Foreword

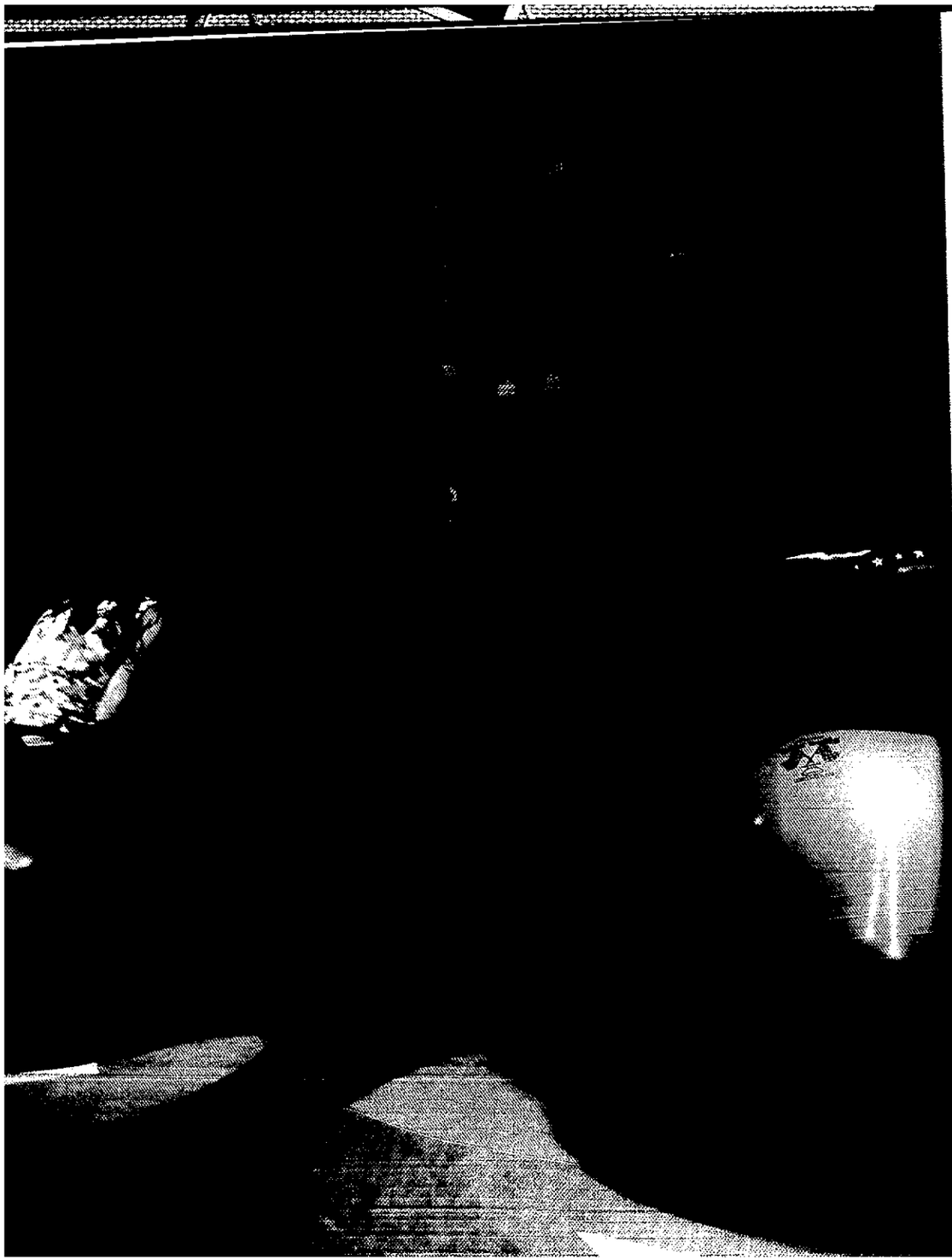
What Is Interaction Design?

by Gillian Crampton Smith

Photo Ivan Gasparini

Gillian Crampton Smith

Gillian Crampton Smith, the director of Interaction Design Institute Ivrea, is the foremost academic in the emerging discipline of interaction design. She studied philosophy and art history at Cambridge University, graduating in 1968. She spent the next decade as a designer, first in book publishing, and then on the *Sunday Times* and *Times Literary Supplement*. In 1981, at the leading edge of desktop publishing, she designed and implemented a page layout program to help her with magazine design. This experience convinced her that designers have an important role to play in creating information technologies. In 1983 she joined the faculty of London's St Martin's School of Art and established a graduate program in graphic design and computers for practicing designers. In 1989 she moved to the Royal College of Art, Britain's only purely graduate school of art and design, and set up the Computer Related Design Department with advice from Bill Moggridge, the external assessor for the program. Now called the Interaction Design Department, this was the first program in the world where graduate designers could learn to apply their skills to interactive products and systems. Under her guidance, the CRD Research Studio achieved an international reputation as a leading center for interaction design. In 2001 she moved to Ivrea—the Italian town in the foothills of the Alps famous as the home of Olivetti—to establish Interaction Design Institute Ivrea,[1] which offers the world's first post-experience interaction design program.



"Collabolla," a video game with Spacehopper balls as input devices. Interaction Design Institute Ivrea, Triennale di Milano, 2004.

Photo
Ivan Gasparini

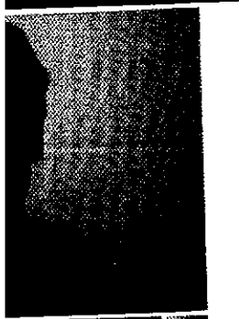
In the same way that industrial designers have shaped our everyday life through objects that they design for our offices and for our homes, interaction design is shaping our life with interactive technologies—computers, telecommunications, mobile phones, and so on. If I were to sum up interaction design in a sentence, I would say that it's about shaping our everyday life through digital artifacts—for work, for play, and for entertainment.

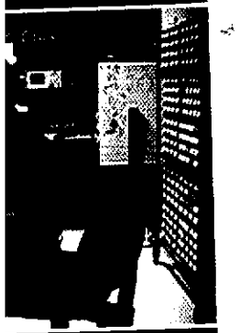
Gillian Crampton Smith, interview of January 30, 2002[2]

Designing for Everyday Life

TWENTY YEARS AGO, when personal computers were first becoming popular, they were mostly used as professional tools, or games machines for teenagers. The situation has changed radically. Now everybody—kids, parents, grandparents—uses them every day, at work, at school, and at home. So today we need to design computer technology differently, to make it a graceful part of everyday life, like the other things we own: our clothes, the plates we eat off, the furniture we buy for our houses. We've come to a stage when computer technology needs to be designed as part of everyday culture, so that it's beautiful and intriguing, so that it has emotive as well as functional qualities.

This book traces how the design of the way people interact with computer technology developed: from the earliest days of Star, the first screen-based graphical user interface and the precursor of the Apple and Windows interfaces, to the plethora of mobile multimedia devices and systems we use now. It describes the challenges designers face in making this powerful technology fit easily into people's everyday lives, rather than forcing their lives to fit the dictates of technology.





idle
enthusiast—EE Lab at IDEO
professional—medical equipment
consumer—iPod at Apple Store

Three Stages of Technology Use

DAVID LIDDLE,[3] WHO led the team that designed the Star graphical user interface, has talked about three stages in the development of a technology—of photography, for instance, or computers—and how people interact with it. The first stage is the *enthusiast* stage. Enthusiasts don't care if the technology is easy or hard to use because they're so excited by the technology itself or by what it will do for them. They want it, however difficult it is to use.

The second stage is the *professional* stage, when those who use the technology are often not those who buy it. Office computers, for example, are usually chosen by a purchasing department, not by their users; the purchasers don't care about the difficulty because they don't experience it, and are anyway more interested in factors like price, performance specifications, or after-sales support. At this stage, indeed, some people even have a vested interest in the technology being difficult because they're selling their ability to use it; the harder it is, the more valuable their skills.

The third stage Liddle identifies is the *consumer* stage. People now are less interested in the technology in itself than in what it can do for them. They don't want to spend much time learning how to use it and hate being made to feel stupid. So if it's hard to use, they won't buy it. This is the current stage in the use of computer and telecommunications technology: it's no longer used only by professionals but by a wide range of nonexperts, who just want to use it to pursue their everyday lives.

In the past, those who built interactive systems tended to focus on the technology that makes them possible rather than on the interfaces that allow people to use them. But a system isn't complete without the people who use it. Like it or not, people—irritable, demanding, and often distracted people like ourselves—and their goals are the *point* of our systems, and we must design for them.

Designing for this new broad spectrum of humanity is more challenging than devising specialist tools for technical professionals. Our users are, justifiably, not prepared to spend time mastering tricky new systems. And they're not obliged to use our products: if they can't make them work, they take them back to the store.

From Usability to Sociability

MANY ADVERTISEMENTS NOW boast about computer technologies that are easy to use: *usability* has become a buzzword. But usability is only the first of the qualities we should expect from the systems we use; they also need to be *useful*. This sounds obvious, but too many systems don't really help people do what they want to do. Mitch Kapor, creator of the hugely successful spreadsheet Lotus 1-2-3, proposed an "architectural" model of software design, distinguishing design from engineering and building. His 1990 "Software Design Manifesto"[4] reminded us that we must start by thinking about designing things so that they're right for people, rather than by thinking first about how to build it. He admitted he was no software engineer, no ace at writing code. Rather, his role was to design what Lotus 1-2-3 should be and do, making sure that this was what people needed. Others more skilled in software engineering ensured that it worked. Lotus 1-2-3 wasn't the first spreadsheet, but it was the first that really did what people needed in a way that fitted how they worked. Thus its success.

That said, there's more to living than utilitarian needs and the functions which satisfy them. As computers begin to shape everyday life, we're interested not only in what this technology can do for us, but also in what owning it *means* for us. When we buy something for our home, a toaster for instance, we choose it because it toasts bread, certainly, but maybe also because of how it looks, feels, sounds. What does it *say* to us? Is it satisfying? Does it enrich, by however little, our just-crawled-from-bed state of mind?

And of course we choose the things we surround ourselves with not just because of what they mean to us, but also because of what they mean to other people. Most Italians have a mobile phone but many young Sicilians, for instance, can't afford the calls. They still buy the phones, though, because sporting one says, fairly explicitly, "I'm connected to a network of family and friends." The symbolic function is as important as the practical one, perhaps more.

The interactive systems we design have implicit as well as explicit meanings. A design may communicate its purpose clearly, so that it's obvious what it is and what we should do with it. But its *qualities*, its aesthetic qualities particularly, speak to people in a different way. Consciously or not, people read meanings into artifacts. A chapel speaks a different architectural language than a supermarket, and everybody can read the difference. In a drugstore we can usually distinguish a medicine bottle from a perfume bottle even if we can't read the label. Artists and designers are trained to use the language of implicit meanings to add a rich communicative element over and above direct functional communication. If we only design the function of something, not what it also communicates, we risk our design being misinterpreted. Worse, we waste an opportunity to enhance everyday life.

To designing for usability, utility, satisfaction, and communicative qualities, we should add a fifth imperative: designing for *sociability*. When IT systems fail to support the social aspect of work and leisure, when they dehumanize and de-civilize our relationship with each other, they impoverish the rich social web in which we live and operate, essential for both well-being and efficiency.

The technologies we design can erode or enhance this social web, so we must design for this explicitly, because technologically driven social changes can be creative. When young people are out socializing they are reluctant to make appointments. They say "Oh yeah, I'm around tomorrow. Don't know when. Give me a call, I'll see where I am."

The mobile phone has brought to hanging out—indeed, to time itself—an altogether more fluid and relaxed approach.

Good Interaction Design

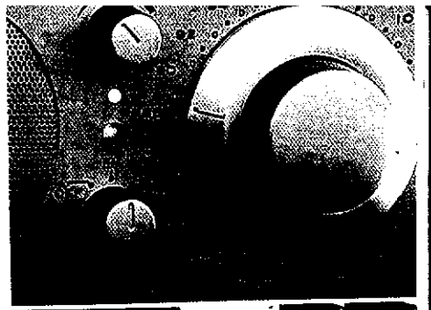
AN ELECTROMECHANICAL OBJECT, a radio say, links its physical mechanical components to its electronic elements in a fairly direct way. When we turn the dial, our fingertips and muscles can almost "feel" the stations being scanned. With computers, however, the distance between, on one hand, keystrokes and screen image, and, on the other, what's happening inside the computer, is usually much less direct. Our physical world and the computer's virtual world seem miles apart.

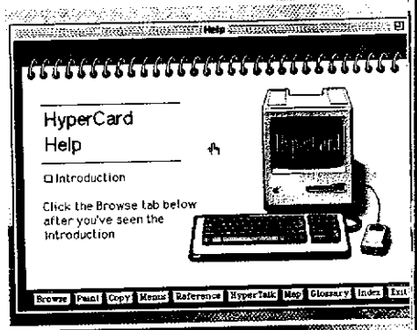
In this (historically unprecedented) situation we need a *clear mental model* of what we're interacting with. HyperCard,[5] for instance, an early scripting system on the Apple, had a very clear mental model, a stack of cards: a precise analogy of what and how the program worked. It was obvious to its users that in effect they were flipping through a stack of cards: everything about the design reinforced this metaphor. Sadly, the same can't be said of many other applications.

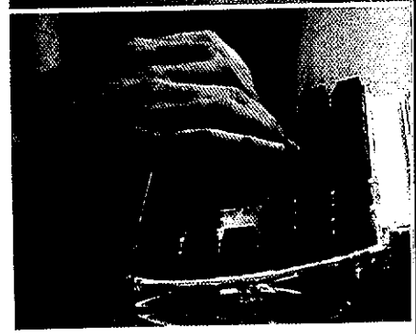
A well-designed system has *reassuring feedback*, so that we know what we've done when we've done it. On a keyboard, for example, we can tell what we've just done because not only do characters appear on the screen but we can the feel the travel of the key itself and hear the little click it makes. Using an early word processor to do something repetitive, I often had to do a sequence of key commands that went "tetick, tick, tick-tick; tetick, tick, tick-tick." If it went "tick, tetick, tock," I'd know I'd made a mistake. The aural feedback let me go faster than if I'd relied just on my eyes.

Navigability is also essential, particularly with things that are primarily on screen. You need to know where you are in the system, what you can do there, where you can go next, and how to get back. The Star and Macintosh interfaces were very influential in this way. The menu at the top of the screen lays out all the possibilities; it's clear how you access them and what will happen when you do.

Equally crucial is *consistency*. A certain command in one part of the system should have the same effect in another part. An example, again from some time ago, was Appleworks, one of the







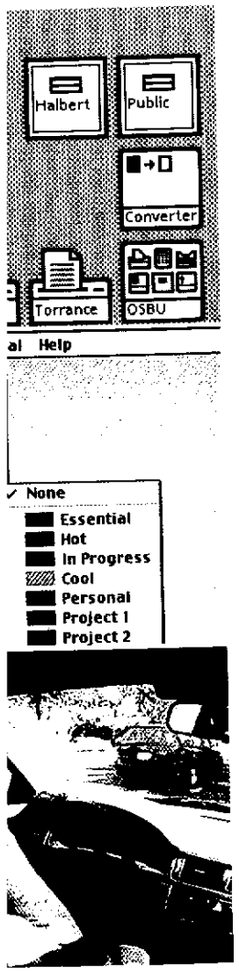
Radio—Henry Kloss ■
Radio dial mechanism ■
HyperCard ■
Keyboard ■

first integrated office programs on the Apple II. Those were the days of green "ransom-note" characters on a black screen, and very limited functionality. But Appleworks was beautifully, satisfyingly, consistent. You knew exactly what to do. A command in the database did exactly the same in the word processor; wherever you were, the escape key took you back up a level. You never got lost and rarely made a mistake. Compare that with modern "integrated" applications. Consistency, like all forms of satisfying simplicity, is very difficult to achieve.

When we interact with everyday artifacts, like a car, we don't spend too much time thinking about the interaction: we think about where we're heading and what we want to do. *Intuitive interaction* minimizes the burden of conscious thought needed to operate the system, leaving us to concentrate on our goals. A good example was Quark Express, which let you almost unconsciously zoom in on your image by holding down two keys and clicking on what you wanted to see better. It was like shifting your gaze: you didn't have to march off somewhere to find the right tool. But too many systems still keep demanding too much attention, like incompetent bosses, distracting us from getting on with the job.

When we design a computer-based system or device, we're designing not just what it looks like but how it *behaves*. We're designing the *quality* of how we and it interact. This is the skill of the interaction designer. It's partly responsiveness: when you move your mouse, for instance, does it feel sluggish, or nippy and sprightly? When you manipulate your iPod dial, the combination of sound and feel, as well as telling you what you're doing, is subtle and satisfying. We can design those qualities of interaction, relating what we see to what we hear or feel with the same refinement with which typographers adjust the spacing of type, or product designers the radius of a curve.

But the qualities of interaction must be appropriate to the context. An adventure game needs an interaction offering subtlety of atmosphere and intriguingly challenging navigation; central-heating control systems offering these qualities, however, would be as welcome as a fire alarm with a snooze button.



r interface
cintosh menus

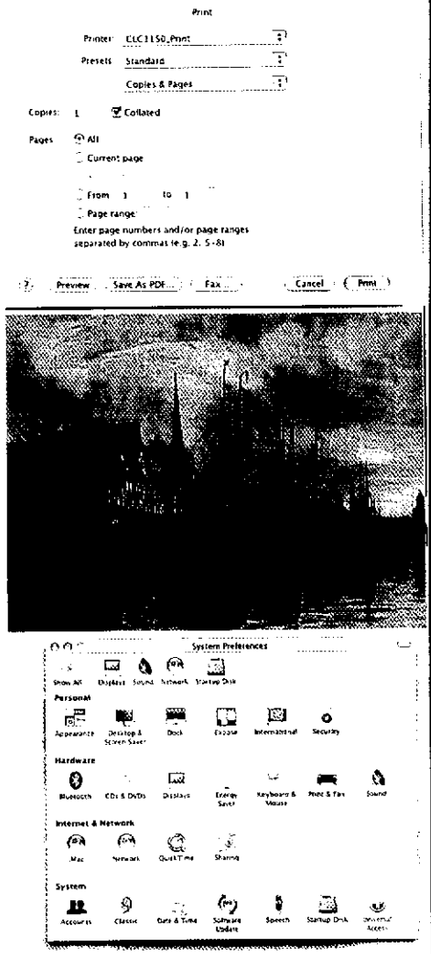
Languages of Interaction Design

When new technologies are born, we tend to think of the new in terms of the familiar. When cinema started, people thought of it as pointing a camera at a theater stage, and divided silent films with "chapter headings" as if they were books. New "languages" eventually emerged that were true to, and fully exploited, the unique qualities of cinema itself—Eisenstein's language of montage, for instance. But the old analogies never lose their validity: films continue to use the conventions of the theater and the novel. They are just augmented by the new languages.

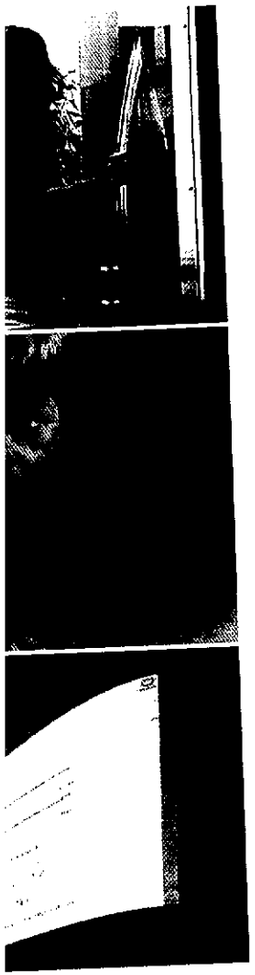
I believe that interaction design is still in the equivalent of the early stages of cinema. As yet, we have no fully developed language unique to interactive technology. So we are still drawing on the language of previous creative modes. It may help to categorize these languages according to their "dimensions": 1-D, 2-D, 3-D, and 4-D.

1-D includes words and poetry. Are the words in a menu the most accurate encapsulations of the action they denote? Are they used consistently? And the "tone of voice" of the dialog boxes in your system: Are they too abrupt and imperious, or too cloyingly conversational?

The 2-D languages that interaction design can borrow from include painting, typography, diagrams, and icons. When we look at a painting, even if it's not representational, it's difficult not to interpret it as a perspectival space; we can use such compositional tropes to layer the screen in apparent depth or to foreground its currently most important element. We can use the familiar hierarchical conventions of typography to structure the screen, and our shared sensitivity to minute differences in letter forms to add distinctions of tone and meaning. We can also use the language of diagrams and information graphics to communicate a complexity which can't be intelligibly rendered in standard text, particularly on a small screen. Another specialist 2-D language, much used in



Dialog box ■
Painting ■
Icons ■



handle
ver
tion

computer interfaces, is of course that of icons: tiny simplified images that stand for a larger idea or a thing.

3-D languages are those of physical, sculptural form. One movement in product design, "product semantics," explores how people understand what the different elements of a product represent. If something has a handle, for example, we know we are meant to grab it; if something has a base bigger than its apex, our experience of gravity suggests that we should keep the base downward. Designers use this language to make things clear, but sometimes also to play with expectations, inserting an element of surprise and wit in what otherwise might have been mundane.

The fourth dimension is time. The 4-D languages include sound, film, and animation. In the 1980s Bill Gaver[6] designed a beautiful sonic interface, the SonicFinder, an augmentation of the Apple Desktop: when you dropped a folder into another folder, it made a sound according to its size: an almost-empty folder went "pink," a fuller one "plonk." It gave good feedback, but the sounds were also poetic and appropriate for their purpose. Another important 4-D language is film: in twenty seconds a TV advertisement can tell a complex story understood by everyone. And animators have been developing their spare language for more than a century, so that with very limited means they can express plot, emotion, anticipation, and action.

We're designing for a public that understands the richness of all these different languages: dialog, graphics, typography, 3-D form, sound, film, and animation. This makes things difficult because nobody can be fluent in all these languages. We must collaborate with those who have other skills and experience. An interaction designer can never be a hermit.

However, after twenty years of drawing on existing expressive languages, we now need to develop an independent language of interaction with "smart systems and devices, a language true to the medium of computation, networks, and telecommunications. In terms of perceptual psychology, we're starting to understand the functional limits of interaction between people and devices or systems: speed of response, say, or the communicative capacity of a small screen. But at the symbolic level of mood and meaning, of sociability and civility, we haven't quite achieved the breathtaking innovativeness, the subtlety and intuitive "rightness," of Eisenstein's language of montage.

By telling the stories of those who have been committed to making interactive products useful, meaningful and joyful, however, this important book nevertheless suggests that we are on our way.