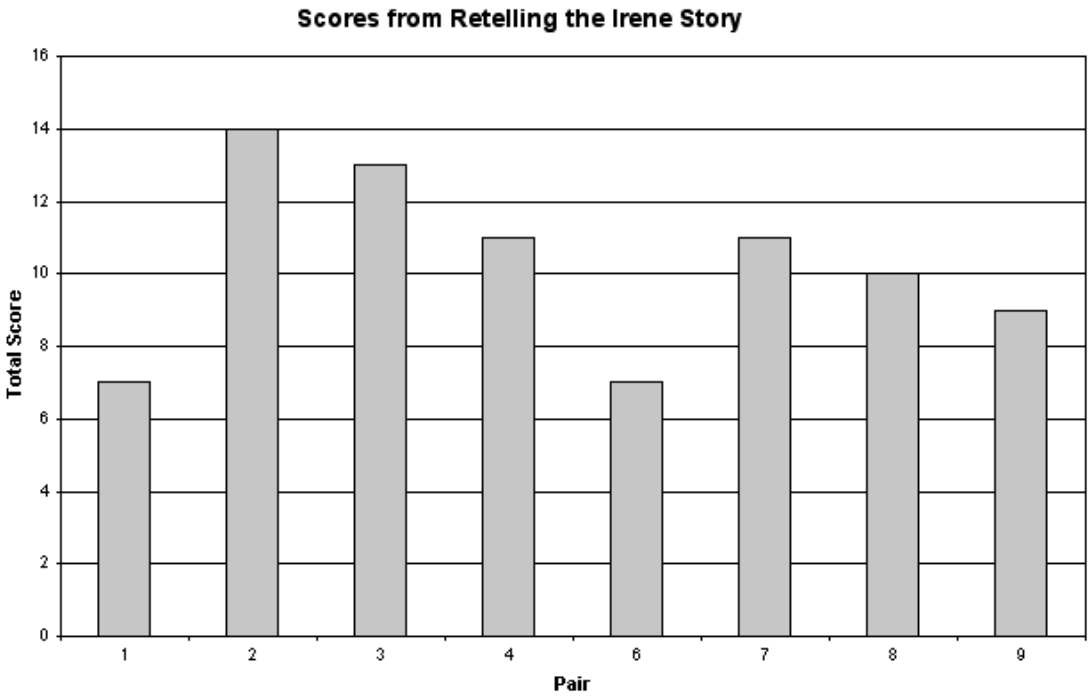

Figure 8.2: The number of total points (out of 16) that each pair scored on retelling the Irene story. The range of scores is from 7 – 14, showing that children were able to retell the story with varying degrees of adult support.

From the the remaining 8 pairs, I saw a greater variance of ability to use the StoryRoom on retelling than on participating. The children scored from 7 to 14 out of a possible 16 points. It appeared then that all pairs were able to show some understanding of how to use the StoryRoom to retell a story (figure 8.2). An analysis of the two frequency charts (activities by icons and by props) revealed a more interesting picture. In figure 8.3, most of the children could retell a story using a prop. In figure 8.4, ALMOST ALL the children correctly retold the story with the icons. The unusually large percent of adult help for the Story On and Story Off activities were mostly due to the children not remembering to change into the play mode. I had expected that the children could tell stories with props and that some could involve the use of icons. So it was a pleasant surprise to see the overwhelming number of storytelling was with physical icons.

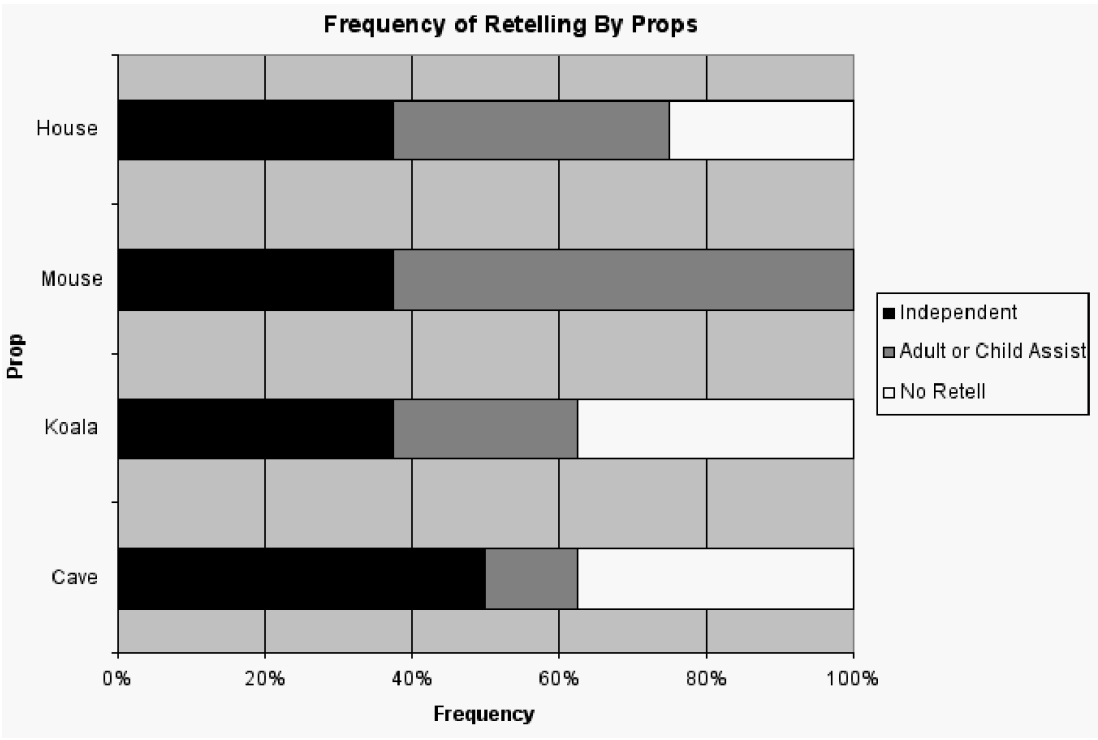

Figure 8.3: Frequency count of retelling by props. 75% of the activities were correctly completed.

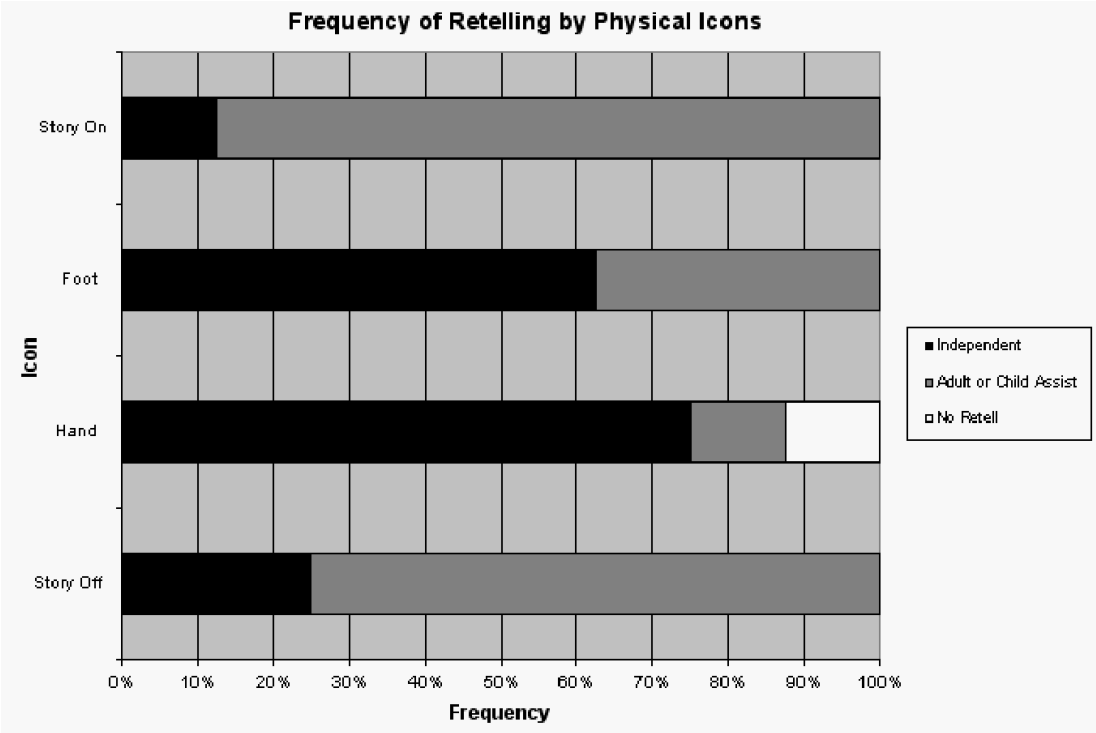

Figure 8.4: Frequency count of retelling by icons. The large number of adult assistance during Story On and Story Off events were likely because the children had to enter and exit the play mode.

It should be noted that adult prompting, such as “What’s next?” was not coded as adult guidance as it was not related specifically to the children’s ability to use the StoryRoom. More specific prompting, such as “What do you do with that foot?” was coded as adult guidance as this prompt was specific to the use of the StoryRoom.

8.2.2 Can Children Program Using Physical Programming?

In order to answer this question, the adults placed a large pile of StoryRoom actuators and sensors on the floor along with the “once-upon-a-time lever” switch. The magic wand and the wizard hat were placed on a table. I began the sessions by showing the physical programming technique to the children.

Children were then each given the opportunity to define interactions for a StoryRoom. Because each child was given a turn to define these interactions on his or her own, the coding for this activity was done by individual rather than by pair. One pair was absent, so 16 children participated in this phase.

Again, videotape of each child was analyzed to determine if he or she took the five necessary steps (section 8.1.2) to create an interaction rule for a StoryRoom.

The ATM also tried to determine if the child understood the interaction rules that he or she had created. Because every child (*A*) was working with a peer (*B*) nearby, there were times when *B* stepped in and completing a task before *A* had a chance to do so. In that situation, an *h* designation was given to *A* on the part of the programming completed by *B*. Any task designated *h* was given a score of 1. One time an adult stepped in to complete a task. This was designated *H*. The videotape did not capture one child’s (pair 1, child b) Story Off activity. This activity was given a score of 0. Therefore child 1b’s total score was at least 50% (table 8.3).

All children scored between 50% and 90% on programming (figure 8.5). These numbers showed that most of the children were able to program with some degree of adult guidance, and all had some understanding of what to do in order to control the interactions of sensors and actuators in a StoryRoom. Inter-rater reliability was established for the coding of this activity by having two adult researchers compare 25% of their coded data. There were no discrepancies; therefore one team member finished the coding.

Table 8.3: Physical programming scores.

Activity	1A	1B	2A	2B	4A	4B	5A	5B	6A	6B	7A	7B	8A	8B	9A	9B
Hat On	2	1	1	2	2	2	1	2	2	2	2	2	2	2	2	2
Use Wand	2	2	2	2	1	2	2	2	1	2	1	2	2	2	2	2
Hat Off	1	1	1	1	2	2	2	1	1	1	2	1	1	1	2	1
Story On	1	1	h	2	1	1	1	1	1	1	1	1	2	1	h	1
Story Off	1	*	0	0	h	h	1	H	h	h	1	1	2	1	1	1
Score/10	7	5	5	7	7	8	7	7	6	7	7	7	9	7	8	7

Analysis of the frequency count of the 5 programming activities led me to three important observations (figure 8.6). First, an overwhelming majority of the children could "become wizards" without help. This may indicate that they accepted the "magic" metaphor. Second, more than 95% of the activities could be completed, whether independently or with help. This level of success might be due to the simplicity of the individual activities. With the exception of the magic wand, the other four activities were single step, direct physical manipulation of real objects. Third, the wide range of user competence (50% to 90%) in chart 8.5 could be partly explained by the frequency count: adults often needed to supply hints to remind the children to **negotiate**

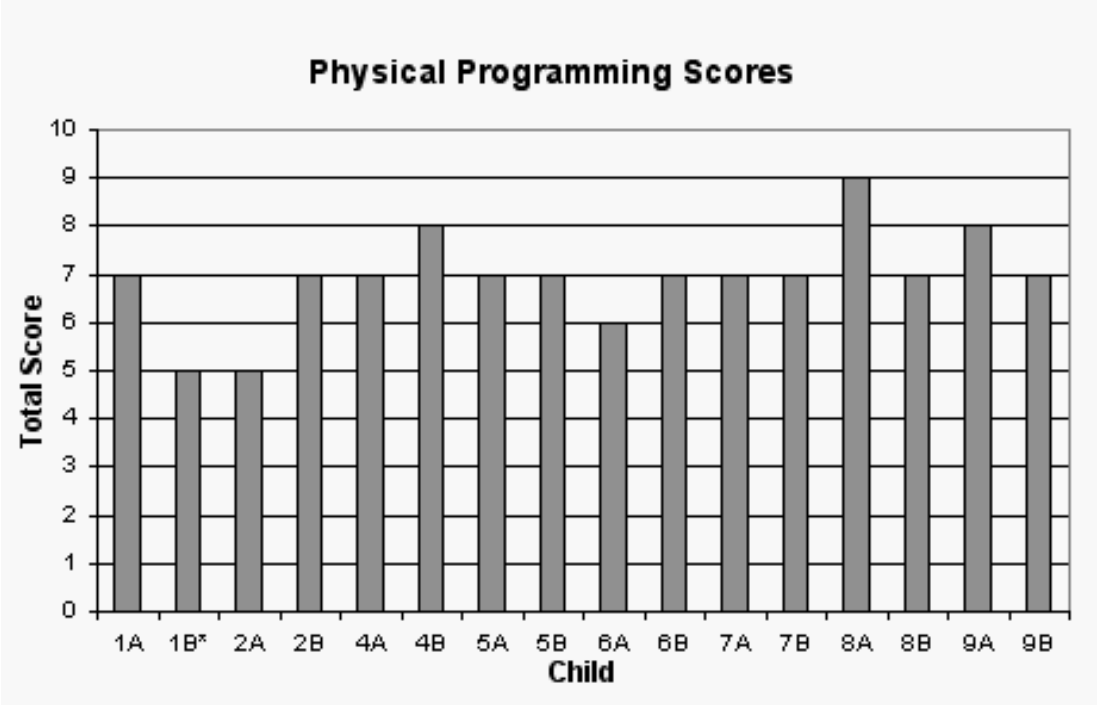

Figure 8.5: Percentage of possible points that each child scored on physical programming tasks.

the modal changes.

Both by direct observations[2] and inspections of the interaction rules captured by the StoryRoom application, I was able to see that **delineating two different interaction rules was a particularly difficult concept for the children**. One common action was to hold the button down while they waved the wand over different icons. Another was to push the button once for each icon they wanted to include into an interaction rule. There are two obvious choices to address this problem: 1) find a different interaction, or 2) find a way to teach the interaction. I did not attempt the first approach. However, as the study progressed I began to pick up some of the children's own language to describe the interactions. When I used the "invisible wire" metaphor 3.6.2 to explain

[2]The video data could not reveal the use of the *new-spell* button because we did not have a way to capture close-up activities of the hand.

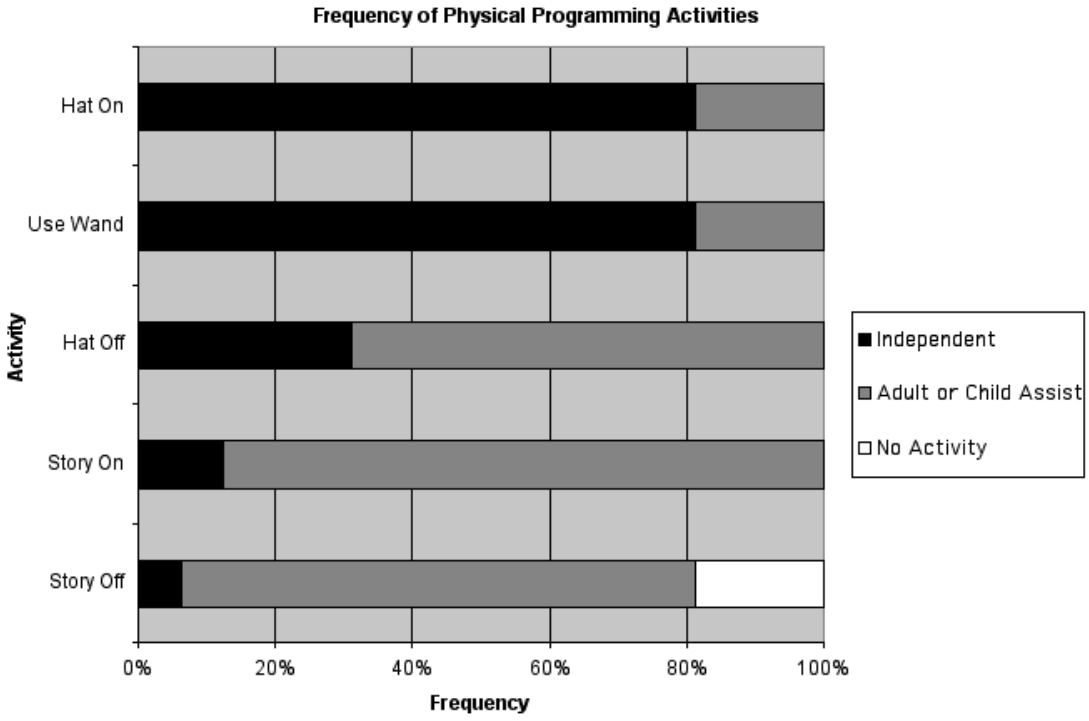

Figure 8.6: Frequency count of programming activities.

when to use the *new-spell* button , they appeared to understand the concept more easily.

In addition to the confusions that stemmed from the record and play modes, I was surprised to see that the physical icons themselves were the cause of unexpected interactions too. The icons were all made of foam and covered with felt. This apparently created an affordance that children could not resist: they wanted to squeeze everything, and not just contact sensors. While the children were able to learn the different uses of the physical icons, a better solution would be to work with the kindergarten children to find better representations that can clearly indidate input and output features.

8.2.3 Can the Children Use Physical Programming to Create an Original StoryRoom?

This question necessitated a much more in-depth approach than the previous two. In order to create a StoryRoom, each pair would have to come up with a story, create the necessary relevant props out of common art supplies, set up their story in the room, and program the interactions. The ATM decided to only work with two pairs for this activity, but to work with these pairs in an in-depth manner. Because the earlier data showed that almost all of the children could program with some degree of adult support, results from the retelling section of activity one were analyzed in order to select pairs for this case study. The two pairs chosen were the pairs with the highest and lowest scores on retelling the Irene story. In this way, I hoped to better understand the abilities of children at each end of the spectrum of StoryRooms use.

Because of the case study-like nature of this task, the adults observed each pair in detail. To analyze the children's stories I asked five questions about the process of creating a StoryRoom.

1. Can the children create a story with a plot, characters, and a setting?
2. Can the children make appropriate props for their story?
3. Can the children program their Story Room?
4. Can the children appropriately integrate the StoryRooms technology into their story?
5. Can the children play or retell the StoryRoom they created by telling a story involving props and aided by the use of the StoryRooms technology?

8.2.4 Case Study One: Bobby and Dennis

The first pair chosen was the highest scoring pair on activity one. These two Caucasian boys, *Bobby* and *Dennis*[3], are both age five. These children both come from two-parent homes and have no siblings. It was both Bobby and Dennis's third year at the CYC. They worked with us for four consecutive days for approximately 45 minutes each day on the task of creating a story using StoryRooms.

On the first day, Bobby and Dennis came up with a plot for their story which follows. The letters in parenthesis show who conceived each part of the story. Italics indicate a mention of how physical programming devices could be integrated into the story.

> A little girl was combing her hair when the sink came on all by itself (Dennis). She knew her dog could help her find what was wrong with the sink (researcher). The dog's name was Rocket (researcher). Rocket didn't know what happened (Dennis). *We could put the purple arrow next to the sink so everybody will know that the sink is broken (Bobby)*. There was a bad ghost in the sink (Dennis). The girl scared the ghost away with the mask (Dennis). The ghost ran away to a cave (Bobby).

Bobby and Dennis decided that they would need a brush, a ghost, a dog, a sink, a cave, and a mask as props for their story (figure 8.7). Using low-tech art supplies, Bobby and Dennis worked with us to construct these props. On the second day, they used their props and StoryRooms icons to set up the story. Both Bobby and Dennis were given a chance to set up and program the story. Each child had a different way that they wanted to arrange the props and icons. After coming to a consensus, they then programmed the story and practiced telling it to us.

[3] All names have been changed to protect the identity of the children.

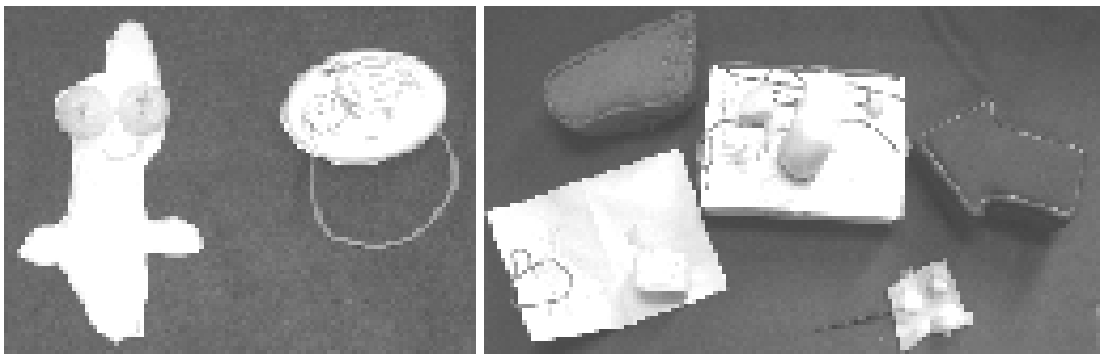

Figure 8.7: Example props in Bobby and Dennis' story. From left to right, the ghost, the mask for scaring the ghost away, Rocket the dog, the sink, and the comb. Adjacent to the sink are the foot and arrow icons.

On the third and fourth days, Bobby and Dennis set up their StoryRoom and told the story to selected classmates and their teachers (figure 8.8). The classmates often got involved in the StoryRoom by asking questions about the story (e.g., "Once upon a time what?") or by pushing icons themselves, and by participating on the floor with Bobby and Dennis. The teachers remained seated in chairs when listening to the story, but were quite engaged with their students' work.

8.2.5 Case Study Two: Mary and Shelly

Mary and *Shelly* were chosen so the ATM could understand the potential for StoryRooms with a pair of children who scored lower on the retelling. Both Mary and Shelly are females and are 5 years old. Both Mary and Shelly come from two-parent homes. Mary is Chinese-American and speaks Chinese at home, is bilingual, and has an older brother. It is her second year at the CYC. Shelly was born in Korea. She moved to the U.S. with her parents and her younger brother one month before the school year began, and is in the process of learning English. It is Shelly's first year at the CYC. The ATM worked with Mary and Shelly for three consecutive days for

Figure 8.8: Bobby and Dennis share their story with classmates.

approximately 45 minutes each day.

On the first day, Mary and Shelly were given the same prompts as Bobby and Dennis and were asked to come up with a story that they could tell using the StoryRooms technology. The story they chose was a retelling of a story that they saw on *Dragon Tales*, a popular animated television show for children in the United States. Although Mary and Shelly were asked several times to tell an original story, instead they wanted to tell the Dragon Tales story. Here is their story.

Max and Emmy moved to a new house and they found a magic wish in a drawer (M). They made the wish come true by saying "I wish I wish with all my heart to fly with dragons in a land apart" (S). This wish took them to Dragon Land (M). There they went to Dragon School (M). At the dragon school they met lots of dragons like Zack, Weezie and Ord (M). *(Note: there was no mention of how physical programming devices could be integrated into the story)*

Mary and Shelly decided that in order to tell this story, the props they would need were

Max and Emmy's house, a drawer for a magic box, the dragon school, and dragons. They then worked with low-tech art supplies and with the adults' help the children made the drawer with the magic box, the dragon school, and some dragons.

On day two, Mary and Shelly were given their props and asked to program the StoryRoom. Mary and Shelly set up the props and icons around the room, but in two different places: one for the props that they made and one for the StoryRoom icons. There was no apparent connection between the prop group and the icon group. Mary flipped the play story switch and expected the icons to work before either she or Shelly programmed them. Mary needed to be prompted explicitly by adults to remember that she needed to program the icons. One adult asked, "How are you going to get magic in those (the icons)?" and another hummed the music that plays during programming before Mary remembered that she needed to use the hat and wand to program. When Mary did use the wand to program, she connected all of the sensors and actuators in one command, which meant that pushing all of the actuators at one time would cause all of the sensors to go off. In this situation, pushing one sensor will not cause any actuator to go off. Mary did remember to flip the play story switch in order to test the icons, but did not realize that she had connected all of the sensors to all of the actuators. She pushed on one sensor at a time expecting something to happen. During this time, Shelly was not paying attention to the StoryRooms task.

When she tried programming again, Mary connected two hands, a foot, and two arrows. At this point, an adult asked, "What do you do if you're done with that spell?" to prompt Mary to use the new spell button to create a new command. At this point, Mary put the hat and wand away, ending the programming mode. Because of the manner in which Mary connected the sensors and actuators, she would have to push on both hands and the foot in order to activate both arrows. She tried pushing the hand

and the foot and then tried just a hand. Therefore it was assumed that Mary did not understand how to "play" the StoryRoom she had just made.

When again prompted to use the props and icons together, Mary put an arrow pointing to the school prop that the children made but also put another arrow pointing to a foot, which is a StoryRoom icon and not a prop. This is significant because it shows that Mary was not distinguishing between the functions of props and icons.

On day three, Mary and Shelly again had trouble programming. Shelly was more engaged on this day, but when magic was mentioned, she pantomimed sprinkling magic dust on the icons. She also said "abracadabra" when using the magic wand with the icons, and said that this was making them work. Mary and Shelly spent time on this day repeatedly picking up and down the magic hat, which turns the programming music on and off, and turning the play story switch on and off, which caused it to repeatedly say "once upon a time" and "the end". Shelly also appeared to enjoy when the StoryRoom gave auditory feedback (such as "yellow foot" when the yellow foot was pressed). When asked again to tell their story, the girls used their props but not the StoryRoom icons to tell a Dragon Tales story, this time telling a different story than the one they had planned.

8.3 Analysis

The two pairs of children in the case study performed very differently in their attempts to create StoryRooms. Both groups created stories with plots and created appropriate props that suggest a setting and characters. However, the disparities between the groups became apparent when it came to programming and integrating the technology with their narratives. Bobby and Dennis were both able to create independent

interaction rules. Mary and Shelly were unable to perform this task; instead, they programmed all of the actuators to go on at once. Furthermore, Bobby and Dennis were able to integrate the StoryRooms technology into their story. Mary and Shelly were not – they programmed the icons separately from the story and did not relate the StoryRooms icons to the props or events in their story. Finally, Bobby and Dennis were able to retell their story to their peers and teachers using the StoryRooms technology. Mary and Shelly did not progress that far in their storytelling experience.

8.4 Lessons Learned from this Usability Study

Through this research, I learned valuable information that will help to direct my future work with StoryRooms. Children may need more prompting when using physical programming technologies such as StoryRooms. My experience with Mary and Shelly taught me that providing more feedback during physical programming can help children to be more successful. This could be supported in future versions of the StoryRoom technologies. For example, the magic wand could provide an audible cue when children are finished with a spell or starting a new spell. In addition, the icons could visually show a child if she had connected icons together, or a new tool could allow children to see which icons had connected in a physical interaction instruction. On the other hand, the novelty of the StoryRoom can sometimes hinder children from using the StoryRoom for its intended purpose of telling stories. For example, Mary and Shelly spent a lot of time picking up and putting down the wizard hat in order to make the ambient music start and stop.

I also learned that for children to understand, predict, and control the interactions in their environment, it may be necessary to expose the system components (i .e., give

them symbolic objects for sensor and actuator tasks). However, Mary and Shelly had challenges integrating the sensors and actuators with the props into one physical story. This may be due to their inability to abstract certain programming concepts, but this may also have to do with the system characteristics. The icons may have been too easily identifiable as say a foot or hand. Some children such as Mary and Shelly tended to focus on those characteristics (as in, "I see a foot") and forget that the item had another purpose, which was to be an interactive proxy for a prop. I think in some cases, this may have been due to the relationship between sensing devices and the physical environment. For instance, a child may place a large icon next to small cottage, making the icon more visually important than the prop.

On the other hand, exposing the system components may not have been quite as obvious as it needed to be in some cases. For instance, Mary just could not relate the *new-spell* button to the programming activity of creating a new interaction group. I believed that this button was perhaps not an appropriate metaphor for more challenged children. Perhaps the visual metaphors for sensors and actuators need to be carefully reconsidered. So while I have revised the physical interfaces many times, further revision to this system needs to be accomplished.

Another obstacle to consider for children using these technologies may have been the system's ruggedness and reliability. There were times that our current RFID system could not respond correctly to children's natural movements (e.g., heavy punching of sensors, constant repetition). A lack of timely feedback often led to unpredictable technology behaviors, which we found could confuse children quite quickly. This study further confirmed that these technologies must be extremely rugged and flexible for children to control in ways that are cognitively and physically appropriate.

In summary, a tool for children to control ubicomp environments demands extremely

reliable, rugged, and flexible technologies they can control. In addition, a balance needs to be struck between visible concrete metaphors for these technologies and integrating these technologies into the environment for storytelling.

Chapter 9

Final Words

I have shown that physical programming is a simple and direct way for children to program interactions in a special ubiquitous computing environment called StoryRooms. While the data show that not all the children were completely capable of the creative process (creating a story) or the programming process, that some children were more than competent users of this approach is extremely encouraging.

I have also shown that the physical programming language and its resultant StoryRooms are equivalent to deterministic finite automata. Moreover, I suggested that the single most important improvement to the language would be the addition of finite memory.

The goal of this dissertation was to understand the relationships among children, ubicomp environments, and control. Thanks to my experience with StoryRooms, I can now offer some insights about general interactive systems. First, when children can place sensors and actuators in their surroundings, the system can better conform its behavior. Second, children do not have to be confused by modal changes, such as between programming and playback, if they are given unambiguous signals by the system (the wizard's hat). Third, extremely simple physical interactions (waving, pushing

button), in conjunction with a compelling tool (magic wand) are sufficient for children to indicate programming intentions.

This means that a ubiquitous computing environment for children should contain these features:

1. Sensors and actuators that can be attached anywhere in the space.
2. Tools and ambient signals to unambiguously indicate mode.
3. Tools with a compelling underlying metaphor to create interaction rules.

I have learned that children CAN, without any adult assistance, control the behavior of a ubiquitous computing environment.

9.1 Revisit the Questions on Control and Tools

Here are the answers the questions that I first posed in the Introduction: What do children need to control the interactions of a physical interactive environment (table 9.1).

Table 9.1: Answers to control and tools.

Question	Answer
What kind of tools are needed?	The tools should 1) be concrete objects to minimize abstractions, and 2) clearly define and set modes.
What do the tools look like?	They should utilize metaphor that is child appropriate (e.g., magic).
How are they used?	The tools are operated by simple gestures within the metaphor.

Table 9.1: Answers to control and tools, continued.

Question	Answer
Do the tools require new interaction models?	Yes. The tools rely on physical syntax that are combinations of physical gesture primitives.
Can children in fact use the tools?	Yes. Refer to the Usability chapter (8)
Implicitly, the ability to control may require a programming model. What is that model?	Physical programming and *PIE*.

9.2 Limitations of the Research

Although I was pleased by my work on StoryRooms, physical programming, and the results of the usability study, I believe the results would have been more useful if I had spent more than one month with the children at the CYC. It was clear that Bobby and Dennis cognitively understood the programming tools and the concept of a room that can express stories. But I am more concerned about Mary and Shelly. Did the two pairs represent the boundary conditions of the cognitive abilities of six year old children, in terms of storytelling and programming? Or, did Mary and Shelly just need more time. And all the other children in between, would they have also benefited from more time learning about my system? These are all questions that require a more significant investment in time and personnel.

Much of the early designs on the physical programming tools and interactions were inspired by the 7-11 year old design partners. I did not have the chance to perform a usability on THIS population. I would be extremely excited to see the result of such a study.

The premise of my research was to understand the relationship between young children and ubicomp environments. I created a deliberately narrow storytelling environment so that children could latch on to something familiar. The natural follow-up question is whether the same tangible tools (perhaps with a different look) and interaction metaphors could carry over to any general ubicomp environment. One future direction might be to collaborate with researchers of a well-established ubicomp environment, perhaps with ties to universal accessibility and universal control, enhance their devices with the minimum software and hardware requirements of StoryRoom/physical programming, and introduce the IDT's young designers into the mix.

My work has been consistently been a tension between what I can make (because of hardware limitations) and what I should make. So am I imposing a solution onto the children, even though it may not be optimal? Thankfully, this is where the magic of the IDT comes in. Because the children and the adults are in the design TOGETHER, everyone is aware of the issues involved. So if a compromise had to be made, it was with the agreement of the entire team.

9.3 The Lab's On-going Work: HazardRoom

Soon after I completed my StoryRooms project, another graduate student embarked on a project to extend the StoryRoom architecture. This project, HazardRoom, is being used to teach children about the many hazardous materials in their environment. Unlike the StoryRoom, which is a free form creative environment, the HazardRoom is intended to be a constrained content-based learning experience. That is, the environment would already be setup to contain the knowledge that teachers would want students to learn. The technologies and concepts for this work is being developed

today.

9.4 My Future Work

In the near term, my work would include the realization of the tools I mentioned in Chapter 9.4.1: Sound, Magic Lens, and Counter. I want to engage the IDT to think about the idea of memory and what its symbols look like in the physical environment. I would like to review the current state of technology and see whether smaller versions of the physical objects could be built.

9.4.1 Additional Physical Programming Interfaces

In prior chapters I described the tools and icons of the physical programming system. Conceptual prototypes were developed for several tools to both extend the usability of the system as well as the power of the language.

The Sound Board

Recorded narration and sound effects are critical elements of StoryRooms. I designed a prototype of the **Sound Board** and the **Sound Stick** for children to easily record sounds to enhance the storytelling experience. The Sound Board was a colorful platform with many different colored patches (**Sound Patch**) on the top. Within each patch was a hole to hold the Sound Stick (figure 9.1). To record a narrative, children would place the Sound Stick into a hole and speak into the stick. The color patch associated with the hole would then become the symbolic holder of the sound.

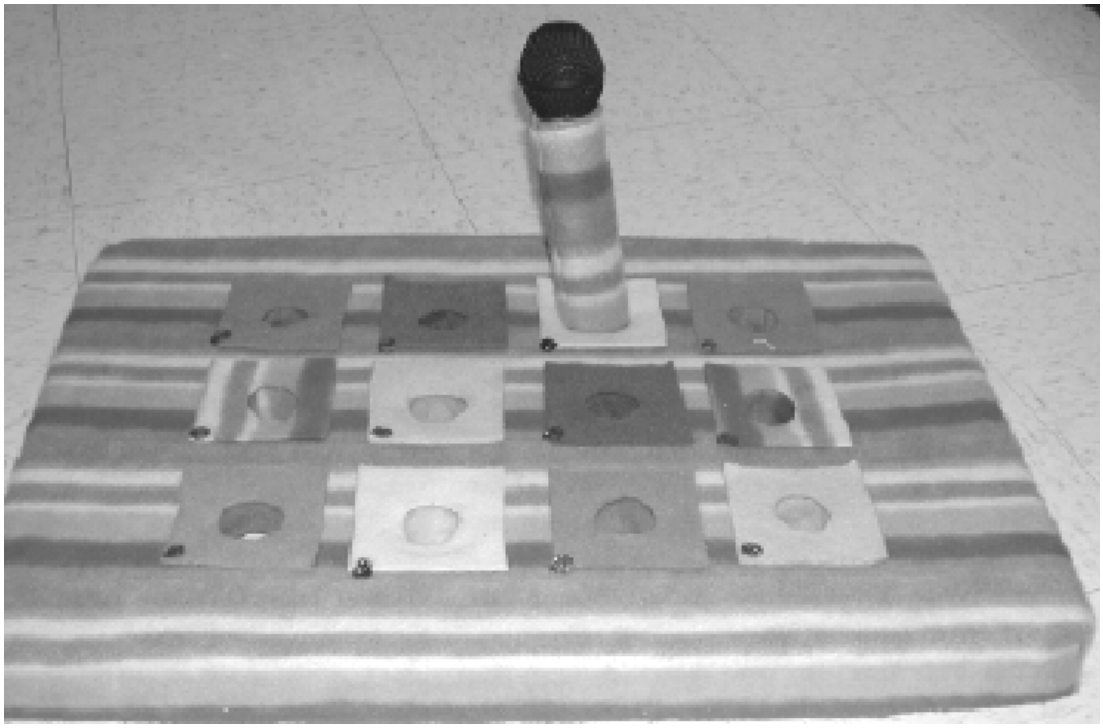

Figure 9.1: A Conceptual Sound Board. A Sound Stick is currently placed inside a Sound Patch.

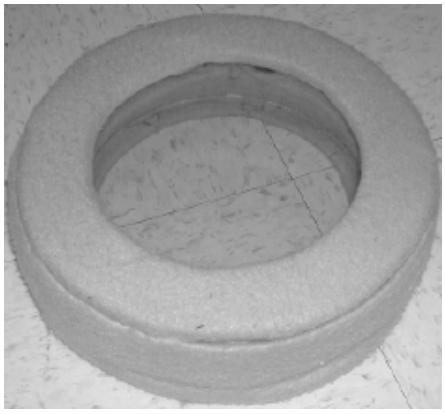

Figure 9.2: A Conceptual Magic Lens. A child can query the state of an icon by holding the disk over it.

Here is a scenario to better illustrate the use of the Sound Board. Suppose I want to hear the words, “I am happy you are here.” when someone presses the blue hand physical icon. I would do the following:

1. Associate a Sound Patch to contain the words, “I am happy you are here.”
2. Wear the wizard’s hat to invoke the programming mode.
3. Press on the *new-spell* button on the magic wand to begin a new rule.
4. Wave the wand over the blue hand physical icon.
5. Wave the wand over the Sound Patch used in step 1.

The Magic Lens

Programming and debugging go hand-in-hand. The **Magic Lens** may be a direct method for children to get information about objects within a StoryRoom (figures 9.2, 9.3, and 9.4).

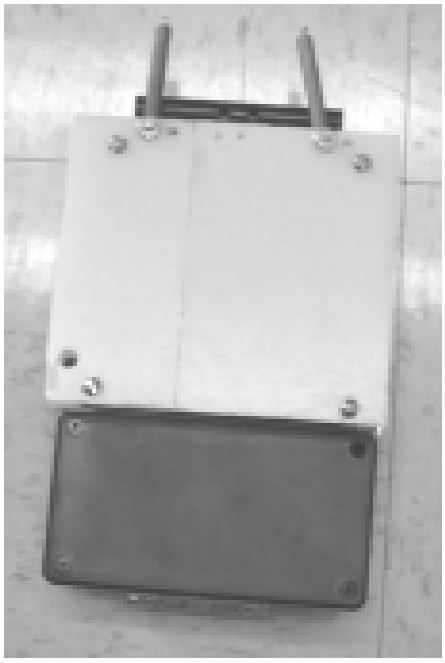

Figure 9.3: Another conceptual magic lens. This lens is in the closed position.

A Magic Lens can be extremely useful. For example, when children move the lens over an object, the lens tool combines the images captured by the camera and the information gathered by the RFID reader to display a composite drawing of the object and its relations to other objects in the room.

Clock, Timer, or Counter

Counting is a natural element of any PIE. It is useful because we often want to control devices relative to time. A physical counter, perhaps in the form of a clock, can offer a simple approach for children to include time into physical programming.

For example, after 10 seconds, turn on the light icon and leave it on for 15 seconds. To program this rule, I would:

1. Wear the wizard's hat to invoke the programming mode.

Figure 9.4: The open position of the magic lens in (figure 9.3). The top third of the assembly contains an RFID reader. The middle section might hold a palm-sized computer with a screen to display relevant information. The bottom third contains the power supply, communication, and microcontroller units. The side facing the object (not shown) would contain a camera.

2. Press on the *new-spell* button on the magic wand to begin a new rule.

3. Wave the wand over the clock tool 10 times.

4. Wave the wand over the light.

5. Wave the wand over the clock tool 15 times.

9.4.2 Collaboration Potentials

The current success of physical interface components such as iStuff [7], Phidgets [42], and X10 [118], and their lack of a physical programming interface may be an opportunity. Here the idea would be to separate physical programming from the StoryRooms environment, and to allow physical interactions to generate rules for the above systems.

9.4.3 Connection to Universal Accessibility and Universal Control

In the long term, I would like to connect my work to the universal control and universal access areas. This could open up the opportunity of a physical, simple to use device for use by other special needs populations (e. g., senior citizens) to dictate their special requirements to the environment.

BIBLIOGRAPHY

[1] ABOWD, G. D., AND MYNATT, E. D. Charting past, present and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction, Special Issue on HCI in the New Millenium 7*, 1 (March 2000), 29–58.

[2] ALBORZI, H., DRUIN, A., MONTEMAYOR, J., PLATNER, M., PORTEOUS, J., SHERMAN, L., BOLTMAN, A., TAXÉN, G., BEST, J., HAMMER, J., KRUSKAL, A., LAL, A., PLAISANT-SCHWENN, T., SUMIDA, L., WAGNER, R., AND HENDLER, J. Designing StoryRooms: Interactive storytelling spaces for children. In *Proceedings of Designing Interactive Systems (DIS-2000)* (2000), ACM Press, pp. 95–104.

[3] ALLIANCE FOR CHILDHOOD. Fool's gold: A critical look at computers in childhood. url: www.allianceforchildhood.net/projects/computers/computers_reports_fools_gold_contents.htm, 2001.

[4] ANJANEYULU, K. S. R., AND ANDERSON, J. R. The advantages of data flow diagrams for beginning programming. In *Intelligent Tutoring Systems: Second International Conference* (1992), C. Frasson, G. G., and G. I. McCalla, Eds., pp. 585–592.

[5] ANNANY, M., AND CASSELL, J. TellTale: A toy to encourage written literacy skills through oral storytelling. In *Winter Conference on Text, Discourse and Cognition* (2001).

[6] BACK, M., COHEN, J., GOLD, R., HARRISON, S., AND MINNEMAN, S. Listen Reader: An electronically augmented paper-based book. In *Proceedings of Human Factors in Computing Systems* (2001), ACM Press, pp. 23–29.

[7] BALLAGAS, R., RINGEL, M., STONE, M., AND BORCHERS, J. iStuff: A physical user interface toolkit for ubiquitous computing environments. In *Proceedings of Human Factors in Computing Systems* (2003), ACM Press, pp. 537–544.

[8] BEIGL, M. Using spatial co-location for coordination in ubiquitous computing environments. In *Handheld and Ubiquitous Computing, First International Symposium, HUC'99, Karlsruhe, Germany, September 27-29, 1999, Proceedings* (1999), H.-W. Gellersen, Ed., vol. 1707 of *Lecture Notes in Computer Science*, Springer.

[9] BELLOTTI, V. M. E., BACK, M. J., EDWARDS, W. K., GRINTER, R. E., LOPES, C. V., AND HENDERSON, A. Making sense of sensing systems: Five questions for designers and researchers. In *Proceedings of Human Factors in Computing Systems* (2002), ACM Press, pp. 415–422.

[10] BERMAN, R. Preschool knowledge of language: What five-year olds know about language structure and language use. In *Writing development: An interdisciplinary view*, C. Pontecorvo, Ed. John Benjamins Publishing, 1977.

[11] BILLINGHURST, M., KATO, H., AND POUPYREV, I. The MagicBook - Moving seamlessly between reality and virtuality. *Computer Graphics and Applications 21*, 3 (2001), 2–4.

[12] BLACKWELL, A. F., AND HAGUE, R. AutoHAN: An architecture for programming the home. In *Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments* (2001), pp. 150–157.

[13] BOBICK, A., INTILLE, S. S., DAVIS, J. W., BAIRD, F., PINHANEZ, C. S., CAMPBELL, L. W., IVANOV, Y. A., SCHÜTTE, A., AND WILSON, A. The KidsRoom: A perceptually-based interactive and immersive story environment. In *PRESENCE: Teleoperators and Virtual Environments* (August 1999), pp. 367–391.

[14] BONASSO, R. P., FIRBY, R. J., GAT, E., KORTENKAMP, D., MILLER, D., AND SLACK, M. Experiences with architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence* (1997), 237–256.

[15] BROOKS JR., F. P. No silver bullet: essence and accidents of software engineering. *Computer 20*, 4 (1987), 10–19.

[16] BROSTERMAN, N. *Inventing Kindergarten*. Harry N. Adams Inc., 1997.

[17] BRUCHAC, J. *Survival this way: Interviews with American Indian poets*. University of Arizona Press, Tuscson, Arizona, 1987.

[18] BRUNER, J. *Toward a theory of instruction*. Harvard University Press, 1966.

[19] BURNETT, M. M., BAKER, M. J., BOHUS, C., CARLSON, P., YANG, S., AND VAN ZEE, P. Scaling up visual programming languages. *IEEE Computer 28*, 3 (1995), 45–54.

[20] CURTIS, B., SHEPPARD, S., KRUESI-BAILEY, E., BAILEY, J., AND BOEHM-DAVIS, D. Experimental evalutation of software documentation formats. *Journal of Systems and Software 9*, 2 (1989), 167–207.

[21] CYPHER, A., AND SMITH, D. Kidsim: End-user programming of simulations. In *Proceedings of Human Factors in Computing Systems* (1995), ACM Press, pp. 27–34.

[22] DRUIN, A. Research notes. 1998.

[23] DRUIN, A. Cooperative inquiry: Developing new technologies for children with children. In *Proceedings of Human Factors in Computing Systems* (1999), ACM Press, pp. 592–599.

[24] DRUIN, A. The role of children in the design of new technology. *Behaviour and Information Technology (BIT) 21*, 1 (2002), 1–25.

[25] DRUIN, A. When technology does not serve children. *SIGCHI Bulletin 34*, 4 (July/August 2002).

[26] DRUIN, A., BEDERSON, B., BOLTMAN, A., MIURA, A., KNOTTS-CALLAHAN, D., AND PLATT, M. Children as our technology design partners. In *The design of children's technology*, A. Druin, Ed. Morgan Kaufmann, 1999, pp. 51–72.

[27] DRUIN, A., BEDERSON, B., HOURCADE, J. P., SHERMAN, L., REVELLE, G., PLATNER, M., AND WENG, S. Designing a digital library for young chil-

dren: An intergenerational partnership. In *Proceedings of ACM/IEEE Joint Conference on Digital Libraries (JCDL 2001)* (2001), pp. 398–405.

[28] DRUIN, A., MONTEMAYOR, J., HENDLER, J., MCALISTER, B., BOLTMAN, A., FITERMAN, E., PLAISANT, A., KRUSKAL, A., OLSEN, H., REVETT, I., PLAISANT-SCHWENN, T., SUMIDA, L., AND WAGNER, R. Designing PETS: A personal electronic teller of stories. In *Proceedings of Human Factors in Computing Systems* (1999), ACM Press, pp. 326–329.

[29] DRUIN, A., AND PERLIN, K. Immersive environments: A physical approach to the computer interface. In *Proceedings of Human Factors in Computing Systems* (1994), vol. 2, ACM Press, pp. 325–326.

[30] Edwin Schlossberg Incorporated, 641 Sixth Avenue, New York, NY 10011. url: http://www.esidesign.com/.

[31] EHN, P. Scandinavian design: On participation and skill. In *Participatory design: Principles and practices*, D. Schuler and A. Namioka, Eds. Lawrence Erlbaum, 1993, pp. 41–77.

[32] FARBER, A., DRUIN, A., CHIPMAN, G., JULIAN, D., AND SOMASHEKHAR, S. How young can our design partners be? Tech. rep., University of Maryland Insitute of Advanced Computer Studies. UMIACS-TR-2002-76, 2002.

[33] FITZMAURICE, G. W., ISHII, H., AND BUXTON, W. Bricks: Laying the foundations for graspable user interfaces. In *Proceedings of Human Factors in Computing Systems* (1995), pp. 442–449.

[34] FLOYD, R. W., AND BEIGEL, R. *The Language of Machines: An introduction to Computability and Formal Languages.* Computer Science Press, 1994.

[35] FREI, P., SU, V., MIKHAK, B., AND ISHII, H. Curlybot: Designing a new class of computational toys. In *Proceedings of Human Factors in Computing Systems* (2000), ACM Press, pp. 129–136.

[36] GEISEL, T. *The Sneetches, and Other Stories*. Random House, New York, 1961.

[37] GISH, R. F. *Beyond bounds: Cross–Cultural essays on Anglo, American Indian, and Chicano literature*. University of New Mexico Press, Albuquerque, NM, 1996.

[38] GREEN, T. R. G. Noddy's guide to ... visual programming. *Interfaces* (1995).

[39] GREEN, T. R. G., AND PETRE, M. Usability analysis of visual programming environemnts: a 'cognitive dimensions' framework. *Journal of Visual Languages and Computing* (1996), 131–174.

[40] GREENBAUM, J. A design of one's own: Toward participatory design in the united states. In *Participatory design: Principles and practices*, D. Schuler and A. Namioka, Eds. Lawrence Erlbaum, 1993, pp. 27–37.

[41] GREENBAUM, J., AND KYNG, M. *Design at work: Cooperative design of computer systems*. Lawrence Erlbaum, 1991.

[42] GREENBERG, S., AND FITCHETT, C. Phidgets: Easy development of physical interfaces through physical widgets. In *Proceedings of UIST* (2001), pp. 209–218.

[43] GREENFIELD, P. M. *Mind and media*. Harvard University Press, 1984.

[44] HILS, D. D. Visual languages and computing survey: Data flow visual programming languages. *Journal of Visual Languages and Computing 3* (1992), 69–101.

[45] HOURCADE, J. P. *User Interface Technologies and Guidelines to Support Children's Creativity, Collaboration, and Learning*. PhD thesis, University of Maryland, College Park, 2003.

[46] HOURCADE, J. P., BEDERSON, B., DRUIN, A., ROSE, A., FARBER, A., AND TAKAYAMA, Y. The international childrens digital library: Viewing digital books online. In *Proceedings of Interaction Design and Children International Workshop* (2002), Shaker Publishing, pp. 125–128.

[47] HOURCADE, J. P., BEDERSON, B., DRUIN, A., AND TAXÉN, G. Kidpad: Collaborative storytelling for children. In *Proceedings of Human Factors in Computing Systems, Extended Abstracts* (2002), ACM Press, pp. 500–501.

[48] HOYLES, C., NOSS, R., ADAMSON, R., AND LOWE, S. Programming rules: what do children understand? In *Proceedings of the Twenty Fifth Annual Conference of the International Group for the Psychology of Mathematics* (2001).

[49] INGEN-HOUSZ, T. Available at http://www.apple.com/education/LTReview/spring99/elephant/.

[50] ISHII, H., AND ULLMER, B. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proceedings of Human Factors in Computing Systems* (1997), ACM Press, pp. 234–241.

[51] KAHN, K. Generalizing by removing detail: How any program can be created by working with examples, 2000. Available at http://www.animated-programs.com/PBD/index.html.

[52] KELLEHER, C., AND PAUSCH, R. Lowering the barriers to programming: a survey of programming environments and languages for novice programmers. Tech. Rep. CMU-CS-03-137, School of Computer Science, Carnegie Mellon University, 2003.

[53] KURLANDER, D. Characterizing pbd systems. In *Watch What I Do: Programming by Demonstration*, A. Cypher, D. C. Halbert, D. Kurlander, H. Lieberman, D. Maulsby, B. A. Myers, and A. Turransky, Eds. MIT Press, 1993, ch. 12.

[54] LAMORISSE, A. The Red Balloon, 1956.

[55] LAVE, J. *Cognition in practice*. Cambridge University Press, 1992.

[56] MACKAY, W., VELAY, G., CARTER, K., MA, C., AND PAGANI, D. Augmenting reality: Adding computational dimensions to paper. *Computer-Augmented Environ-ments: Back to the Real World. Special issue of Communications of the ACM 36*, 7 (1993).

[57] MACKAY, W. E. Augmented reality: Linking real and virtual worlds (Keynote Address). In *Proceedings of Conference on Advanced Visual Interfaces* (1998), ACM Press, pp. 1–9.

[58] MALONE, T. W. Heuristics for designing enjoyable user interfaces. In *Proceedings of Human Factors in Computing SystemsGathersburg Conference* (1982), ACM Press, pp. 63–68.

[59] MARTIN, F., MIKHAK, B., RESNICK, M., SILVERMAN, B., AND BERG, R. To mindstorms and beyond: Evolution of a construction kit for magical machines. In *Robots for kids: New technologies for learning*, A. Druin and J. Hendler, Eds. Morgan Kaufmann, San Francisco CA, 2000, pp. 9–33.

[60] MARTIN, F. G. The handy board technical reference. Available at http://www.handyboard.com/techdocs/hbmanual.pdf, 1998.

[61] MCNERNEY, T. S. Tangible programming bricks: An approach to making programming accessible to everyone. Master's thesis, MIT Media Lab, 2000.

[62] http://www.microchip.com.

[63] MODLEY, R. *Handbook of Pictorial Symbols: 3,250 Examples from International Sources.* Dover Publications, 1976.

[64] MONTEMAYOR, J., DRUIN, A., CHIPMAN, G., FARBER, A., AND GUHA, M. L. Sensing, storytelling, and children: Putting users in control. Tech. rep., University of Maryland. UMIACS-TR-2003-16, January 2003.

[65] MONTEMAYOR, J., DRUIN, A., FARBER, A., SIMMS, S., CHURAMAN, W., AND D'AMOUR, A. Physical Programming: Designing tools for children to create physical interactive environments. In *Proceedings of Human Factors in Computing Systems* (2002), ACM Press, pp. 299–306.

[66] MONTEMAYOR, J., DRUIN, A., AND HENDLER, J. PETS: A personal electronic teller of stories. In *Robots for kids: New technologies for learning*, A. Druin and J. Hendler, Eds. Morgan Kaufmann, San Francisco CA, 2000, pp. 367–391.

[67] MONTEMAYOR, J., DRUIN, A., AND HENDLER, J. From PETS to storyrooms, constructive storytelling systems designed with children, for children. In *Socially Intelligent Agents - creating relationships with computers and robots*, K. Dautenhahn, A. Bond, L. Canamero, and B. Edmonds, Eds. Kluwer Academic Publishers, 2002, ch. 25, pp. 205 – 212.

[68] MUMFORD, E., AND HENSHALL, D. *Designing participatively: A participative approach to computer systems design*. Manchester Business School, 1979/1983.

[69] MYERS, B. A. Taxonomies of visual programming and program visualization. *Journal of Visual Languages and Computing 1*, 1 (1990), 97–123.

[70] MYERS, B. A. Using hand-held devices and pcs together. In *Communications of the ACM*, vol. 44. ACM Press, November 2001, pp. 34–41.

[71] NARDI, B. *A Small Matter of Programming: Perspectives on End-User Computing*. MIT Press, 1993.

[72] NISHI, T., SATO, Y., AND KOIKE., H. Interactive object registration and recognition for augmented desk interface. In *ACM SIGCHI 2001 (short talk)*. ACM Press, April 2001, pp. 371–372.

[73] OBERLANDER, J., BRNA, P., COX, R., AND GOOD, J. *The Match-Mismatch Conjecture and Learning to Use Data-Flow Visual Programming Languages*. The University of Leeds, 1999. Available at http://www.cbl.leeds.ac.uk/ paul/grip.html.

[74] ORTIZ, S. J. *Speaking for generations: Native writers on writing*. University of Arizona Press, Tuscson, AR, 1998.

[75] PAPERT, S. *Mindstorms: Children, computers and powerful ideas*. Basic Books, New York, 1980.

[76] PAPERT, S. *The Children's Machine: Rethinking School in the Age of the Computer*. Basic Books, 1993.

[77] PAUSCH, R., VOGTLE, L., AND CONWAY, M. One dimensional motion tailoring for the disabled: A user study. In *Proceedings of Human Factors in Computing Systems* (1992), ACM Press, pp. 405–411.

[78] PIAGET, J. *Psychology and Epistemology: Towards a theory of knowledge*. Viking Press, 1971.

[79] PIAGET, J. *To understand is to invent: The future of education*. Grossman, 1973.

[80] PINHANEZ, C. S., DAVIS, J. W., INTILLE, S., JOHNSON, M. P., WILSON, A. D., BOBICK, A. F., AND BLUMBERG, B. Physically interactive story environments. *IBM Systems Journal 39*, 3–4 (2000), 438–455.

[81] PLAISANT, C., DRUIN, A., LATHAN, C., DAKHANE, K., EDWARDS, K., VICE, J. M., AND MONTEMAYOR, J. A storytelling robot for pediatric rehabilitation. In *Proceedings of ASSETS'2000* (2000), ACM Press, pp. 50–55.

[82] http://www.portdiscovery.com.

[83] http://www.realsoftware.com.

[84] REKIMOTO, J., ULLMER, B., AND OBA, H. DataTiles: A modular platform for mixed physical and graphical interactions. In *Proceedings of Human Factors in Computing Systems* (2001), ACM Press, pp. 269–276.

[85] RESNICK, M. Technologies for lifelong kindergarten. *Educational Technology Research and Development 46*, 4 (1998), 16–19.

[86] RINGEL, M., BERG, H., JIN, Y., AND WINOGRAD, T. Barehands: Implement-free interaction with a wall-mounted display. In *Proceedings of Human Factors in Computing Systems, Extended Abstracts* (2001), ACM Press, pp. 367–368.

[87] SALBER, D., DEY, A., AND ABOWD, G. Ubiquitous computing: Defining an hci research agenda for an emerging interaction paradigm. Tech. rep., Georgia Institute of Technology, 1998.

[88] SCAIFE, M., ROGERS, Y., ALDRICH, F., AND DAVIES, M. Designing for or designing with? informant design for interactive learning environments. In *Proceedings of Human Factors in Computing Systems* (1997), ACM Press, pp. 343–350.

[89] SCARLATOS, L., DUSHKINA, Y., AND LANDY, S. Ticle: A tangible interface for collaborative learning environments. *Proceedings of Human Factors in Computing Systems, Extended Abstracts* (1999), 260–261.

[90] SCHELL, J., AND SHOCHET, J. Designing interactive theme park rides: Lessons from disney's battle for the buccaneer gold. available at: http://www.gamasutra.com/features/20010706/schell_01.htm., 2001.

[91] SCHILIT, W. N. *A Context-Aware System Architecture for Mobile Distributed Computing*. PhD thesis, Columbia University, 1995.

[92] SEMPER, R. J. Science museums as environments for learning. *Physics Today* (November 1990), 50–56.

[93] SHAFER, S. A. N., BRUMITT, B., AND CADIZ, J. Interaction issues in context-aware intelligent environments. *Human-Computer Interaction 16* (2001), 363–378.

[94] SHNEIDERMAN, B. Direct manipulation: A step beyond programming languages. *IEEE Computer 16*, 8 (1983), 56–69.

[95] SHU, N. *Visual Programming*. Van Nostrand Reinhold, 1988.

[96] SLOMAN, A. Interactions between philosophy and artificial intelligence: The role of intuition and non-logical reasoning in intelligence. In *Proceedings of the Second International Joint Conference on Artificial Intelligence* (1971), pp. 270–278.

[97] SMITH, D. C. Pygmalion: An executable electronic blackboard. In *Watch What I Do: Programming by Demonstration*, A. Cypher, D. C. Halbert, D. Kurlander, H. Lieberman, D. Maulsby, B. A. Myers, and A. Turransky, Eds. MIT Press, 1993, ch. 1.

[98] http://www.parallax.com.

[99] STROMMEN, E. Children's use of mouse-based interfaces to control virtual travel. In *Proceedings of Human Factors in Computing Systems* (1994), ACM Press, pp. 405–410.

[100] STROMMEN, E. When the interface is a talking dinosaur: Learning across media with actimates barney. In *Proceedings of Human Factors in Computing Systems* (1998), ACM Press, pp. 288–295.

[101] SUNDBLAD, Y. Quality and interaction in computer-aided graphic design (Utopia Report 15). Tech. rep., Stockholm: Arbetslivscentrum, 1987.

[102] SUPPES, P. Computer tchnology and the future of education. In *Computer-assisted instruction: A book of readings*, R. Atkinson and H. A. Wilson, Eds. Academic Press, 1969, pp. 41–47.

[103] SUZUKI, H., AND KATO, H. Interaction-level support for collaborative learning: *AlgoBlock* — an open programming language. In *Proceedings of CSCL'95* (October 1995), J. L. Schnase, Ed., pp. 349–355.

[104] TANENBAUM, A. S. *Computer Networks*, 3 ed. Prentice Hall, 1996.

[105] TANIMOTO, S. L. Tutorial notes on visual languages for computer based communication. Human Centered Computing Conference (2002).

[106] THE PRESIDENT'S COMMITTEE OF ADVISORS ON SCIENCE AND TECHNOLOGY. *Report to the President on the use of technology to strengthen K-12 education in the United States.* The Executive Office of the President of the United States, 1997.

[107] UMASCHI, M. Soft toys with computer hearts: Building personal storytelling environments. In *Proceedings of Human Factors in Computing Systems* (1997), ACM Press, pp. 20–21.

[108] WANT, R., HOPPER, A., FALCAO, V., AND GIBBONS, J. The active badge location system. *ACM Transactions on Information Systems 10*, 1 (1992), 91–102.

[109] WANT, R., PERING, T., BORRIELLO, G., AND FARKAS, K. Disappearing hardware. In *IEEE Pervasive Computing* (2002), pp. 36–47.

[110] WANT, R., SCHILIT, B. N., ADAMS, N. I., GOLD, R., PETERSEN, K., GOLDBERG, D., ELLIS, J. R., , AND WEISER, M. An overview of the parctab ubiq-

uitous computing experiment. *IEEE Personal Communications 2*, 6 (1995), 28–43.

[111] WARD, A., JONES, A., AND HOPPER, A. A new location technique for the active office. *IEEE Personal Communications 4*, 5 (October 1997), 42–47.

[112] WEISER, M. The computer for the twenty-first century. *Scientific American* (September 1991), 94–104.

[113] WEISER, M. Some computer science problems in ubiquitous computing. *Communications of the ACM* (1993).

[114] WILSON, A., AND SHAFER, S. XWand: UI for intelligence spaces. In *Proceedings of Human Factors in Computing Systems* (2003), ACM Press, pp. 545–552.

[115] WOODRUFF, A., AOKI, P., HURST, A., AND SZYMANSKI, P. The guidebook, the friend, and the room: Visitor experience in a historic house. In *Proceedings of Human Factors in Computing Systems, Extended Abstracts* (2001), pp. 273–274.

[116] WYETH, P., AND PURCHASE, H. C. Programming without a computer: A new interface for children under eight. *Australian Computer Science Conference 22*, 5 (2000), 141–148.

[117] WYETH, P., AND WYETH, G. Electronic blocks: Tangible programming elements for preschoolers. In *Human-Computer Interaction - INTERACT'01* (2001), IOS Press, pp. 496–503.

[118] http://www.x10.com.