Beyond Input Devices
A New Conceptual Framework for the Design of Physical-Digital Objects

Matthew G. Gorbet

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning, in Partial Fulfillment of the Requirements for the Degree of Master of Science in Media Arts and Sciences at the Massachusetts Institute of Technology

June 1998

Author _____

Matthew G. Gorbet
Program in Media Arts and Sciences
May 8, 1998

Certified By _____

Hiroshi Ishii
Associate Professor of Media Technology
Tangible Media Group
MIT Media Laboratory

Accepted By _____

Stephen A. Benton
Chair, Departmental Committee for Graduate Students
Program in Media Arts and Sciences

Beyond Input Devices
A New Conceptual Framework for the Design of Physical-Digital Objects

Matthew G. Gorbet

Submitted to the Program in Media Arts and Sciences
School of Architecture and Planning
on May 8, 1998
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences at the
Massachusetts Institute of Technology

Abstract

This work introduces the concept of physical-digital objects: physical objects which allow people to interact with digital information as though it were tangible. I treat the design of physical-digital objects as a new field, and establish a conceptual framework within which to approach this design task. My design strategy treats objects as having both a physical and a digital identity, related to one another by three design principles: *coupling*, *transparency*, and *mapping*. With these principles as a guide, designers can take advantage of emerging digital technologies to create entirely new physical-digital objects. This new design perspective encourages a conceptual shift away from discrete input and output devices as gateways to a digital world, and towards a more seamless interaction with information, enabled by our knowledge and understanding of the physical world. I illustrate this by introducing and discussing seven actual physical-digital object systems, including two which I developed: Bottles and Triangles.

Thesis Advisor:
Prof. Hiroshi Ishii
Director, Tangible Media Group
MIT Media Laboratory

Thesis Committee

Thesis Advisor _____

Hiroshi Ishii
Associate Professor of Media Technology
Tangible Media Group
MIT Media Laboratory

Thesis Reader _____

William J. Mitchell
Professor of Architecture and Media Arts and Sciences
Dean, MIT School of Architecture and Planning

Thesis Reader _____

William Buxton
Professor, Computer Systems Research Institute
University of Toronto

Chief Scientist
Alias | Wavefront Inc.

Chief Scientist
Silicon Graphics Inc.

Acknowledgements

I am grateful to all those whose support, encouragement and guidance helped shape this work. I would especially like to thank:

Contents

1   Introduction

1.0   Bits or Atoms? Both.

This century has seen the development and phenomenal effects of digital information technology. The world has become smaller and faster, and our attention has increasingly shifted from the concrete to the intangible. We work with data. We depend on email, fax, digital media, satellite and cellular communication. We use the Internet for information exchange, entertainment, and expression. We leave traces of our lives in bits; our bank accounts, credit histories, education and demographic information are all digitally encoded, instantly and globally accessible. Although our bodies are firmly rooted in the physical world, our actions extend into an information world without distance or mass, where items can exist in many places at once and bits can be easily, infinitely, and perfectly reproduced.

Yet we are still physical beings, and all around us are the objects which define our lives. As infants we reach for and hold onto whatever we can get our hands on as we begin to make sense of the world around us. As we develop intellectually, the physical world informs our thoughts, language and comprehension. We use metaphors of the physical to describe abstract concepts like emotional states ("I'm *in* a great mood.") or social systems ("You're *moving up* in the world!"). These metaphors extend to digital systems as well.

We imagine information as "contained" in our networks. We *enter* records *into* the system. We think of and speak of data "*in* the computer". We "*get* stuff *off* the web".

Human-Computer Interaction designers have long recognized this metaphorical connection between the physical and digital world, and have made good use of it as the basis of the drag-and-drop desktop and hierarchical file systems of the conventional Graphical User Interface (GUI). In this thesis, I present and discuss work that further integrates our physical and digital worlds. I define the concept of **physical-digital objects** – *physical objects whose form and materials enable people to access and interact with digital information as though it were tangible.*

Research in physical-digital objects is still in a nascent state, and the field is broad and unexplored. The first step in defining an area of design research is to establish a conceptual framework on which to ground our assumptions and build a common perspective. My aim is to develop a new conceptual framework for physical-digital objects that presents them as distinct from computer input devices. To this end, I map out a broad design space by introducing three specific principles: *coupling, transparency,* and *mapping*. These terms will be introduced and fully discussed in chapters three and four.

1.1 Motivation

Digital technology is generally equated with computers. And when most people today are asked to draw a picture of a computer, they sketch out a box representing a monitor and some lines indicating a keyboard, with a mouse alongside it (fig. 1.1) [Bux96, McCol96]. Of course, these input/output transducers are simply the most visible parts of the most visible kind of computer in 1998 – the desktop PC. We don't tend to think about the processor in the box and its associated electronics. Similarly, we don't generally think of the digital technology in our cars, ovens, watches and telephones. For now, our conscious interaction with digital information is usually limited to the things we
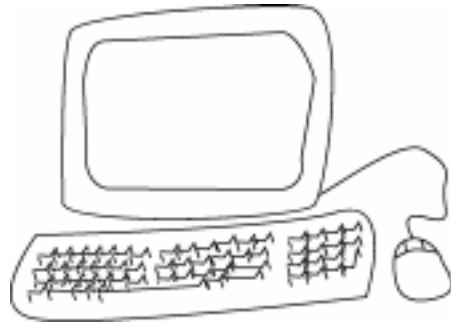
Figure 1.1 (after Buxton, 1996, McCollough, 1996)

Is this a computer? No, but it is the popular conception of one, and of digital technology in general. As such, it limits our imagination with respect to how digital technology can be used.

can do with that most generic of information appliances – the PC. Consequently, our popular conception of what digital technology can be used for is severely constrained by the image of the physical interface elements as "the computer'.

To change this, a new way of thinking is required; one that recognizes the unique properties of digital information without the constraints of our current user interfaces. What new uses of digital systems could we imagine if only we thought of information slightly differently?

Microprocessors consist of components which are far smaller and operate at speeds far faster than we can perceive naturally. The information in our networks is imperceptible to humans, and must be mediated by devices for us to interact with it at a scale which we can comprehend. By drawing on metaphors of the physical, real-world objects can be created which provide access to the digital world. It is a reasonable conjecture that our interaction with information would be greatly enhanced by mapping it as directly as possible to ideas of scale, speed, mass, or other physical phenomena with which we are familiar. This is the basis for my investigation of physical-digital objects.

1.2 Thesis Overview

Chapter Two presents a historical, philosophical and design context to support the development of physical-digital objects. Chapter Three defines the idealized physical-

digital object as well presenting three design principles which inform its design (*coupling, transparency,* and *mapping*). This chapter also introduces and describes seven systems that begin to approach this ideal. I chose these systems as influential explorations that were created in a variety of contexts by researchers, artists and designers and represent a broad range of design strategies for physical-digital objects. Two of these systems, Bottles and Triangles, are work that I undertook as part of my research with the MIT Media Laboratory's Tangible Media Group. In Chapter Four, I establish a cohesive design space for these physical-digital objects by discussing how the three design principles help merge their physical and digital identities. Chapter Four closes with a brief discussion of the interdisciplinary nature of dual-identity object design.   Next, Chapter Five presents several applications of one of my physical-digital object systems: the Triangles. Chapter Six offers a summary and conclusion of the work presented herein, and is followed by three technical appendices. Appendix A discusses some available technologies relevant to physical-digital objects, and Appendix B and C describe the technical implementation of the Bottles and Triangles projects.

2   Background

In this chapter I discuss a body of work in philosophy, design, and the social sciences which supports the concept of physical-digital objects. Philosophically, there is a growing recognition of the importance of the physical world in our understanding of complex abstract concepts. The design background which I present first introduces the concept of *physical affordance* as a means for relating the form of an object to its tasks. Next, I draw a parallel between the evolution of digital technology and the development of plastic. I then discuss the role of *design languages* and *product semantics* in shaping the nature of designed objects. This chapter concludes with an exploration of new applications for digital technology in society and in education, and how these are leading towards physical-digital objects.

2.0   Philosophy

In investigating the role and use of the physical-digital object, it is helpful to ground our thinking in some philosophical writings on the nature of physicality and the relationship between mind and body. This topic has been at the heart of philosophy for centuries, going back to Renée Descartes' *cogito ergo sum.*

2.0.1 The Mind-Body Link

Modern Western thought has been deeply influenced by a traditional philosophical split between the mind (reason) and body (perception). Although present in the philosophical works of Aristotle and Plato, the articulation of this split is commonly attributed to Descartes, whose writings on the nature of thought and reason presented "a world where man became a detached observer of himself"[Svan98]. Descartes suggested an objective, universal truth to be found outside of the realm of personal experience through mathematical controls and scientific rigor. This third-party, objective view of reason has for centuries played a key role in our intellectual pursuits as the basis of the modern scientific method.

Recently, there has been a significant movement in linguistics, cognitive science and philosophy which questions the Cartesian view of reason. Modern American philosophers George Lakoff and Mark Johnson suggest in three separate books that our physical bodies are not only strongly influential to our capacity to find meaning in the world, but that cognitively, the mind and body are inextricably linked. In *The Body in the Mind* [Joh9X], Johnson offers the notion of *image schema*, a set of low-level, physicality-based thought devices which we use to categorize and comprehend the world around us. For example, he speaks of the *in-out* schema, by which we categorize ideas and concepts as though they were physically in a box with other ideas. He describes many other such schema, including the *balance* schema as applied to justice and mathematics and the *force-blocking* schema used in arguments or debates. He shows how physical principles like transitivity and inertia are clearly extended to apply to our intellectual image schema, through metaphor.

Berkeley researcher George Lakoff, a colleague of Johnson's, uses similar arguments in his book *Women, Fire, and Dangerous Things* [Lak87], an

exploration of the relations between categorization, imagination and meaning. Lakoff's principal notion is that we make sense of the world only through classifying things into categories, and that our mechanisms of categorization are firmly linked to the physicality of our bodies, again through image schema. Prior to these two books, Lakoff and Johnson co-authored *Metaphors We Live By* [Lak80], a sweeping examination of the role of metaphor in thought and language which has become a standard text in the field of human-computer interaction research.

2.0.2 Merleau-Ponty and Embodiment

Some of Johnson and Lakoff's work extends previous thoughts by French philosopher Maurice Merleau-Ponty. His best-known work, *The Phenomenology of Perception* [Mer62], breaks with Cartesian dualism by presenting perception itself as an "embodied, active and acquired skill" [Sva98]. Merleau-Ponty presents bodily experiences as essential to the way we perceive the world around us, writing that "The body is our general medium for having a world" [Mer62]. He also describes *embodied* tool-use, whereby we integrated tools into our bodily mechanism as a way to further understand the world.

As we reach for and grasp an object, it makes a transition from its own *object-space*, into our *body-space*, becoming an extension of our body. The degree to which it falls under our complete physical control determines the degree to which it is seamlessly integrated into body-space. For example, if a pen is sitting on a desk, it is in its own object-space, being affected by external forces independent of my body. As I proceed to pick it up it is transferred completely into my body-space. I remove all ties to the constraints of its own space and extend the conditions of my body onto the pen. However, if the pen were tethered to the table by a string, it

could only partially become part of my body-space, as its motion would still be governed by the string, which ties it to table's object-space.

Merleau-Ponty's writings on body-space and object-space have led me to explore the concept of *data-space*, a perceptual space divided from our body-space, within which most of our current interaction with information occurs. Data-space is perceptually separated from body-space by fact that all of our interactions with data must be consciously mediated by input and output devices. Unlike the physical objects which surround us, items in data-space cannot easily cross into our body-space to be manipulated as embodied objects. One goal of physical-digital objects is to allow a more seamless transition between data-space and body-space. I discuss this further in section 4.0.2.

The philosophical research discussed in the preceding subsection points to the common-sense notion that humans are very good at understanding complexity through physical phenomena and by extension, metaphors and illusions of the physical. A well-designed physical-digital object creates a clear mapping of digital properties to physical attributes, making digital information seem as tangible as objects in the physical world.

2.1   Design

As sensors, actuators, memory, processors, and communication technologies become smaller and cheaper, it becomes possible for designers to work with the digital properties of an object in the same way as they consider its material or mechanical properties. New objects can be created from these "digital materials" which capture, transmit, process or react to the ways in which they are physically manipulated.

Digital materials enable objects whose form is left entirely open, free of the constraints of a specific medium to contain the information. For example, by putting a tiny unique digital ID chip into an object, it can be used to access on-line music. The songs no longer need to be packaged on tape or CD or even diskette — the data is stored on the net, so the object and the device used to retrieve it can take whatever form we like. This leaves us free to think about how one might act to retrieve or manipulate information from an object. We can create a new artifact whose form embodies the unique properties of digital information. The object's form might represent the content, or be designed to facilitate a particular interaction, but no longer needs to be wed to the specifics of a given transport medium. For example, Durrell Bishop's *Marble Answering Machine* (see section 3.1.4) uses small marbles to represent individual phone messages, giving digital content a physical form that can be easily handled.

2.1.1   Affordances

One key concept in the design of physical objects has to do with the functionality made possible and communicated by their form. Borrowing from and extending J.J. Gibson's term, Donald Norman defines the means by which an object expresses its potential use as its *physical affordances*. For example, a glass *affords* holding liquid, a chair *affords* sitting, and depending on its design, a door handle *affords* pushing or pulling. These physical affordances of objects are easily discovered by simple examination or experimentation with the object. It is this "legibility" that enables people to invent new uses for objects in a pinch – using a book to prop up a shaky table or a wine bottle as a candle holder. With current digital systems, such creative adaptation is seldom possible. The functionality of most digital systems follows specific protocols and is only made available to the user through a very strict set of symbolic controls, rather than being expressed as affordances.

Physical-digital objects can be created which respond to certain manipulations, enabling designers to metaphorically express their *digital* capabilities through their physical design. For instance, the Triangles project uses a tiling shape and magnetic connectors, which *afford* their being connected to one another, pulled apart, and easily rearranged. This is a metaphorical expression of the ability of a digital dataset to be easily reorganized (see S. 3.1.2 for more details on the Triangles system). The physical design of the Triangles and their connectors makes it possible to discover the digital capabilities of the Triangles system through experimentation.

2.1.2   The Plastics Analogy

Digital materials offer opportunities and challenges similar to those created by the introduction of plastics to industrial design. Like plastic, digital information is entirely a creation of human technological advances. It is infinitely malleable, has no natural form, yet it has been pervasively integrated into our daily lives to the extent that most of us depend on it daily. And, like plastic, digital information systems have had an ambiguous cultural identity, alternately seen as heralding a new utopic era of unbridled possibility or poisoning our culture and cheapening humanity. The similarity between the technologies of plastic and digital information systems suggests that we might learn from examining their parallel histories.

When the first celluloid plastics were introduced in the late 1800s, they were used as imitations for ivory, tortoiseshell, and other natural

Figure 2.1

Celluloid, one of the first plastics, was used primarily to imitate existing natural materials like ivory and tortoiseshell, as in these items of apparel, c. 1890

materials (fig. 2.1). In his engaging book *American Plastic*, cultural historian Jeffrey L. Meickle describes celluloid's transition from imitation to innovation[Mei97]: "Imitation and substitution dominated most nineteenth-century discussions of celluloid. Later, as celluloid and newer plastics gained wider use in the early decades of the twentieth century, the balance gradually shifted from imitation to innovation, and promoters began to celebrate the frankly artificial." Initially, innovative items like the polyethylene bottle or the formica countertop not only substituted for their natural glass or wood counterparts, but took advantage of the properties of plastic to augment these items, making them unbreakable and stain-resistant. In the 1950s the cultural acceptance and technical understanding of plastic had reached the point where its properties could drive the design of entirely new products. Gradually designers recognized and were able to take advantage of plastic's full potential as a design material in its own right. Furniture, equipment, packaging, clothing, and sporting goods that are now commonplace would have been inconceivable without this conceptual shift from imitation to innovation and finally, to invention (fig. 2.2).

A parallel can be drawn to the development of digital computer applications. Early digital computers were used in commerce and defense and perceived mainly as highly efficient replacements for manual calculators. As technology advanced, it became possible for bits to be combined into letters that could be edited and re-edited with a keyboard, providing a substitute for existing typewriters.   This substitution enabled many advances, such as the ability to save and reprint documents, or correct errors before they

Figure 2.2

These medical containers are flexible, disposable, and transparent. Recognizing the unique properties of plastic has led to new inventions that could not have been possible with other materials.

were printed. Just as polyethylene bottles provided advantages over glass ones, documents constructed from bits have advantages over paper documents. Gradually, thinking about bits shifted from imitation and innovation towards inventing new uses for the unique properties of digital information. In 1995, Nicholas Negroponte proclaimed: "Better and more efficient delivery of what already exists is what most media executives think and talk about in the context of being digital. But like the Trojan horse, the consequence of this gift will be surprising. Wholly new content will emerge from being digital…"[Neg95]

The plastics analogy points to an important cycle of imitation, innovation, and finally invention that comes from taking full advantage of the properties of a new material. In the case of plastics design and manufacturing, the material is plastic and the properties are physical: flexibility, luster, translucency, color, and strength, to name a few. The properties of bit-based information are conceptual: the ability to be infinitely reproduced, remapped into the physical world as sound, motion or images, or flawlessly transmitted over great distances. Understanding these properties has enabled us to invent new applications for digital information.

We are currently experiencing the first of these new digital inventions – virtual chat spaces, nonlinear narratives, shared remote workspaces, 3D computer animation and home banking, to name a few – applications that could not have been possible in the pre-digital era. We can now dream up and create applications to take full advantage of the unique nature of bits. However many of the stumbling blocks for users of these applications can be traced to the document-centric Graphical User Interface (GUI) of personal computers. This ubiquitous keyboard-mouse-monitor interface has us stuck in the realm of imitation, pretending that we are still experiencing "documents". What should be new inventions – our digital homesteads, person-to-person communication, and realtime content, for example –

are cast instead as mere innovations to the "pages" of "documents" for lack of a better interface. And increasingly, designers of these new applications are struggling to overcome the constraints of our existing physical interfaces. How do you navigate a virtual 3D space with a mouse and keyboard? What about generating and mixing complex synthesized audio tracks? Or creating lifelike, fluid computer animation? Digital technology mediated by personal computers allows us to invent these new things. But our current physical interfaces to digital information, stuck in the imitative mode, restrict us.

2.1.3 How "Documents" have Driven the Design of the Digital

Computer interfaces based on windows, icons, menus and a pointer (WIMP) have evolved only slightly in the past 20 years. The Star Workstation, originally developed at Xerox Palo Alto Research Center (PARC), was the predecessor to the Apple Macintosh and is widely considered to have originated the Graphical User Interface (GUI) as we are now familiar with it (Fig. X). The designers of the Star thought of it as an office document processing machine, and chose metaphors and design elements to suit that purpose:

> "The document is the heart of the world and unifies it… [The Xerox Star] assumes that the primary use of the system is to create and maintain documents. The document editor is thus the primary application. All other applications exist mainly to provide or manipulate information whose ultimate destination is the document" [Joh89]

Since the Star, certain conventions have changed slightly, but the general document-oriented nature of personal computation has become ingrained in our

conception of digital technology. It is important to remember that the design language and physical devices that make up current interfaces were consciously chosen in the context of business document processing, a task to which they are quite well-suited. It is my hope that as we move further towards new uses of digital technology, we will explore entirely new design languages. Physical-digital objects are one direction to take in this exploration.

2.1.4　　　　Design Languages

In their essay *Design Languages*, John Rheinfrank and Shelley Evanston describe the *design language* of an object as

"…the means by which

- Designers build meaning into objects, so that objects express themselves and their meanings to people.

- People learn to understand and use objects.

- Objects become assimilated into people's experiences and activities."

They describe a process for developing a language for the design for a particular product or product family. Their five-stage prescription includes a step that they call *re-registration*, in which, "a new assumption set and design framework" are consciously created by designers:

"Creating a new set of assumptions fosters creativity by allowing a design team to move away from designing according to preframed and preanalyzed sets of assumptions, and by encouraging members to move toward designing according to the patterns that they construct collaboratively from current contexts of use." (Winograd1996, p. 76)

Rheinfrank and Evanston suggest that such a re-registration process, applied to personal computation, might result in new physical artifacts for interacting with digital technology:

> "…the first cellular telephones were *wireless telephones*, in much the same way as we referred to the first automobiles as *horseless carriages*... The potential of cellular communication almost certainly has as little to do with today's telephone as today's automobile has to do with buggies and buggy whips. We will one day look back at today's personal computers (TV sets with typewriter keyboards) and software as historic curiosities of the same ilk." (p.74)

2.1.5 Product Semantics

In the 1980s, the application of new *design languages* was the focus of an influential design practice at the Cranbrook Academy of Art in Michigan. Under the joint direction of Katherine and Micheal McCoy, Cranbrook students methodically explored *product semantics*, designing the form of objects to literally illustrate their meaning and use. Concentrating primarily on household appliances and electronic devices, whose featureless boxes had previously made them nearly indistinguishable from one another (e.g. Bang & Olufsen hi-fi designs), the

Figure 2.3

The design of Lisa Krohn's *Phonebook* telephone/answering machine clearly displays its semantic content and its context of use. The various modes of operation of the device are activated by turning its 'pages' to reveal specific functionality.

McCoys' students consciously developed products whose meaning could be "read" in their form. For example, Lisa Krohn's 1987 *Phonebook* telephone answering machine, designed to look like an address book, has plastic "pages" that can be turned to select its mode of operation (fig 2.3). Similarly, Van Hong Tsai's 1986 *Toaster* is designed to look like two pieces of bread, molded with ridges illustrating the "friendly waves of heat" which transform the bread into toast (fig 2.4). The lever used in operating the toaster is shaped like a conical arrow pointing down, clearly indicating its use.
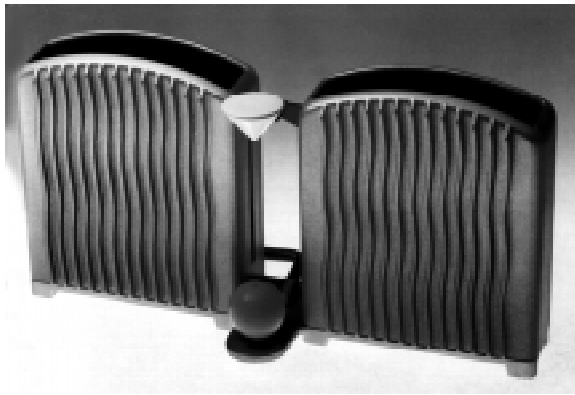
Figure 2.4

Van Hong Tsai's *Toaster* describes its purpose through 'heat-waves' molded into the bread-shaped segments.

In his essay *The Mannerists of Micro-Electronics*, Hugh Aldersey-Williams states that, "[product semantics] aims to extend human factors from the physical and psychophysical into the cultural, psychological, and social domains."[Ald90] The Cranbrook students find intellectual support in Merleau-Ponty's phenomenology, focusing on the fusion of the object and its concrete meaning as perceived by its user, and in the process divorcing its form from the often complex minutia of its inner workings.

At its simplest extreme, *product semantics* turned objects into cartoonish props for everyday life, adding illustrated meaning to the design of everything from microwaves in the form of lunchboxes to computers that looked like library book

stacks. However, these explicit visual puns soon gave way to a more elegant, mature integration of product   of objects. Examples like Peter Stathis' 199X *Bedside Television* (fig. 2.5) show this transition. The device is designed to lie curled up at the bedside like a cat, awakening when stroked. Its form clearly suggests a domesticated appliance, but without the overtly literal references of the earlier Cranbrook work.

In many ways, initial explorations in the design of physical-digital object interaction parallel the exaggerated, overstated metaphors seen in designs like Krohn's *Phonebook*. David Small's *LEGO Helicopter* [Sma96], for example, presents the user with a tiny model of a helicopter which can be used to "fly" through a digitally represented 3D world (see section 3.1.7). Brygg Ullmer's *Tangible Geospace* system [Ull97] uses small plastic figurines representing buildings that allow interaction with a digital map (see section 3.1.6). Although these systems at first present easy-to-understand alternatives to traditional information interfaces, they seem too literal to be practical. Their rigid forms limit their scalability and their novelty soon wears off. However, just as the early

product semantics work at Cranbrook awakened an awareness of the human, immediate meaning of objects, these early physical-digital objects suggest the potential benefits of integrating high-level digital meaning into physical objects.

Figure 2.5

Peter Stathis' *Bedside Television* maintains an elegantly abstract form while subtly echoing Product Semantics in its legibility as a 'domesticated' appliance.

2.2   Culture

Developing and designing new physical-digital objects will also suggest new ways to use their digital capabilities. My hope is that by giving people a new way of thinking about digital content – as tangible – we will find new ways of creating, using, and manipulating that content. New types of content and new applications will arise that expand on and go beyond what can currently be done with a computer. It is likely that these applications will be in domains that did not play a driving role in the development of the GUI interface – artistic expression, games and social dynamics, for example. Especially exciting are application domains like education, toys and storytelling, which can help make digital technology an integrated part of growing up. These are fields in which computers have shown great potential, but which have traditionally been poorly dealt with by the GUI.

As these new types of applications develop, it will always be possible to use the GUI to model and represent them, of course, but interaction mediated by a mouse will suffer compared to interaction with the actual physical objects. Leafing through a book or walking through a building remains a much richer interaction than browsing pages or navigating 3D worlds with a mouse and monitor. This is discussed further in section 4.0.2.

2.2.1   New Ways to Play and Learn

Evidence of these new application domains is already beginning to emerge. Microsoft, Mattel and Motorola are creating products which focus on the social uses of computers – from pagers used by teenagers at school to Mattel's *Talk With Me Barbie*, information technology is beginning to move out of the proverbial beige box and into our lives. For years, the use of digital technology in education has involved children "playing on the computer" or "learning the computer".

Digital devices like Microsoft's *ActiMates* toys, which communicate with computers via infra-red and respond to being touched or spoken to, provide children with an early introduction to digital technology that centers more on the applications than on the computer.

At the MIT Media Lab, Mitchell Resnick's Epistemology and Learning research group has introduced many new physical interfaces and computational devices into learning environments. His *Behavior Construction Kits*[Res93], oversized LEGO blocks which can be programmed using LOGO and incorporate a variety of sensors and actuators, have enabled children to explore logic and programming concepts in the physical world. More recently, Resnick introduced the term *digital manipulatives* [Res98]. The educational term *manipulatives* is used to describe a variety of physical teaching and learning aids used in the classroom, such as Cuisenaire Rods and Pattern Blocks. Resnick's digital versions are balls, beads, and blocks which can be programmed and manipulated by children. They can be used to create autonomous creatures, complex chaotic interactions, or rule-based games.

The digital manipulatives are designed to bring ideas of construction and *constructionist* learning[Pap80] into the digital realm, allowing students to explore rich and complex concepts like feedback and emergence in digital systems. Digital manipulatives offer children new interfaces to digital information, but the effect of their novel use of technology goes much farther than this: "Our primary goal is not to help users accomplish some task faster or more effectively, but rather to engage them in new ways of thinking."[Res98]

In this chapter, I have provided overviews of pertinent philosophical, design and cultural elements which pave the road towards physical-digital objects. These fields inform and support my exploration of the design space of the physical-digital object as well as its role in future interaction with digital technology.

3   Physical-Digital Objects

This chapter provides a concrete definition of physical-digital objects as well as design principles for examining them. It then introduces and describes seven existing projects which begin to approach the design qualities of the idealized physical-digital object.

3.0   Definition

> *physical-digital object:* **a physical object whose form and materials are designed to enable people to directly access and manipulate digital information as though it were tangible.**

This definition focuses on the end-user's perspective. It articulates, as a fundamental quality of physical-digital objects, that they enable users to act as though they are indeed physically manipulating information. The definition is purposefully silent on questions of technology, system architecture, and application. It makes no mention of objects as controls or containers, or of the degree to which these objects are representational. I will examine these issues in depth in the next chapter. Instead, the definition suggests a *dual-identity* object,

with a physical and digital identity so tightly coupled that interaction with the physical implies interaction with the digital.

Having defined the boundaries of the arena so loosely, it is now necessary to clarify the ambiguous cases of objects that lie along its borders. It has been pointed out to me, for instance, that every time we use a plastic credit card we manipulate our bank balance, that floppy disks allow us to physically transport information, and what are pagers, if not objects that allow us to exchange and manipulate digital messages?

Credit cards, floppy disks, pagers and many other media allow us to access and modify digital data only through a secondary interface – a card reader, keyboard, or keypad. These traditional interfaces require an intellectual buffer between the user and the data, as the user must translate a particular manipulation of the data into commands to be carried out by the system.   Typing on the keypad of a bank machine is essentially telling the machine (or more realistically, *asking* the machine) to perform a certain function using your money, using an often arbitrary language imposed by the machine. Conversely, the exchange of actual cash is a means for metaphorically transferring wealth from one person to another. Using cash requires much less of an intellectual work-out than the use of a credit card or even a cheque. Cash transactions are even more second nature in countries where currency designers have used metaphors of physicality, making more valuable bills larger or varying colors appropriately.

3.0.1   Design Principles

The issues which separate physical-digital objects from other devices can be examined through the lens of the following three closely related principles. Taken together, they define the design space of physical-digital objects:

1. *Coupling*– the interaction mechanism directly couples specific physical action with digital functionality. This results in a clear expression of the results of a user's actions, and is achieved through a perceptual convergence of input and output. Coupling places the interaction clearly within a user's body-space, as opposed to simply providing a "remote control" for interactions within data-space.

2. *Transparency* – the interface is made transparent by using physical affordances and physical interaction metaphors to suggest appropriate ways to use the object's digital capabilities. The need for conscious interpretation of arbitrary lexical commands or symbols is reduced through a design language which appeals to a user's past experiences with the physical. This also allows the digital properties of the object to be used in new and creative ways.

3. *Mapping*– the design of physical-digital objects suggests the context and scope of the digital information that they provide access to. Bits can be chunked into data items at a variety of scales, with a wide range of meaning. The interaction with these items might be the control of parameters of an individual item, navigation or identification in a group of items, or exploration of the relationships between data items. The form that the physical-digital object takes can also suggest the semantic content and the scope of interaction with these data items. The physical form of an object can also be used to metaphorically describe its semantic content or digital meaning.

3.1   Actual Systems

Up to this point, I have been discussing a somewhat idealized notion of physical-digital objects as distinct from existing computer interfaces, in order to offer a new way of thinking about digital information – a conceptual starting point for design. In truth, the design of these objects is strongly tied to factors of available technology, application domain, target users and existing interfaces. As such, there exists a continuum between the current design space of human-computer interface and that of the physical-digital object. At one end lie the motivations and issues faced by interface designers today, and at the other extreme is the physical-digital

object in its purest form. Every physical-digital object system lies somewhere on this spectrum.

This section introduces seven physical-digital object systems created in various contexts, beginning with two that I designed. The systems were each developed in their own context, by artists, designers, and researchers exploring a variety of interests. There are many ways in which they could be grouped or defined, and indeed most of them have previously been published in design journals, conference proceedings, and technical reports. My aim in bringing them together here is to suggest that despite their varied origins they share the common seed of the "pure" physical-digital object.

3.1.1   Bottles

Figure 3.1 The Bottles project uses small glass bottles with corks.

In the Fall of 1996, the Tangible Media Group embarked on a project exploring *ambient media*[Ish97, Wis98a], a means for augmenting an architectural space (the *ambientROOM*) so as to provide peripheral, background representations of information. One issue that we faced was user control over such an environment. How could a user seamlessly activate or deactivate the display of audio media in their periphery? The Bottles system was designed to provide a simple technique for allowing a user to select which content is present in the ambientROOM. A real glass bottle with a cork serves as the physical container for an invisible source of digital information (fig. 3.1). In our demonstration, this information is the status of traffic on the computer network in the Media Laboratory. As the bottle's cork is removed, sound of cars and trucks can be heard in the ambientROOM, metaphorically mapped to the activity of network traffic.

The bottle can be left uncorked, leaving the information audible. When it is closed, the sound ceases as the information source is once again "contained".

3.1.2 Triangles

The Triangles project was a joint exploration by myself and PhD. candidate Maggie Orth, of the *Opera of the Future Group* at the MIT Media Lab [Gor98]. The system consists of a set of identical flat, plastic equilateral triangles, each with a microprocessor inside and a unique digital ID. The Triangles each have different images or markings on them, which can be changed depending on the intended application. They have magnetic connectors on their edges which allow easy physical interconnection, as shown in figure 3.2. The connectors also pass electricity, and the Triangles use them to communicate digital information to each other and to a desktop computer. Thus, when the pieces contact one another, specific information about that connection is sent back to the computer, which keeps track of the history of connections and current topography of the system.
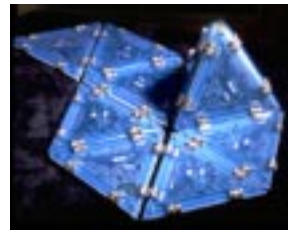
Figure 3.2 The triangular tiles connect both physically and digitally, forming 2D patterns and 3D forms.

The Triangles can be used to make two- and three-dimensional objects whose exact configuration is known to the computer. Changes in this configuration can trigger specific computer-controlled events. For example, connecting or disconnecting two specific Triangles could provide access to a specific web page, or cause a digitized audio sample to play. Events can also be associated with specific groupings of Triangles, rather than simple connection or disconnection of individual Triangles. The actual output event that results from a given interaction depends on the application being used, but can be practically anything that a computer can control.

3.1.3 Bricks

Bricks, an exploration by Bill Buxton et al. at the University of Toronto's Input Research Group, sought to allow two-handed, *graspable* interaction with information [Fitz95]. The Bricks system consists of a flat, desk-like projection surface on which users manipulate objects resembling LEGO® blocks. These Bricks have 6 DOF electromagnetic sensors in them, allowing the system to know their position and orientation at all times. Putting the *bricks* onto the surface binds them to the virtual element upon which they are placed. For example, a user can move windows around on a virtual desktop by placing the brick on the window and sliding it. Lifting the Brick off of the surface of the desk releases the window, allowing the brick to be moved freely thereafter. The potential of Bricks was highlighted in experiments using a virtual drawing toolkit (fig. 3.3). Using two Bricks and a menu system resembling a muffin tray (into which the bricks were dipped to change their operating mode), a user could easily create, position and modify graphical elements as though they were physical.

Figure 3.3 The *Bricks* system being used with a drawing application.

3.1.4 Marble Answering Machine

Durell Bishop's *Marble Answering Machine* is an exploration that the artist undertook as part of a product design program at the Royal College of Art in London, England [Cra95]. The answering system consists of a stylized flat device with a bulging corner which contains many marbles. Whenever a caller leaves a message, the machine releases one of its marbles, which rolls down a gentle slope into a tray at the front of the device. The marbles are each coded with a resistor-based unique ID, which the system associates with the (digitally recorded) message

data. The presence of marbles in the tray indicates that messages have been left, and the user can listen to these

Figure 3.4 Messages can be played back by placing their marble on a small indentation on the machine.

messages by simply picking up a marble and placing it in a small indentation on the machine (fig. 3.4). "Saving" the message simply involves keeping the marble. "Erasing" the message is done by putting the marble back into the machine for re-use. The overall effect is that the marbles represent, or seem to contain, the messages.

3.1.5   Voice Box

Media artist Natalie Jemijenko's *Voice Box* [Jer96] is a small, cube-shaped aluminum box with rounded corners and a miniature speaker on its front (fig. 3.5). The top of the box is a lid which

Figure 3.5 The *Voice Box* can be opened to record, and plays back when handled.

can be pulled open, and is hinged so that it snaps closed when released. Inside the box are electronic components which enable the box to record several seconds of digital audio. By opening the box and speaking into it, the user essentially traps their voice within the box. Picking up the box or otherwise shaking it causes the contents to be replayed. The artist has exhibited the Voice Boxes in sets of dozens or more, allowing them to be used as a system with many elements. For example, children were observed recording phrases into the Voice Boxes and then stacking them in a specific order to tell a story. Gallery visitors would also use the Voice Boxes to leave messages for friends or to make announcements to future visitors.

3.1.6   Tangible Geospace

Tangible Geospace is an
application of the Tangible
Media Group's *metaDesk*
system, a flat desk-like
projection surface with an
assortment of video, contact,

Figure 3.6 By placing and
moving objects in the
shape of buildings, a user
can scale, rotate, and
translate the digital map
information.

and radio-frequency sensing technologies embedded within it [Ull97]. To interact
with the Tangible Geospace application, users place small acrylic objects in the
shape of buildings (called physical icons or *phicons*) onto the surface of the
metaDesk (fig. 3.6). The building objects are recognized by the system, and digital
map data is projected onto the desk, positioned so that the buildings are aligned
with the map. The phicons can then be used to manipulate the map image,
translating, rotating, or scaling it by simply dragging the phicons relative to each
other on the surface of the desk. The digital image remains aligned.

3.1.7   LEGO Helicopter

The *LEGO Helicopter* was a small part of a larger project by David Small at the
MIT Media Lab's Visible Language Workshop [Sma96]. His work with LEGO
bricks and virtual three-dimensional *information spaces* led him to develop a
LEGO-based computer-aided design (CAD) system. He constructed a physical

LEGO world and, using his CAD
system, created a virtual
counterpart which is rendered in
3D on a computer monitor. The
on-screen world is embellished

Figure 3.7
'Flying' the
LEGO Helicopter
in the real world
allows easy
exploration of the
on-screen 3D
LEGO world.

with extra digital information such as animated objects and text. Small then built a LEGO helicopter which seamlessly integrates the real and virtual spaces (fig. 3.7). The helicopter's position and orientation are tracked by the system and coupled to the viewpoint of the virtual camera. Users can "look around" in the virtual world by simply flying the helicopter around the physical LEGO model. The helicopter seems to have a tiny camera inside it, an effect which is so convincing that first-time users sometimes ask to see the camera.

In this chapter, I have offered a broad definition of the physical-digital object from the user's point of view. I also introduced three design principles by which physical-digital objects can be analyzed, and described seven projects which resonate with my definition. There are, of course, many other good examples of physical-digital object explorations, including the Chameleon [Fitz93], Flip-Brick, LEGOwall [Fitz96], and physical programming systems like AlgoBlock [Suz93] and Maeda's Solid Programming [Mae93] systems. I will reserve the investigation of these projects as physical-digital objects for future work in this field. In the next chapter I will elaborate upon the three design principles, *coupling, transparency,* and *mapping*, drawing on the seven projects introduced herein when needed as examples.

4　A Conceptual Framework

In this chapter I present a conceptual framework for the design of physical-digital objects. This framework begins with the notion that the objects have two identities from their conception—that is, they are *both* physical objects *and* digital objects. To create objects that clearly bring these identities together into a single, coherent object, I propose three guiding design principles. These three principles are *coupling, transparency,* and *mapping.*

The three principles are very closely related, and each informs the other two. For example, a seamlessly *coupled* object will provide a more *transparent* interaction, allowing more complete and appropriate *mapping* to a given application's context and content. The three principles can be closely tied to three aspects of an object's design: technical/physical design, interaction design, and application design. Figure 4.1 illustrates this relationship.
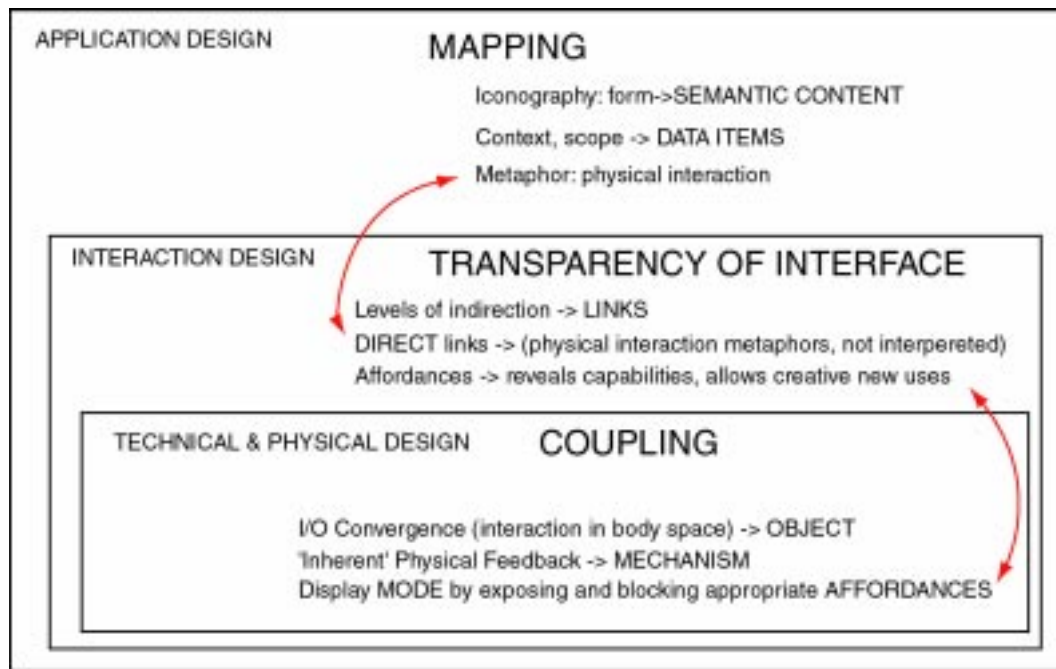
Figure 4.1. The three principles that guide the design of physical-digital objects, as related to three aspects of their design. Each layer depends on and informs the others.

Within each of the three nested layers, design decisions need to be made, and design direction will depend on the specific needs of the project—its emphasis, context, and constraints. The conceptual framework I present seeks to provide a good understanding of coupling, transparency, and mapping, so that appropriate design decisions can be made. The following sections describe these principles in detail.

4.0 Coupling

I use the term *coupling* to refer to specific physical actions which are directly linked to digital functionality and vice-versa. In the physical world, noticing that a door is closed or that a pen is covered also usually reveals the mechanism for opening it (changing modes), through its specific physical affordances. When a

user perceives a physical mechanism as both an indicator of state and a means for changing that state, concepts of input and output are meaningless. Such a convergence of input and output into simple observation and action places the interaction clearly within a user's body-space.

4.0.1　　　　Linking Physical State and Digital Mode

Many digital systems and some electronic devices have several mutually exclusive *modes* of operation, such as the various applications that can be run under a given GUI-based operating system or the VCR/CAMERA modes of a camcorder. *Mode errors* occur when the user thinks that a certain function is available but it is not. This is a potential source of confusion and frustration, forcing the user to think about the operation of the device, rather than the particulars of their task. To illustrate, let us consider the design of the Kodak DC120 digital camera. When this camera is on, it has two mutually exclusive modes of operation: a *photo mode* for taking pictures and a *playback mode* for reviewing them. To switch between modes, there is a momentary slide-switch marked "play/stop". The current mode is indicated by an icon on a separate LCD panel (fig. 4.2). There are two things about this design which contribute to mode errors:

Figure 4.2

The Kodak DC120 features a mode switch does not provide any indication of mode or alter the physical affordances of the camera in such a way as to suggest which mode it is in. This is inconsistent with its power switch, which doubles as a lens-cover.

1. Action and feedback are decoupled. The switch snaps back to a neutral position after use, so there is no inherent feedback of the device's mode. Instead, an icon appears at a different location. This means that the user must know what the icon means, and where to look for it, to determine the mode of operation. Conversely, the act of determining the current mode offers no insight as to how to change modes. If the switch were simply a toggle, it would clearly present its mode as well as providing an obvious means for changing modes.

2. All of the camera's physical functions *appear* to be available in both modes. The switch and icon change the digital properties of the camera, but its physical affordances remain constant – even in playback mode, it can be aimed and its trigger can be depressed, but no photo will result. If the mode-changing mechanism was actually a cover which toggled between the lens and the playback screen, this would remove or expose appropriate physical elements, providing a clear indication that picture-taking functionality is unavailable in playback mode, and vice-versa.

To make matters worse, the design of this particular camera *does* couple a physical lens-cover with the on/off switch, inconsistently suggesting that picture-taking is possible whenever the lens is exposed.

In contrast, the modes of operation of most purely physical objects are easily selected and intrinsically indicated through their physical *state*. As a simplistic example, a pencil with an eraser on one end cannot be used for writing and erasing simultaneously. The means for achieving a certain mode of use (turning the pencil one way or the other) is directly coupled to the means for determining what mode the pencil is in (examining its orientation). Thus, the pencil provides consistent and

unambiguous feedback of its mode of operation, and only exposes functionality appropriate for a given mode in each state.

Fig. 4.3 The StarTac's folding design acts as a switch and mode indicator, exposing appropriate physical affordances and reducing the potential for mode errors.

Some digital devices do couple physical and digital mode-switching. For example, the Motorola StarTac™ cellular telephones must be *unfolded* to be used (fig. 4.3). This exposes the keypad, speaker and microphone for use, simultaneously enabling them. To end a call, the phone can simply be closed. However, the Star-Tac has another mode which adds to its complexity. There is no clear physical representation for the distinction between the device's *standby mode*, in which it is ready to receive calls, and its *off mode*. To indicate this mode, its designers chose to use a blinking green LED on its exterior.

Almost all of the physical-digital objects described in s. 3.1 closely couple physical interactions with digital mode changes, so as to provide inherent and unambiguous feedback of state. For example, the Bottles' two modes can be thought of as *playing* and *stopped*. Opening a Bottle puts it into playing mode, which is clearly indicated by the open bottle. The physical states of the Bottles are tightly coupled to their digital modes. Of course, it is much more difficult to create clear and

meaningful couplings for a multi-purpose or general digital system than for special-function physical-digital objects. Still, complexity can be achieved in physical-digital objects like Triangles, which nonetheless provide a clear indication of their digital state through their physical relationships and reinforce particular operations with magnetic polarity and geometric shape.

4.0.2 Interaction in Body-Space

The coupling of physical to digital implies a convergence of the perceptual spaces of input and output. Most existing computation systems present the user with physically distinct input mechanisms and output devices. With a Graphical User Interface (GUI), for example, the keyboard and mouse are physically and perceptually distinct from the monitor. This separation has partly historical roots. Interaction with early computer systems was necessarily asynchronous and command-driven.   Bit-switches or punch-cards were used to first *put* data and commands *into* the system, which would process and only then *output* results. Command-line interaction languages also follow this serialized *input-> process-> output* interaction. In contrast, systems which use windows, icons, menus, and
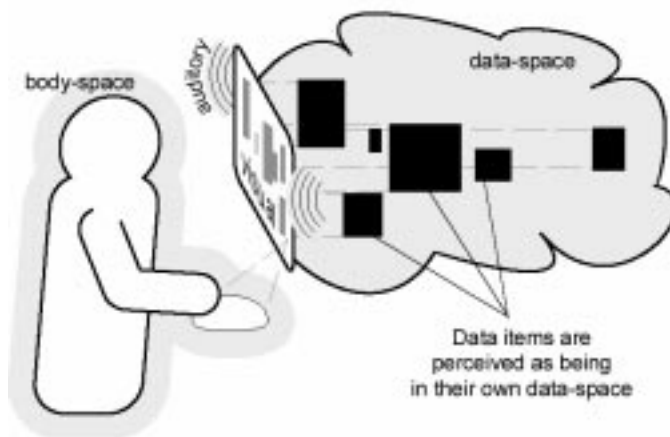
Figure 4.4: Remotely Controlling Data Items from Body-Space

Working with a GUI, the user's perception is that operations take place inside or behind the screen. The mouse provides an extension of the user's hand or finger into data-space. Thus, operating with the mouse is perceived as a kind of remote control.

pointers (WIMP) in their interaction tightly couple manual input (via the mouse) with continuously refreshing visual feedback to enable what has been called direct manipulation[Sch92] of on-screen items. But the separation of input and output hardware persists, creating a visually dominated *data-space* akin to Merleau-Ponty's object-space (fig. 4.4).

As an input device, the mouse acts as a perceptually transparent tool for extending our physical touch into the graphical data-space, but our interaction is constrained to that realm. The mouse is a remote control device for controlling items in data-space. This is in contrast to most of our real-world interactions, in which there is no split between the input and output, and seamless transitions can occur between our body-space and object-space.
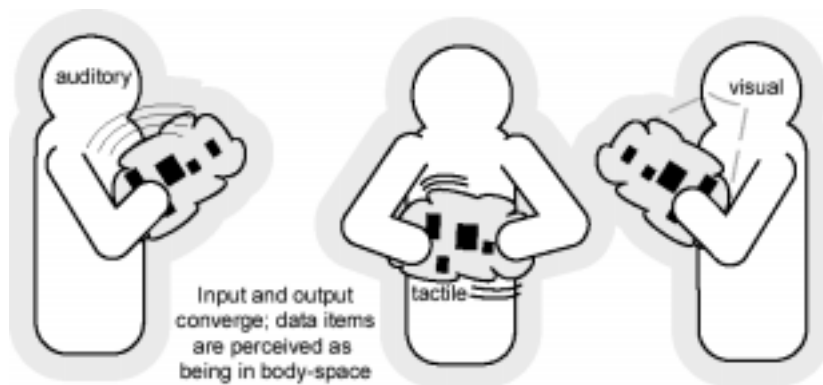
Figure 4.5: Using Data Items in Body-Space

Ideal physical-digital objects would seamlessly join input and output, shifting interaction into body-space.

Physical-digital objects, on the other hand, map data-space back into their object-space, enabling the user to grasp and manipulate data through them. Users can bring physical-digital objects into their body-space, experiencing a more directly coupled interaction with data (fig. 4.5). The specific nature of the interaction will depend highly on the mapping of the physical-digital object and its application (see S. 4.2.)

It is important to note that even as input device technology changes and improves with new devices like 2-handed mice, "data-gloves" or 3D pointers, as long as input is *perceptually* separated from output, the split between body-space and data-space will still be reinforced. The actual output and input devices need not necessarily be colocated or identical to avoid this split, but the user must perceive them as being so. For example, if a user connects two Triangles tiles together and a sound is triggered from a nearby speaker, the perceptual space is made up of the visual and tactile interaction with the Triangles (colocated input and output) as well as the audio feedback. Since none of these modalities are mutually exclusive, they are not perceived as separate and the interaction remains perceptually in body-space.

The use of "direct" couplings between the physical and digital properties of an object also opens up possibilities for creativity. It suggests the ability to create digital systems whose functionality is available to be built upon and reused in unexpected ways by end users. Stuart Brand describes how architectural spaces can be designed to grow and change over many years of use, adapting to the unforeseen future needs of their inhabitants [Bra94]. Similarly, people often use the physical affordances of a bottle (designed for drinking) as a vase for holding flowers, or pots and pans as musical instruments. The ability to adapt an object can be key to subsequent innovation. In their compelling study on breakthrough products and services, the Doblin Group identified one major feature of a breakthrough as having "surprising" uses:

> "[such products] generate unforeseen innovations by end users, some of whom are more imaginative about new applications than development engineers."[Dob96]

Physical-digital objects which use close coupling and a *transparent* interface (see section 4.1), will potentially enable surprising and creative new uses of digital technology.

4.1 Transparency

I use the term *transparency* to refer to the degree to which the interaction occurs without the need for conscious thought and interpretation. This is the property referred to by Merleau-Ponty as *embodiment*[Mer62], by Svanæs as *pre-conscious interaction*[Sva98], and by Heidegger as *transparency of equipment* [Win86]. The physical affordances of an object can suggest appropriate actions without requiring lexical or symbolic cues for interaction. Using a design language which directly maps a user's understanding of physics and physical systems to digital capabilities enables interactions which require fewer foreground, intellectually interpreted actions and consequently render the interface more transparent. For example, a mouse or joystick act as a physical extension of the user's hand and in normal use, they require very little foreground intellectual effort to operate.

The *Transparency* of an interface depends heavily on good *coupling*, as described in the previous section. Direct and consistent coupling of a physical interaction and its digital result is very important to the transparency of the physical-digital object interface, because it reduces *breakdown conditions* and *mode errors* which would force the interaction mechanism to the foreground[Win86].

4.1.1 Degrees of Transparency

Figure 4.6 illustrates transparency of interface in a trivial interaction. If a user is given a black square (presumably representative of some digital element), and asked to cause a change in this square's position, the figure presents various
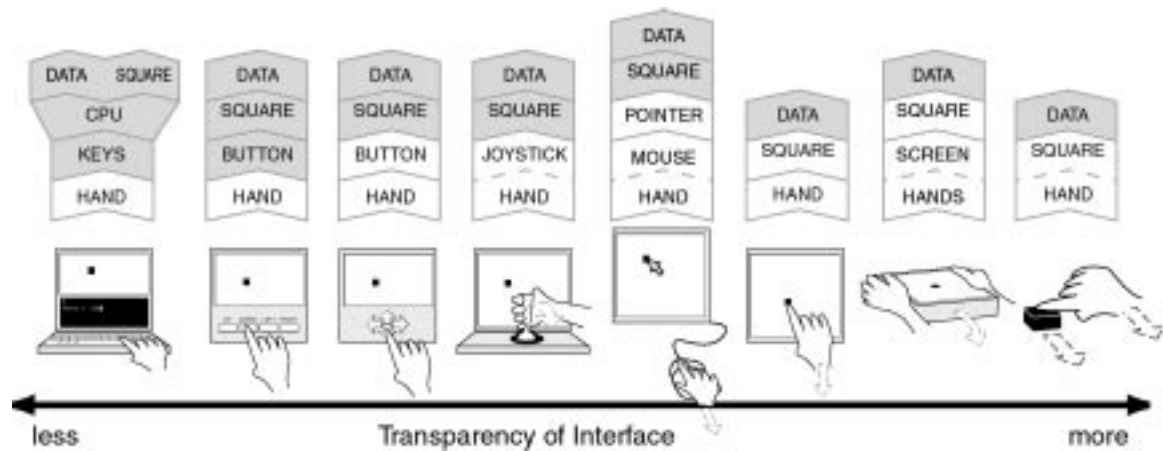
Figure 4.6 Transparency of various interfaces for a trivial task of moving a black square, representing some data. Chevrons indicate levels of linkage, where white is a direct link based on a physical metaphor, and grey is a link which depends on an intellectual interpretation. Dotted lines indicate a link which is a physical extension between the user's body and the physical interface.

common (and some less common) possibilities for interaction with this square. It illustrates the intellectually interpreted (gray) and direct (white) links between the user's actions and the resulting change. The table is arranged from least transparent interaction (on the left) to most transparent (on the right), and suggests that fewer links and a larger proportion of direct links vs. intellectually interpreted links lead to a more transparent interaction.

4.1.2 Functional Affordance

The physical affordances of a physical-digital object can suggest appropriate ways for it to be used. Because the physical world has more degrees of freedom then can be monitored by a digital system, designing objects which suggest appropriate or recognizable functions through their physical design will enhance the transparency of a physical interface. For example, a physical-digital object designed to be held in one's hands should be small, light, and comfortable. A physical-digital object which is designed to be slid on a desk surface should sit flat on the surface and be

harder to pick up. The following table describes some of the physical affordances which contribute to the transparency of the physical-digital objects described in section 3.1:

Table 4.1. *Affordances* of Some Physical-Digital Objects	
Object	Affordance
 Triangles	Number of Triangles affords use by more than one person. Shape affords edge-to-edge tiling in many combinations, as well as specific 3D shapes. Magnets afford easy, impermanent connection, easy disconnection. Flat surface affords pictorial marking.
 Bottles	Cork affords opening or closing and leaving open or closed (toggle).
 Voice Box	Spring-hinge lid affords opening lid only temporarily. Small box affords holding, shaking.
 Marble Answering Machine	Round marbles afford being moved by gravity. Indentation on machine suggests placing marbles there temporarily (listening to messages). Hole in top of machine suggests putting marbles away there permanently (erasing message).

LEGO Helicopter	Size and weight affords holding in 3D space, helicopter design affords pointing, hovering.
Tangible Geospace	Size and shape afford sliding along desk surface (as handles for physical data).
Bricks	Size and shape afford sliding along or lifting off the surface (dragging, releasing graphical elements)

4.1.3   Choosing Transducers

Choosing appropriate transducer technology can help render an interface more transparent. The key to this is to take advantage of real-world phenomena that are well understood. Using sensors that can detect changes in inertia, heat, deformation, or pressure, physical-digital objects can be made to react to manipulation in appropriately informative ways. For example, shaking a small box might trigger a closely coupled sound, offering insight into the quantity of data that is metaphorically contained in it. Sensors which track position or state electromagnetically or through video can respond to changes in the relative orientation of objects, and direct electrical contacts, infra-red, or capacitive coupling are examples of technologies that can track relative proximity of objects to one another. These sensing technologies and complementary actuators are further discussed in Appendix A.

4.2   Mapping

I use the tem *mapping* to refer to the design of two aspects of physical-digital objects: the *semantic content* and the *application scope*. By *semantic content* I refer to the digital meaning of an object as expressed by metaphor. The form of the physical-digital object can be representational, metaphorically depicting the object's meaning. For example, the Tangible Geospace application described in section 3.1.x uses representational *phicons* in the shape of specific buildings to give access to mapping information about those buildings. One can also think of *mapping* in the context of the *application scope* in which the physical-digital object is used – a single object representing a single piece of data, an assortment of objects used to define a data-space, or a set of communicating, interacting objects as physical-digital building blocks.

4.2.1   Design for Semantic Content

The degree to which a physical-digital object's form represents its semantic content will depend highly upon its intended use. Very iconic objects of people or things provide concrete stand-ins for the concepts which they represent. The *phicons* used in the Tangible Geospace application are specifically and inextricably linked with the specific buildings which they represent. At the other extreme are systems like Bricks or Triangles, which provide a generic form which can be mapped to a variety of meanings depending on the context of its use. For certain applications, each of these might be preferable. Figure 4.7 illustrates the varying degree of semantic representation present in some of the physical-digital objects described in section 3.1.

Representational Form   Generic Form

Figure 4.7 The form of physical-digital objects can range from highly representational to very generic. The LEGO Helicopter and Tangible Geospace use highly representational forms. Objects like Bottles, Voice Box, and Triangle*s* can acquire semantic representation through their materials or surfacing, while the form of Bricks and the Marble Answering Machine marbles focus more on the physical interaction than on semantic representation. Generic forms can be more versatile, but depending on the context of interaction, representational forms can lead to greater transparency of interaction (See S.4.1).

Even at the representational end of the scale, the specificity of objects can vary widely. Scott McCloud's insight on abstraction in comics [McCl93], reprinted here as figure 4.8, can also be applied to physical-digital objects. One could imagine a naval planning application, for example, which used physical-digital objects to represent various boats on a physical map, providing access to digital information about the location, capacity, and type of equipment available on each boat. In such a situation, the level of representation of the physical-digital objects should be enough that different types of boat could be differentiated from one another, but not so specific that every boat in the fleet require an exact representation on the map.
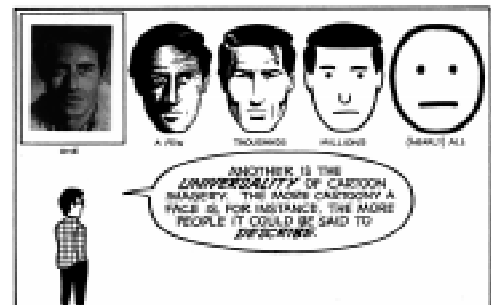
Figure 4.8 Like comic drawings, even representational physical-digital objects can have varying degrees of specificity. This can be useful when one type of object is required to play different specific roles at different times.

In addition to the absolute form of an object, its materials, construction, and ornament can represent semantic content. For instance, a hand-carved wooden block suggests a different mood than a stainless steel block of the same dimensions. This can be a useful consideration when semantic representation is secondary

design issue, allowing the form of the objects to provide clear functional affordances while maintaining a desired mood.

4.2.2 Design for Application Scope

The mapping of physical-digital objects must match the application context in which the user is interacting with digital information. Put another way, at what scale does the user *think about* the data that he or she is interacting with? The primitives of digital information are, of course, the bits (the "DNA of information" [Neg95]). But in the context of an application these bits combine hierarchically to become numbers, words, records, models, pictures, and much more. I refer to the user's perception of the lowest contextually relevant level of information as the **data items** of an application. What constitutes the data items will change depending on the application, and even within a given application or for different users. For example, if I am looking through a digital address book for a particular record, the appropriate contextual data item is most likely the *record*. If I then begin editing a given record, the context of my interaction has shifted, and the level at which I think about the data becomes the *entry* – name, telephone number, email address, and so on.

This user-centered perception of the scope of data items becomes important when metaphorically mapping a certain physical system to a given body of information. I have identified the following three fundamental contexts in which we can interact with data items:

- *Manipulating parameters of a single data item*—If we view each data item as an entity with a series of parameters, we can imagine reviewing, adjusting or editing these parameters using an object. For instance, if we have an object that is metaphorically representing a telephone message, the direction from which we

observe this object might give us information about its length, speed or importance. The vigor with which we manipulate it might affect its playback speed or longevity.

- *Exploring identity or meaning of individual data items*—In a given user context, we might have many similar items which we wish to navigate or identify. In keeping with our example, answering machine messages left using Durrell Bishop's *Marble Answering Machine* (see section 3.1.X) can be manipulated in this way. The context of use of this device involves identifying and sorting data items as marbles. Users do not need or have access to the parameters of each individual data item, as they would in the previous example.

- *Exploring combinations of many items—*The user context might involve many data items whose relationships are important to the interaction. For example a world built out of data items describing objects and their behaviors, or a data-space built from a query result where the arguments of the query make up the data items.

Each of these three fundamental interaction contexts suggests a different kind of physical-digital object. For instance, a single Voice Box or Bottle could be used in the first context, Bishop's marbles from the Marble Answering Machine in the second, and a system like the Triangles in the third. These fundamental contexts can be combined to form more complex contexts, suggesting appropriate design strategies for physical-digital objects. For example, if Triangles could be deformed or stretched, they might allow users to manipulate parameters of the data items as well as their interconnections.

Because the latter of the three contexts can involve creating new digital objects from components, a simple shift of perspective can lead back to the first context. Data structures enable us to make this shift in purely digital programming environments. In the physical world, this is analogous to matter built from its constituent molecules, atoms, and quarks. Each particular task will suggest an

appropriate scale at which to undertake its design. (I think of this as a conceptual version of the film *Powers of Ten* by Charles and Ray Eames [Eam78].) Figure 4.9 illustrates the contextual design space described by these primitives as a recursive spiral, with the physical-digital objects introduced in section 3.1 positioned around its arc.
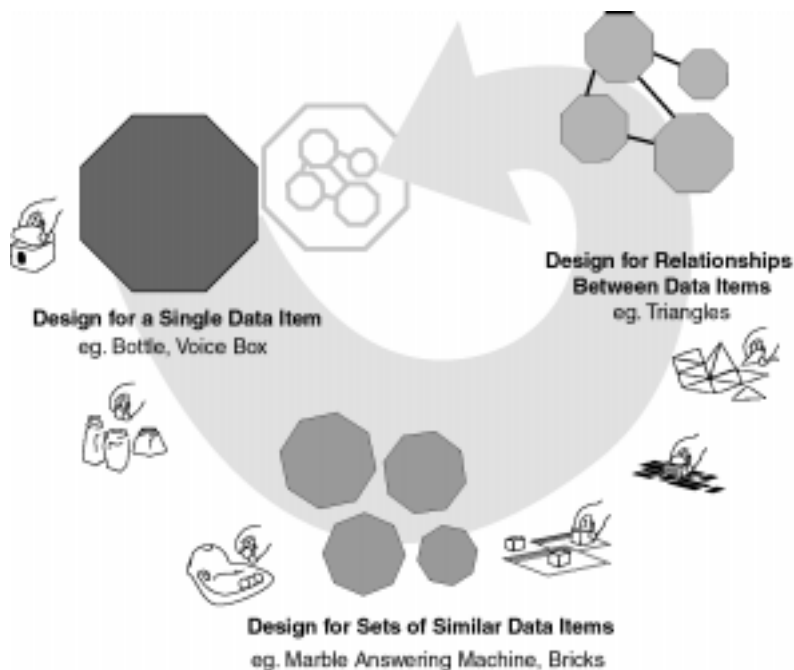
Figure 4.9

The contextual design space of physical-digital objects is much like the building blocks of the physical world – application contexts dealing with a single element combine into contexts involving groups, and then the relationship between these groups, which can in turn be seen as a new single object. Locating a particular interaction context on this spiral can help determine appropriate interaction metaphors for a physical-digital object system.

4.3   An Interdisciplinary Design Challenge

The dual identity of physical-digital objects necessitates an interdisciplinary approach to their design. Electronics, software, mechanical, industrial, and interaction design are all necessary and complementary disciplines, each informing and guiding the others. Many design problems have solutions in several of these

domains. Finding the best solution requires a good knowledge of the possibilities across disciplines.

The design of the Triangles system illustrated this on many occasions. For example, the number of electrical connections on each edge of the Triangles was affected by the decision to provide power externally rather than using batteries. This in turn affected the software architecture of the system in that the message-passing scheme was able to rely on separate lines for both local and global communication on each Triangle edge. As an alternative, we considered modulating the digital communication signals over the same connector which provided power, which would have reduced the mechanical complexity of the connectors, but resulted in a more complex circuit. It was only through active consideration of all aspects of the design that we were able to make decisions which resulted in a working system. (For more information on the technical implementation of the Triangles, see Appendix B).

As the design of physical-digital objects matures, it is important that design teams incorporate interdisciplinary skills. An iterative, exploratory design practice which allows rapid prototyping, good communication and an overall understanding of issues across disciplines will be a key factor in the design of successful physical-digital objects.

In this chapter I have described a conceptual framework for thinking about the design of physical-digital objects. This framework rests on the concept that the objects have physical and digital identities from their conception, brought together by three key principles: *coupling, transparency,* and *mapping.* These design principles each inform the others, and together they provide direction in the design of physical-digital objects.

5   Applications of Triangles

The Triangles system is significant among the physical-digital projects that I have discussed in this thesis, in that it was designed as a general-purpose platform for exploring many applications of physical-digital objects. The generic triangular shape of the tiles and their ability to be resurfaced with new images or textures allows a rich range of possible applications to be designed for Triangles. This chapter describes four applications of the Triangles system, in the areas of storytelling, media system control, and artistic expression.

5.0   Galapagos! – A World-Wide Web Story

In *Galapagos!*, partial illustrations of characters, places and events are placed on the faces of the Triangles in such a way that one or more users connecting two edges together can complete these images. As the two halves of a character or event are connected, web pages containing the content of the story appear on the user's computer screen. Which Triangles are chosen and the order in which they are connected to one another determine aspects of the progression of the story. The

result is a non-linear narrative that is told partially by a comic book-like arrangement of physical tiles, and partially by animated images and text on a computer screen (fig. 6.1).

One problem with *Galapagos!* is that its content is entirely visual, requiring the user to split their visual focus between the images and text on the computer screen and the Triangles themselves. The perceived digital interaction in this case is clearly not in body-space (see section 4.0.2). Children who played with the application did not always know where they should be looking, and expected audio feedback. This issue was addressed in the next storytelling application that we created, *Cinderella 2000*.

Figure 6.1

By arranging the Triangles to complete images of characters, places and events, a user navigates the *Galapagos!* story.

5.1 Cinderella 2000 -- An Audio Comic Book

Cinderella 2000 presents a modern version of the Cinderella fairy tale. Interactively arranging seven Triangles that depict various aspects of the story, a user can trigger audio samples stored on a desktop computer, creating a soundtrack of sound effects, narration and dialogue in the voices of the characters. These sounds are synchronized with the progression of the story, because they are triggered by specific connection events and Triangle configurations. Using audio

for the output avoided the split-focus experienced with *Galapagos!*, creating a storytelling experience that was perceptually more centered in the user's body-space.

The images for *Cinderella 2000* were more varied in their arrangement and design than those in *Galapagos!* The design of the visuals was greatly influenced by the techniques and visual language of comics [McCl93], making use of framing, scale, implied action and composition to create a narrative progression through still frames (fig 6.2).

Also, two specific Triangles were created as interaction devices:

- *Event Triangles*, symbolizing specific events in the story, for example the arrival of the invitation to the ball. Attaching an Event Triangle changes the context of the story, and thus the behavior of the characters.
- An *Info Triangle*, depicting three comic-book voice-bubbles. Attaching a specific edge of this Triangle to a character would cause that character to reveal certain personal information.

Figure 6.2

The images used in *Cinderella 2000* draw on the visual style and language of Comics, using framing, a variety of scales and viewpoints, and 'thought bubbles'.

5.1.1 Lessons Learned from Storytelling Applications

With *Galapagos!* and *Cinderella 2000*, we showed the potential of the Triangles system as a general interface for non-linear storytelling. In creating these

applications, we also explored techniques and developed general authoring tools that would be useful for others creating nonlinear Triangles content.

However, one thing that became clear from implementing these two applications was the difficulty inherent in authoring unique content for the astonishing number of configurations possible with the Triangles system. The *Galapagos!* and *Cinderella 2000* applications each used seven Triangles, offering literally millions of possible unique configurations. It was clear that limiting the number of *appropriate* connections was necessary, in order to avoid having to create a huge number of unique content events. This was addressed in *Galapagos!* by using the partial illustrations to suggest which connections would be appropriate. Still, interaction issues arose around what would happen if incorrect connections were made. For example, connecting half of a turtle to half of a bird might seem reasonable in a fantasy story about mythical animals.

If the application could respond appropriately to any of hundreds of thousands of possible connections, some extremely compelling and interesting storytelling applications might be possible. Advanced artificial intelligence and emergent behavior research [Blu95, Res97] suggests that such applications could actually be written, generating or modifying content on the fly and thus making full use of the Triangles system's potential. In the future, we hope to collaborate with experts in this field to further investigate this possibility.

Another critical lesson learned from the storytelling applications was the importance of providing a single focus for the user's attention. The use of audio feedback was much more effective with children than pure visual content, as discussed above.

Triangles provide a very simple means for interacting with a potentially very complicated set of character relationships and storytelling situations. This ease of manipulation can also be applied to other sorts of information relationships. The next applications we developed use Triangles as an interface for configuring complicated media systems.

5.2 TriMediaManager

Triangles' potential as a control system for information was further explored in *TriMediaManager*, an application in which the Triangles system is used to select and configure various media during a broadcast lecture or presentation. Triangles are given markings representing content – audio and video clips, 3D datasets, images and other documents to be used during the session. The Triangle that is directly attached to the computer is labeled as the *display*. During the session, the presenter can interactively decide which media is being shown by physically rearranging the positions of the Triangles in relation to each other and to the display Triangle.

For example, if a presenter has access to a live video feed of herself, an audio clip, two video clips and a variety of images (presentation slides, for example), these can each be selected at any time by simply attaching the appropriate tile to the display Triangle. If the presenter wishes to present several of these media in parallel, this can be achieved by joining the content Triangles with one another. When this occurs, *TriMediaManager* attempts to simultaneously display as much of the total content as possible, giving precedence according to the proximity of each tile to the display Triangle. In our example, if the tile representing the live feed of the presenter were directly connected to the display Triangle, it would be broadcast as a full-screen image with audio. If an *image* tile were then connected to another edge

of the *live video* Triangle, it would appear as a smaller image, inset into the video feed (fig. 6.3). Connecting a *video clip* Triangle on the remaining edge of the live video tile would inset the appropriate video clip in another corner of the screen (the audio streams from the live feed and the clip would be mixed).
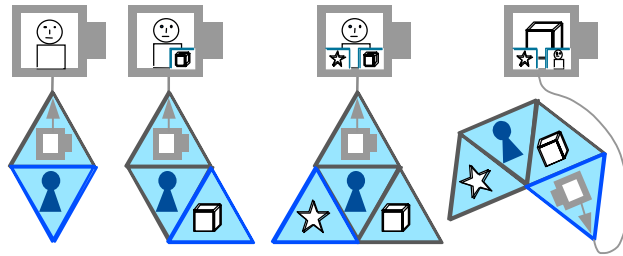
Figure 6.3

A user can control the display hierarchy of video imagery and other media by manipulating the physical *TriMediaManager* tiles.

At any point in the presentation, the presenter could easily change the display priorities of the various media. Moving the display Triangle so that it was connected directly to the video clip tile would cause the hierarchy to shift, and the output to respond accordingly. The clip would take over the full screen, with the image and the live video feed each inset as smaller windows.

The presenter might attach still more Triangles to the configuration, which might or might not be immediately displayed, depending on available output resources and proximity to the display Triangle. The *TriMediaManager* application keeps track of what display resources are available (inset windows, audio channels, volume control, etc) and how each type of information is being displayed (i.e. moving video, still image, audio, or rendered object) and allocates resources for as much of the data as possible.

TriMediaManager takes advantage of the close coupling of physical interactions to digital events, and the subsequent feedback of the persistent physical Triangles to indicate mode. Using the Triangles to represent high-level content selection like "vacation clip" or "earnings slide" as opposed to using traditional patching controls

like "VCR1→MON2" or "carousel advance" allows the presenter to manipulate the digital elements of the presentation as though they were physical. This puts the focus on content, allowing the presenter to dynamically rearrange their presentation if the need arises.

5.3   **The Digital Veil**

In creating *Galapagos!*, *Cinderella 2000*, and *TriMediaManager*, we explored many of the benefits of the Triangles system, including the exploratory nature of rearranging Triangles, the combinatorial potential of Triangles configurations, and the body-space interaction that results from using audio output with the physical system.

One drawback common to all of these applications is their use of pre-defined mappings of information to Triangles. This requires extensive content authoring before each application can be used. *The Digital Veil*, the next Triangles project that was undertaken, allows users to control not only the output generated by

specific Triangles interactions, but to assign and reassign meaning to groupings of Triangles during the course of an interaction. *The Digital Veil* was created as an art installation for the 1997 *Ars Electronica* festival in Linz, Austria[Orth97].

Figure 6.4 The images used in *The Digital Veil* were chosen to be ambiguous yet evocative. Users can combine them and then give meaning to their combination by recording a digital audio sample.

The piece consists of a table on which are laid out 35 Triangles. Each tile has a photograph, illustration, graphic symbol or

physical texture applied to its surface(fig. 6.4). These elements were designed to be beautiful, evocative and meaningful, both individually and in combination with each other. The public is invited to interact with the tiles in two locations on the table (fig. 6.5), creating arrangements of Triangles and connecting them to the *input station* or the *output station* – two small boxes on the table with exposed Triangle-edge connectors. The input station has a light-up button and a microphone on it. When a user connects up to four Triangles together and attaches this arrangement to the input station, the button lights up, and the user can push it and speak into the microphone. Their voice is sampled and linked with the specific arrangement of Triangles that they created. In this way, participants can assign meaning to their configurations, creating illustrated phrases and small narratives that metaphorically hold personal digital meaning.

Figure 6.5 Members of the public interacting with *The Digital Veil* could collaboratively explore the data space created by other visitors, or add to it with their own voice.

At the output station, participants can create large configurations of Triangles, building a visual and tactile texture on the table in front of them. As they do so, each individual Triangle that they add triggers its own evocative audio sample, building an aural texture to accompany the configuration that they create. In creating the large configuration, if the user arranges any of the tiles to form one of the "phrases" that had been recorded by a previous participant, that audio recording is also played back. In this way, the piece grows and changes over the course of its presentation, keeping a memory of the meanings and associations that users have created.

The Digital Veil was created to address the reconfigurability of meaning that is inherent in digital information systems. Its use of Triangles allows participants to explore this in an interactive and creative way by taking advantage of the *transparency* of the Triangles interface (see section 4.1). The Triangles provide the feeling that users are actually holding and rearranging the information itself, without any intermediate intellectually interpreted interface.

Audience members greatly enjoyed being able to quickly make new groupings of Triangles that they could personalize with their voice. They found their own uses for the application, sometimes leaving secret messages in the system to be retrieved later, or even singing rounds that could be controlled by adding new Triangles at the right time.

In this chapter, I have described four unique applications of the Triangles system, which illustrate how *coupling* and *transparency* enabled the perception of Triangles as physical-digital objects. The semantic *mappings* used in these applications varied, but they each had an emphasis on the rearrangement and connections between data items as their application contexts. This was facilitated and made transparent by the affordances of the Triangles' design—namely, their tiling shape and magnetic edges.

6   Conclusion

In this thesis I have presented my vision of a new type of dual-identity object, and articulated three basic principles to serve as a foundation for physical-digital object design. More importantly, I have suggested that these objects will encourage a new way of thinking about our interactions with information as free from the constraints of existing computer systems.

Physical-digital objects enable us to conceive of digital information as something other than words and pictures on a computer screen. They allow us to pick up, contain, shake, polish or even *break* our data. We can imagine *compressing* a file system, or keeping data in our pockets, or assembling data structures with our bare hands.

The three design principles which I have presented guide the creation of flexible physical-digital objects whose functioning can be explored, discovered and adapted by their owners, rather than being constrained by the arbitrary lexical command sets and input/output devices of current information interfaces.

- **Consistent coupling** enables objects to express their digital state clearly and persistently, and allows us to manipulate that state just as easily as we can manipulate the objects themselves. It also

allows us to explore and adapt both the physical and digital aspects of an object in novel ways if the need arises.

- A **transparent design** uses physical affordances and clear coupling to reduce the degree to which an object's digital functioning requires intellectual interpretation. We can work directly with content, rather than "working a computer".

- Designing **appropriate mapping** yields specific physical-digital objects for given functionality or subject matter. "The right tool for the right job."

Designing a physical-digital object is more than simply creating an ergonomic enclosure for existing circuitry, or defining a communication protocol for stuffed animals. Physical-digital objects have two equally important identities from their conception. Designing them well depends on understanding them as both physical *and* digital, and merging these identities into a coherent single object, through application of these three guiding principles.

A powerful concept presents itself in objects with dual identities. New social, cultural and creative everyday use of digital technology will result from these objects, changing the way we think and learn. An interdisciplinary mindset and a clear conceptual framework will enable the design of future interaction with digital information to move beyond input devices, and into our lives.

Appendix A: Implementation Technology

This appendix provides an overview of some basic technologies used by digital systems to sense and act upon the physical world, for it is these technologies that are at the heart of the physical-digital object. I have not emphasized technology very much in the body of this thesis because I wanted to discuss the conceptual design of the physical-digital object as divorced from its implementation. Technology can often play an important and deciding factor in design, however, so it is useful to acknowledge and review the commonly-used technologies of sensing and actuation. The list I present here is by no means intended to be a complete technology catalog or taxonomy, but rather to serve as a technical overview to complement the conceptual matter discussed in previous chapters.

There has been quite a bit of prior work on complete taxonomies and systematic comparison of input devices [e.g. Bux83, Bux86, Card90, Lip93] which could conceivably be extended towards classifying physical-digital objects, but this lies outside of the scope and timeline of this thesis and is therefore regrettably reserved for future work.

Although a primary goal of the physical-digital object is to lessen the perceptual separation between input and output, the underlying technologies used can still best be classified as *sensors* or *actuators*.

Sensors

Sensing Position/Orientation

Position and orientation can each have several meanings. Sensors exist which track relative position, absolute position, and changes in position to various degrees of accuracy. Position sensors can act as orientation sensors, if the relative positions of parts of an object are measured. Rotation can also be sensed in absolute or relative ways, and can be bounded or unbounded. In addition, sensors have varying degrees of freedom (DOF): some sensors track position and rotation in three dimensions each (6DOF), while others are constrained to just one or two degrees of freedom. In addition, sensors operate at a variety of distances, and require a range of bandwidth and CPU processing power. The following list describes a variety of position- and orientation-sensing technologies:

- **Direct electrical contacts or switches**
  Simple foil, wire, or fabric electrical connectors, as well as toggle or momentary switches, can be used to determine and object's position. Position is sensed relative to another object such as a table, game-board or another object by completing or breaking a circuit. This is a very low-bandwidth and simple way of detecting physical changes.

- **Rotary sensors (potentiometers and optical encoders)**
  Potentiometers or optical encoders convert rotation of a shaft to voltage levels or pulses. Coupled with mechanical systems for converting translation to rotation, they can be used to measure movement of objects and thus relative position. Potentiometers

can only turn through a limited number of degrees or turns in any direction, while optical encoders are unbounded.

- **Acoustic**
Ultrasound can be used to detect the distance between a transmitter and acoustically reflective objects. Microphones embedded in surfaces can also be used to detect the position of objects as they contact the surface [Wis98b]. These techniques require fast computation, but can operate at a distance on passive objects.

- **Radio-Frequency**
Transmitted radio-frequency pulses can be very precisely detected by a remote receiver, allowing a system to determine the relative distance of the receiver from the transmitter. Very small receivers can be arranged such that their position and orientation can each be inferred in three dimensions. Polhemus™ and Ascension Technologies' Flock of Birds™ are popular examples of such systems. Radio-frequency tracking devices are generally expensive and high-bandwidth, requiring fast computation. They have a limited range, and are also easily affected by large metal objects within the range of the transmitter.

- **Infra-Red Optical**
Infra-red diodes and can be used to generate beacons which are sensed by receiver elements. Examples of such arrangements include objects which determine their position with reference to a fixed beacon in the environment or with respect to other objects, or arrangements of IR beams which register the presence of objects as they are crossed. Occlusion becomes an issue when obstacles are present in the environment.

- **Video Optical**
Using machine-vision techniques, video frames can be analyzed to detect the color, motion, or form of distant objects with varying degrees of accuracy. Such technology generally requires specialized hardware or software capable of high bandwidth and computationally expensive digital signal processing.

- **Laser Tracking**
Reflected laser light can be analyzed to determine the exact distance of objects from the laser source, allowing very precise determination of position even at great distances. Scanning range-finders can detect the position of many different objects in a space.

- **Hall-Effect**
  Hall-effect sensors provide precise measurement of magnetic fields and can be used to track magnetic objects at close range. They are often used to measure the rotation of shafts or the extent of linear actuators, but arrays of hall-effect sensors could be used to measure the position of objects on a surface.

- **Accelerometer**
  Accelerometers provide voltages which change with acceleration. This can be used to accurately infer the speed and position of a moving object which starts from a known location. Accelerometers can be arranged orthogonally to one another in order to measure acceleration in three dimensions.

- **Electric Field Sensing**
  A very versatile means for sensing the position of *people* involves measuring the change in small electric fields generated by a special circuit [Zim95]. This has many applications, and is particularly effective for sensing the approach and movement of human hands near physical-digital objects.

Sensing Identity

Physical-digital object systems which use more than one object often need to be able to differentiate between the objects. Following are a few techniques that can allow digital systems to determine the identity of an object:

- **Resistor ID**
  Resistors are probably the simplest way to determine identity. Embedding a resistor in an object enables the object to be identified when it is directly connected to a "reader" device, completing a simple voltage divider circuit. Resistors are cheap and readily available, but can be problematic due to noisy electrical connections between the object and the reader.

- **Serial ID chips**
  Dallas Semiconductor, Inc. manufactures several types of tiny electronic components with trade names like I-Button[®], Silicon Serial Number[®], and Touch Memory[®]. They each store a unique digital ID, which can be directly read using a serial protocol [Dal95]. These chips are a reliable and easily implemented means

for determining the identity of an object through direct electrical contact.

- **Barcodes**
  Barcodes are ubiquitous patterns of high-contrast stripes (or more recently, dots) that can be scanned with a laser. Barcodes can be applied to almost any object, but take some space and can interfere with the aesthetic of an object. Many mechanisms exist for scanning, including pens, wands, and supermarket-style scanners. Generally scanning requires close proximity (0-6 inches), but in controlled circumstances this method can be used to identify objects at a distance.

- **Infra-Red**
  Infra-red diodes can transmit pulses of invisible light which can be used to convey identity or any other information. Occlusion and noise can be issues with IR.

- **Radio-Frequency Tagging**
  Many schemes exist for identifying objects via radio-frequency. Generally, they consist of some passive metal strip or coil which oscillates at a particular frequency [Fle97]. A transmitter is then used to "sweep" through various frequencies, and oscillations are measured. Radio transmission can also be modulated to carry data, allowing objects to transmit their identities directly to a remote receiver.

- **Video**
  Video analysis can be performed to identify color, shape and other features of an object. This technology works best in conjunction with special color-coding or patterns applied to the objects which are to be identified. Video processing requires high-bandwidth, computationally intensive hardware.

Sensing Deformation or Physical Alteration

Sensors exist for sensing pressure, shear, torque, or stretching. Generally, these are embedded in an object rather than being sensed remotely:

- **Piezoelectric Sensors**
  Piezoelectric materials generate high voltages, with low current,

when compressed or hit. They can be used to detect vibration, as switches, or as pressure gauges, and they can be embedded inside polyurethane bumpers to measure bouncing or bumping.

- **Fiber Optics**
  Optical fibers can be used to sense bending or deflection by passing light through the fiber and measuring how much is transmitted. The more the fiber flexes, the more light will escape through its edges. This technique is used in DataGlove™-type applications [Rhein91]. By creating crossed loops of optical fiber, bending can be sensed in two orthogonal directions and twist can be inferred [Dan97].

- Strain Gauges
  Some alloys change their resistive properties when they are flexed or strained, allowing them to be used as deformation sensors. There are hundreds of different types of strain gauges available, made from different alloys and useful in various contexts.

Actuators

Of the five senses, the primary senses of sight, hearing and touch are most relevant to physical-digital object interactions. There has been some exploration of smell as a controlled output for digital systems [Bar96], but it is difficult to control accurately and is quite subjective. The olfactory sense certainly has potential for creating expressive physical-digital objects, however, and should be actively pursued as an area of further research

Audio

Loudspeakers are the most common means for creating sound, and come in a variety of forms, from tiny piezoelectric buzzers to room-shaking subwoofers. There are also as many ways of making sounds using mechanical or physical motion or airflow. For instance, Matt Heckert's *Munich Samba* [Heck97], which won the grand prize for computer music at the 1997 Ars Electronica festival, uses

large machines with solenoids and motors to make music by banging, swinging, clicking, twisting and shaking.

Visual

Visual display technologies are many and varied, from standard computer monitors to exotic projection technologies:

- **Cathode Ray Tube (Monitor)**
Although large, bulky and power-hungry, monitors are quite likely the easiest type of visual display to implement, as they make up the standard primary output for current computer systems. Often monitors can be combined with mirrors or built into objects so as to give new physical context to the screen and its imagery.

- **Video projection**
Video projectors are getting smaller, brighter and less expensive every year, and it is possible to use them creatively to bring life to a surface or a group of objects, by projecting "virtual shadows" or very accurately and quickly changing the patterns of light on the objects [Und98].

- **Flat panel displays**
Liquid Crystal Display (LCD) panels are still the standard for high-resolution portable displays such as those found in laptop computers. There are several LCD technologies ranging from Cholesteric displays, which require very little power and are flexible (but only display in black or silver) to the custom LCD displays of digital watches or consumer appliances, to high-resolution, full-color displays which can be very expensive, require special power regulation, and backlighting. Currently one promising area of research is that of microdisplays—miniature full-color LCD displays the size of postage stamps. One problem with LCD panels traditionally has been the angle of view, which generally ranges from about 15 to 40 degrees. Plasma panels, on the other hand, have extremely good near-180-degree viewing angle, and very sharp color. Currently plasma panels are thicker, heavier, and more expensive than LCD panels, however.

- **Light-Emitting Diode (LED)**
  The LED is the most common means for providing visual feedback in small electronic objects. Available in red, yellow, green, orange, and recently blue, LEDs come in many shapes and sizes. They draw very little current and do not burn out with normal use, and they stay cold during operation. LEDs are also available in arrays that enable the display of digits or patterns.

- **Electroluminescence**
  Electroluminescent materials glow (usually green or orange) when power is provided to them. They are available as flat sheets of plasticized material, and often used to provide backlit LCD displays for palmtops and digital watches. They are relatively expensive, and draw a great deal of current.

- **Light Bulbs**
  Light bulbs are available in a bewildering array of sizes, power requirements and brightness. Their light is generated by current passing through a filament, so they tend to get warm with use and have a limited lifetime. Halogen bulbs are extremely bright for their size and power requirements, but can be dangerous as they get very hot.

- **Neon and other gasses**
  Gas-filled tubes emit light if a voltage is put across two electrodes within the tube. Shapes can be created by bending the glass tubes (e.g. neon signs) or by bending the electrodes within the tubes (e.g. nixie tube displays). Neon displays generally require very little current but high DC voltage (>100V) to light.

- **Laser**
  Laser light can be used to project a spot of (usually red) light over a great distance. Small, inexpensive laser diodes are available prepackaged with driver circuitry and are as easy-to-use as LEDs. By combining laser diodes and some means for deflecting the laser beam, it is possible to build vector-based "drawing" displays such as those used in planetarium shows or concerts. Some drawbacks of using lasers are that they must be carefully controlled to avoid creating a safety hazard from people looking directly into the beam, and that there are currently no inexpensive laser diodes in colors other than red.

- **Mechanical motion**
  Visual display can be accomplished through the mechanical movement of solenoids, motors, gearworks or other physical actuators, of course. These mechanisms can be extremely effective for giving objects lifelike movements.

Tactile

The sense of touch has not been as prevalent a display medium as vision or hearing, but there are a few techniques for creating compelling tactile displays.

- **Fluids (with pump)**
  Objects can exert pressure using telescoping hydraulics or bladders full of liquid or gas, with pumps. Airflow in space can also be regulated using fans.

- **Piezo**
  Piezoelectric materials vibrate when a high current is passed through them.

- **Heating Elements**
  Pelletier junctions are small, flat elements which become warm on one side and cool on the other very quickly when a 5VDC voltage is applied across them. Resistors and filaments can also be used to generate heat, but are less easy to control.

- **Shape-Memory Alloy**
  Shape-Memory Alloys (SMA) are metals (usually wire or strips) which can be "trained" to take a specific shape when heated. A popular application of SMA is in Muscle-Wire™, a linear actuator made out of a piece of wire which contracts when a current is passed through it. Unfortunately, the amount of contraction in this type of application is usually quite small (3-5%). Precise control over SMA actuators is difficult due to their thermal nature.

- **Motors**
  Motors are the most common means for creating mechanical action from electricity. Several types of motors exist which can augment physical-digital object, including stepper motors whose relative position can be controlled with digital pulses, servos whose absolute position can be controlled using pulses of varying lengths and DC motors, whose speed can be controlled with varying DC

voltage. Motors are often used in conjunction with gears and other mechanical systems to create linear motion, vibration, or precise mechanical positioning.

- **Solenoids (Magnetism)**
  Solenoids are electromagnets with moveable iron cores. When current is applied through the electromagnet's coil, the core is forced in one direction or the other. Solenoids can be used to produce vibration or direct percussive pressure, but they draw quite a bit of current, and can usually only be turned on or off, offering no fine position control.

Appendix B: Bottles Technical Implementation

This appendix provides an overview of the technical implementation of the Bottles system. The system was part of a larger prototype called the ambientROOM, which demonstrated the concept of ambient media [Ish97, Wis98a]. The ambientROOM is based on the Personal Harbor™ product from Steelcase Corp., a 6'x8' enclosed mini-office installation (fig. B.1). The ambientROOM surrounds the user within an augmented environment, providing subtle augmentations to activities conducted within the room. In this demonstration setting, the Bottles were used to change the ambient sounds present in the ambientROOM.

Figure B.1

The ambientROOM is based on the Personal Harbor™ product from Steelcase Corp. A 6'x8' mini-office, it is augmented with sensors, ambient sound sound, light, airflow and projection.

Initial Prototyping

Functional Sketch

Because the ambientROOM project is a prototyping environment, the demonstrations installed therein were initially simply conceptual design sketches. The initial Bottle implementation used small, clear glass bottles (fig. 3.6) with corks. Two strips of copper tape were applied to the inner rim of each bottle's mouths, and the cork was wrapped with a thin strip of copper as well. Fine-gauge enameled magnet wire was affixed to the copper tape at the mouth of the bottle, creating a circuit which would be complete when the Bottle was closed and broken when it was opened (fig. B.2). The magnet wires were fine enough that they were not noticeable to people observing the interaction with the Bottles.

Figure B.2

Copper strips applied to the rim of the bottle and the cork close an electrical circuit when the Bottle is closed.

The other ends of the magnet wire were used as "input transducers" to an I-Cube™ MIDI interface kit, by Infusion Systems, Inc. This system is a basic analog-to-digital converter designed to interface with Opcode® MAX™ MIDI control language on a Macintosh platform. We use MAX to control various sound samples which are triggered when the Bottles are opened or closed, and played using the ambientROOM's

Figure B.3

When the bottle is opened, an analog signal is to a MIDI event by the I-Cube system. A MAX program running on a computer then triggers a sampler to create ambient audio.
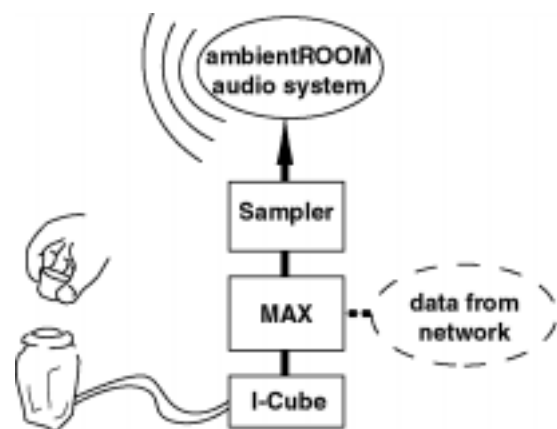
embedded audio system (fig. B.3).

The samples are chosen and controlled by the MAX program, and thus mapped to appropriate information. In the initial demonstration of the Bottles system, we mapped the status of network traffic in our building to sampled sounds of real vehicular traffic. The concept sketch version of the Bottles system always triggered the same sound sample when one particular bottle was opened. By using actual network data and controlling the volume of several layered samples, the working prototype could create an appropriate mapping of the vehicular traffic flow to the network traffic flow. Specific techniques for creating such mappings are currently an active area of ambient media research by members of the Tangible Media Group.

Future Implementation

Wireless Bottles

Current Bottles research is focusing on a more robust technical implementation of the Bottle. Embedded radio-frequency identification (RFID) tags[Fle97] in the Bottles and their corks, and a sensing coil embedded in the surface of the ambientROOM's desk allow remote, wireless sensing of the Bottles' states with very little impact on the appearance of the physical Bottles.

Visual Feedback

One other avenue of interest that is currently being pursued is the use of embedded neon tubes in the Bottles to provide a visual indication of the "quantity" of information which they metaphorically contain. These tubes can be activated at a distance using a controlled electromagnetic field.

Bottles Accessories

Bottles allow access to specific information which is metaphorically "contained" inside the object. The question arises of how did the information get there and how could one map new information sources to Bottles? One way to address this is to develop physical-digital objects specifically designed for these tasks, like information eyedroppers, tweezers or funnels. This raises many interesting conceptual and technical questions which are worth pursuing.

Appendix C: Triangles Technical Implementation

This appendix provides an overview of the technical implementation of the Triangles system. It is based largely on material that can be found on the Triangles web site (as of June 1998) at http://tangible.media.mit.edu/projects/triangles/main.html.
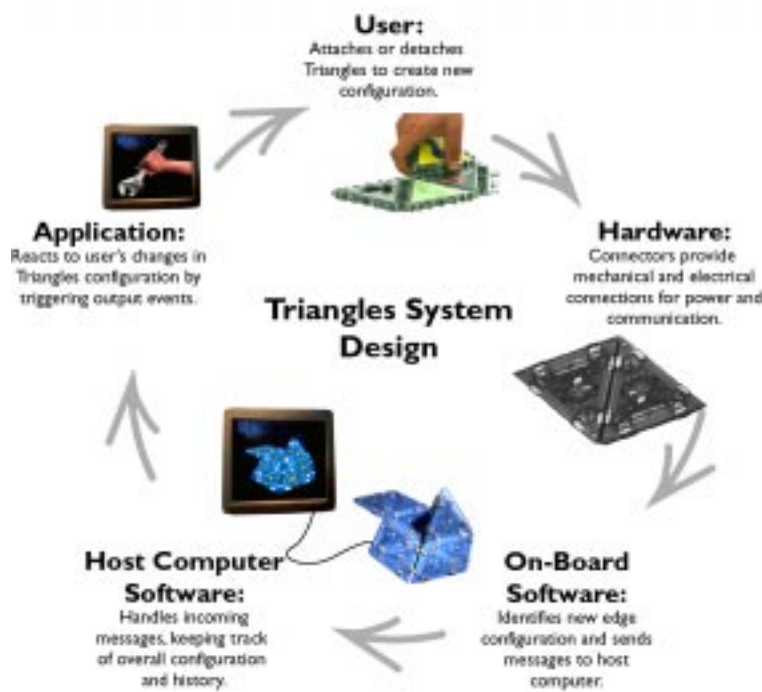

Figure C.1

The Triangle*s* system consists of 5 parts, as illustrated in this diagram. Each discrete user interaction follows the cycle from the user's action through the electrical and mechanical connections, microcontroller software, host computer engine, and finally the application layer, which responds to the user's action and generally prompts further action from the user.

Triangles Hardware

Housing

Each Triangle is housed in a specially designed acrylic casing. The original casings were cut in-house using a computer controlled CNC milling machine. This was extremely time- and labor-intensive. Later versions of the Triangles housing were created from three pieces using a laser cutter. This enabled us to easily outsource this aspect of the production, emailing CAD drawings to Maley Laser, Inc. in Providence, Rhode Island, and receiving the finished housings by courier.

Connectors

Many prototypes of various connector designs were investigated, including zippers, snaps, Velcro®, and tab-and-slot designs. It was determined that magnets provided the best feedback of connection and disconnection, and they allowed robust attachment even after repeated reconnections.

Two connector designs were actually implemented. The first used the magnets themselves as electrical contacts, and to ensure consistent contact during flexing and manipulation, ring-shaped magnets were mounted so as to slide on an axle (fig. C.2).

Printed Circuit Board ⎯⎯⎯

Nylon Washer

Sliding Magnet

Electrical Lug

Nylon Axle
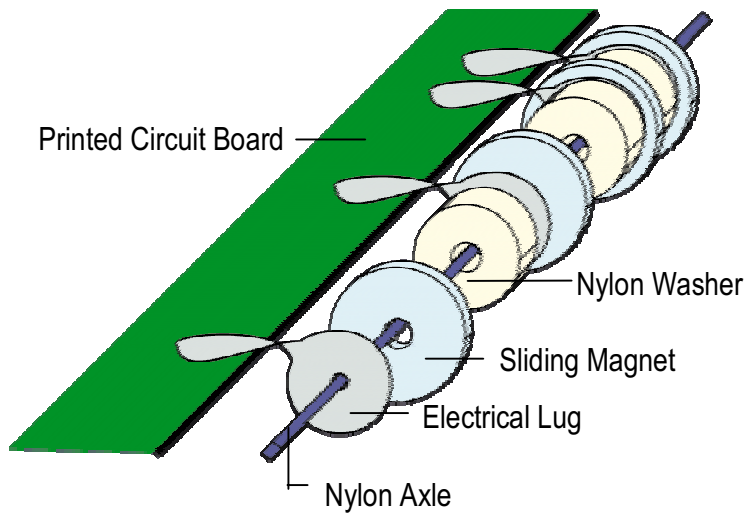
Figure C.2

The original Triangles connectors used conductive ring-shaped magnets which could slide on a non-conductive axle, to provide better contact during flexing.

These connectors worked, but they still sometimes interrupted the electrical connection during flexing. Our second connector design facilitated a more robust electrical contact, using fixed magnets to hold the Triangles edges together, and conductive Velcro® strips to make the electrical connection. The brush-like fibers of the electrical contacts provided a more continuous connection as the Triangles were manipulated and flexed (Fig. C.3).

Printed Circuit Board ⎯⎯⎯

Conductive Velcro

Fixed Magnet

Acrylic Frame
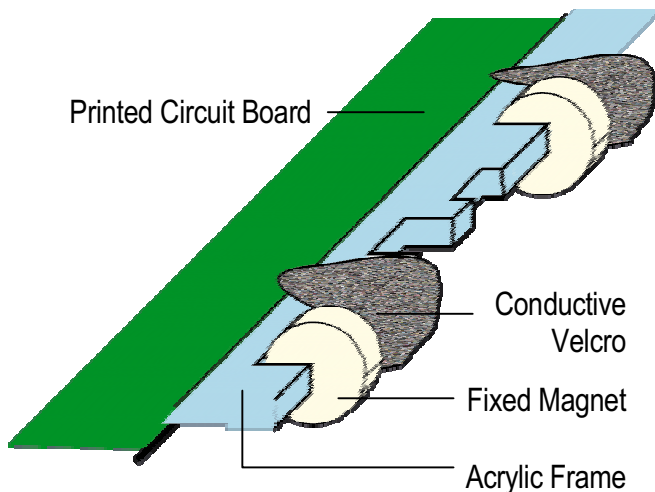
Figure C.3

The second version of the connectors used magnets which were rigidly fixed to an acrylic housing, and conductive Velcro® strips as electrical connectors.

Electronics

The Triangles circuit board is based around a Microchip® PIC16F84 processor. Power to the circuit is provided by a shared 12VDC power bus which is established by connecting Triangles to the system. A 5V power regulator on each Triangles circuit board regulates the power to the processor. Capacitors filter the power signal and the electrical connections, which include 2 local connections (send and receive) between each Triangle and its 3 neighbors, as well as a global 2-way communication bus which enables each Triangle to communicate with the host computer. Figure C.4 shows the Triangles circuit board.
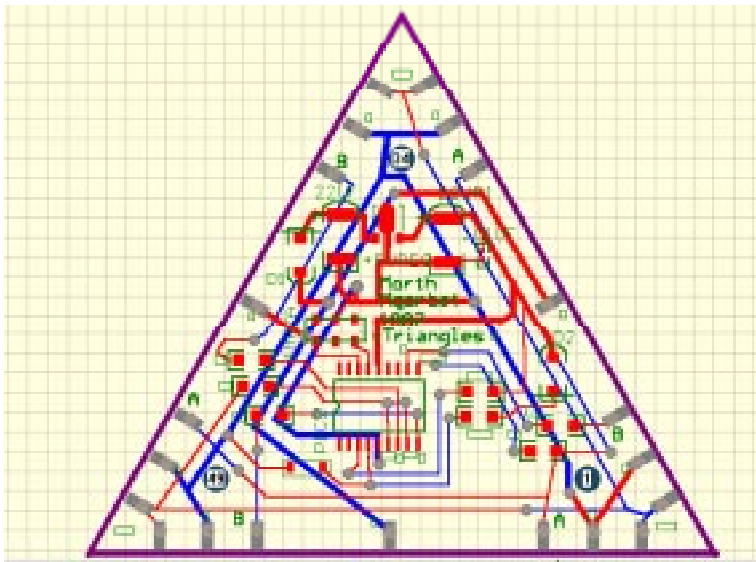
Figure C.4

The Triangle circuit board shows the PIC processor near the middle of the board, with power regulation circuitry above it and traces leading to the edge connectors.

An additional interface module is connected between the Triangles in the system and the host computer. This board consists of another PIC processor, a Maxim® RS232 communication chip, and a momentary reset switch which can be used to cut power to the entire Triangles system. The PIC processor on the interface module is responsible for generating timing codes for the Triangles to use in their global communication. Figure C.5 shows the Triangles interface module circuitry.
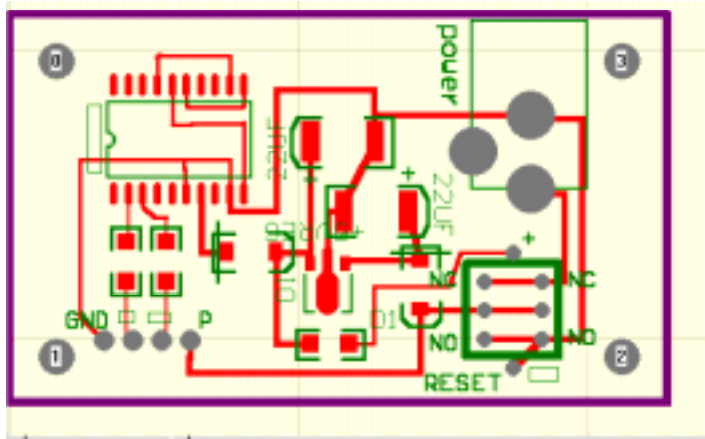
Figure C.5

The interface module circuit board shows four inputs for the power and global data busses (on the lower left) and the reset switch (on the lower right). 12VDC plugs in on the upper right, and the MAX serial communication chip is on the upper left.

The interface module can be customized with hardware and software additions, to provide application-specific functionality. This is further discussed later in this appendix.

Microprocessor Software

Triangles is a distributed system, with each triangle running primarily identical code. Each triangle knows only is position relative to its nearest neighbors, those connected directly its sides. This local communication between neighbors is facilitated by the host triangle which globally synchronizes communication. Triangles also take turns transmitting information about connection and disconnection events on a common serial bus, which is also shared by the API.

The code run by each triangle differs only by ID number assignment. This is necessary for the API to correctly identify configuration. Also, these ID numbers establish a precedence ordering for serial bus communication.

Development

This system was designed to minimize latency between a connection/disconnection event and the time the API receives information about it. The present ID swapping routine takes about 30ms to complete. Since communication is synchronous, this time is *constant* and *independent of the number of Triangles in the system*. Message relay time scales linearly with the number of connection/disconnection events being reported on a particular communication cycle. Messages are transmitted using RS232 at 9600 baud.

A previous communication routine relied on asynchronous communication between neighboring Triangles. Rather than using a common bus, each triangle would establish how far it was away from the host and then pass its messages to the neighbor along the shortest path. Each triangle attempted to establish communication with a neighbor on one side at a time. Successful communication would require that both Triangles at a connection be trying to establish communication at the same time. When this happened, Triangles would exchange ID information and pass messages about connection events to their lowest neighbor. This system proved to be quite slow, as Triangles successfully established communication with their neighbors relatively infrequently. Messages that had to be passed down long paths to the host took several seconds to arrive. This latency prompted redesign of the communication protocol.

Communication Routine

When the Triangles system is first turned on, the host Triangle will send `*x!` on the communication bus. This signals the API that the system has been turned on. This message is only sent once on power-up. The reset switch will momentarily

---

disconnect the system from power. When the switch is released, the host will respond with *x!, again signaling power-on.

The host initiates a communication cycle by sending g (as in go) on the bus. The host then falls into a loop where it clocks each communication bit of the ID swap. Each Triangle drives the specified bit of its 8-bit ID onto its "toAll" line. For each bit, a Triangle asserts its ID bit, checks the input lines from each of its neighbors while continuing to assert its ID bit, and then checks these lines again, still asserting its ID bit. (This method of bit-splicing is a plate spinning algorithm which allows the Triangles to send and receive bits simultaneously.) Each bit of ID takes seven clock pulses from the host to complete. This is repeated for all 8 bits.

The system then goes into a 12 clock-pulse error checking routine. During this period, each Triangle asserts those sides on which it has heard a neighbor and listens to verify that a neighbor heard it on that side. If this is not verified, the received bits are discarded.

Following the ID swap, there is a period where the Triangles take turn driving the communication bus. A triangle will only transmit a message when there has been a connection/disconnection event on one or more of its sides. A triangle will verify a connection/disconnection event 3 times (3 consecutive communication cycles) before reporting it. Contention is avoided by a race condition imposed on the system, whereby Triangles command the bus in order of increasing ID number. Each Triangle has a countdown timer based on its unique ID. If the Triangle sees any activity on the bus (a Triangle with a lower ID has taken control of the bus), it will wait until the bus clears and then reset its clock. When its clock runs out, it will control the bus.

Customized Host Units

The Host Triangle and Interface Module

The host Triangle utilizes the same circuit board as the other triangles, although it runs a different program. One can connect to it on two sides, as the third is used to connect it to the interface module.

The interface module has three main functions. First, it converts TTL level RS232 (0, +5V) to standard -12V, +1V to interface with a computer. Secondly, it distributes 9V DC power to system. This power is regulated to 5V on each Triangle board. Third, it provides a reset button. Upon reset, the system will transmit information about all current connections among triangles.

For most applications, the host module will consist of the host Triangle hardwired to the interface module. This configuration allows the API to listen to the bus, but not transmit to the Triangles system.

Example with Button

Customized units can also be designed for specific applications. Communication to the Triangles system is facilitated by the design of (PIN_B0) as an external interrupt pin. In hardware, this means that this pin has an external, weak pull-up resistor. Normally this pin is held at a logic high. When the line is driven low, as by communication from the API or by an external sensor (e.g. button) the host triangle can interrupt the communication routine and jump to a special point in its program to deal with the incoming signal. How this interrupt is handled depends on the software running on the host triangle.

One useful customization is the addition of a button, which may be used to record information to a particular configuration. For the "Digital Veil" application, (see section 6.4), we used this button to record audio samples. This specific example is a good demonstration of how the interrupt can be used and the code used for this can be used for a number of other applications using input devices other than a microphone. (This will require appropriate changes to the API). This application required the addition of button hardware between the host triangle and the interface module. When the button is pushed, the following sequence takes place:

HOST Triangle suspends current communication routine. Will resume when button sequence is finished.

BUTTON sends (*R!) to API signaling that it should start recording.

BUTTON flashes the LED in the button and waits until the button is released.

When button is released, BUTTON sends (*P!) to the computer and turns the LED off. This signals the API to stop recording.

The API sends (p) to BUTTON, confirming the receipt of (*P!).

BUTTON waits for (r) from the API. When is received, BUTTON lights the LED, signaling that another sample can be recorded.

Note: For this cycle to take place, the LED in the Button must be lit, signaling that BUTTON is ready to record. Also, if BUTTON ever receives (x) from the API, it will reset itself.

System Engine

Each Triangle in the system generates messages regarding changes to the system as they occur. The messages are timed and triggered by the host module, which communicates with the Triangles over a shared bus. The **system engine** is software module running on the PC or SGI machine to which the system is connected, which listens to this bus, interpereting the messages as they are received, and provides access to the state of the system through a library of Application Programming Interface (API) function calls. For more information on the host module and the individual microprocessor code running in each Triangle, see the Microprocessor Software section of this appendix. For more information on the API, please see the API Specification.

Message Format

The messages sent by the Triangles are of the form:

```
* version my-id:type neighbor-id edge!
```
where:

> **version** is the version #. We are still on version 1.

> **my-id** is the Triangle's unique ID

> **type** is either C or D for Connect or Disconnect.

> **neighbor-id** is the ID of the neighboring triangle being reported.

> **edge** is the side on which said event has happened.

For example:

```
* 1 12:C 6 0!
```

is sent by triangle #12 to say that it has just connected to triangle #6 on side 0.

The clock pulses that the host module sends (g!) on the communication bus are irrelevant to the system engine. Therefore, messages that are meant for the engine are packaged starting with an asterisk (*) and ending with an exclamation point (!), as in *message-goes-here!.

The engine parses messages by waiting for a !, then going back to the last * and looking at what lies between. Anything before the * is discarded.

Message Consolidation

Because each Triangle acts autonomously and is unaware of the messages sent by its neighbors, any connection event should cause the creation of *two* complimentary messages. Our above example had Triangles 12 and 2 being connected. In this event, the engine would actually parse the following two messages:

```
* 1 12:C 6 0!
```

and

```
* 1 6:C 12 2!
```

In this example, side 0 of Triangle 12 is connected to, or "sees" side 2 of Triangle 6.

The system engine takes such pairs of messages and consolidates them into specific *events*, which it uses to reconstruct the state of the system at any given time.

When Triangles are *disconnected* from each other, however, the Triangle bieng removed generally loses its power, so only one message is received by the engine. For example, if Triangle 12 were still attached, and Triangle 2 were now removed, the following message would appear:

```
* 1 12:D 6 0!
```

The engine would use this message to invalidate the previous connection event that it had built.

Time-Stamping

The system engine also time-stamps each message that is received, and only consolidates complimentary messages if they arrive within a specific time-threshold of each other. Currently, that time-threshold is set at 0.1 seconds, but this is adjustable using the API. When messages are consolidated, the resulting overall event is also time-stamped, and this allows the engine to reconstruct the entire history of interactions with the Triangles system for any given session. Access to this functionality is handled through the function calls which make up the API.

API Specification

The Triangles application programming interface is a C++ library that runs on SGI Irix and Windows platforms. It consists of the following core classes, described in detail hereafter: TsSystem, TlList, TlItem, TrTriangle, TrEdge, TeEvent, and TeTime.

TsSystem

This class represents the API serial polling system. It gathers data about the real triangles from the physical world, and synchronizes the internal virtual triangles with that information.

constructors:

```
TsSystem();

~TsSystem();
```

method index:

```
void Init(char *port);
void Close();
TlList * TsGetEventList();
TlList * TsGetTriangleList();
TlList * TsGetTriangleList(TeTime *when);
Bool TsRegisterEvent(TeEvent *event,
Callback callback,
void *userdata);
void TsSysUpdate();
void TsSysUpdate(char *message);
```

1.0.1   TsSystem Methods:

void Init(char *port)

Initializes the **TsSystem** object, opening the serial port and allowing the system to read and handle incoming messages.
Arguments: the serial port to read from, COM1 or COM2 etc.
Returns: void

---

void Close()

Shuts down the **TsSystem** object, clearing the triangle and event lists, and closing the serial connection.
Arguments: none
Returns: void

TlList * TsGetEventList();

Gets the complete list of events that have occurred since the Init() call.
Arguments: none
Returns: pointer to a TlList object of type TL_EVENT

TlList * TsGetTriangleList();

TlList * TsGetTriangleList(TeTime *when);

Gets the current list of all triangles in the system. Optionally gets the list of triangles in the system at any given time.
Arguments: an optional pointer to a TeTime object specifying the time for which the triangle data is requested
Returns: pointer to a TlList object of TlType TL_TRIANGLE

Bool TsRegisterEvent(TeEvent *event, Callback callback, void *userdata);

Registers an event that the system should watch for. If an event matching this event occurs, the specified callback function is called and passed a pointer to the event that occurred and a pointer to specific user data. The user specifies an event mask to watch for by creating a TeEvent object with TeType and TrEdge fields to be matched. The user may opt to leave some fields blank (-1) and they will be treated as wildcards. For example, if an empty TeEvent object is passed to **TsRegisterEvent**, all events trigger the callback. If the TeEvent specifies TE_CONNECT as its TeType, connection events trigger the callback, but disconnects are ignored. Since TeEvents are symmetrical (i.e. *A side 1 + B side 2 = B side 2 + A side 1*), the event's TrEdge objects can be specified in any order.
Arguments: pointer to a TeEvent object to use as an event mask; pointer to a callback function; user-defined data to pass back to the callback function.
Returns: Boolean -- True if the event was registered properly, False if there was an error.

void TsSysUpdate();

void TsSysUpdate(char *message);

Forces the system to update its triangle and event lists.
Arguments: optionally takes a simulated message of the same form as that which would come over the serial port. Otherwise reads from the serial port.
Returns: void

TlList

This class represents a linked list of TlItems. It is fairly standard linked-list fare.

constructors:

```
TlList(TlType type);

~TlList();
```

related data type index:

```
enum        TlType
enum        TlErr
class       TlItem
```

method index:

```
TlType       TlGetType();
int          TlGetSize();


TlErr        TlAddItem(TrTriangle  * toadd);
TlErr        TlAddItem(TeEvent     * toadd);
TlErr        TlAddItem(TlList      * toadd);
TlErr        TlAddItem(TlItem      * toadd);
TlErr        TlAddItem(TrId          toadd);
```

```
TlErr            TlAddItem(TrEdge      * toadd);


TlErr            TlInsertItem(TrTriangle * toadd, int index);
TlErr            TlInsertItem(TeEvent    * toadd, int index);
TlErr            TlInsertItem(TlList     * toadd, int index);
TlErr            TlInsertItem(TlItem     * toadd, int index);
TlErr            TlInsertItem(TrId         toadd, int index);
TlErr            TlInsertItem(TrEdge     * toadd, int index);


TlErr            TlRemoveItem(int index);


int              TlGetIndex(TlItem *item);
TlItem*          TlGetByIndex(int index);


TlItem*          TlGetNext();
TlItem*          TlGetNext(TlItem *i);
TlItem*          TlGetNext(int index);


TlItem*          TlGetPrev();
TlItem*          TlGetPrev(TlItem *i);
TlItem*          TlGetPrev(int index);


TlItem*          TlGetCurrent();
int              TlGetCurrentIndex();
TlErr            TlSetCurrent(int index);
TlErr            TlSetCurrent(TlItem *i);
```

TlList Related Data Types:

enum TlType

This enum specifies the various types of lists and hence the types of the TlItems
that make up those lists. Lists are homogeneous, so that if an attempt is made to
add, insert, etc... an item of a different **TlType**, that operation will return a TlErr of

"TL_TYPE_MISMATCH". **Note:** This is different from TeType, which specifies a type of TeEvent.

```
enum tltype {

TL_NOTYPE,                  no type specified

TL_LIST,                    list whose TlItems point to other TlLists

TL_TRIANGLE,                list whose TlItems point to TrTriangles

TL_EVENT,                   list whose TlItems point to TeEvents

TL_ID,                      list whose TlItems contain TrIds

TL_EDGE                     list whose TlItems point to TrEdges

} TlType;
```

enum TlErr

This enum specifies various return values for certain TlList functions.

```
enum tlerr {

TL_NO_ERROR,                the operation was successful

TL_TYPE_MISMATCH,           the TlItem specified was of a different
                                    TlType than its TlList

TL_INDEX_OUT_OF_RANGE       the TlList does not have an item
                                    corresponding to the index
                                    specified

} TlErr;
```

TlList Methods:

TlType TlGetType();

Gets the type of the list.
Arguments: none.
Returns: TlType

int TlGetSize();

Gets the number of items in the list.
Arguments: none.
Returns: int.

TlErr TlAddItem(TrTriangle * toadd);

TlErr TlAddItem(TeEvent * toadd);

TlErr TlAddItem(TlList * toadd);

TlErr TlAddItem(TlItem * toadd);

TlErr TlAddItem(TrId toadd);

TlErr TlAddItem(TrEdge * toadd);

Methods for adding items to a list. They perform error checking to ensure that the
list is of the same type (TlType) as the item being added.
Arguments: Pointer to a TrTriangle, TeEvent, TlItem, TlList, or TrEdge object, or
a TrId.
Returns: TlErr, (its value will be TL_NOERROR if it was successful)

TlErr TlInsertItem(TrTriangle * toadd, int index);

TlErr TlInsertItem(TeEvent * toadd, int index);

TlErr TlInsertItem(TlList * toadd, int index);

TlErr TlInsertItem(TlItem * toadd, int index);

TlErr TlInsertItem(TrId toadd, int index);

TlErr TlInsertItem(TrEdge * toadd, int index);

Methods for inserting items into a list. They perform error checking to ensure that
the list is of the same type (TlType) as the item being inserted.
Arguments: Pointer to a TrTriangle, TeEvent, TlItem, TlList, or TrEdge object, or
a TrId; int specifying where in the list to add the specified item.
Returns: TlErr, (its value will be TL_NOERROR if it was successful)

TlErr TlRemoveItem(int index);

Removes the item at specified index.
Arguments: int specifying zero-based index of item to remove from the list.
Returns: TlErr, (its value will be TL_NOERROR if it was successful)

int TlGetIndex(TlItem *item);

Gets the zero-based index of an item.
Arguments: Pointer to a TlItem whose index is desired.
Returns: integer index of TlItem specified, or -1 if the item is not in the list.

TlItem* TlGetByIndex(int index);

Gets the item at the index specified.
Arguments: int specifying zero-based index of the desired item.
Returns: Pointer to a TlItem of the same TlType as the list, which in turn contains a pointer to the desired object.

TlItem* TlGetNext();

TlItem* TlGetNext(TlItem *i);

TlItem* TlGetNext(int index);

Methods for getting the next item in a list. If an argument is given, the item returned will be the next item *after the one specified.*
Arguments: int specifying the index of the item from which the next item will be retrieved, or a pointer to a TlItem specifying the same.
Returns: Pointer to a TlItem of the same TlType as the list, which in turn contains a pointer to the desired object.

TlItem* TlGetPrev();

TlItem* TlGetPrev(TlItem *i);

TlItem* TlGetPrev(int index);

Methods for getting the previous item in a list. If an argument is given, the item returned will be the item *before the one specified.*
Arguments: int specifying the index of the item whose next item will be retrieved, or a pointer to a TlItem specifying the same.
Returns: Pointer to a TlItem of the same TlType as the list, which in turn contains a pointer to the desired object.

```
TlItem* TlGetCurrent();
```

Gets the current item in the list.
Arguments: none.
Returns: Pointer to a TlItem of the same TlType as the list, which in turn contains a pointer to the desired object.

```
int TlGetCurrentIndex();
```

Gets the index of the current item in the list.
Arguments: none.
Returns: The zero-based index of the current list item.

```
TlErr TlSetCurrent(int index);
```

```
TlErr TlSetCurrent(TlItem *i);
```

Sets the current item in the list.
Arguments: int specifying the zero-based index of the item to make current, or pointer to a TlItem to make current.
Returns: TlErr, (its value will be TL_NOERROR if it was successful)

TlItem

This class represents an item that can be stored in a linked list. An item can be another TlList, a TrTriangle, a TrEdge, a TrId, or a TeEvent, as defined by the TlType of the list.

constructors:

```
TlItem(TrTriangle *t);
TlItem(TeEvent *e);
TlItem(TlList *l);
TlItem(TrId t);
TlItem(TrEdge *e);
~TlItem();
```

related data type index:

```
enum  TlType
enum  TlErr
```

method index:

```
TlType        GetType();


TrTriangle *  GetTriangle();
TeEvent *     GetEvent();
TlList *      GetList();
TrId          GetId();
TrEdge *      GetEdge();
```

TlItem Related Data Types:

See: `TlList Related Data Types`

TlItem Methods:

TlType GetType();

Gets the type of this object, so that its content can be properly retrieved using
GetTriangle, **GetEvent**, etc.
Arguments: none.
Returns: A TlType specifying the type of information referenced by the item.

TrTriangle * GetTriangle();

TeEvent * GetEvent();

TlList * GetList();

TrId GetId();

TrEdge * GetEdge();

Methods to retrieve the actual information from a TlItem.
Arguments: none.
Return: Either a TrId, or a pointer to a TrTriangle, TeEvent, TlList or TrEdge
object, depending on the contents of the item. **Note:** This function handles errors by
returning Null (or -1 in the case of *GetId()*) and sending a message to cout if an
inappropriate get operation is attempted. Be careful to first check the TlType of the
item before using this function.

TrTriangle

This class represents a triangle, which consists of an ID, a list of TeEvents relevant
to this triangle, and 3 TrEdges.

constructors:

```
TrTriangle(TrId myid);

~TrTriangle();
```

related data type index:

```
int          TrId
class        TrEdge
```

method index:

```
TrId         GetId();
TlList *     GetEventList();
```

```
TrEdge *        GetEdge(int whichedge);
```

TrTriangle Related Data Types:

int TrId

This typedef provides an int (8-bit) unique ID for each triangle. Each triangle broadcasts its hard-coded unique ID to the TsSystem, which assigns the IDs to corresponding **TrTriangle** objects in their constructor. **TrId**s can be stored and manipulated in TlLists as can pointers to the **TrTriangle**s themselves, by ensuring that the TlList has the appropriate TlType.

TrTriangle Methods:

TrId GetId();

Gets the ID of the triangle object. Generally, this will correspond with the hard-coded ID of the physical triangle's microprocessor.
Arguments: none.
Returns: TrId equal to the ID of the triangle.

TlList * GetEventList();

Gets a pointer to a list containing all of the TeEvents in which this triangle has been involved since the TsSystem::Init() call.
Arguments: none.
Returns: TrId equal to the ID of the triangle.

TrEdge * GetEdge(int whichedge);

Gets a pointer to an edge object which contains information about the which triangle (and which edge of that triangle) is connected to the specified edge of this triangle. **Note:** Only accepts values between zero and two (edges 1-3). Otherwise, returns NULL.
Arguments: none.
Returns: TrEdge pointer containing the id and edge of the triangle connected to the specified edge, or NULL if the specified edge is out of range (either less than zero or greater than two).

TrEdge

This class represents a triangle's ID/Edge pair, and is used in referring to a specific edge of a specific triangle.

This class represents a triangle's ID/Edge pair, and is used in referring to a specific edge of a specific triangle.

constructors:

```
TrEdge();
TrEdge(TrId t, int e);
~TrEdge();
```

related data type index:

```
int          TrId
```

method index:

```
void         SetId(TrId t);
TrId         GetId();
void         SetEdge(int e);
int          GetEdge();
```

TrEdge Related Data Types:

See: `TrTriangle Related Data Types`

TrEdge Methods:

Bool SetId(TrId t);

Associates a triangle ID with this edge object.
Arguments: the TrId, of this edge's triangle.
Returns: True if the operation was successful, False if not.

TrId GetId();

Gets the ID of the triangle associated with this edge object.
Arguments: none.
Returns: the TrId of this edge's triangle; -1 if no TrId has been assigned yet.

Bool SetEdge(int e);

Sets the edge (0, 1, or 2) associated with this edge object.
Arguments: an int (0, 1 or 2) describing which edge of a TrTriangle is associated with this object.
Returns: True if the operation was successful, False if it was not (i.e. if the value passed was out of range).

int GetEdge();

Gets the zero-based edge number of the triangle associated with this edge object.
Arguments: none.
Returns: the number of the edge which this object represents (0, 1, or 2); -1 if no edge has been assigned yet.

TeEvent

This class represents an event that can occur to a triangle: either a connection or a disconnection. It stores the time the event occurred, and the 2 id/edge pairs (TrEdges) relevant to the event.

constructors:

```
TeEvent(TeTime *ti, TeType ty);
```

related data type index:

```
enum            TeType
class           TeTime
```

method index:

```
TeTime *        GetTime();
TeType          GetType();
TrEdge *        GetEdge(int which);
```

TeEvent Related Data Types:

enum TeType

This enum provides named types for the different kinds of events that can occur.
Note: This is different from TlType, which specifies a type of TlList.

```
enum tetype {
TE_NOTYPE,                  no type defined yet
TE_CONNECT,                 this is a connection event
TE_DISCONNECT               this is a disconnection event
} TeType;
```

TeType Methods:

TeTime * GetTime();

Gets the time at which the event occurred.
Arguments: none.
Returns: a TeTime object describing the time at which the event occurred.

TeType * GetType();

Gets the type of this event.
Arguments: none.
Returns: a TeType describing what kind of event this was.

TrEdge * GetEdge(int which);

Gets the specified id/triangle pair (either zero or one). **Note:** Only accepts zero or one as arguments. Otherwise, returns NULL.
Arguments: an int (either 0 or 1) denoting which of the two TrEdge objects to return.
Returns: a TrEdge object specifying one of the triangle/edge pairs involved in this event; NULL if the index passed was out of bounds.

TeTime

This class represents a quantity of time. It is used to time-stamp TeEvents.

constructors:

```
TeTime(int d, int h, int m, int s, int ms);

TeTime();

~TeTime();
```

method index:

```
static TeTime *     now();

Bool          closeTo(TeTime *t,int threshold);
```

TeTime Methods:

static TeTime * now();

This *static* function returns a TeTime object representing the current time.
Arguments: none.
Returns: a **TeTime** object that represents the present.

Bool closeTo(TeTime *t, int threshold);

This method checks the calling object against another **TeTime** object to see if they

occurred within a certain threshold.

Arguments: A **TeTime** object to compare, and a number of milliseconds to use as

the threshold.

Returns: A Bool: True if the specified instant is within the threshold of the object,

and False if it is not.

Image Credits

All of the photographs and illustrations in this document were created by the author except:

P. 16:   From [Spa93], P. 15

P. 17:   From [Man86], P. 173

P. 21:   From [Ald90], P. 47

P. 22:   From [Ald90], P. 93

P. 23:   From [Ald90], P. 106

P. 31:   Photo ©1997 Web Chapell

P. 32:   From http://www.dgp.toronto.edu/people/GeorgeFitzmaurice/papers/gwf_bdy.html
                (as of June 8, 1998)

P. 33 (top):   From [Cra95], P. 63

P. 34 (top):   From [Ull97], P. 225

P. 34 (bot):   From http://www/people/dsmall/images/web.jpg
                (as of June 8, 1998)

P. 40:   From http://www.mot.com/GSS/CSG/Europe/English/Promotion/index.html
                (as of June 8, 1998)

P. 49 (bot):   From [McCl93], P. 31

P. 60:   Photo ©1997 rubra

P. 72:   From [Wis98a], P. 25

References

[Ald90]   Aldersey-Williams, Hugh. (1990). The Mannerists of MicroElectronics. *The New Cranbrook Design Discourse*. New York: Rizzoli, pp. 20-26.

[Bar96]   Barfield, W., and Danas, E. (1996). Comments on Olfactory Displays for Virtual Environments, *Presence: Teleoperators and Virtual Environments*, 5 (1).

[Blu95]   Blumberg, B. and T. Galyean (1995). Multi-level Direction of Autonomous Creatures for Real-Time Virtual Environments. *Proceedings of ACM SIGGRAPH 95*.

[Bra94]   Brand, Stuart. (1994). *How Buildings Learn: What Happens After They're Built?* Viking Press.

[Bux83]   Buxton, W. (1983). Lexical and Pragmatic Considerations of Input Structures. *Computer Graphics*, 17(1), pp. 31-37.

[Bux86]   Buxton, W. (1986). Chunking and Phrasing and the Design of Human-Computer Dialogues, *Proceedings of the IFIP World Computer Congress*, Dublin, Ireland, pp. 475-480.

[Bux96]   Buxton, W. (1996). Absorbing and Squeezing Out: On Sponges and Ubiquitous Computing, *Proceedings of the International Broadcasting Symposium*, November 13-16, Tokyo, 91-96.

[Car90]   Card, S. K., Mackinlay, J.D and G. G. Robertson. (1990). The Design Space of Input Deices. *Proceedings of ACM CHI'90*, New York: ACM Press, pp. 117-124.

[Cra95]   Crampton Smith, G. (1995). The Hand That Rocks the Cradle. *I.D., May/June 1995*, pp. 60-65.

[Dal95]   Dallas Semiconductor. (1995). *Automatic Identification Data Book*. Dallas, TX: Dallas Semiconductor.

[Dan97]   Danisch, L.A., "Fiber optic bending and positioning sensor," US Patent No. 5,633,494, May, 1997.

[Dob96]   Doblin Group. (1995). Breakthroughs. *Design Quarterly 167*, Winter 1996. Cambridge, MA: MIT Press.

[Eam78]   Eames, Charles and Ray Eames. (1978). *Powers of Ten.* (Video) IBM Coroporation/Office of Charles and Ray Eames.

[Fitz93]   Fitzmaurice, G. (1993). Situated Information spaces and Spatially Aware Palmtop computers. *Communication of the ACM*, July 1993, V. 36(7). pp 38-49.

[Fitz95]   Fitzmaurice, G., Ishii, H., and Buxton, W. (1995). Bricks: Laying the Foundations for Graspable User Interfaces. *Proceedings of ACM CHI'95*, New York: ACM Press, pp. 442-449.

[Fitz96]   Fitzmaurice, G. Graspable User Interfaces. Ph.D.Thesis, University of Toronto, 1996.

[Fle97]   Fletcher, R. (1997). *A Low-Cost electromagnetic Tagging Technology for Wireless Identification, Sensing, and Tracking of Objects*, Masters' Thesis, Program in Media Arts and Sciences, MIT Media Laboratory, Cambrige, MA.

[Gor98]   Gorbet, M. Orth, M., and Ishii, H. (1998). Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography. *Proceedings of ACM CHI '98*, New York: ACM Press, pp. 49-56.

[Heck97]   Heckert, Matt (1997). Munich Samba. *FleshFactor: Informationsmacschine mensch (1997 Ars Electronica Festival Catalog)*, New York: SpringerWeinNewYork, p. 311.

[Ish97]   Ishii, H. and Brygg Ullmer. (1997). Tangible Bits: Towards Seamless Interfaces Between People, Bits, and Atoms. *Proceedings of ACM CHI '97*, pp. 234-241.

[Jer96]   Jeremijenko, N. (1996). Voice Boxes. *Memesis (1996 Ars Electronica Festival Catalog)*, New York: SpringerWeinNewYork, p. 403.

[Joh87]   Johnson, Mark. (1987). *The Body in the Mind.* Chicago: The University of Chicago Press.

[Joh89]   Johnson, J., Roberts, T., Verplank, W., et al. (1989). The Xerox Star: A Retrospective. *IEEE Computer* 22(9), September 1989, pp. 11-29.

[Kne95]   Knep, B. et al. (1995). Dinosaur Input Device, *Proceedings of ACM CHI'95*, New York: ACM Press, pp. 304-309.

[Lak80]   Lakoff, George and Mark Johnson. (1980). *Metaphors We Live By.* Chicago: The University of Chicago Press.

[Lak87]   Lakoff, George. (1987). *Women, Fire, and Dangerous Things.* Chicago: The University of Chicago Press.

[Lip93]   Lipscomb, James S. and Michael E. Pique. (1993). *Analog Input Device Physical Characteristics.* ACM SIGCHI Bulletin, 25(3), pp. 40-48.

[Mac93]   Macann, Christopher (1993). *Four Phenomenological Philosophers: Husserl, Heidegger, Sartre, Merleau-Ponty.* New York: Routledge.

[Mae93]   Maeda, John and Kevin McGee (1993). Dynamic Form. Japan: International Media Research Foundation.

[Man86]   Manzini, Ezio (1986). *The Material of Invention.* Cambridge, MA: MIT Press.

[McCl93]   McCloud, Scott. (1993). *Understanding Comics: The Invisible Art.* New York: HarperCollins.

[Mei97]   Meickle, Jeffrey L. (1995). *American Plastic: A Cultural History.* London: Rutgers University Press.

[Mer62]   Merleau-Ponty, Maurice. (1962). *Phenomenology of Perception* (C. Smith, Trans.). New York: Humanities Press.

[Neg95]   Negroponte, Nicholas. (1995). *Being Digital.* New York: Alfred A. Knopf.

[Orth97]   Orth, M. and Gorbet, M. (1997). Triangles and The Digital Veil. *FleshFactor: Informationsmacschine mensch (1997 Ars Electronica Festival Catalog)*, New York: SpringerWeinNewYork, pp. 280-283.

[Pap80]   Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books.

[Red79]   Reddy, Micheal J. (1979). The Conduit Metaphor: A Case of Frame Conflict in our Language About Language. *Metaphor and Thought* (A. Ortony, Ed.). Cambridge, England: Cambridge University Press, Ch. 10 (2nd ed.)

[Res93]   Resnick, M. (1993). Behavior Construction Kits. *Communications of the ACM*, 36(7), July 1993, pp. 64-71.

[Res97]   Resnick, M. (1997). *Turtles, Termites and Traffic Jams.* Cambridge, MA: MIT Press.

[Res98]   Resnick, M. et al. (1998). Digital Manipulatives: New Toys to Think With. *Proceedings of ACM CHI'98*, pp. 281-287.

[Rhein91]   Rheingold, Howard. (1991). *Virtual Reality.* New York: Summit Books.

[Sch92]   Schneiderman, Ben (1992). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. New York: Addison-Wesley Publishing Company.

[Sha49]   Shannon, Claude E. and Warren Weaver (1949). *The Mathematical Theory of Communication*. Urbana: University of Illinois Press.

[Sma96]   Small, David. (1996). Navigating Large Bodies of Text. *IBM Systems Journal* 35(3&4), 1996, pp. 516-525.

[Spa93]   Sparkle, Penny (Ed.) (1993). *The Plastics Age.* New York: Overlook Press.

[Suz93]   Suzuki, H. and H. Kato (1993). AlgoBlock: a Tangible Programming Language, a Tool for Collaborative Learning. *Proceedings of the 4th European Logo Conference,* August 1993. pp. 297-303.

[Svan98]   Svanæs, Dag. *Understanding Interactivity.* Draft monograph of upcoming book.

[Ull97]   Ullmer, B. and Ishii, H. (1997). The metaDESK: Models and Prototypes for Tangible User Interfaces. *Proc. of UIST '97*, ACM, October 1997, pp. 223-232.

[Und98]   Underkoffler, J. and Ishii, H. (1998). Illuminating Light: An Optical Design Tool with a Luminous-Tangible Interface. *Proceedings of ACM CHI '98.* New York: ACM Press, pp. 542-549.

[Win86]   Winnograd, T., and F. Flores. (1986). *Understanding Computers and Cognition.* Reading, MA: Addison-Wesley, 1986.

[Win96]   Winnograd, T. (ed.) (1996). *Bringing Design to Software.* New York: ACM Press.

[Wis98a]   Wisneski, Craig et al. (1998). Ambient Displays: Turning Architectural Space Into an Interface Between People and Digital Information. *Proceedings of International Workshop on Cooperative Buildings*, GMD, Darmstadt, February 1998, pp. 22-32.

[Wis98b]   Wisneski, C., Orbanes, J., and Ishii, H. (1998). PingPongPlus: Augmentation and Transformation of Athletic Interpersonal Interaction. *Conference Summary of ACM CHI '98*, ACM Press, pp. 327-328.

[Zim95]   Zimmerman, T. et al. (1995). Applying Electric Field Sensing to Human-Computer Interfaces. *Proceedings of ACM CHI'95*. New York: ACM Press, pp. 280-298.