1

A History of Interaction

It is a truism that computers are becoming faster and more powerful all the time. They play an ever larger role in our lives, giving us access to more and more information, being incorporated into more and more of our devices, and creating whole new forms of interaction and activity that we would never otherwise have imagined. From desktop computers to laptops to personal digital assistants, not to mention bank teller machines, microwave ovens, cellular telephones, and ticket machines, we encounter computers in all aspects of everyday life. The ever-expanding province of computation is a commonplace, the topic of a million coffee-shop conversations, television reports, and newspaper headlines. We talk about how fast it is changing, but we talk much less about the ways in which it is not. Many things about computers are not changing at all. Our basic ideas about what a computer is, what it does, and how it does it, for instance, have hardly changed for decades. Nor have the difficulties we encounter actually using computers.

Our experience using computers reflects a trade-off that was made fifty years ago or more. When computers were first being developed commercially, they were extremely expensive devices. Computer time was much more expensive than your time or mine. In that context, efficiency dictated that we minimize the amount of computer time any job or activity needed, even if that meant burdening the people who wanted to submit the job. If a rigid, formalized input language was easier for the system to process, for example, then the cost in people's time to format their data in that language was more than offset by the savings in processing time that would result. Because most uses of computers were

military and commercial rather than personal, it was hard to disagree with this sort of economic argument. It gave rise to a model that favors performance over convenience, and places a premium on the computer's time rather than people's time. This model is still with us today.

However, in light of those commonly observed transformations in computer power, we are now in a position to reconsider the trade-off. Arguably, we *must*. Computers are now so much faster and more powerful, giving us access to so much more information that we are simply no longer able to manage and assimilate it. At the same time, those powerful computers spend 95 percent of their time doing absolutely nothing. Modern personal computers perform very few tasks that use their full capacity for longer than a second or two. Outside these brief bursts of activity, most of the time they do nothing at all, generally while we try to figure out what to make of what just happened or what we want to do next.

At the same time, we increasingly see computers incorporated into devices other than the traditional PC sitting on the desk. Computation is part of your cellular telephone, your microwave oven, your car, and a host of other technologies. The rise of so-called embedded computing reflects the fact that computation can be usefully harnessed for more than just traditional desktop computing. It can also help us as we get up and move about in the world, which we generally do more of than sitting at desks (or would, if the computers didn't shackle us to them). However, this new form of computation exacerbates the effects of the trade-off between the work that the user and the system do. As I sit at my desktop computer, it occupies the whole of my attention; but that would be a terrible idea in a computer I'm using while driving, or crossing the street, or trying to enjoy a conversation with friends.

These two trends—the massive increase in computational power and the expanding context in which we put that power to use—both suggest that we need new ways of interacting with computers, ways that are better tuned to our needs and abilities. Over the last few years, research into Human-Computer Interaction (HCI) has begun to explore ways to control and interact with a new breed of computer systems. Prototype systems have been developed; new forms of interaction explored; new research groups established; new designs developed and tested.

This book is a contribution to the emerging literature on this new approach to interacting with computers, one that I call "Embodied Interaction." Embodied Interaction is interaction with computer systems that occupy our world, a world of physical and social reality, and that exploit this fact in how they interact with us.

There are two ways in which the material I want to present in this book differs from other explorations in HCI. The first difference concerns the set of entities that will appear here. In particular, although computer interfaces are the general topic, interfaces themselves will not appear too often. Here, I am more concerned with interaction than I am with interfaces, and more concerned with computation than I am with computers. When I say that I am more concerned with interaction than with interfaces, I mean that I will be dealing with the ways in which interactive systems are manifest in our environment and are incorporated into our everyday activities, rather than with the specific design of one user interface or another. Similarly, when I say that I am more concerned with computation than with computers, I mean that I want to address the idea of computation per se—of active representations embodied in hardware and software systems—rather than the specific capabilities of systems available at the start of the new millennium. So, gigabytes and megahertz will not be at issue, but representational power will be.

The second difference is in the way that those topics will be addressed. In particular, as you might guess on the basis of my concern with interaction and computation, I want to address a set of topics that are more foundational than technical. This is not a source book of design solutions, or a how-to manual for interface developers—although these practical matters will certainly arise, and I hope that designers will find something useful here. In fact, the very reason for exploring foundations is to support the design and evaluation of new systems, tools, and interaction modalities. The goal of this foundational exploration is to provide resources to designers and system developers, by giving them tools they can use to understand and analyze their designs.

Traditionally, the central component of any account of computation has been algorithms or procedures—step-by-step models that specify the sequential behavior of a computer system. In turn, because they are based on an analogy between mental phenomena and computation,

cognitive science and AI have also predominantly espoused a step-by-step model of procedural execution. In the last few years, though, this procedural approach has been challenged by a new conceptualization of computational phenomena that places the emphasis not on *procedures* but on *interaction* (Wegner 1997). Interactional approaches conceptualize computation as the interplay between different components, rather than the fixed and prespecified paths that a single, monolithic computational engine might follow. These models of computation have more in common with ecosystems than with the vast mechanisms we used to imagine. They emphasize diversity and specialization rather than unity and generality. Perhaps there is, in this, something of the spirit of the times; perhaps, too, the rise of new computational paradigms such as parallel systems, object-oriented programming, and Internet-style software design is implicated in this change. The change, though, has occurred across a wide range of areas of computational investigation. It has affected how we think about computation from a mathematical perspective, leading to new theoretical accounts of systems such as Hoare's CSP (Hoare 1985) or Milner's work on CCS and the Pi Calculus (Milner 1995, 2000); it has affected how we think about computational models of mind, as reflected by Minsky's "Society of Mind" (Minsky 1988), Agre's critique of computational reasoning (Agre 1997), or Brooks's approach to robotics (Brooks 1999); and it has led to new accounts of the practice of programming (Stein 1998).

You might think that studies of how people use computers must always have been built around a model of the world that gives pride of place to interaction, but in fact HCI has traditionally been built on a procedural foundation. HCI, from its very beginning, took on the trappings of the traditional computational model and set out its account of the world in terms of plans, procedures, tasks, and goals. In contrast, the model of HCI I set out here is one that places interaction at the center of the picture. By this I mean that it considers interaction not only as *what* is being done, but also as *how* it is being done. Interaction is the means by which work is accomplished, dynamically and in context.

Some background will help to clarify what this means and to set the stage for the argument this book will develop. The context is the historical evolution of the idea of interaction and the technology of HCI.

A Historical Model of Interaction

Just as computers have evolved considerably in their short history, so have styles of human-computer interaction. There are many ways to conceptualize the history of interaction with computer systems. The purely technological view, for example, would recount the history of the input and output devices that have characterized different stages of interface development, and would describe their computational demands. A political view would consider the movement of ideas from one laboratory to another as researchers respond to the demands and interests of funding agencies and so forth, while an economic view would consider how user interface development has influenced, and been influenced by, the growth of the high-tech industry and PC economy. Grudin (1990) describes the history of interaction as the story of the "computer reaching out," in which interaction moves from being directly focused on the physical machine to incorporating more and more of the user's world and the social setting in which the user is embedded. Although Grudin's analysis is now a decade old, it is interesting to see the ways in which later trends in HCI design—including some that are of particular interest in this book—have followed quite closely the directions that he laid out.

I want to explore a slightly different view here, in order to set some context for the discussion that will follow. In particular, I want to present the stages in the historical development of user interfaces in terms of the different sets of human skills they are designed to exploit. This is not a different history of HCI, of course, but merely a different telling of the history, with the emphasis in a slightly different place. As is perhaps appropriate for a discipline that concerns itself as much with human abilities as with technological opportunities, it draws attention to the human experience of computation. The are four separate phases of development to discuss. I characterize them as electrical, symbolic, textual, and graphical forms of interaction.

Electrical

Today, when we talk of "computers," we invariably mean digital devices. The computer as we know it is inescapably bound up with the ones and zeros of digital logic. It was not always this way. Originally, the

word "computers" referred to human beings—people whose daily work was the figuring of calculations, such as for producing engineering tables. However, even when "computers" became electronic devices, they were not necessarily digital ones. Before digital computers came analog computers. Analog computers did not rely on the discrete logic that characterizes modern computing devices; instead, they relied on the use of standard components such as resistors and capacitors to create electronic models of continuous natural phenomena (such as wave motion, the interaction of electronic forces, or the movements of objects under gravity). Essentially, the analog computer was the apparatus for laboratory simulations that took place not in the physical world, but in an analogous electronic reality. To set up a new experiment, the machine would have to be reconfigured, possibly quite radically, through the incorporation of new circuits. This task-specificity was shared by the early digital computers, too. Even after we had made the move from analog electronics to digital logic, the earliest digital computers were special purpose devices, designed as automatic calculators to solve specific problems—often, inevitably, in military domains (such as calculating missile trajectories or exploring patterns in coded messages).

Although there is some debate about precisely who was the first to make the move—perhaps Eckert and Mauchley with EDVAC in Philadelphia, or Williams and Kilburn building the Small-Scale Experimental Machine, known as "Baby," at Manchester, or one of the other contenders—what *is* generally accepted is that the critical development in digital computing was that of the *stored program computer.* In contrast to earlier designs, a stored program computer is a machine whose operation is not directly encoded in its circuits, but rather is determined by a sequence of instructions held in its memory—instructions that can, clearly, be changed or replaced much more easily than the electrical circuits could be reconfigured. Nonetheless, the first age of computing, around the time that this transition took place, relied heavily on an understanding of the electronics that made up any given machine. Every machine was a prototype; every program, uniquely designed for a specific computer (and perhaps even a specific version or configuration of that computer). What we currently refer to as "instruction sets"—the set of low-level operations that processors such as the Pentium or PowerPC can understand—were, at

that stage in the history of computation, intimately tied to the individual details of the circuitry of any particular computer. So, even as we made the transition from hardware configuration to digitally stored programs, the dominant paradigm for interaction with the computer was electronic. Entering a new program, even if that program was to be stored digitally in the memory of the computer, could still bear a remarkable resemblance to electronic reconfiguration, involving plugboards and patch cables. Indeed, such programming activity was often accompanied with the development of new circuits that could extend the operation of the system. The boundary that we now take for granted between hardware and software was much fuzzier then; interacting with the system, and developing new programs, relied on a thorough understanding of the electronic design.

Symbolic

The next stage of development is characterized by the emergence of symbolic forms of interaction. The movement from one stage to another is not a sudden and clear transition; instead, it is a general trend that emerges in a number of different ways. We can see it in the basic models offered for programming systems, which was the primary form of interaction between human and computer at a time when "users" as we now know them did not yet exist.

As the transition from electrical to symbolic approaches gradually took hold, programming computers came to require less understanding of the detailed construction of each particular machine, and relied increasingly on regularized and well-understood capacities that would be available across a wide range of machines—register files, index registers, accumulators, and so forth. At the same time, the primary form of programs moved from a numeric form (that is, the "machine language" of raw instructions that a machine would understand) to other symbolic forms that were more readily understandable to human beings. So-called assembly languages are essentially symbolic forms of machine language, using mnemonic codes that stand in one-to-one correspondence with the machine level instructions, so that a sequence of instruction codes such as "a9 62 82 2c" is rendered as a symbolic expression such as "movl (r1+), r2."[1]

Since assembly languages are simply a different rendering of machine languages—symbolic forms that describe sets of specific instructions—they are just as tied as machine languages to particular systems, although, by this stage, computer systems were being produced industrially rather than developed as one-off prototypes in laboratories. But they are in no way portable between machines of different sorts, even today—assembly programs for an Intel processor yield machine instructions that will run only on Intel processors, and not on other processors made by Motorola. A further progression along the symbolic path, though, came with the development of the early programming languages such as LISP and FORTRAN. Essentially, these lay down two sets of rules. The first set describes what structural properties a set of instructions will have to be valid programs—what rules must be followed when creating something that is a FORTRAN program rather than simply gibberish. The second describe how programs can be turned into a set of (machine language) instructions for the computer to execute. The important point is that, whereas programs would previously be specified with relation to a specific machine language (perhaps encoded as assembly instructions, but still tied to a particular sort of computer), the programmer's activity was now lifted to a more abstract level that was simultaneously a more natural form of expression and independent of the precise details of any specific computer, its implementation and configuration.

The introduction of programming systems such as assemblers and programming languages moved computer interaction, then, from an electronic level to a symbolic one. It introduced a set of symbolic representations of computer system operation as the primary modality by which interaction was conducted. Interestingly, this was also reflected in the physical interaction with systems. Punched cards, for example, can be regarded as a primitive form of symbolic interaction, especially because punched card systems quickly came to incorporate both *data* cards (that is, cards that carried information for programs to process) and *control* cards (instructing the system to begin and end jobs, etc.) The control cards, then, provide a symbolic language for controlling the behavior of the system.

The reason I want to cast the history of interactive computing in terms of these different sorts of interaction modalities is that it draws our attention to the fact that they exploit quite different sets of skills. We are all highly skilled at various forms of symbolic interaction; language and communication, for us, are largely symbolic in nature, whether these symbols take the forms of icons, traffic signs, flags, maps, or marks on paper. Symbolic interaction is a much more natural and intuitive form of interaction for us than the electronic form that had previously been necessary; and it allows us to bring to bear a much more powerful set of intuitions and abilities to the interactive task. So, finding errors in assembly language programs is much less error-prone than trying to do the same in machine language; and debugging programs written in so-called high level languages is easier still (although, as any programmer will tell you, it is still the most time-consuming and intricate part of the process of developing software). We are generally able to exploit a greater range of skills—visual, cognitive, and so on—as we move from electrical to symbolic forms of interaction.

Textual

The best-developed form of symbolic interaction with which we are familiar is, of course, written language and textual interaction. So it is only natural that symbolic interaction with computers should gradually extend into the textual domain.

Of course, most of the examples I provided for symbolic interaction were textual in nature, one way or another. For my purposes, a distinction can be made between symbolic and textual interaction by looking at the *actual interaction* with the computer. So, although programs written in assembly language are clearly textual, the form in which they arrive at the computer might not be textual at all, but might be encoded on punched cards or other symbolic media. However, the modes of interaction with technology are continually shifting as technology develops and new opportunities present themselves, and before long the primary form of direct interaction with computers was, indeed, textual interaction, at teletype machines and video terminals.

When this transition took place, textual interaction was no longer simply a means to describe computer operations, but became the primary

form of interaction. Arguably, this is the origin of "interactive" computing, because textual interfaces also meant the appearance of the "interactive loop," in which interaction became an endless back-and-forth of instruction and response between user and system. Even in these days of graphical and virtual reality interfaces, this model is still often the only recourse for some operations.

One reason that textual interaction remains so powerful is that it draws not only on the use of textual characters but on how those characters can be combined into words and sets of words. In other words, along with textual interaction came a "grammar" of interaction, one that broke input text into commands, parameters, arguments, and options. So, just as the move from electrical to symbolic interaction meant that interface designers could draw upon a new set of human skills and abilities, so too did textual interaction. Textual interaction can draw on our linguistic skills, not by letting us simply "talk" to computers (at least, outside of science fiction films), but rather by drawing on our abilities to create meaningful sentences by combining elements each of which contributes to the sense of the whole.

The compositional character of textual interaction has proven hard to replace as interfaces have developed. The value, as we will see, of later interaction modalities such as graphical user interfaces is that they make the abstract entities of computation into "real," individuable objects supporting direct interaction. However, because our programs are still constructed in terms of abstract entities, textual interaction still proves its value by giving us the ability to create instructions that operate in terms of generalities—loops, conditions, patterns, and more.

The other significant feature of the textual interface paradigm is that it brought the idea of "interaction" to the fore. Textual interaction drew upon language much more explicitly than before, and at the same time it was accompanied by a transition to a new model of computing, in which a user would actually sit in front of a computer terminal, entering commands and reading responses. With this combination of language use and direct interaction, it was natural to look on the result as a "conversation" or "dialogue." These days, this idea of dialogue is central to our notion of "interaction" with the computer, replacing configuration, programming, or the other ideas that had largely characterized the interplay

between users and systems in the past. So, although the notion of "interaction" with computers had important predecessors before this period—such as Ivan Sutherland's hugely influential work on Sketchpad (Sutherland 1963)—it was arguably from the paradigm of text-based dialogue that people drew the idea of "interacting with the machine." And interacting was something that we already knew how to do.

Graphical

Probably the most significant transition, in terms of the development of the user interface models that are familiar to us today, was the transition from textual to graphical interaction. Graphical interaction developed from the work of many people, including Sketchpad on the TX-2 (Sutherland 1963), and the work of Alan Kay and his colleagues at PARC, based in turn on the developmental psychology of Piaget, Bruner, and others (Kay 1993).

Just as the move from symbolic to textual interaction did more than simply replace one symbolic language with another, the move from textual to graphical interaction did not simply replace words with icons, but instead opened up whole new dimensions for interaction—quite literally, in fact, by turning interaction into something that happened in a two-dimensional space rather than a one-dimensional stream of characters. Traditional textual interaction took place at teletype machines or serial terminals, where information appeared at the bottom of the screen and scrolled up to disappear off the top. The user's input and the system's output together formed a single stream of information, arranged linearly, character by character. In contrast, graphical interaction is characterized by its use of space; information is spread out over a larger screen area, so that the locus of action and attention can move around the screen from place to place or can even be in multiple places simultaneously (e.g., in different windows). The task of managing information becomes one of managing space.

Moving from one-dimensional to two-dimensional interaction made it possible, again, to exploit further areas of human ability as part of the interactive experience. These included:

Peripheral Attention Distributing information around a two-dimensional space allows us to arrange it so that it can be selectively attended to.

For example, many applications divide the screen (or window) into two areas—a large area taking up most of the space in which the primary interaction takes place, and a smaller area, at one edge or off to the side, in which messages are displayed about the current progress of other tasks, or other ancillary information. My word processor uses this approach. It has a status bar at the bottom of the screen that shows when the document is being updated, saved, printed, and so forth and provides various pieces of information that might be helpful in managing my activity but are not central to it. By placing them in the periphery, the application exploits my ability to focus on one area while passively attending to other activity in the edge of my visual field.

Pattern Recognition and Spatial Reasoning Laying out information in two dimensions lets us apply the skills we use managing visual information in the everyday environment. Actions as simple as walking across the room or picking up a cup involve spatial reasoning skills, and these can be exploited in two-dimensional interfaces. In particular, our ability to recognize patterns in the spatial organization of information provides new ways to convey information, and opportunities to arrange data elements so that they convey information as a whole. The same techniques that allow graphs, charts, and other visual information designs to provide insight into collections of information can also be exploited when we move computational information and interaction into a two-dimensional space.

Information Density Pattern recognition draws upon the way in which certain arrangements of data can draw attention to patterns and other items of "meta-information." In turn, this raises a question of "information density." Some information can be conveyed more succinctly in graphical form than in lists of numbers or other textual representations. A picture really can be worth a thousand words; it can often be displayed more compactly and apprehended more rapidly than can its thousand-word equivalent. Of course, there are also forms of information for which a textual presentation is either desirable or required, but graphical interaction has never been *purely* graphical; instead, it extends the vocabulary of interaction to incorporate graphical as well as textual

presentation forms and allows textual information to be presented within a framework that incorporates graphical elements and two-dimensional layout.

Visual Metaphors As well as giving new ways to depict data, the graphical approach can also add value by providing new ways to represent actions and the context in which actions take place. This leads to the development of visual metaphors for information management. The most widespread is the office or desktop metaphor, in which information management tasks are based around a metaphorical model incorporating filing cabinets and trashcans, graphically displayed on the screen along with the basic data elements, and so conveying a sense of the activities that can be performed over the data. In more recent systems, this has been extended. General Magic's "Magic Cap" interface, used a metaphorical depiction of an office featuring a desk (along with various desktop tools), a telephone, and a door open to a world outside; note-taking applications often feature graphical depictions of notebooks or index cards; and so on.

The development of graphical interaction techniques led to a model of interface design known as *direct manipulation,* in which these elements are combined and extended. The fundamental principle in direct manipulation interfaces is to represent explicitly the objects that users will deal with and to allow users to operate on these objects directly. Uploading a file to a server by naming it, or even by selecting it from an "open file" dialog, is not a direct manipulation approach; direct manipulation would advocate selecting the file icon, dragging it and dropping it onto a representation of the server. The direct manipulation style of interface extends the idea of the visual metaphor to a richer model in which the abstract objects that make up the system's conceptual model—be they records, files, connections, servers, transactions, or whatever—are realized in a metaphorical world that also defines how they interact with each other. From these separate elements, the designer builds an inhabited world in which users act. Direct manipulation interfaces exploit and extend the benefits of graphical interaction. Because the system can be controlled entirely through the manipulation of on-screen objects, all opportunities for action are "out in the open." This eliminates (or, at

least, reduces) the need for long sequences of action, paths that might be difficult to recognize or hard to follow.

Progress

It has been a long transition from interacting with computers using a soldering iron to interacting using a mouse. It has been neither smooth nor planned. Instead, the evolution of interaction models has gone hand in hand with the evolution of technologies, models of computation, and perceptions of the roles that computers will play in our lives.

Despite the rather chaotic evolution of interaction, it is still possible to draw out some general trends. The trend I have emphasized here is the gradual incorporation of a wider range of human skills and abilities. This allows computation to be made ever more widely accessible to people without requiring extensive training, and to be more easily integrated into our daily lives by reducing the complexity of those interactions. The "skills and abilities" perspective also offers a model for what sorts of opportunities new research directions might offer.

New Models for Interactive System Design

Graphical interaction remains the dominant paradigm for interaction with computers. In 1981 Xerox's Star was the first personal computer to ship with the features of a graphical user interface as we recognize them today—windows, menus, and a mouse—and the Macintosh, three years later, was the first to ship in volume at an affordable price. Perhaps more significantly, the release of Macintosh signaled a sea change in the way in which we interacted with computers. It simply became clear that this new paradigm was how we would interact with computers from then on.[2] Other manufacturers started shipping their machines with mice and with displays capable of supporting windowed interfaces, and the graphical user interface became the familiar face of computing.

Twenty years later, this is still true. As I write this, there are four computers here in my office, running three different operating systems; but they all display similar graphical user interfaces comprising windows, menus, and widgets such as buttons and scroll bars, controlled by a mouse sitting next to the keyboard. Although the Macintosh is arguably

the only one that was designed that way from Day 1, the style that it introduced has remained largely unchallenged. In fact, the graphical interface predominates even in those areas where its application is more questionable, from wall-sized electronic whiteboards to small handheld computers.

However, recent research programs have begun to explore new paradigms for interaction and interactive system design. Some of these will be the topics of the next few chapters, but a quick sketch is in order here.

Tangible and Social Approaches to Computing

This chapter opened by discussing how we are increasingly encountering computation that moves beyond the traditional confines of the desk and attempts to incorporate itself more richly into our daily experience of the physical and social world. Each of these areas—physical and social—has been a focus of research attention.

Work on physical interaction has been a particularly active topic in the last few years. A variety of terms have been used to encompass the different activities being carried out and concerns being addressed. I use "tangible computing" here as an umbrella term.[3]

Tangible computing encompasses a number of different activities. One general trend is to distribute computation across a variety of devices, which are spread throughout the physical environment and are sensitive to their location and their proximity to other devices. In these sorts of environments, printers and fax machines might advertise their presence to handheld computers, which can then reconfigure themselves around the set of services available in the local environment; or tags identifying individuals might signal their presence to each other so that their wearers can find out which people in a meeting room share their interests, or even just who the people are. A second trend is to augment the everyday world with computational power, so that pieces of paper, cups, pens, ornaments, and toys can be made active entities that respond to their environment and people's activities. A toy might know when it has been picked up and change the computer display to reflect the fact that its owner is clearly feeling more playful rather than concentrating on work. Or picking up a piece of paper might cause my computer to show me related documents or remind me about other things I was working on

when I last worked on it. A third topic of investigation in tangible computing is how these sorts of approaches can be harnessed to create environments for computational activity in which we interact directly through physical artifacts rather than traditional graphical interfaces and interface devices such as mice. Mice provide only simple information about movement in two dimensions, while in the everyday world we can manipulate many objects at once, using both hands and three dimensions to arrange the environment for our purposes and the activities at hand. A child playing with blocks engages with them in quite different ways than we could provide in a screen-based virtual equivalent; so tangible computing is exploring how to get the computer "out of the way" and provide people with a much more direct—tangible—interaction experience.

Although perhaps less focused as a research activity than tangible computing, the last decade or so has also seen increasing attempts to incorporate understandings of the social world into interactive systems. By analogy with tangible computing, I refer to this as "social computing."

Again, it encompasses a range of different activities that are more or less aligned. One set of activities involves incorporating social understandings into the design of interaction itself. That is, it attempts to understand how the "dialogue" between users and computers can be seen as similar and dissimilar to the way in which we interact with each other. Social science offers models of social action and the establishment of social meaning, which provide insight into the design of interaction with software systems. At the same time, anthropological and sociological approaches have been applied to uncovering the mechanisms through which people organize their activity, and the role that social and organizational settings play in this process. These investigations have yielded both prototype systems and generalized understandings of the influence that social and organizational settings can have on the organization of activities around computer systems. Finally, here, a third set of investigations has explored how what we normally consider to be "single-user" interaction—one person sitting in front of one computer—can be enhanced by incorporating information about others and the activity of others. This information can, in turn, assist individuals in exploring the electronic world of a computer application in the same way that the real

world reveals to us signs and indications of the activities of others that can help us find our way around and carry on our actions—whether by "following the crowd" to find an event, sizing up the clientele when deciding on a restaurant, or knowing that a hotel is a good place to catch a taxi.

These are brief sketches of research areas, to be explored in more detail later on. However, even these overviews show that Human-Computer Interaction research is responding to the challenges of computation that inhabits our world, rather than forcing us to inhabit its own.

From Tangible and Social Computing to Embodied Interaction

My reason for viewing the history of interaction as a gradual expansion of the range of human skills and abilities that can be incorporated into interaction with computers is that I believe that it provides a valuable perspective on activities such as tangible and social computing. In particular, it shows that these two areas draw on *the same* sets of skills and abilities. Tangible and social computing are arguably aspects of one and the same research program.

This is the hypothesis that this book sets out to explore. The rest of the book will discuss the hypothesis and its implications in more detail, but I will set the argument out briefly here. It has four parts.

First, I want to argue that social and tangible interaction are based on the same underlying principles. This is not to deny their obvious differences, both in the approaches they adopt and the ways in which they apply to the design of interactive systems. Nonetheless, they share some important elements in common. In particular, they both exploit our familiarity and facility with the everyday world—whether it is a world of social interaction or physical artifacts. This role of the everyday world here is more than simply the metaphorical approach used in traditional graphical interface design. It's not simply a new way of using ideas like desktops, windows, and buttons to make computation accessible. Instead of drawing on artifacts in the everyday world, it draws on *the way the everyday world works* or, perhaps more accurately, *the ways we experience the everyday world*. Both approaches draw on the fact that the ways in which we experience the world are through directly interacting

with it, and that we act in the world by exploring the opportunities for action that it provides to us—whether through its physical configuration, or through socially constructed meanings. In other words, they share an understanding that you cannot separate the individual from the world in which that individual lives and acts.

This comes about in contrast to a narrowly cognitive perspective that, for some time, dominated the thinking of computer system designers and still persists to a considerable degree. The positivist, Cartesian "naive cognitivism" approach makes a strong separation between, on the one hand, the mind as the seat of consciousness and rational decision making, with an abstract model of the world that can be operated upon to form plans of action; and, on the other, the objective, external world as a largely stable collection of objects and events to be observed and manipulated according to the internal mental states of the individual. From this perspective, a disembodied brain could think about the world just as we do, although it might lack the ability to affect it by acting in it. In contrast, the new perspective on which tangible and social computing rest argues that a disembodied brain could not experience the world in the same ways that we do, because our experience of the world is intimately tied to the ways in which we act in it. Physically, our experiences cannot be separated from the reality of our bodily presence in the world; and socially, too, the same relationship holds because our nature as social beings is based on the ways in which we act and interact, in real time, all the time. So, just as this perspective argues that we act in the world by exploring its physical affordances, it also argues that our social actions are ones that we jointly construct as we go along. A conversation between two people is shaped in response to the moment rather than abstractly planned, in much the same way as a juggler has to respond dynamically to the way in which each ball falls.

This leads to the second part of my argument, which is that the central element of this alternative perspective is the idea of *embodiment*. By embodiment, I do not mean simply physical reality, although that is often one way in which it appears. Embodiment, instead, denotes a form of participative status. Embodiment is about the fact that things are embedded in the world, and the ways in which their reality depends on being embedded. So it applies to spoken conversations just as much as to

apples or bookshelves; but it's also the dividing line between an apple and the *idea* of an apple.

Why is embodiment relevant to these sorts of interactions with computers? It is relevant in at least three ways.

First, the designers of interactive systems have increasingly come to understand that interaction is intimately connected with the settings in which it occurs. In adopting anthropological techniques as ways to uncover the details of work and develop requirements for interactive systems to support that work, we have begun to realize just how important a role is played by the environment in which the work takes place.[4] This is true of both physical environments and social or organizational ones. Physical environments are arranged so as to make certain kinds of activities easier (or more difficult), and in turn, those activities are tailored to the details of the environment in which they take place. The same thing happens at an organizational level; the nature of the organization in which the work takes place will affect the work itself and the ways it is done. The increasing sensitivity to settings leads naturally to a concern with how work and interaction are embodied within those settings, because that embodiment determines how it is that computation and the setting will fit together.

Second, this focus on settings reflects a more general turn to consider work activities and artifacts in concrete terms rather than abstract ones. Instead of developing abstract accounts of mythical users, HCI increasingly employs field studies and observational techniques to stage "encounters" with real users, in real settings, doing real work. These encounters are often very revealing, as they show that the ways the work gets done are not the ways that are listed in procedural manuals, or even in the accounts that the people themselves would tell you if you asked. Attention to detail, to specifics, and to actual cases, leads in turn to thinking about computation in similar terms. In particular, it leads to a concern with how interaction is manifest in the interface. Tangible computing reflects this concern by exploring the opportunities for us to manifest computation and interaction in radically new forms, while social computing seeks ways for interaction to manifest more than simply the programmer's abstract model of the task, but also the specifics of how the work comes to be done. In the real world, where the artifacts through

which interaction is conducted are directly embodied in the everyday environment, these are all manifested alongside each other, inseparably. Tangible and social computing are trying to stitch them back together after traditional interactive system design approaches ripped them apart.

Third, there is a recognition that, through their direct embodiment in the world we occupy, the artifacts of daily interaction can play many different roles. As an example, consider the revealing studies of the role of medical record cards in hospitals (Nygren, Johnson, and Henriksson 1992). From a technical perspective, patient record cards are simply carriers of well-defined information concerning the patient's diagnosis and treatment, and, as embodied on paper, present various problems: they can be lost, they can be hard to read, and they can only be in one place at a time. From this perspective, it seems both straightforward and beneficial to replace the paper records with electronic versions. However, in practice, such straightforward replacements are rarely successful. Studies of the failure of such systems show that the paper records are more than simply carriers of information about patients. They carry other important information as a result of the way that they are used in the work of the hospital. For example, handwriting on the forms reveals who performed different parts of the treatment; wear and tear on the form indicates heavy use; and the use of pencil marks rather than pen informally indicates tentative information. To trained eyes, a card conveys information not just about the patient, but also about the history of activities over the card and around the patient. It can do this because it not only represents the world of the patient, but it also participates in that world—it is an embodied artifact, and it participates in the embodied activities of those administering medical care. So, one relevance of embodiment for interaction with computational systems is that, for many tasks, it is relevant to consider how computation participates in the world it represents. Computation is fundamentally a representational medium, but as we attempt to expand the ways in which we interact with computation, we need to pay attention to the duality of representation and participation.

The third element of this book's argument is that the idea of embodiment as a common foundation points us to other schools of thought. Embodiment is not a new phenomenon, or a new area for intellectual

endeavor. In fact, it is a common theme running through much twentieth century thought. The notion of embodiment plays a special role in one particular school of philosophical thought, phenomenology.

Phenomenology is primarily concerned with how we perceive, experience, and act in the world around us. What differentiates it from other approaches is its central emphasis on the actual phenomena of experience, where other approaches might be concerned with abstract world models. Traditional approaches would suggest that we each have an understanding of the elements of which our world is constructed, and an abstract mental model of how these concepts are related. We understand that there are entities we can drink from, and that cups, glasses, and mugs are examples; we understand that we can sit on things like sofas and stools, and that people might keep cats and rabbits as house-pets, but rarely elephants or seals. This information, abstractly encoded in our heads, guides our actions in the world. Armed with a model of appropriate concepts and relations—an ontology—we can look around us and recognize what we see. So, the traditional model supposes that when I encounter a glass of wine, even though I have never seen this particular one before, I can still recognize it as being a glass of wine because of the way in which it fits into my model as an instance of the abstract class of glasses and other drinking vessels.

In contrast, the phenomenologists argue that the separation between mind and matter, or between what Descartes called the *res cogitans* and the *res extensa*, has no basis in reality. Thinking does not occur separately from being and acting. Certainly, there is nothing in our experience to support such a separation. In every case, we encounter them together, as aspects of the same existence. Consequently, phenomenology has attempted to reconstruct the relationship between experience and action without this separation. Rather than the Cartesians' theory- or model-driven approach to perception, the phenomenological approach argues for what we might call a *preontological* apprehension of the world. Perception begins with what is experienced, rather than beginning with what is expected; the model is to "see and understand" rather than "understand and see."

To say that phenomenology is all about perception is to limit it unfairly. In addition to perception, it is also concerned with action, with

understanding, and with how these are all related to each other, as part and parcel of our daily experience as participants in the world. In the hands of some, such as Alfred Schutz, phenomenology has also been a tool to understand social action and practice; others such as Wittgenstein, while not phenomenologists, have developed allied approaches to topics such as language and meaning. As we will see, these approaches provide an extensive set of investigations of the questions of presence, embodiment, and action.

In turn, the fourth element of the book's argument is that we can build on the phenomenological understandings to create a foundational approach to embodied interaction. Such a foundation should do two things. First, it should account for the ways in which social and tangible computing—and, perhaps, further areas to be defined—are related to each other, showing how they can be draw upon each other's work and provide a unified model for Human-Computer Interaction. Second, it should inform and support the design, analysis and evaluation of interactive systems, providing us with ways of understanding how they work, from the perspective of embodiment.[5]

This, then, is the four-part hypothesis that this book sets out to explore: that tangible and social computing have a common basis; that embodiment is the core element they have in common; that embodiment is not a new idea, but has been a primary topic for phenomenology; and that phenomenology and related investigations of embodiment can provide material for developing a foundation for embodied interaction.

This has all been presented so far in very broad strokes. The chapters to come will explore the issues in more depth and provide much more background. The two chapters that follow describe the recent trends in HCI research that are the starting point for this work. Chapter 2 deals with tangible computing, while chapter 3 explores social computing. Each presents both the research and the context in which it emerged. However, they present tangible and social computing as self-contained; in chapter 4, we begin to examine how they might be brought together, and how ideas from phenomenology and other philosophies of presence and experience can be brought to bear to understand the relationships between them. Just as chapters 2 and 3 try to introduce the set of ideas from tangible and social computing that will inform the later discussion,

so chapter 4 provides an introduction to the phenomenological work that we will draw upon later. With this background, chapter 5 explores the notion of embodiment in more depth, drawing out a number of constituent elements whose relationships can be used to analyse interaction case studies. Chapter 6 builds on this and presents a framework that arranges these foundational elements to be able to draw on them for design, and chapter 7 points to some future directions.

2

Getting in Touch

For a device whose fundamental properties have changed so radically over the past thirty years, the personal computer itself—the familiar beige box sitting by the desk—has changed remarkably little.

The personal computer (PC) as we currently know it has its origins in work carried out at Xerox's Palo Alto Research Center in the early 1970s. The forerunner of the modern PC was, arguably, the Alto workstation developed by researchers there; it pioneered such now-common features as bitmapped displays with overlapping windows, graphical interfaces with multiple fonts and pop-up menus, and computers linked together over local-area networks. Although underpowered by today's standards (it was clocked at 6 MHz rather than the many hundreds of today's PCs), it nonetheless set the stage for what was to come, and its basic feature set, built around "the three 'M's"—millions of pixels, a megabyte of memory, and a million instructions per second—is still with us today.

On the other hand, an Alto in those days cost around $16,000 to build, scarcely affordable enough to put "a computer on every desk," as Microsoft would later set out to do. A more affordable option in 1977 (by which time the PARC researchers were working on the Dorado, a considerably faster and more powerful machine) was the Apple II, the device which, arguably, kick-started the personal computer industry. The Apple II was powered by a 6502 8-bit processor running at 1.5 MHz. It had 8 kilobytes of semiconductor memory and stored programs on cassette tape; optional floppy disk drives stored around 150 kilobytes each. Compare that to the modern personal computer. The laptop computer on which I'm writing this is certainly not top-of-the-line; it wasn't even top-of-the-line when I bought it a year ago. It has a 166 MHz 32-bit

processor, 64 megabytes of memory, and a 13-inch color display and can store up to 6 Gb on an internal hard disk; and it cost under $4,000.[1]

Imagine what it would be like if any other technology had undergone such rapid advances in price/performance. A car would cost a few dollars; airplanes would travel at hundreds of times the speed of sound; televisions would weigh a few ounces. More to the point, if cars, airplanes, and televisions had been so radically transformed, they would not be cars, airplanes, and televisions any more. They would have transformed themselves into something else altogether.

Computers, though, remain computers. As we enter the twenty-first century, today's PC still looks remarkably similar to that of the late 1970s (and perhaps even more like the Alto of the earlier part of that decade; see figure 2.1). This is not simply a matter of packaging and

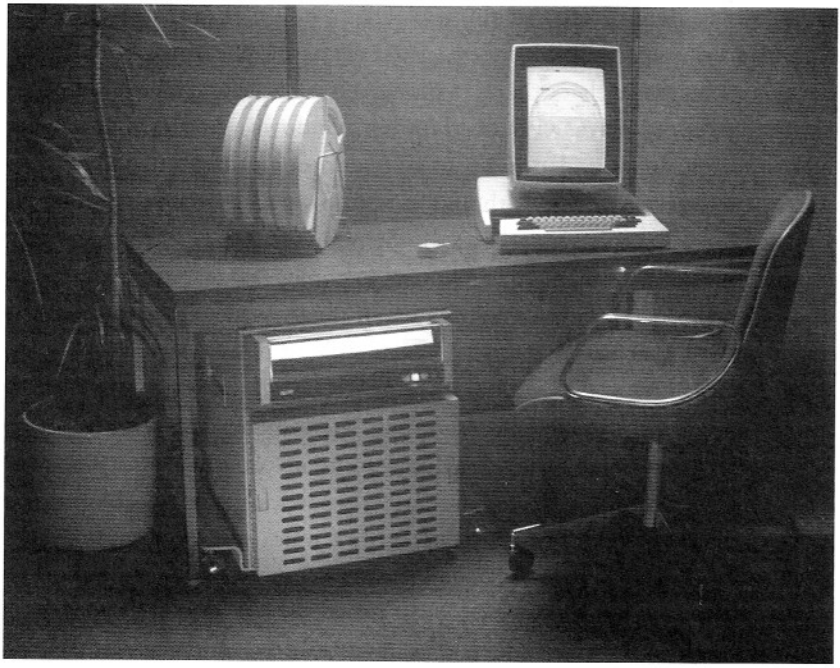


Figure 2.1
Xerox's Alto (1974). This early personal computer is somewhat bulkier than today's, but is otherwise very recognizable in form. Reprinted by permission of Xerox Palo Alto Research Center.

industrial design, although it is certainly the case that with a few notable exceptions, we seem to be firmly stuck in an age of beige boxes. My concern is not so much about the boxes themselves as about the relationship of the user to the box. Despite the fact that computers are so radically different from the computers of twenty years ago, and that their capabilities are so vastly different, we interact with them in just the same way; we sit at a desk, watching the screen and typing on the keyboard. If you were to look at a photograph of people using computers some time over the last twenty years, their clothes and hairstyle might give you a clue to the date when the picture was taken, but the style of interaction with the computer certainly would not.

Similarly, the style of interaction concerns not simply the set of physical devices (keyboards, screens, and mice) or the set of virtual devices (dialog boxes, scroll bars, and menus) through which we interact, but also the ways in which the computer fits into our environments and our lives. Interaction with screen and keyboard, for instance, tends to demand our direct attention; we have to look at the screen to see what we're doing, which involves looking away from whatever other elements are in our environment, including other people. Interaction with the keyboard requires both of our hands. The computer sits by the desk and ties us to the desk, too. So, it is not simply the form of the computer that has changed remarkably little over the last thirty years; it is also the forms of computer-based activity and the roles that we imagine computers playing in our everyday lives.

Although this model of everyday computing might be conventional, it is not inevitable. The rise of the personal computer—and, more broadly, of personal computing—was an attempt to break away from the then-dominant paradigm of mainframe computing. Similarly, while personal computing may now be established as the dominant model, a variety of alternatives have been explored in the research community; departures from the world of the conventional PC as radical as the PC was from the world of the mainframe. In this chapter, I will take a brief tour through some of the research laboratories where these alternatives are being explored. In particular, I will focus on an approach that looks at the relationship between computers on the desktop and the world in which they (and we) operate. This is a model of interaction that I refer to as "tangible

computing." Although it is only lately that the tangible computing paradigm has become broadly established, its has emerged from a research program that stretches back over a decade.

Ubiquitous Computing

We begin the tour, ironically enough, in the Computer Science Lab at Xerox PARC—the same place that gave us the desktop PC. In the 1970s, Xerox had set up PARC to explore "the architecture of information," and the Computer Science Lab, under the guidance of former ARPA manager Bob Taylor, had delivered what was to become the basic elements of office information technology in the decades to follow—powerful personal workstations, laser printers, and shared servers, linked together on local area networks. Xerox, famously, had failed to recognize its own future in PARC's vision, so today's office technology generally doesn't carry a Xerox label (Smith and Alexander 1988).

By the start of the 1990s, the situation was different. PARC's vision of the architecture of information had, largely, come to pass; and, in the opinion of the new manager of the Computer Science Lab, Mark Weiser, it was time for a new and equally radical vision of the future of technology.

What Weiser proposed was a research program that he dubbed "Ubiquitous Computing." Weiser saw that the development and diffusion of general-purpose computers, and in particular PC's, had resulted in a focus on the computer rather than on the tasks that the computer was used to accomplish. He argued that ongoing technological developments, particularly in mobile and low-power devices, would transform the nature of computers and the way we interact with them. Why deal with a single, large, expensive computer when you could harness many tiny, low-cost devices spread throughout the environment? Instead of always taking work to the computer, why not put computation wherever it might be needed? Through the technical developments that supported this new model, he saw an opportunity to turn attention away from the dominating focus on the computer sitting on the desktop and back to the applications, and to the artifacts around which those applications were structured. Weiser's vision of "ubiquitous computing" was one of computationally enhanced walls, floors, pens, and desks, in which the power

of computation could be seamlessly integrated into the objects and activities of everyday life.

One analogy that Weiser proposed as a way of understanding his vision for the new role of computation was that of solenoids, the electronically actuated switches that are part of the fabric of many everyday technologies. For example, he observed, a modern car has a vast number of solenoids, invisibly controlling everything from the air conditioning to the fuel intake. Solenoids are a critical component of modern technological design and are used in all sorts of settings. And yet, we don't deal directly with solenoids in the way we do with computers. We don't have to think about the design of the "human-solenoid interface"; we don't have programs on "solenoid literacy" in schools; you can't take a degree in "solenoid science," and nobody had to upgrade to "Solenoids 2000."

Why have computers and solenoids followed different paths? Various possibilities present themselves. Perhaps it is because of the nature of computers as multipurpose devices; or perhaps it is a historical accident, a feature of how computer technology was introduced into the home and work environments. And to be sure, there are all sorts of computer technologies surrounding us that are far more like solenoids than they are like PCs, such as the computer processors inside my television set, microwave oven, and car. The difference between my PC and those other devices is that those other devices are organized around human needs and functions.

Weiser's model of ubiquitous computing was also, paradoxically, one of invisible computers. He argued for a vision of computers in which the computer had become so ubiquitous that it had, essentially, disappeared. He proposed that the computer of the twenty-first century would have proceeded further along the path from the mainframe to the processor in my microwave oven, and that the intermediate step—the desktop PC—would be all but gone. However, in this world, although there might be no more computers as we understand them today, there would certainly be computation. In fact, there might be a great deal more computation than there is now. Computational devices would be embedded in all sorts of technologies, Weiser argued, creating a variety of specialized devices augmented with computational power. Computers would

disappear into the woodwork; computers would be nowhere to be seen, but computation would be everywhere.

Computation by the Inch, Foot, and Yard

In the Computer Science Lab at Xerox PARC, Weiser initiated a wide-ranging research program around his vision of Ubiquitous Computing, fostering the development of new computational technologies, the infrastructure necessary to support them, and new application models. PARC's ubiquitous computing strategy followed three tracks: they were known as computation by the inch, the foot and the yard (see figure 2.2).

"Computation by the inch" focused on the development of small devices, like electronic tags or computational "Post-It" notes. One focus of attention was the use of devices called "Active badges," originally developed at the Olivetti Research Centre in Cambridge, England (Want et al. 1992). Active badges are devices measuring roughly 1.5 inches square that are intended to be worn like normal identity badges. However, they house some simple electronics and emit a fixed, coded infrared signal every thirty seconds or so (or whenever a button on the badge is pressed). These signals are detected by a network of infrared receivers located in the environment, and which are connected to a computational server process. Because each badge emits an individual code, and because its signal will generally only be received by the closest detector, the server can maintain a map of the location of each badge within the sensor network, which in turn can locate the badge's wearer within the environment.

When people wear active badges, then applications can help make the environment responsive to their movements. The system can route telephone calls to the current location of the person being called, display relevant information on nearby monitors as they pass by, or customize the behavior of a computer system to the needs of the person sitting at it. In Weiser's model, badges or similar tags could also be attached to books and other artifacts, so that their location and mutual proximity could become a resource to computer-based applications.

If computation "by the inch" sought a model of computationally enhanced Post-It Notes, the computation "by the foot" was concerned with computationally enhanced pads of paper. The primary focus of this

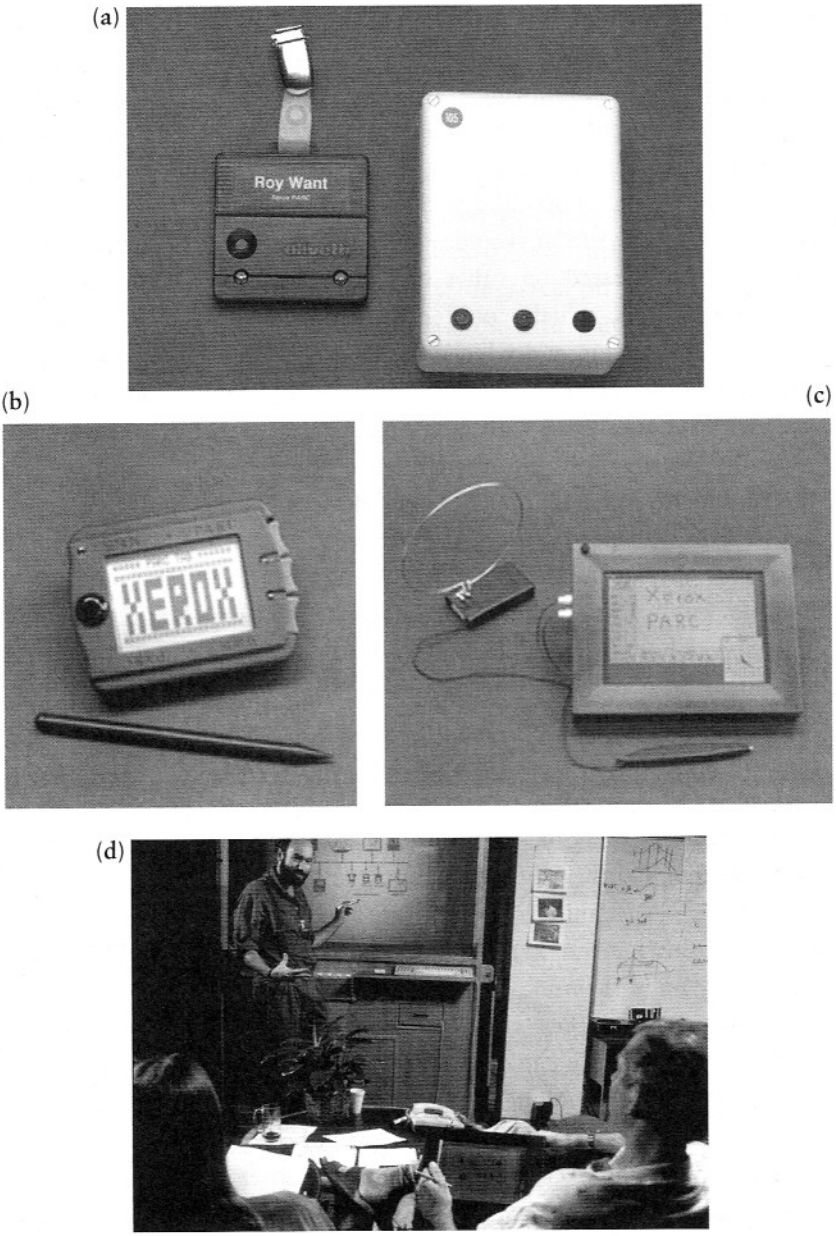


Figure 2.2
Computing by the inch, the foot, and the yard: (*a*) an active badge, (*b*) the PARC Tab, (*c*) the PARC Pad, and (*d*) a meeting at the Liveboard. Reprinted by permission of Xerox Palo Alto Research Center.

area of work was the development and use of computational devices of about the size and power of recent laptop computers. Laptop computers were, of course, already widely available at this point, but they tended (as they still do) to function simply as scaled-down versions of their desktop cousins. In contrast, the goal of ubiquitous computing research was not simply on the size and packaging of the devices, but of how they would fit into a world of everyday activities and interaction. As a result, research concentrated on other concerns. Examples included stylus-based interaction, which could eliminate keyboards as the primary source of interaction, and which could support note-taking and sketching, and mobile operation, so that devices could be moved from place to place without interfering with their operation.

Finally, investigations into computation "by the yard" introduced the opportunity to consider much larger devices. In particular, attention focussed on wall-sized devices such as the LiveBoard. LiveBoard was a large-scale display (approximately five feet by three feet) supporting multiple pens, a sort of computationally enhanced whiteboard. Researchers observed how the very physical form of this device was an important component in structuring interactions with it. On the one hand, the use of pen input meant that collaborative activities (such as brainstorming in a meeting) would be implicitly structured by the fact that the board was large enough for everyone to see at once, but that two people could not stand in front of the same part of the board or write in the same area at the same time. On the other hand, the board's large size also meant that new interaction techniques would have to be developed; using a scroll bar or pull-down menu on a board a board five feet wide could be, quite literally, a pain in the neck.

Discussing each of these components of PARC's ubiquitous computing strategy independently can mask the critical integration of the various facets of the program. None of these devices was intended to operate on its own. The focus, after all, was on a form of computation more deeply integrated with the everyday environment, and the everyday environment is filled with a variety of objects and devices. So it was with the ubiquitous computing vision. A single user might have, at his or her disposal, tens or more of the inch-sized devices, just as we might have many Post-It notes dotted around, stuck to computer screens, walls, books,

and sheets of paper; at the same time, they might also have three or four foot-sized devices, just as I might have a number of notebooks for different topics or projects; but just as I probably only have one or maybe two whiteboards in my office, there will be fewer of the devices at the larger scale. What is more, information is expected to be able to move around between the different devices. Notes that I have prepared on an electronic pad might be beamed onto the board for group consideration in a meeting; while action items might be migrated off into a hand-held device that stores my calendar and to-do list. In the everyday environment, information continually undergoes transformations and translations, and we should expect the same in a computationally enhanced version of that environment such as might be delivered to us by ubiquitous computing.

The Digital Desk

At much the same time as Weiser and his PARC colleagues were developing the ubiquitous computing program, related activity was going on in another Xerox lab, in Cambridge, England. EuroPARC had been set up as a European satellite laboratory of PARC. It was a much smaller lab (with a research complement of around twenty) with a focus on interdisciplinary research into Human-Computer Interaction and Computer-Supported Cooperative Work.

EuroPARC was home to a variety of technological developments, but the particular technology that concerns us here is the Digital Desk, designed and developed by Pierre Wellner (Wellner 1991; Newman and Wellner 1992). In common with many people, Wellner had observed that the "paperless office" envisioned by many in the 1970s and early 1980s had manifestly failed to develop. However, that was not to say that the development of personal computers, and increasingly networked personal computers, had not caused an massive increase in the number of digital or online documents that we all have to deal with everyday. Wellner was concerned with how we could work with both paper and electronic documents in a much more fluid and seamless way than is normally the case. The traditional approach to these problems was either to scan in the paper documents to bring them into the

electronic realm, or to print out the electronic documents to bring them into the physical realm. By moving across the boundary from online documents to paper documents and back again, users could take exploit the advantages of each; the digital malleability and computational power of electronic documents with the portability, readability, and informal interaction of paper ones. As many studies have attested, paper has many properties that are hard to reproduce in the electronic world (Sellen and Harper 1997; Henderson 1998), while, at the same time, electronic documents increasingly exploit features (such as animation, hyperlinks, or interactive elements) that paper documents cannot capture. So, the move back and forth between electronic and paper forms is not only inconvenient but also impoverished, since some features always remain behind. Taking his cue from Weiser's ubiquitous computing work, Wellner wondered if there wasn't a way to combine the two worlds more effectively by augmenting the physical world with computational properties.

Wellner's Digital Desk (figure 2.3) combines elements of each. The Digital Desk was a physical desktop, much like any other, holding papers, pens, coffee cups, and other traditional office accoutrements. However, it was also augmented with some distinctly nontraditional components. Above the desk were placed a video projector and a video camera. Both of these were pointed down toward the desktop; the projector would project images onto the desk, over whatever objects were lying there, and the camera could watch what happened on the desktop. These devices were connected to a nearby computer. Image processing software running on the computer could analyze the signal from the video camera to read documents on the desk and watch the user's activity. At the same time, the computer could also make images appear on the desk by displaying them via the video projector.

The result was a computationally enhanced desktop supporting interaction with both paper and electronic documents (Wellner 1993). Electronic documents could be projected onto the desktop by the video projector, but then could be moved around the (physical) desktop by hand (using the video camera to track the user's hand movements and then "moving" the displayed document in coordination). Similarly, physical documents could be given computational abilities on the same

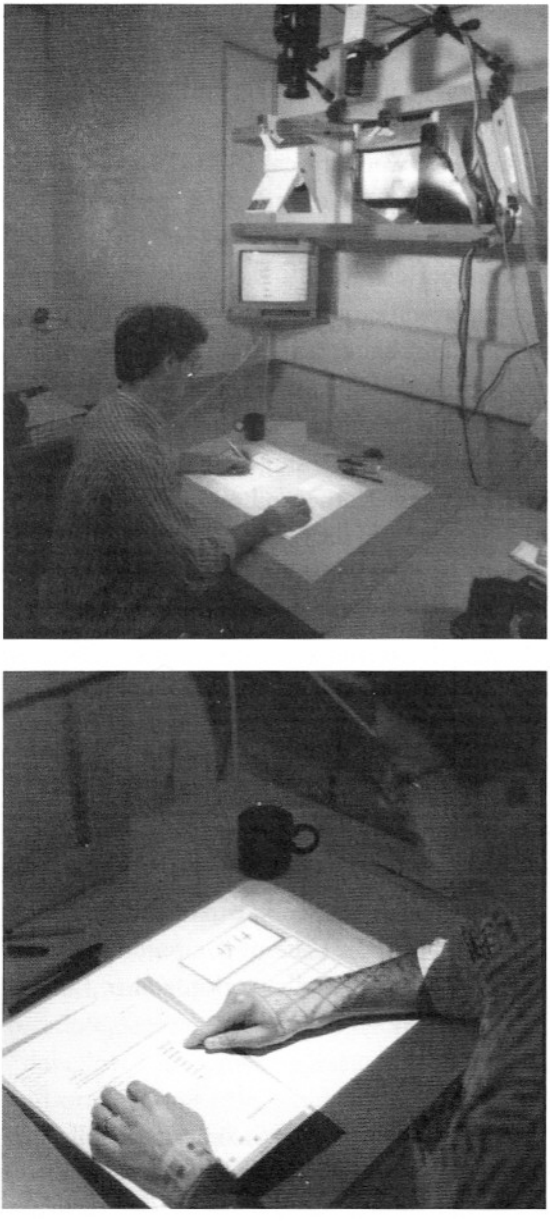


Figure 2.3
Wellner's Digital Desk allowed interaction with paper and electronic documents on the same desktop. Reprinted by permission of Xerox Research Centre Europe.

desktop. For example, a paper document containing a list of numbers could be used as input to a virtual calculator; the computer could use the camera to "read" the numbers off the printed page, and then project the result of a calculation over those figures.

Two features of the Digital Desk were critical to its design. The first was its support for manipulation. In Wellner's first prototype, one moved objects around on the desk with one's fingers; in contrast with the prevailing approach to interface design, this was *really* direct manipulation. What's more, of course, while our computer systems typically have only one mouse, we have *two* hands and *ten* fingers. By tracking the position and movements of both hands or of multiple fingers, the Digital Desk could naturally support other behaviors that were more complicated in traditional systems, such as using both hands at once to express scaling or rotation of objects. The second critical design feature was the way in which electronic and physical worlds were integrated. A document on the digital desk could consist of both physical content (printed on a page) and electronic content (projected onto it), and printers and cameras allowed material to move from one domain to the other fluidly so that objects created on paper could be manipulated electronically. The Digital Desk offered developers and researchers an opportunity to think about the boundary between the physical and virtual worlds as a permeable one.

While the work on ubiquitous computing had shown how computation could be brought out of the "box on the desk" and into the everyday world, Wellner's work on the digital desk expanded on this by considering how, once the real world was a site of computational activity, the real and electronic worlds could actually work together.

Virtual Reality and Augmented Reality

Weiser and Wellner shared the goal of creating computationally augmented reality. They both attempted to take computation and embed it in the everyday world. This follows in the trend, outlined earlier, to expand the range of human skills and abilities on which interaction can draw. In this case, the abilities to be exploited are those familiar ways in which we interact with the everyday world; drawing on whiteboards,

moving around our environments, shuffling pieces of paper, and so on. One of the interesting feature of these approaches, at the time, was the way in which they developed in opposition to another major trend—immersive virtual reality.

Virtual reality (VR) is, at least in the popular consciousness, a technology of recent times; it became particularly prominent in the 1990s. Immersive VR as we know it today came about through the increase in computer power, and particularly graphics processing, that became available in the late 1980s, as well as some radical sensor developments that gave us data gloves and body suits. The technical developments supporting immersive VR became widespread at around the same time as William Gibson's notion of "cyberspace"—a technically mediated consensual hallucination in which people and technology interacted—also entered the popular consciousness. Virtual reality has been around a good deal longer than that, however. Ivan Sutherland, the father of interactive computer graphics, went on to investigate what we now recognize as virtual reality technology back in the 1960s, and the use of digital technology to create environments such as flight training simulators is well-known. Howard Rheingold's book *Virtual Reality* (1992) documents some of the early history of this seemingly recent technology.

Virtual reality immerses the user in a computationally generated reality. Users don head-mounted displays, which present slightly different computer-generated images to each eye, giving the illusion of a three-dimensional space. By monitoring the user's head movements and adjusting the image appropriately, this three-dimensional space can be extended beyond the immediate field of view; the user can move his head around, and the image moves to match. With appropriating sensing technologies, the user can enter the virtual space and act within it. A "dataglove" is a glove augmented with sensors that report the position and orientation of the hand and fingers to a computer; the hand of the user wearing the glove is projected as a virtual hand into the same computer-generated three dimensional space that the virtual reality system generates, so that the user can pick up virtual objects, examine them, move them around, and act in the space.

The ubiquitous computing program was getting under way at about the point when virtual reality technology began to make its way out of

research laboratories and into newspaper articles. Both approaches to the future of computing are based on similarly science-fiction notions; immersion in a computer-generated reality, on the one hand, and computers in doorknobs and pens on the other. They embody, however, fundamentally different approaches to the relationship between computers, people and the world. In the virtual reality approach, interaction takes place in a fictional, computer-generated world; the user moves into that world, either through immersion or, more commonly these days, through a window onto the world on a computer screen. The world of interaction is the world of the computer. The ubiquitous computing approach to interaction—what Weiser dubbed "physical virtuality" and would become known as augmented reality—does just the opposite. It moves the computer into the real world. The site of interaction is the world of the user, not that of the system. That world, in the augmented reality vision, may be imbued with computation, but the computer itself takes a back seat.

The Reactive Room

The ubiquitous computing model distributes computation throughout the environment. All sorts of objects, from walls to pens, might have computational power embedded in them. For someone concerned with interaction, this raises one enormous question—how can all this computation be controlled?

At the University of Toronto, Jeremy Cooperstock and colleagues explored this question in an environment they called the Reactive Room (Cooperstock et al. 1995). The Reactive Room was a meeting room supporting a variety of physical and virtual encounters. It grew out of both the ubiquitous computing perspective and the "media space" tradition, an approach to supporting collaboration and interaction through a combination of audio, video, and computational technology (Bly, Harrison, and Irwin 1993). The room was designed to support not only normal, face-to-face meetings, but also meetings distributed in space (where some participants are in remote locations) and time (recording meeting activity to be viewed later by someone else). To that end, it also featured a shared computer display, for electronic presentations and application-

based work; a variety of video and audio recorders; and audio and video units connected to a distributed analog A/V network that could be connected to similar "nodes" in people's offices, so they could remotely "attend" meetings.

However, such a complex and highly configurable environment presented considerable challenges for control and management. To configure the room for any given situation (such as a presentation to be attended by remote participants), each device in the room would have to be configured independently, and adjusting the configuration to support the dynamics of the meeting was even more challenging. The design of the Reactive Room sought to use ubiquitous computing technology as a means to manage this problem. The critical move here was to see ubiquitous computing as a technology of *context;* where traditional interactive systems focus on what the user does, ubiquitous computing technologies allow the system to explore *who* the user is, *when* and *where* they are acting, and so on.

In the case of the reactive room, contextual information could be used to disambiguate the potential forms of action in which a user might engage. For example, by using an active badge or similar system, the room's control software can be informed of who is in the room and can configure itself appropriately to them. Similarly, if the room "knows" that there is a meeting in progress, then it can take that information into account to generate an appropriate configuration. If a user presses the "meeting record" button on a VCR, to record a meeting in progress, the Reactive Room can determine whether or not there are any remote participants connected to the audio/video nodes and, if so, ensure that it adds those signals to the recording. When someone in the room makes use of the document camera or the projected computer display, the room software can detect these activities and automatically make the document camera view or the computer display available to those people attending the presentation, either locally or remotely.

In other words, the design of the Reactive Room attempts to exploit the fact that the people's activities happen in a context, which can be made available to the software in order to disambiguate action. Clearly, of course, the sort of context that can be gathered with current technology is limited; the Reactive Room would make use of motion in

particular parts of the room, presence and activity as detected using active badges or pressure sensors, and so on. The other, perhaps most important, piece of context it made use of was *the fact that it was the Reactive Room*. That is, the room was designed for meetings and presentations, and so much activity in the room could be interpreted as being appropriate to meetings and presentations. The same sorts of inferences would probably be inappropriate in other settings, such as a private office, or a home. The "meeting" context, then, also serves to disambiguate the user's goals.

The Reactive Room demonstrated the way that ubiquitous computing did not simply move out of the box on the desk and into the environment but, at the same time, also got involved in the relationship between the environment and the activities that took place there. The topic of "setting-ed" behavior will come back into focus in the next chapter; for the moment, however, we will continue to explore the development of tangible computing.

Design Trends

The systems that have been described—the vision of Ubiquitous Computing, and the Digital Desk and Reactive Room prototypes—have been firmly located in the domain of Computer Science research. However, "academic science" has by no means been the only contributor to the development of Tangible Computing. In fact, one striking aspect of the development of this line of investigation has been the contributions from the perspectives of art and design. Two pieces that have proved to be particularly inspirational to a number of researchers in this area were Durrell Bishop's Marble Answering Machine, and Natalie Jeremijenko's Live Wire.

The Marble Answering Machine was a design exercise undertaken by Bishop in the Computer-Related Design department at the Royal College of Art in London (Crampton-Smith 1995). It explored possible approaches to physical interaction for a telephone answering machine. Rather than the traditional array of lights and buttons, Bishop's answering machine has a stock of marbles. Whenever a caller leaves a message on the answering machine, it associates that message with a marble from the

stock, and the marble rolls down a track to the bottom, where it sits along with the marbles representing previous messages. When the owner of the machine comes home, a glance at the track shows, easily and distinctly, how many messages are waiting—the number of marbles arrayed at the bottom of the track. To play a message, the owner picks up one of the marbles and drops it in a depression at the top of the answering machine; because each marble is associated with a particular message, it knows which message to play. Once the message has been played, the owner can decide what to do; either return the marble to the common stock for reuse (so deleting the message), or returning it to the track (saving it to play again later).

The Marble Answering Machine uses physical reality to model the virtual or electronic world. In Bishop's design, marbles act as physical proxies for digital audio messages. By introducing this equivalence, it also enriches the opportunities for interacting with the device. The problem of interacting with the virtual has been translated into interacting with the physical, and so we can rely on the natural structure of the everyday world and our casual familiarity with it. So, counting the number of messages is easy, because we can rapidly assess the visual scene; and operations such as playing messages out of order, deleting messages selectively, or storing them in a different sequence, all of which would require any number of buttons, dials, and controls on a normal digital answering machine, all become simple and straightforward because we can rely on the affordances of the everyday world.

Natalie Jeremijenko's piece "Live Wire," also sometimes known as "the Dangling String" and described by Weiser and Brown (1996), was developed and installed at Xerox PARC in 1994 and explored similar questions of the boundary between the virtual and physical worlds. Physically, Live Wire was a length of plastic "string" around eight feet long, hanging from the ceiling at the end of a corridor. Above the ceiling tiles, the wire was connected to a small stepper motor, which in turn was connected to a device on the local ethernet. Every time a data "packet" passed by on the ethernet, the stepper motor would move, and its movements would be passed on to the string. Ethernet, in its classic form, is a "shared medium" technology—all the traffic, no matter which machine sends it or which machine is to receive it, travels along the same cable.

The busier the network, the more data packets would pass by, and the more the stepper motor would move. The ethernet can carry thousands of packets per second, and so when the network was busy the motor would whir and the string would spin around at high speed, its loose end whipping against the wall nearby.

Others have followed in the footsteps of Bishop and Jeremijenko and continued to explore the design "space" around these issues of the borders between physical and virtual worlds. Feather, Scent, and Shaker (Strong and Gaver 1996) are devices for "simple intimacy." "Feather" features a feather that is gently lifted on a column of air, to indicate to its owner that, perhaps, a photograph of them has been picked up somewhere else; it is designed to convey a sense of fondness across distance. Scent, similarly, releases a pleasant, sweet smell in similar circumstances providing an awareness of distant action.

The topic of "awareness" is one that has concerned the developers of technologies for group working, who want their systems to be able to support the casual and passive awareness of group activity that coworkers achieve in a shared physical space. Strong and Gaver turn this around, though, and give us technologies for supporting shared intimacy rather than shared work. Their pieces are designed to be evocative and emotive rather than "efficient." What is particularly interesting about this group of devices is that they originate not from a technical or scientific perspective, but from a design perspective. The result of this shift in perspective is that they a reflect a very different set of concerns. It is not simply that they reflect an aesthetic component where the scientific developments are marked more by engineering concerns. That is certainly one part of it, of course; the design examples certainly do reflect a different set of principles at work. However, there is more than this.

First, the design examples discussed here reflect a concern with *communication*. What is important is not simply what they *do*, but what they *convey*, and how they convey it; and the communicative function that they carry is very much on the surface. There is an "at-a-glance readability" to these artifacts that stands in marked contrast to the "invisibility" of ubiquitous computing. Second, they reflect a holistic approach that takes full account of their physicality. The physical nature of these pieces is not simply a consequence of their design; it is funda-

mental to it. While it was a tenet of ubiquitous computing, for example, that the technology would move out into the world, the design pieces reflect a recognition that the technology *is* the world, and so its physicality and its presence is a deeply important part of its nature. Third, they reflect a different perspective on the role of computation, in which computation is integrated much more directly with the artifacts themselves. In the other examples, while they have aimed to distribute computation throughout the environment, there has always been a distinct "seam" between the computational and the physical worlds at the points where they meet. In these examples, however, the computational and physical worlds are much more directly connected.

The result is an approach to tangible computing that sees computation within a wider context. Ubiquitous Computing pioneers saw that, in order to support human activity, computation needs to move into the environment in which that activity unfolds. These design explorations take the next step of considering how computation is to be manifest when it moves into the physical environment, and recognizing that this move makes the physicality of computation central.

Tangible Bits

Most recently, perhaps the most prominent site for development of these ideas has been the Tangible Media group at the MIT Media Lab. A group of researchers led by Hiroshi Ishii has been exploring what they call "Tangible Bits," a program of research that incorporates aspects of both the Ubiquitous Computing program and the design perspective explored by people like Jeremijenko.

The term "Tangible Bits" reveals a direct focus on the interface between the physical and virtual worlds. The rhetoric of the computer revolution has, pretty consistently, focused on a transition from physical (the world of atoms) to the virtual (the world of bits). We talk of the future in terms of "electronic cash" to replace the paper bills and coins we carry about with us, or we speak of the "paperless office" in which paper documents have disappeared in favor of electronic documents stored on servers and displayed on screens. We envision a world in which we communicate by electronic mail and video conferencing, in

which we read from "e-books," telecommute over great distances via digital communication lines, and play in virtual worlds. What these visions have in common is the triumph of the virtual over the physical. They suggest that we will overcome the inherent limitations of the everyday world (such as the need to be in the same place to see each other, or that a thousand books actually take up real shelf space) by separating the "information content" from the physical form, distilling the digital essence and decanting it into a virtual world.

The MIT Media Lab, where Ishii and his colleagues are based, is one of the most prominent proponents of this vision, especially, perhaps, in the writings of its founding director, Nicholas Negroponte. His collection of essays *Being Digital* (Negroponte 1995), explores the relationship between atoms and bits and how the development and deployment of Internet technologies is changing that relationship.

The work on Tangible Bits provides some balance to the idea that a transition from atoms to bits is inevitable and uniformally positive. It is certainly not defined in opposition to the gradual and ongoing movement of traditionally physical forms into digital media. However, it observes that while digital and physical media might be *informationally* equivalent, they are not *interactionally* equivalent. By building information artifacts based on physical manipulation, the Tangible Bits programme attempts to reinvest these distilled digital essences with some of the physical features that support natural interaction in the real world.

metaDESK, Phicons, and Tangible Geospace

Let's take an example from the work of the Tangible Bits group. The metaDESK (Ullmer and Ishii 1997) is a platform for tangible interaction. It consists of a horizontal back-projected surface that serves as the top of the physical desk itself; an "active lens," which is a small flat-panel display mounted on an arm; a "passive lens," which is transparent, also digitally instrumented; and a variety of physical objects called *phicons* (for "physical icons"). The metaDESK is shown in figure 2.4.

The functions of the various components of the metaDESK platform are best seen in terms of an application running on the desk. Tangible Geospace is a geographical information system augmented with tangible UI features and running on the metaDESK. It allows users to explore a

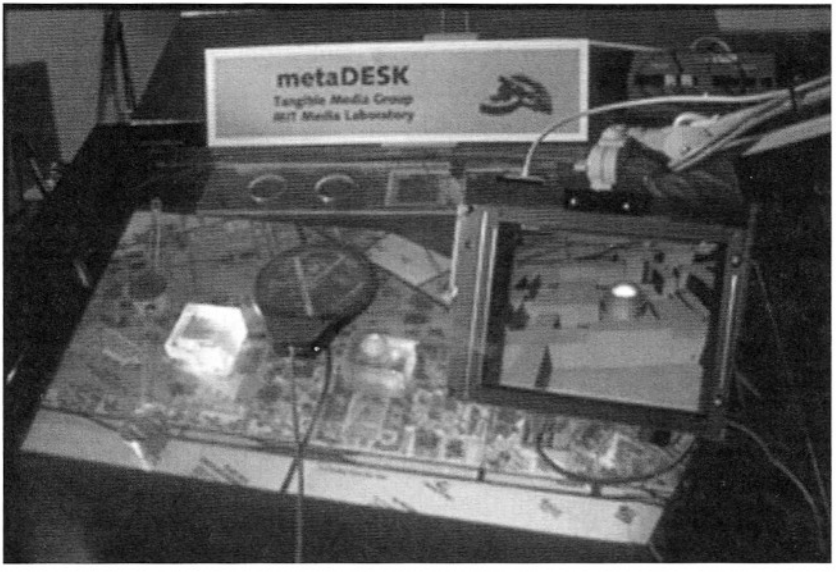


Figure 2.4
Interactions with geographical information on the metaDESK, using phicons, the passive lens, and the active lens. Reprinted by permission of The MIT Media Lab.

visualization of a geographical space, such as the area of Cambridge, Massachusetts, around MIT.

The geographical information, in the form of a two-dimensional map, is back-projected onto the desk, so that the user seated at the desk can see it. The user can move and orient the map using phicons. One of the phicons represents MIT's Great Dome, and when it is placed on the desk, the map is adjusted so that the position of the Great Dome corresponds to that of the phicon. As the user moves the phicon, the system adjusts the map to ensure that the phicon is always aligned with the point on the map that it represents. By moving the phicon around on the desk, the user can cause the map to move too, "scrolling" around in the geographical space. By rotating the phicon on the desk, the user can cause the map to rotate.

If a second phicon is added to the desk, say one representing the Media Lab building itself, then another degree of freedom can be constrained. The two icons, together, can be used to control the scale of the

map display. If the metaDESK always ensures that the virtual Great Dome always co-occurs with the Great Dome phicon, and the virtual Media Lab always co-occurs with the Media Lab phicon, then the user can control the scale of the map by moving these two phicons closer together or further apart.

The active and passive lenses can be used to provide access to other sorts of information. In the Tangible Geospace example, the active lens is used to view a three dimensional model of the MIT Campus. The active lens is a computer display mounted on an arm over the desk. It is instrumented so that the metaDESK computer system can determine the position and orientation of the display. When this information is coordinated with the current position, scaling, and orientation of the map being displayed on the desk, the result is that the active lens can be used to control a "virtual camera" moving through the geographical space being displayed on the metaDESK. When this is combined with a three dimensional model of the campus, then the active lens can be used to give a three-dimensional viewport onto the two-dimensional map. The illusion is of "looking through" the lens and seeing a transformed view of the map underneath.

The passive lens works in a similar way, although it rests on the desk surface. The passive lens is simply a piece of transparent plastic. As it is moved around the desk, the computer system can track its current location. On the desk area directly underneath the lens, the metaDESK replaces the map with a view onto a photographic aerial record of the campus. As before, this is correlated with the current position, scaling, and orientation of the basic map, as well as the position of the lens. The effect is that it seems to the user that the lens reveals the photographic model underneath as it moves across the desk. This is similar to a user interface technique known as "magic lenses" (Bier et al. 1993), user interface components that selectively transform the content of interfaces as they are moved across the screen, although, of course, in the case of the metaDESK the lens has a physical manifestation.

The Ambient Room

Tangible interfaces such as the metaDESK explore interaction that is situated in the environment, rather than on a screen. This is even more

clearly demonstrated by another of the MIT prototypes, called the Ambient Room (Wisneski et al. 1998).

The Ambient Room is a small office cubicle that has been augmented with a variety of "ambient displays," designed to provide peripheral, background information to the occupant of the room without being overwhelming or distracting. Examples of ambient displays include projected light patterns, non-speech sounds, and objects that respond to changes in air flow.

The information that the Ambient Room conveys is typically information about activities in either physical or virtual space, such as the presence or activity of others, e-mail arriving, people logging in and out, and so forth. These can be mapped onto the displays available in the room. For instance, light patterns projected on the wall can respond to the activities of a networked computer system, conveying information about network traffic and hence activity in the virtual space; or movements in a shared project room can be mapped onto subtle sounds in the Ambient Room so that the occupant can be aware of comings and goings in the project space. Reminiscent of the Feather, Scent, and Shaker work of Strong and Gaver, these ambient displays can be used to project the actions in one space (either physical or virtual) into another; like the technologies of the Reactive Room, they can also respond to the activity of the room's occupant, providing a display that is appropriate to the context in which they are working.

It is tempting to think of the metaDESK as exploring the potential for tangible media as input technologies, and the Ambient Room as exploring their potential for output. To do so, though, would be to miss an important point, which is that, in the everyday environment, "input" and "output" are fundamentally interconnected. This is a critical feature of the tangible media explorations. They should be characterized not in terms of "input" and "output," but in terms of the coordination between phenomena; between activity in a space and the pattern of light on a wall, or between the movement of objects on the desk and the information presented there. This sort of coordination, or coupling, is fundamental to the explorations presented here; they depend upon it for the causal illusion they want to maintain.

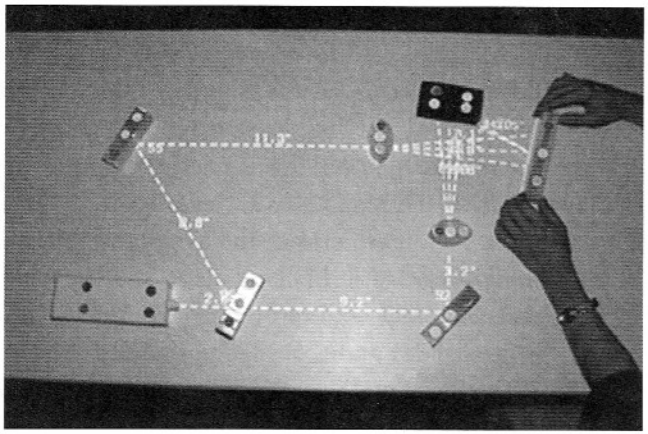
Illuminating Light and Urp

Two other applications developed in the MIT group echo the Digital Desk in their creation of mixed physical/virtual environments for task-focused work. These are Illuminating Light and Urp, both developed principally by John Underkoffler (and illustrated in figure 2.5).

Illuminating Light (Underkoffler and Ishii 1998) is a simulation of an optics workbench, aimed particularly at students of laser holography. The interface is based on a combination of phicons and a camera/projector arrangement (which Underkoffler dubs the "I/O Bulb") similar to that of the Digital Desk. The application allows users to experiment with and explore configurations of equipment for laser holography. Real laser holography is a complex business, conducted using delicate and expensive instruments. Setting up and fine-tuning an experimental configuration can be extremely time-consuming, especially for novices. Illuminating Light allows holographers to simulate the effects of particular configurations and to explore them so as to develop a better intuitive sense for the interaction of their elements. Phicons represent physical elements such as lasers, lenses, mirrors, and beam-splitters, while the system provides a simulation of light paths through the experimental equipment, showing light emitted by the laser, redirected by mirrors, and so on. As the phicons are moved around a physical surface, the system continually updates its projection of the simulated light paths to reflect the moment-by-moment physical configuration. In addition to the simulated light beams, the system can also provide numerical descriptions of the configuration; incidence angles, distances, and so forth. In this way, users can rapidly explore a variety of configurations and develop an understanding of the consequences of different changes on the set-up.

Urp (Underkoffler and Ishii 1999) is an urban planning workbench in which physical models of buildings are combined with electronic simulations of features such as air flow, cast shadows, reflectance, and so forth. The underlying technology is similar to that of Illuminating Light but applied to a different domain. There are two sorts of phicons used in Urp. The first represent building structures. By placing these on the surface, the user can obtain a visualization of the shadows that the buildings will cast, or the wind patterns around them. Combining multiple structures allows urban planners and architects to explore the

(a)



(b)

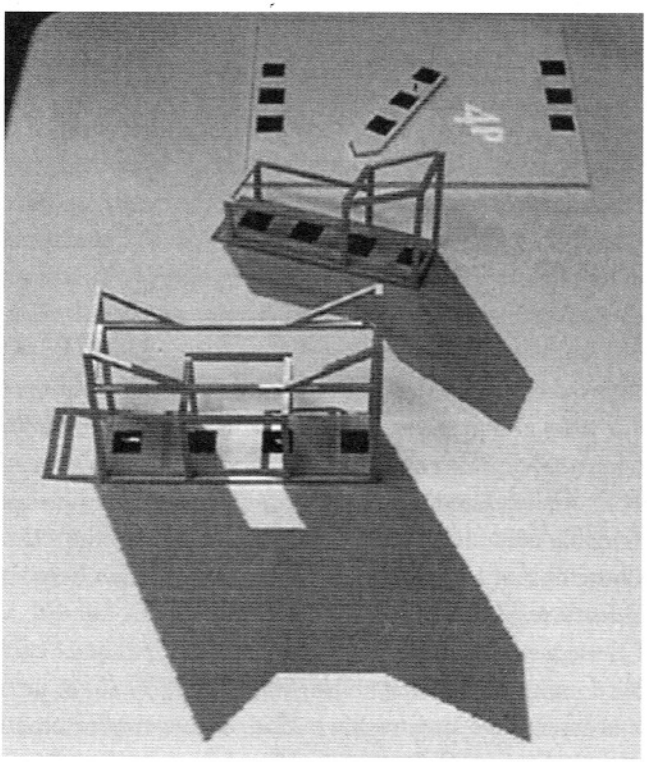


Figure 2.5
Illuminating Light (*a*) and Urp (*b*) apply tangible interaction techniques to the domains of optics and urban planning. Reprinted by permission of The MIT Media Lab.

interactions of wind, reflection, and shadow effects in an urban landscape. As with Illuminating Light, real-time tracking of the position and orientation of these phicons allows the system to update the display continuously, so that users can move the buildings around or rotate them until they find a satisfactory arrangement. The second set of phicons act as controls for the simulation. For example, a "wand" can be used to change the material of the buildings, so that the computed reflectance patterns will simulate buildings clad in brick or glass, another controls the direction of the simulated wind, while a "clock" has hands that can be moved to specify the time of day and hence the position of the sun for the shadow simulation. In this way, the simulator's controls are introduced into the same space that is the focus of the system's primary input and output.

Interacting with Tangible Computing

Tangible computing takes a wide range of forms. It might be used to address problems in highly focused and task-specific work, or in more passive awareness of activities in the real world or the electronic. It might attempt to take familiar objects and invest them with computation, or it might present us with entirely new artifacts that disclose something of the hidden world inside the software system. The bulk of this chapter has explored a range of tangible computing systems, but the survey has been far from comprehensive; indeed, I have said nothing about whole areas, such as wearable computing and context-based computing, that are clearly strongly related. My goal, however, was not to provide a catalogue of tangible computing technologies, but rather to introduce a sample of the systems that have been developed, and to begin to look for some common features of their design.

The first of these general issues that we see across a range of cases is that, in tangible computing, there is no single point of control or interaction. Traditional interactive systems have a single center of interaction, or at least a small number. Only one window has the "focus" at any given moment; the cursor is always in exactly one place, and that place defines where my actions will be carried out. Cursors and window focus insure that the system always maintains a distinguished component

within the interface, which is the current locus of interaction. To do something else, one must move the focus elsewhere. When computation moves out into the environment, as in the tangible computing approach, this is lost. Not only is there not a single point of interaction, there is not even a single device that is the object of interaction. The same action might be distributed across multiple devices, or, more accurately, achieved through the coordinated use of those artifacts. Imagine sitting at your desk to write a note. The writing comes about through the coordinated use of pen, paper, and ink, not to mention the desk itself and the chair you sit in; you might write on the page with your dominant hand while your nondominant hand is used to orient the page appropriately. These are all brought together to achieve a task; you act at multiple points at once. In the same way, ubiquitous computing distributes computation through the environment, and, at one and the same time, distributes activity across many different computational devices, which have to be coordinated in order to achieve a unified effect.

A related issue is how tangible interaction transforms the sequential nature of interaction at the interface. The single point of control that traditional interfaces adopt leads naturally to a sequential organization for interaction—one thing at a time, with each step leading inevitably to the next. This ordering is used both to manage the interface and to simplify system development. For instance, "modal" dialog boxes—ones that will stubbornly refuse to let you do anything else until you click "okay," "cancel," or whatever they need—both structure your interaction with the computer, and save the programmer from the need to handle the complexity of worrying about other actions that might transform the system's state while the dialog box is displayed. When we move from traditional models to tangible computing, sequential ordering does not hold. It is not simply that interaction with the physical world is "parallel" (a poor mapping of a computational metaphor onto real life), but that there is no way to tell quite what I might do next, because there are many different ways in which I might map my task onto the features of the environment.

These two issues are particularly challenging from a technical perspective, because they address the programming models we use to develop systems, embedded in software toolkits and applications. The

third feature of tangible interaction may, however, provide some relief. This is the fact that, in tangible design, we use the physical properties of the interface to suggest its use. This is nothing new; arguably, it is what product design or other forms of physical design are all about. Kettles are designed so that we can tell how to safely pick them up; remote controls are designed to sit comfortably in the hand when oriented for correct use (at least when we're lucky). What is more, this sort of design that recognizes the interaction between the physical configuration of the environment and the activities that take place within it can also be a way to manage the sequential issues raised earlier. For instance, Gaver (1991), in his discussion of "sequential affordances" (which will be presented in more detail in chapter 4), gives the example of a door handle, which, in its normal position, lends itself naturally to turning and then, in its turned position, lends itself naturally to pulling; the whole arrangement helps "guide" one through the sequential process of opening the door through careful management of the physical configuration of the artifact. Taking this approach, designers can create artifacts that lead users through the process of using them, with each stage leading naturally to the next through the ways in which the physical configuration at each moment suggests the appropriate action to take. The relationship between physical form and possible action can give designers some purchase on the problems of unbounded parallel action.

Interacting with tangible computing opens up a new set of challenges and a new set of design problems. Our understanding of the nature of these problems is, so far, quite limited, certainly in comparison to the more traditional interactional style that characterizes most interactive systems today. The theories that govern traditional interaction have only limited applicability to this new domain. At the same time, tangible computing has been explored, largely, as a practical exercise. Most prototypes have been developed opportunistically, driven as much by the availability of sensor technology and the emergence of new control devices as by a reasoned understanding of the role of physicality in interaction. We have various clues and pointers, but there is no theory of tangible interaction. Why does tangible interaction work? Which features are important, which are merely convenient and which are simply

wrong? How does tangible computing mediate between the environment and the activity that unfolds in it?

This book is about developing answers to these questions. The interpretation that it will offer is one that is concerned not just with what kind of technology we use, or with what sorts of interactions we can engage in with that technology, but about what makes those interactions meaningful to us. From this perspective, the essence of tangible computing lies in the way in which it allows computation to be manifest for us in the everyday world; a world that is available for our interpretation, and one which is meaningful for us in the ways in which we can understand and act in it. That might seem to be quite far removed from looking at application prototypes, reactive rooms, and digital desks. The path from practice to theory will be easier to see after looking at the second aspect of embodied interaction—social computing.