Design and Magic Cap™

April 24, 2000

Design and Magic Cap

Copyright © 1998-2000 Icras, Inc. Portions copyright © 1992-1998 General Magic, Inc.

All rights reserved.

No portion of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means—electronic, mechanical, photocopying, recording, or otherwise—without the written permission of Icras, Inc. (“Icras”)

(version 4/4/00)

License

Your use of the software discussed in this document is permitted only pursuant to the terms in a software license between you and Icras.

Trademarks

Icras, the Icras logo, DataRover, the DataRover logo, Magic Cap, the Magic Cap logo, and the rabbit-from-a-hat logo are trademarks of Icras which may be registered in certain jurisdictions. The Magic Cap technology is the property of General Magic, Inc., and is used under license to Icras, Inc. Apple, the Apple logo, Mac, Macintosh, and MPW are registered trademarks of Apple Computer, Inc.

All other trademarks and service marks are the property of their respective owners.

Limit of Liability/Disclaimer of Warranty

THIS BOOK IS SOLD “AS IS.” Even though Icras has reviewed this book in detail, **ICRAS MAKES NO REPRESENTATION OR WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS BOOK. ICRAS SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OR MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE AND SHALL IN NO EVENT BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGE, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES**. Some states do not allow for the exclusion or limitation of implied warranties or incidental or consequential damage, so the exclusions in this paragraph may not apply to you.

Magic

David Hendler wrought this. He was abetted by the Magic Cap team and committed Magic Cap developers.

Flattery is very long-winded in certain cultures

This product is “commercial item” as that term is defined at 48 C.F.R. 2.101 (OCT 1995) consisting of “commercial computer software” and “Commercial computer software documentation,” as such terms are used in 48 C.F.R. 12.212 (SEPT 1995) and is provided to the U.S. Government only as a commercial end item. Consistent with 46 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (JUNE 1995), all U.S. Government End Users acquire this product only with those rights set forth therein.

Icras, Inc.

955 Benecia Avenue
Sunnyvale, CA 94086 USA

Tel.: 408 530 2900
E-mail: info@icras.com
Fax: 408 530 2950
URL: http://www.icras.com/

Contents

1

Introduction

Design and Magic Cap is for software developers and people who are interested in how to make software using Magic Cap. The book's different sections cover ways to make your software look good, feel approachable, and work well for your users.

What this book contains

This book captures some of the things that people have learned in designing Magic Cap software. You might find that the ideas here help you in creating the interface for your packages and objects. You'll likely have to extend your work beyond the topics covered here to tune and refine your particular package. As you do, you'll discover new things that you can do and good ways of doing them.

send your comments

Please let Icras know about things you learn during development, because this document also contains comments and ideas from software developers outside Icras—people working on the Magic Cap platform and trying out things that no one has before.

Skip around in this document as you will. Some of the sections may have nothing to do with your work; some of the later sections might be worth reading early on.

related works

This document assumes you are familiar with Magic Cap software and with the way it works. If you aren't, you might want to consult the user's manual for a particular Magic Cap device, like the book *Magic Cap Complete*. This guide describes the general ideas behind the elements of Magic Cap software, not the elements themselves. *Design and Magic Cap* covers principles and examples, not instructions for implementation. See the comprehensive technical documentation for details about implementing the things you read about here.

2

Essential principles

Good ideas are often not rules. Keep in mind that this is a collection of ideas, not a rule book. The quality of the software you make depends unequivocally on the work you do. You should go beyond these guidelines whenever you find you can change your design to make users more comfortable, to make their interaction with Magic Cap faster, more convenient, or more efficient.

When you encounter a rule or suggestion written this way, don't follow it blindly. But do know that people who write Magic Cap software believe what it says.

You can gain by following the spirit of the guidelines discussed here. Many people who are writing Magic Cap software—including the manufacturers and the designers at Icras—will have this guide at hand, so your software will seem more familiar to people who have used Magic Cap if it takes advantage of some of the techniques described here.

There are three central themes in pursuing any user-interface project: consistency, usability, and simplicity. Although the three categories overlap—a simple window containing one button is likely very usable—you'll find that they are often at odds. For example, you might be writing a grocery list program for Magic Cap and want to have the exact layout of the local grocery store represented on the screen. But the grocery store probably has too many aisles to fit comfortably on the screen. To use the software, the user might need a two-level hierarchy—much more complicated than a simple map. Constructing a grocery list might be easiest if an inconsistency is introduced: staples are marked on a map but obscure items—cocktail sausages and wasabi—are only to be found by searching.

other developers

Magic Cap is not yours alone. Your software might require the user to enter a room and to learn special tricks, but the rest of Magic Cap is never more than a single tap away. More than on any traditional computer, you share Magic Cap with other developers. It's worth thinking about how what you do fits into Magic Cap as a whole. What you do to the screen and what you ask your users to do will influence how they see the device as a whole. When you create objects that can move from scene to scene, you exert particularly strong influence on how people understand the way objects work throughout Magic Cap.

users' expectations

The decisions you make about your software affect other software designers and their work, particularly in the users' minds. If your package works consistently with the other software packages for Magic Cap, users will find your package more usable and other Magic Cap packages easier to understand.

Treat others better than you would be treated. Since you are a software designer, you probably have built up a tolerance for all kinds of computer software—good, bad, difficult, and incomprehensible. But when you're writing for Magic Cap, you can expect many of your users to have no such tolerance. A number of them will never have used a computer; many more will have been put off from computing by precisely the kinds of programs that you use every day. When you write software for Magic Cap, you're developing a software *package*— collections of objects and their methods, designed to serve some cogent set of purposes. Packages resemble computer applications in many ways. The Magic Cap datebook package is like a computer planning application, for example. However, because each of the objects within a package is relatively autonomous, it can function outside of the package—the element within which you supply it. Icras uses the word *package* rather than *application* to emphasize to users that Magic Cap software is really a collection of interacting objects, not a monolithic program.

packages

Computing Magic Cap

Magic Cap software won't always be seen as computer software. You know it runs on a computer. Icras knows it runs on a computer. But many of the people for whom Magic Cap is designed are nervous about using computer software—not about turning on the

toaster, not about starting their new car's engine, not about making a long-distance telephone call, even though all these acts often involve using software. They are nervous quite explicitly about using computer software.

The Magic Cap interface, when it is successful, keeps the computer in the wings and puts onto the stage something people feel they can handle without a course in programming. Cars, telephones, and toasters are not intimidating because they hide their computers behind interfaces that suit the tasks of driving, conversing, and browning bread. Even a common consumer electronics device like the fax machine, which contains a computer, has a simple, limited interface that makes that machine a special-purpose device and less threatening than a vaguely powerful computer might be. Software packages within Magic Cap can be tuned to what the user wants to do. The datebook is for scheduling, so its controls have more to do with scheduling than with directly manipulating bits. Your software package will have a particular focus and a particular audience, so you can design the interface around them.

Keep it simple

Because the screen of the standard Magic Cap device is small, you'll find that even simple screen layouts often look cluttered. Your users will remain happy when you present a small amount at a time and present it legibly and clearly. As you think about adding an image, a control, or a piece of text, consider that it will almost certainly make your software look more complicated. Does the element pay for its presence? Will your users tolerate it? Leaving your software simple might make it more powerful, because more people will use it and they might even use it more often.

Adding any element can make a software package visually more complicated. Does the element pay its way by removing complexity in use or in understanding?

Know your audience

Even if you intend to give away your software package for free, your design will work its way into good software only if you constantly think about the people who will use it. Even better, you can involve them from the very beginning of your project. Pick out potential customers and show them your sketches, tell them your ideas. Then listen to what they have to say. They might give you advice that you never take, but their opinions and ideas can shape your work all the same.

You're making your software for people. Let them help you decide what is good.

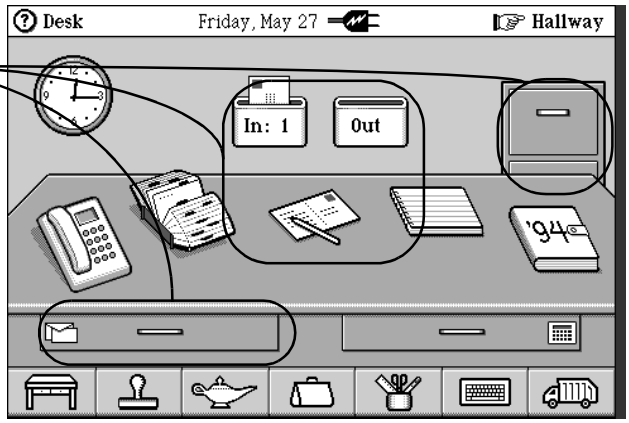
3

Guided tour

This tour of creating, writing, and sending a message points to some of the interface elements and principles that the user encounters in the course of this task. The tour shows what thoughtful design is for: making software that people like using and that they can use again and again.

At the desk, before touching anything to get started with a new message, the user sees a number of objects. Most of them are modelled on things one might expect to ?nd on a real desk. All of them are stylized so they can be recognized readily. A few of them look like they might have something to do with messages.

These objects have a lot to do with messages. They are drawn to look like everyday, physical objects that have something to do with creating and handling messages. These objects thus explain what they do, relying on the users' experiences and their expectations.



These objects remind the user of paper mail. Magic Cap messages aren't usually paper messages, so this is one of those occasions when the software uses a metaphor to make itself clear.

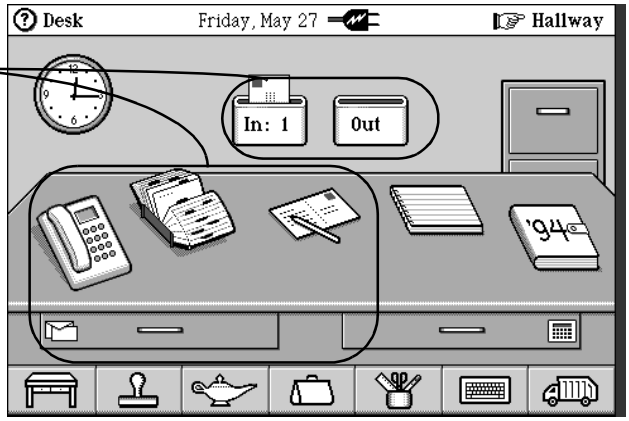
metaphors

- *Rely on analogy with familiar objects rather than your users' ability to learn how new things work.*

position for emphasis

There are five objects called out in the picture of the desk on the previous page. Three of them are essential to getting messages in and out of this Magic Cap device: the in and out boxes and the pen and postcard that create a new message when touched. The other two objects—a stationery drawer and a file cabinet—while important enough to warrant placement at the desk, are slightly less central. The placement of objects in relation to each other reflects their relative importance. In a scene like the desk, the middle of the screen is particularly important visually. In scenes with a lot of text, like a letter seen close up, the top-left corner of the screen is most prominent, since that's where the user starts reading.

These objects focus on communication. Magic Cap is a platform for human communications, so these objects are placed prominently and so they can be used readily.



room for touching

Place important objects prominently—in the center or top left of the screen, for example. Objects in these places draw users' attention immediately. Their placement emphasizes their importance.

To create a new message, the user touches the pen and postcard at the center of the screen.The pen and postcard aren't larger than all the other objects on the screen. However, they are prominently positioned in the screen's center and they have some empty space between them and other objects around them. This space makes it easy for the user to touch the pen and postcard without accidentally hitting another object. The more room around an object, the easier it is to hit with a casual, inexact touch. Most Magic Cap users have to touch the pen and postcard frequently, because touching them is a quick way to create a new message. Sometimes they'll miss. But

The space around the pen and postcard lets the user touch each of them without hitting another object.

the pen and postcard are big enough and there is enough space around them that a miss is unlikely to do anything unintended.

As soon as the user touches the screen, the Magic Cap device makes a short *click* sound. This sound is the device's way of saying *I hear and obey.* Nonetheless, because preparing a new card for the user to address and write might take some time, Magic Cap makes the sound as soon as the user touches the object. The sound is subtle; the device's acknowledgment is more a nod than a fanfare.

Instant feedback assures the user that the device is alert and responsive.

Magic Cap next shows the user that it is creating a new card. A small card hops out of the stationery drawer onto the desk, then expands out to become a large card that fills the screen. The card hops out of the drawer to hint to the user that more stationery will be found in the drawer, not on the desk.

hopping

Hopping is surprisingly important in Magic Cap. It's surprising because most of us manage to get through a day with ordinary walking, occasional skipping, and the rare mad dash. Having objects on the Magic Cap screen hop engages users' eyes by making the objects seem like real things—the trajectories that hopping objects follow is a close simulation of the trajectories that bouncing, hurled, or leaping objects follow in the physical world. Moreover, animation on the screen engages people's attention. Just like frogs searching for their buzzing meals, humans focus automatically on the moving elements in their field of view.

Magic Cap objects can hop to move across the screen quickly and efficiently. Not only does the object's motion explain where it is from and where it is going, but the hopping often demonstrates a way in which the user can drag an object.

After hopping onto the desk, the small piece of stationery zooms to fill the screen. As on television and in the movies, this zoom effect is used to focus the user's attention on the details of a particular item in view.

Visual effects teach users about processes. Seeing an object like a piece of stationery move out of a container indicates that the user can ?nd more of that stationery in the container. Zoom effects help to show that the user is going from one scene to another.

Magic Cap next presents a list of addressees from which the user can choose to whom to write. The list is in a window with a close-*x* to tell it to go away even if the user hasn't chosen anything.

Zoom effects reinforce navigation metaphors. They can be used to show a small object filling the screen, one object—such as a window—emerging from another, or moving from scene to scene.

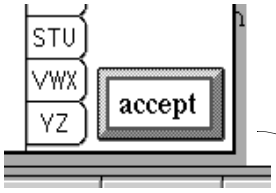
Encouraging choices by strong suggestion is more flexible than forcing those choices through some kind of modality. If your software insists that its users perform a particular act at a particular moment, some of those users will feel bossed around. If instead, you make it clear that it would be a good idea to do something—address a message for example—at a specific moment, you help the human to do what the machine wants without making the machine insist.

Windows and menus in Magic Cap offer the user the option of not making any selection straightaway.

In Magic Cap windows are used to present information that is of interest but won't ?t on the screen or would take up too much space if it were on the screen all at once. Imagine if all of the windows associated with objects at the desk were visible at once: the tote bag would overlap the keyboard, and the stationery drawer might obscure the trash. There just isn't room on the screen to show all the things available within the scene. Windows let some of those things stay in the wings until they are needed.

However, these sorts of windows are unfamiliar to most humans who have spent their lives dealing with objects other than comput-

ers and the occasional Magritte painting. Naïve users sometimes don't understand that the window is meant to look like it is covering what's underneath. To make it clearer that windows are covering something up—and that something will probably still be there when the window goes away—windows are generally drawn with a generous shadow.



The *address to* window has a shadow to show that it is floating over the piece of stationery behind.

Even more important than the shadow in keeping the user's mind within a specific scene is the size of the window. The *address to* window is kept to about a third of the screen size. Many other windows are smaller. These smaller windows are unlikely to shock users into thinking that they have been transported unexpectedly to a new scene.

Shadows should all fall in the same direction. In Magic Cap they care cast down and to the right. Another kind of shadow would imply a second light source in the Magic Cap universe, leading imaginative users to draw conclusions like *Planet Magic Cap orbits a double star*. Most people would just wonder what the funny smudge is for.

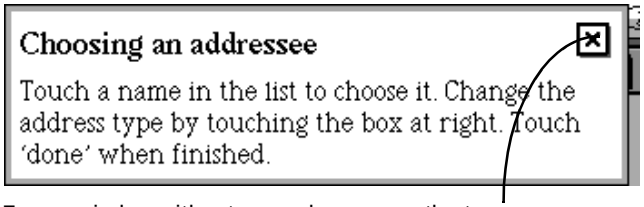
Keep windows small. If you need a lot of space, use a scene.

windows prompt with text

The window for selecting an addressee explains itself by its name, *address to*. The top left corner of a window is a natural place for someone who reads a left-to-right language like English to start

reading to see what the window is for. Putting an instructive name here helps convey the window's point.

Opposite the window's name in the top-right corner, is the close-*x*. Touching the close-*x* makes the window go away. Windows that have gray bars across the top also always have a close-*x*. Some windows have a close-*x* even without the gray bar.



Even a window without a gray bar across the top should have a close-x so that the user knows how to make the window disappear. In this case, a touch anywhere in the window makes the window go away, but the close-*x* is still important to tell users that the window can be closed.

The only windows that shouldn't have a close-*x* are those windows that block all other action while they are on the screen. These windows cannot be closed, but the action they report on needs to finish or be stopped to make the window go away. You might create such a window if your package performs operations that completely take over the communicator, blocking other activities.

The bar across the top of a window is generally gray. Because of this, Magic Cap uses a bold text style for the window's name. At many contrast levels, a name in a text style that isn't bold would be difficult to read in this position. In general, windows without gray bars are the simplest kind, presenting information or a simple two- or three-button choice. Windows that require gray title bars are those that introduce complex choices. A window with attached information needs to have a gray title bar so that the user has a question mark to touch.

User testing

You can see if your suggestions work: give your software to someone to try. User testing is always helpful, is often inexpensive, and

the suggestions that test users give can easily make the difference between a mediocre product and a brilliant one.

> *It pays to try out your software to see if your users recognize pictures and objects as you intend. When you have people test your software, suggest that they talk out loud and tell you what they're thinking, what they expect to happen, and what surprises them.*

basic operations are simple

The *address to* window handles a complicated range of choices by making the most usual selections easy to perform, with few touches. Addressing a message to one person means touching that person's name and then touching *accept*. Expert users can work with the *address to* window in an expert way, holding down OPTION when they touch *accept* to keep the window on the screen.

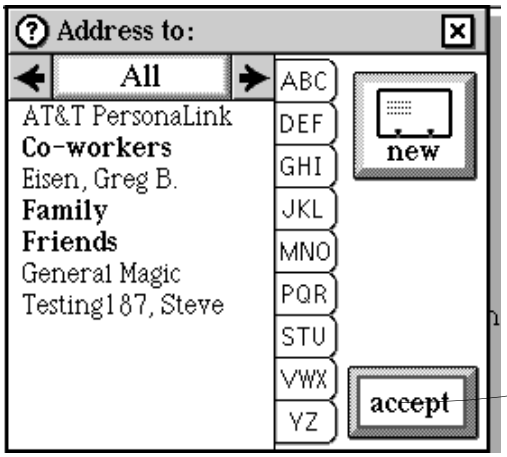
familiar techniques

Addressing a message to more than one person means first addressing it to one person, then touching the word *to:* on the message or the *address* button to go ahead and add an addressee. In business contexts, where electronic communication is probably most often sent to more than one person, this intricate activity might not seem ideal. But Magic Cap is designed for personal messaging first, and personal communication is very often one-to-one. However, even with this optimization for single addressees, the multiple-addressee case is made familiar and straightforward. The same list of people will appear in a similar-looking window.

metaphors can be stretched

The *address to* window, though it appears above the card on the screen, can still contain objects with some visual depth to them.

The *accept* button, for example, is made to look like it sticks a little way out of the window.



The *address to* window includes buttons that look like they stick out of the window's surface.

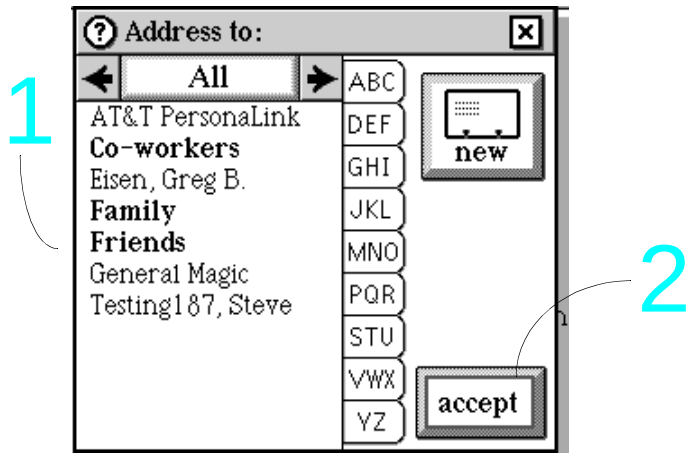
Foolish consistency is the hobgoblin of small minds.
—R.W. Emerson

Magic Cap doesn't adhere rigidly to the physical metaphors it uses. Paper postcards never have buttons sticking out of them. Magic Cap telecards sometimes do. Similarly, though the *address to* window looks like a flat object floating above a card, it can bear objects on it with some physical depth. The physicality of the button encourages the user to think of it as something touchable.

inconsistency can be useful

In choosing a name from the list in the *address to* window, the user touches the name and it becomes highlighted. In order to make that selection stick, the user has to touch the *accept* button in the *address to* window. This select-then-operate technique, familiar from computer interfaces to some people, is at odds with how Magic Cap usually works. The select-then-operate technique was selected for this particular operation after user testing showed that in this particular case, many test subjects wanted to confirm to the Magic Cap device that they did indeed mean to choose the name that they had just chosen. The select-then-operate style is not widely used in Magic Cap, but it does have its place—some choices

are intricate enough that people want to make them—and have the chance to change them—and then confirm with an *accept* button.



The *address to* window works in a select-then-operate way. This technique is not widely used in Magic Cap, since it's pretty uncommon in the physical world and unfamiliar to people who aren't computer users. Nonetheless, it works for this particular window and in this arrangement, which operates top left to bottom right.

The name of the window is placed in the window's top-left corner because the first thing the user needs to know about the window is what it is for. The *accept* button is near the window's bottom-right corner, as it is the last part of the window that the user will manipulate.

- *The order of operation runs from top left to bottom right, as on a page of text.*

accept *and* done

You now know the difference between *accept* and *done* in Magic Cap. *Accept* is confirmation for an individual choice, like the user saying "everything's ready, now go ahead and do it." *Done* is used when the action has already taken place and an entire procedure is complete, as when setting the clock. Once the user changes the clock, it's set. The *done* button provides a way for the user to feel that the action is complete. It also gives the user a way out of a scene or a window. The close-*x* is only appropriate for use in a win-

dow, not for a scene. The *done* button can be used for either. Moreover, *done* can be hidden until the user fills in the necessary information or takes the necessary action, then appear when the procedure is complete. The close-*x* for a window shouldn't disappear and appear.

automatic salutation

When the user has finished choosing to whom to send the card, the *choose a name* window disappears and the card behind is visible for the first time in all its glory. There is a standard formula for beginning most cards, so Magic Cap goes ahead and puts "Dear *addressee*," at the beginning of the messages.

Most of the time the automatic salutation will be correct or close to correct, saving the user some typing. Saving typing is important, because typing on the projected keyboard is pretty slow. But if the automatic salutation is wrong, the user can use the keyboard *delete* key or text selection to get rid of it.

The user can edit the default automatic salutation like other text. Ordinarily, the salutation chooses the addressee's first name. Here the sender has edited the salutation to be more formal.

> *When software does something particularly clever, it offers the user a way to override the cleverness.*

personality can be worth more than efficiency

On the right side of the card are the addressee's name and the name of the device user. The message mixes information that is relatively ?xed—like the word *Dear* and the sender's name—with information the user has changed. In this way, Magic Cap messages are much like the sorts of messages people send frequently, where stock information is mixed with changeable information without special fonts or delimiting characters. Magic Cap text flows clearly, even in cases where the information is added in a mechanical, almost mindless way.

In the layout of the address information, Magic Cap again twists a metaphor. Postcards made of paper have addresses, often three or four lines long, in this position. Electronic mail messages usually have lots of bewildering, but rigidly formatted, header information at the top. The critical parts of any header—the sender, the addressee, and a subject line—are included on the card. But here they are formatted to resemble a traditional paper-postcard address.

The address area of the message includes a subtle, almost hidden control. Touching the word *to* lets the user alter the addressee list. This shortcut is not obvious to most novice Magic Cap users, but it is both direct and practical.

Beginning users might not see hidden controls and are unlikely to try touching text to make something happen. Nonetheless, some controls are worth hiding and some text is worth touching. Hide anything that isn't central. Let users touch text that they want to change.

buttons at the right of the screen

To the right of the message is a set of buttons that control how the message is handled. They are arranged according to their importance, their frequency of use, and, in part, their complexity. Most messages will eventually get sent, so the *send* button comes first. Many messages will need to have addressees added or the list of addressees changed, so the next button down controls these features.

Buttons on the right side of the screen are usually arranged in order of importance or frequency of use. Note here, though, that *extend* is at the bottom of the rank, mostly because it operates on the bottom of the message on the screen.

Many scenes have four or five buttons along the right side of the screen. In these ranks of buttons, the ones that aren't working at a given time are not visible. The most important button is usually at the top.

Each of the buttons along the right side of a Magic Cap scene has a picture and text. The pictures make each button easy to recognize and distinguish from the others. The text is often more explicit than a picture might be at telling beginning users what the button does.

Often the text and picture are complementary. A picture of a card sliding into a slot clearly means the card is going to go somewhere. The verb *send* tells that it is going to be sent.

The text on a button allows users to talk about the buttons unambiguously. *Touch erase* is an easier instruction to understand and to give than *touch the button with a swoosh and a cross on it.*

controls at the bottom of the screen

At the bottom of the screen, Magic Cap uses only pictures. The objects at the bottom of the screen are always visible. Providing them with names might make using Magic Cap the very first time a little easier. But this small advantage doesn't outweigh the cost of these words in visual complexity and in the space they consume. The names on buttons at the side of the screen are like name tags at a conference or a convention. It's useful to have everyone's name showing even though you might recognize the faces. Putting name tags on the buttons at the bottom of the screen would be a little like insisting that everyone in your household wear name tags to breakfast. If the faces aren't familiar, you're probably in the wrong house.

The controls at the bottom of the screen are always visible. They will be familiar to every Magic Cap user and don't need name tags.

The user can touch the *send* button, at the right of the screen, at any time. Even when the card is unaddressed, the *send* button works by displaying the list of potential addressees. Since the rest of preparing a message is up to the user, the *send* button has to be readily available no matter how long the user takes to work on the card.

temporarily stepping away

If the user were to touch the word *Desk* in the top-right corner of the screen at this point, the message would appear as a miniature card on the desk. The user could go ahead and work on a dozen other things before getting back to the message. The miniature card is labeled with the addressee's name to remind the user why it's lying there.

Magic Cap doesn't force users to follow long procedures with no diversions and no way to postpone completion. Because Magic Cap devices are designed to be used on the go, the software has to be prepared for interruptions.

tools, stamps, the keyboard

The tools for doing the fun part of making a message are along the bottom of the screen. Formal notes are likely to be typed. Sappy love letters generally involve lots of hearts. The choice is the user's. A wide selection of items relevant to making messages is at hand: stamps in the stamp collection, drawing and shape tools in the tool holder, a keyboard for typing. The user has ready access to these items at all times, from the buttons along the bottom of the screen. By providing stamps and tools, packages can add to the built-in items.

Once the user is finished with the message, it needs to be sent. Touching *send* places the message in the out box. How it is sent from there depends on the policy that the user has set up. Magic Cap's default policy will suit most users, sending messages when they are connected to a telephone line. But people with special requirements can change the policy directly using rules. Rather than present all users with the many possibilities available at any time, the commonest procedures are the default, and variations can be enabled through rules.

When the message has been placed in the out box, the desk once again fills the screen. Sending the message began and ended in the same scene, with little changed besides the net amount of information in the universe.

The best procedures travel full circle. The user starts out wanting to do something, goes and does it, and lands back where the procedure began.

4

Navigation and organization

Magic Cap has a physical geography modeled on that of the world. This simple fact provides Magic Cap's users with a lot of information about how Magic Cap works: doors open onto rooms, books open up to pages, hallways lead to other doors. Knowing about the individual navigation elements, you can decide what sort of elements to use for your package. Knowing how other developers and the core of Magic Cap use different navigation elements will help you anticipate your customers' expectations so that you can make your software appeal to them.

Rooms, buildings, and icons

Each of Magic Cap's rooms is a complete place appropriate for some certain activity: a room for games is called the *game room*. The room where disused objects are stored is the *storeroom*. Individual rooms contain objects appropriate to the activities performed there. Similarly, different buildings on the street downtown represent different information or communication services. And individual objects in one of those places—the icons in the desk accessories drawer, games in the game room, control panel selector buttons—give the user access to scenes with their own particular purposes.

Having different activities partitioned off in different scenes helps users remember where they need to go to do what they want to do. As a developer, you can use your own scenes, containing nothing but your own objects, to keep close control over what is on the screen while your customers use your package.

Locating your package

As you design your package, you'll need to decide how to locate its parts in the Magic Cap geography.

Some packages are utilities that users might need to get at easily from the desk. These packages, such as calculators, time or money management utilities, and packages that help users prepare certain kinds of messages or documents, go best on the desk or in the desk accessories drawer on the right side of the desk.

Other packages work best as rooms off the hallway. Packages that involve numerous objects or that require that the user have access to an assortment of objects on shelves will need the space that a separate room implies. When your package is installed, the doorway into its room appears in the hallway.

Some packages work well as individual objects in one of the existing Magic Cap rooms. Games, for example, usually install themselves into the game room.

Other packages add to existing collections of objects. A new set of stamps naturally appears in the stamps window. A new kind of stationery goes into the stationery drawer at the left side of the desk. A new global command fits best in the Magic Lamp, where other command buttons are found.

Finally, some packages represent the outpost of a communications service within Magic Cap. These outposts usually belong in buildings downtown, which is intended as a place for services and for software that involves communication with an external network.

Choose a location that is appropriate to your package. Your site should be useful and practical, rather than glaringly prominent. By placing packages consistently, you help users understand the rich geography of their communicator and help them ?nd your software when they want it.

Where should your package's objects go?

location	purpose of objects	examples
through a door on the hallway	complete contexts for an activity	art studio accounting room conferencing system
as an icon or gadget at the desk, including in the desk accessories drawer	utilities that are closely associated with messaging or information management	time management money management contact- or phone call recording
in the phone's *services* panel	telephone-controlled services	voice mail simple airline flight information bank-by-phone panel
as a building downtown	outpost of a communicating service	electronic service building AT&T PersonaLink℠ center
as an icon on the desk	central functionality	datebook, phone (developers only add these as part of an arrangement with a manufacturer)
in the stamps window	stamps and related objects that can be dispensed	decorative animal stamps new kinds of address labels
in the Magic Hat (stamps window, construction mode)	components, such as coupons, that provide construction-mode function	new border coupons sound coupons song coupons new buttons
in the Magic Lamp window	commands, either scene-specific or global rules	search tool rule to tidy up a scene periodically

Different parts of the screen

The top of the screen is the *name bar,* displaying the name of what the user is looking at and where in the Magic Cap universe of places this particular place lies. The name bar usually contains the time, date, battery level, and other information that is useful at all times—precisely what depends on the user and the particular model of communicator. Your package will affect the name bar to

- provide notification about ongoing activity, such as a service connection in progress
- provide arrows to move from card to card or page to page
- display which card or page is currently on the screen
- add a stamp that displays a list of announcements when your package posts an announcement and other, higher-priority announcements are already being displayed

At the bottom of the screen is a collection of buttons called the *control bar*: the buttons constitute a set of global controls and objects. These objects have been chosen to provide the user with consistent tools that might be used anywhere in Magic Cap world. The button bar stays pretty much the same all the time a person is using their communicator. Your package only changes the bottom bar if you

- add a tool to the tools window—when the tool is active, it will be displayed in place of the tool holder
- put an item into the tote bag or trash, which both display whether they are full or empty
- change the system's mode to construction mode, which changes the stamper to the magic hat

The four-fifths of the screen between the bottom bar and the name bar belongs to the current *scene*. This area is the package developer's domain. In users' minds a scene is a particular room, document, or other collection of objects that fills the screen. Some scenes are rooms, streets, or hallways. These scenes are often called *places.* Other scenes include messages, lists in file cabinets, the datebook pages, the clock—scenes seen up close.

The buttons along the right are the datebook's. The buttons along the bottom of the screen always appear.

Within documents and when looking at objects up close, a rank of local command buttons often runs down the right side of the screen. Up to five buttons fit in this space. Sometimes a scene won't

need all the buttons. Blank buttons appear in the unused spaces; they do nothing when touched.

Global additions—Magic Lamp and stamps

Two of the buttons in the control bar provide a place for you to add global commands. If your package provides a new kind of search tool—such as one that performs searches with boolean logic—you might add it as a button in the Magic Lamp. The Magic Lamp will accommodate additional commands.

Your package might be a collection of new stamps, perhaps on a theme. If you have licensed all of the Star Trek™ characters from Paramount™ and create a set of stamps with their images, these stamps could consist of an entire bank of drawers. The bank would be called *Trek;* the individual drawers might be called *crew, Klingons™, starships,* and so forth. In this way your package gives your customers all of the stamps you have created without complicating any of the activities they do without your package.

Scene-specific additions

Ordinarily, objects specific to your package appear in the package scene. However, two system places also hold package-specific objects:

- The Magic Lamp can contain commands that are local to your software package in addition to the global commands it always holds. If your package adds scene-specific commands, they will appear as buttons or icons in the Magic Lamp window when that scene is on screen. The Magic Lamp also contains rules specific to the place on the screen. If your package adds rules to an existing scene, adds a scene that has its own rules, or adds choices to an existing rule, then your package will change what the user can do with the Magic Lamp.

- The bottom-most stamps drawer contains scene-specific stamps and other objects. In many scenes, this drawer opens automatically when the user touches the stamper. If your package has its own scene-specific stamps, you'll install them into the *local* drawer in the stamps collection. The local drawer can take on the name of the scene. To see how this works, look at a name card and touch the stamper. You'll see a set of name card-specific stamps.
- The stamper can contain additional banks of scene-specific banks of drawers. To see how this works, look at a name card and touch the stamper. You'll see a bank of drawers containing stamps you can add to the name card.

Your burden isn't the whole screen

The first-generation Magic Cap has a screen that is 480 pixels wide and 320 pixels tall. The top 24 pixels of the screen are consumed by the name bar. The control bar always takes up the bottom 40 pixels of the screen. This leaves you with 256 pixels of vertical space and the full 480 pixels of horizontal room for your software package to draw in.

You get a lot for giving up a fifth of the screen. Magic Cap uses the name bar at the top of the screen and the bottom bar at the bottom to keep the user grounded and to keep the device functional, no matter what room or object is filling the rest of the screen. The name bar contains a complete set of controls for dealing with places. At the left end is the name of the scene that is currently on the screen. Touching the scene's name shows a small bit of textual and graphical help about the scene. Whoever develops the scene should give it a name and provide this help text and perhaps some graphics.

In addition, announcements that the user must see—indicating sending progress, declining battery power, and similar important activities—are abbreviated by a small picture that appears in the name bar. If your package includes a *stack scene,* such as a set of pages or cards, arrows can appear in the top bar to let the user move from page to page or card to card.

Leave the top and bottom bars pristine and you'll give users the comfort of always having something familiar on the screen. Moreover, your users can take advantage of those familiar buttons at the bottom of the screen to work with your scene.

5

Touchable objects

Magic Cap contains hundreds of kinds of user interface elements. In making a package, you might add to this number. Some elements are visible on the screen and the user can touch them. You can get specific information in the *Magic Cap Class and Method Reference* and *Magic Cap Concepts* books about the particular elements you are using. This chapter describes a few of the basic elements that you might use and how they relate to each other.

Stamps

Stamps are usually intended to affect the way that the device itself, the communications network, or other users process and understand the objects to which the stamps are attached.

Many of the stamps you see on the screen are decorative or purely informational—they don't do anything special when you touch them and they're inert during transit. Magic Cap users can use these simple, decorative stamps to mark cards and documents for sorting. Magic Cap can search for a given stamp or sort messages by their decorative stamps.

smart stamps

Some stamps in Magic Cap are actively intelligent. Many stamps affect messaging. These messaging stamps are the main way users see different transport, delivery, and payment options that communications services offer.

Some messaging stamps only make sense once a message reaches its destination. A stamp that reads *party!* or your corporate logo might not affect delivery of a message. But such a stamp will probably determine how the recipient of that message handles the message.

Touching a stamp tells it to act. Some stamps don't do anything on command. Like most viewable objects, they are highlighted when touched. They will also play any sound they have when touched. But Icras and software developers often make stamps that contain

scripts. Those scripts often make the stamp do something when touched.

Magic Cap has special kinds of stamps for special purposes. The sound recording stamp, when it is empty, starts recording when it is touched. Once the stamp contains a recording, its appearance changes, the balloon filling with symbolized sound. The next time the lips stamp is touched, it will play back the recording. The sound recording stamp thus embodies one of the most important principles for designing Magic Cap software:

The sound recording stamp changes its image as its state changes. With an empty balloon, touching the stamp starts recording. Touching the lips with a full balloon plays the recorded sound.

Show state changes visually, especially when the change in state affects how to use an object.

Buttons

Stamps sometimes work like buttons. Buttons have some added features—they can have a border and are stretchable. However, just to activate a script, a stamp can serve just as well as a button. In general, Magic Cap makes touching a button initiate an action, as buttons do in everyday life. Buttons are generally rectangular and contain some text. Stamps, in contrast, usually mark a message or a document for sorting, sending, or processing.

Stamps and buttons look different and users have different expectations of them. The button (left) tells the communicator to send messages straightaway. The urgent stamp, when attached to a message, indicates to the communications network or to a person seeing it that the message should be sent quickly. The *urgent* stamp can also be used to trigger rules that the recipient has set for handling messages.

People usually expect buttons to do something at the moment pressed—most buttons that people touch in everyday life on their telephones, in their cars, and in their offices have immediate effects. In contrast, stamps might not do anything until sent; or they might never do anything at all. Choose buttons for immediate actions; stamps for deferred behavior.

The sound recording stamp responds immediately when touched.

Icons and gadgets

Icons are objects that often look very much like stamps but are specifically designed to navigate from scene to scene. When the user touches an icon, Magic Cap switches scenes. The phone on the desk is an icon. Many packages that involve multiple scenes use icons to provide users with a way to move from scene to scene.

Gadgets also look like stamps. When touched, a gadget makes a window appear. The window usually zooms from the gadget to its full size. When the window goes away, it generally does so by zooming back to the gadget.

From the user's point of view, icons, stamps, and gadgets aren't necessarily particularly distinct. Buttons look different from these other objects, and users seem to regard them differently.

How individual object types differ

object	use	examples
stamp	display information to user convey information to a communication service provide a marker for searching and sorting can be moved by simple sliding	grinning face *urgent* stamp company logo stamp
button (including icons and gadgets that act like buttons)	perform a function when pressed convey information about the function that the button performs normally can't be moved just by sliding invite pressing by their form	*?nd* button *fax* button
special stamp	display state information respond when touched	sticky note sound recording stamp
gadget	open a window when touched	tote bag
icon	go to a new scene when touched	phone name card file

Coupons and tinkering

Magic Cap uses coupons to change the behavior and appearance of objects. Each coupon is good for one change. The coupon is applied by sliding it and dropping it into a viewable object. Coupons have a consistent graphical theme: a heavy, dashed, black, rectangular border surrounding an image or some text.

Coupons turn invisible or intangible qualities into objects that you can see and manipulate. If your objects have attributes that users can change, consider making coupons for the alternate states of those attributes.

Coupons identify themselves with images or with text.

If the coupon appears with a textual description, the text describes what the coupon will do when you drop it into an object. Visual properties, for example, are all described textually. Visual properties are somewhat difficult to show until they are applied to a view or objects on the screen, but the textual description is more compact. In other cases, a graphical depiction makes more sense. A swatch of color more readily indicates the color that a coupon embodies than does a phrase like *dithered light gray with dark gray*.

Some attributes are well suited to graphical representation

Others are better represented in text.

Your package can provide coupons to allow users to manipulate objects that can't ordinarily be manipulated. Specific chunks of text are put into text coupons as the user moves them around the screen. You can use coupons similarly to capture the contents of visible objects. A text field might contain some text, a color, and a border. Each of these properties can be manipulated with coupons.

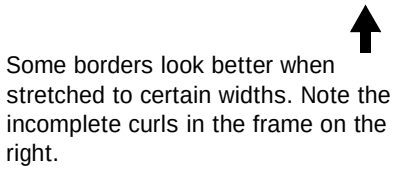
Magic Cap generally uses coupons to change features. Coupons are especially good if the different values or states for a new property are difficult to express well in text. Line thicknesses, for example, are easiest to set if you can see what the different settings will look like. Magic Cap includes coupons to control the line thickness of borders.

Borders

Borders are stretchable edge-and-corner patterns for rectangular objects. Magic Cap makes wide use of borders. Borders have numerous advantages over other ways of attaching visual edges to objects. They tend to be more space efficient than identical-looking bitmaps, which can require storage space proportional to their size. Magic Cap software knows how to draw borders quite quickly. Moreover, borders can stretch with the rectangular objects they surround.

Some coupons hold objects that aren't themselves visible—like the script represented by this coupon.

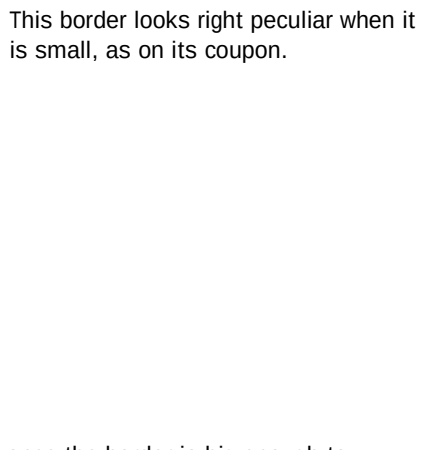
Some borders look better at certain widths and heights because they are formed of patterns that repeat only every few pixels. When you ship software with such borders, you want to make sure that the objects that use those borders have the right widths and heights so the borders look their best. It's true that users might end up stretching the objects so the borders look funny but at least they must share the blame.



Some borders look better when stretched to certain widths. Note the incomplete curls in the frame on the right.

To save memory, use borders rather than bitmaps for repeating edge patterns.

Some borders look strange when small. Books are drawn using a special book border. The border doesn't look much like a book when it's drawn on its coupon. But as the border is stretched to larger sized rectangles, it takes on its bookish appearance. It's more important that a border look good when it's applied to an object—as in the books themselves—than in the small size that will appear on a coupon.



This border looks right peculiar when it is small, as on its coupon.

But once the border is big enough to surround a book, it looks more bookish.

Boxes

You can use boxes to group together objects that are logically associated. Because a box provides a visual grouping, it helps indicate to users that the objects inside have something to do with each other. Moreover, anything inside the box moves when you move the box. Thus, if you have a switch, two labels, and a meter in a particular arrangement, you might put them into a box so that you can move them all together. The box itself can be invisible—transparent with no border—so no one but the user who tries to move the objects will know that the three items are in fact inside another.

A special kind of box called a choose-one box makes check boxes or switches placed within the box respond to one another. If you place a group of switches inside a choose-one box,[1] switching one

[1]Older readers might recall the popular verse:

I never know to have whitefish or lox.
But I shlep them 'round in a choose-one box.
It makes it easy for me to select:
One, I eat; the other, reject.

on turns the others off. If you put check boxes inside a choose-one box, checking one makes all the others unchecked.

These uses of boxes to contain and—in the case of the choose-one box—to control other objects illustrates how creative Magic Cap software can use invisible objects to make the visible objects behave as desired.

Windows

Windows are bordered collections of objects. A window overlays the scene that's currently on the screen. Windows can also overlay other windows. Windows in Magic Cap software share many features with windows in computer interfaces: Magic Cap windows can have a title bar, they generally display a shadow on the objects they overlay, and it's usually easy for the user to move the window out of the way or to make it disappear.

Multiple windows make it possible to arrange objects, controls, and information hierarchically. Windowing computer interfaces often use multiple windows for separate applications, for separate portions within the same application, and for dialogs between applications and the user. However, in the index card-sized world of Magic Cap, multiple windows extract a heavy toll in comprehensibility. You can make your windows close when other windows appear. You can also make sure that one window doesn't depend on objects inside another.

Beginning users often don't understand windows at all and, if a window is large, assume they've gone to a new scene. Windows are potentially confusing, so it's best to have no more than one on the screen at a time.

Category boxes, like drawers, let one window show more than it can hold at one time.

Magic Cap software can avoid multiple windows by letting the same window hold multiple collections of objects and giving the user a way of switching among the collections. The Magic Hat is arranged hierarchically, as other windows might be. As the user moves through the hierarchy, the contents of the window change.

Because the active drawer appears open, the user gets feedback on which part of the hierarchy is currently being displayed.

Many windows have balloon spouts, showing where the window came from. Balloon spouts remind the user how to make a window appear again. Balloon spouts can show the item to which the window refers. While editing a list of four or more addressees on a message, a balloon spout shows whether the *to:* or *cc:* list is being edited. You can use balloon spouts to help your customers understand what a window is for, where it came from, and where it will go when it is put away.

The Magic Hat displays category boxes. Tapping a category box changes the contents of the Magic Hat, rather than bringing up another window on top of the Magic Hat.

Most windows also have a close-*x* in the top-right corner. This indicates to the user that the window is closable at any time. If the contents are uninteresting or unsettling, just touch the *x* and the window will go away.

Books

A Magic Cap book, like a paper book, is a series of pages arranged so the reader can turn back and forth among the pages. Unlike all but the most sophisticated pop-up paper books, Magic Cap books can contain objects besides text and pictures. Switches, buttons, animations, and even on-screen pianos are all strong candidates for inclusion in a book. In your package, you might use a book to provide the user with instructions. Some packages also use books as a good way to browse through information, possibly making choices or touching objects on the way.

Icras has included a few sample books with Magic Cap software. Although the books behave much like the bound volumes in a public library, some have special features. Some books are interactive catalogs. When the reader touches a check box, the particular topic checked gets added to a comprehensive subscription pro?le. This is one of the ways in which Magic Cap objects improve on the real-world things after which they are modeled.

This catalog of information services includes checkboxes. When you touch one of the boxes to check it, the order form for your subscription is automatically updated.

Don't sacrifice functionality just to maintain a metaphor. It is the advantages over the physical world that Magic Cap offers that make it appealing. Besides, your software might set an example: some day maybe paper catalogs will pay attention when you touch the scratch-'n'-sniff items in its pages.

6

Processes

Using Magic Cap is a story of processes: the user switches on the communicator, does some things, and sooner or later the communicator goes off. Many of the user interface elements that Magic Cap provides enable or smooth these processes. They guide the user through a sequence, providing information along the way, allowing for unexpected changes, and trying to make the most common paths through a process the easiest.

Among the commonest processes for the user are navigating, making choices, providing input, and receiving information. Navigating is covered in detail in the chapter on navigation and organization, chapter 4. Input and receiving information are covered in the next chapter, on input and output. This chapter covers making choices.

Category boxes

The Magic Hat is Magic Cap's all-purpose object catalog and dispenser. Category boxes are the section headings in this catalog. To get an object from the Magic Hat, the user touches the Magic Hat to open it, then touches the appropriate category box to go to a certain section, then touches the object desired.

The categories in the Magic Hat are intended to be intuitive rather than an unadulterated reflection of the classes of objects. Buttons and stamps are very close to each other in implementation; to the user they have somewhat different roles in the world. Buttons *invite* the user to press them, and users expect immediate action when they push buttons. Stamps, in contrast, often mark or identify a message, without needing any pushing at all.

Category boxes are best for selecting among disparate objects. Drawers, which use only text to describe their contents, are better suited to selecting among subtly differing groups of similar items.

Drawers are, unlike category boxes, a way to select among different kinds of similar items. This contraption combines a choice box with some drawers.

Menus and choosing text items

Menus are lists of words—occasionally the names of delicious Spanish seafood dishes—presented so the user can choose one of the items from the list. Some menus scroll to show more words than could ?t in a single column on the screen. Menus most commonly appear when the user touches the text of an item in a choice box.

Touch on the words in a choice box and a menu of all the choices appears.

Magic Cap uses menus to let the user make a choice from a list of textual items without leaving the current place. Menus are at their best when the number of choices is small and no scrolling is required.

Provide lots of space in lists that users have to touch. Users can drag their fingers up and down the list until the right name is highlighted. However, it's far easier for them if they can simply touch down on the right name.

spacing and text size

The size of the text on the screen is one of several factors that affect its legibility. After some experimentation, Magic Cap's designers and developers have found that menus are particularly sensitive to the size of the text in them. With text that's big enough and has sufficient space between lines menus are both legible and usable.

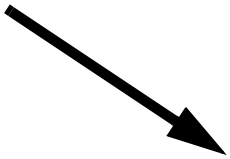
Leading and type size in menus

Font size and spacing pixels at 100 dpi	**Use**
font 12 (upper-case letters about 12 pixels high) *spacing between lines of text* 18 (font height + 6 pixels)	Limited use
font 14 (upper-case letters about 14 pixels high) *spacing between lines of text* 20 (font height + 6 pixels)	Recommended

choosing from long lists

Lists become harder to use and require additional aids, like lettered tabs, when the list of items gets very large. For choosing people, Icras put together a complete window called the people picker. This window has a button for adding a person who isn't on the list. It also includes index tabs to make going through long lists of people easier. If your software involves long lists, lettered tabs or a similar mechanism might make finding and selecting the right line in the list quicker. For people, you can also use the built-in people picker.

The axis of a process



Magic Cap users' eyes scan the screen from the top left to the bottom right. If you arrange objects so that a process goes in this order, the process will be easy to understand.

You can arrange objects to help your customers go through the processes in order. There is really only one thing to remember about making the order of a process apparent. Start at the top or at the left and proceed to the bottom or the right. If you order the crucial elements along this axis it's hard for the user to get lost.

Many of the arrangements you see on the Magic Cap screen derive from this directional ordering. The scene name, for example, appears in the top-left corner of the screen because that is the first place most users look when they are presented with a new, unfamiliar scene and start looking through it systematically to discover its purpose.

The people-picking window, too, follows a top left-to-bottom right order. Select a name at the left, touch *accept* on the right.

This ordering is usually easy, and you'll ?nd yourself following it without thinking—placing a *done* button at the bottom right of a window, for example. Sometimes, though, the user will deviate from the proposed process and from the standard axis. And sometimes even the commonest order will deviate from this direction.

Detours and extra loops

You've probably noticed that the way to put away both windows and scenes in Magic Cap is by touching what is in their top-right corners. In a window, a close-*x* appears in that position. In a scene the pointing hand and a scene name are in that spot. This spot is far off the top left to bottom right axis. The user touches that spot when the process is complete or is to be abandoned. Similarly the

desk button appears in the bottom-left corner of the screen at all times. This is an escape hatch, removed from the normal axis of processes.

Sometimes other constraints intrude and make your design deviate from the standard modes of access. In writing a message, for example, the last thing that the user does in the process is touch *send*. The *send* button is not at the bottom-right corner of the screen, but near the top-right corner. In the message-writing scene, the buttons along the right side of the screen are ordered in relation to each other. Since it affects the bottom of the message, *extend* belongs near the bottom of the message. Similarly, since it gets rid of the message, *discard* belongs fairly close to the trash. *Send* is on top for a couple reasons: the topmost button is the easiest to ?nd, and most of the time users touch *send* rather than any of the other buttons to dispatch their message. Beginning users rarely miss the send button since it's at the top of the rank. In fact, most of them see the button well before they've finished writing their messages.

Extra loops are auxiliary parts of the process that don't happen every time, but should be part of the ?ow when they do. For example, when using the people-picker window the user can decide to look at the members of a particular group by touching the choice box at the top of the list. This appears in the top-left corner of the window, since using the choice box happens first.

Similarly, the *new* button is close to the axis in the people-choosing window. After touching *new* and creating a new name card, the user proceeds downward—in the direction of the axis—to the *accept* button.

7

Input and output

Input is information that flows from the user into Magic Cap. The primary input mechanism is the touch sensor. Magic Cap communicators also have a microphone for audio input. This document is mostly about immediate interaction with the device, so it focuses on the touch sensor for input.

touching is input

Output is the realm of information that the device provides to the user. The primary output device in a first-generation Magic Cap is the liquid-crystal display (LCD), a screen directly underneath the touch sensor. The display can contain images of any kind: drawings, text, scanned photographs, representational art. Along with the screen, Magic Cap uses the speaker to provide information to the user.

what you see is output

Magic Cap also gets information from the telephone network, from peripherals like keyboards, scanners, and sphygmomanometers connected to Magicbus, or out of thin air through its assorted communications channels like its infrared receiver. Other output devices, like printers and welding robots, can potentially be connected through Magicbus. These alternate forms of input and output are covered briefly at the end of this chapter.

Input and output go hand in hand. Every time the user does something in the way of input, the communicator probably responds with some kind of output. The way in which one kind of input produces a certain kind of response is called *feedback*. Providing good feedback is an essential step in making a good user interface.

Responsiveness

The user controls Magic Cap by touching objects on its screen. Objects should respond rapidly, almost instantly.

When your software gives users instant feedback, it feels responsive and useful.

Your software can respond instantly to a touch even when considerable processing has to occur for the software to respond fully. For example, as soon as the user touches the screen, Magic Cap makes a soft tone. The software then goes on to compute which object the user touched. Because the tone came early, it's clear to the user that Magic Cap registered the input.

Your software needs to respond quickly to please your customers. The people who buy Magic Cap communicators usually feel that they have very little time to waste. Like most people, they are impatient with technology, and they demand responsiveness. When Magic Cap has to spend some time doing something—dialing a telephone number, looking up an address, cleaning up a package—it tells the user that it is busy by spinning a small hat on the screen. The spinning hat is concise, consistent way for the communicator to say *just a moment please.*

touch sensor

Touch the screen

Users usually won't distinguish between the touch sensor and the LCD that lies beneath it. The sandwich of the display and the sensor together is called a *touch screen* to remind designers that they are effectively one unit. The resolution of the touch screen is considerably higher than the resolution of first-generation LCDs. Nonetheless, clumsy human fingers and relatively inaccurate calibration of the touch sensor to the display beneath it make touching somewhat inaccurate. When an object is to be touched by fingers, inaccuracy of up to a pinky's width—around a centimeter—is likely.

To put a set of objects on the screen and keep them usable by clumsy fingers, space the visual portions as widely as you can. A good guide is your own finger. If two controls are less than a finger's width apart, users will probably hit the wrong one a frustrating part of the time.

It's easy to move your finger from one side of the screen to the other—much easier than using a mouse to move a pointer across

the screen. If two buttons do opposite things, try placing them on opposite sides of the screen. That way they're not likely to confound your users. For example, the arrows that appear in the name bar at the top of the screen to let you move back and forth among cards are separated by the card-number information that they change.

Touching is rarely precise

Several features of Magic Cap make touching inaccurate. First, fingers are imprecise pointers. Compared to pens and other pointy implements, fingers have broad, mushy tips. It's hard for a person to tell exactly which part of the fingertip is touching the screen hardest. The Magic Cap screen informs the touch-handling software that a particular *point* is being touched, even though a whole fingertip is pressing on the screen.

Second, because there's a small distance between the touch sensor and the display, users sometimes press a place on the sensor that's slightly offset from the point or object on the display that they want to touch.

Finally, the touch sensor doesn't behave well when you touch it in two places at once. If you are trying to write on the Magic Cap screen and you rest the heel of your hand on the bottom of the screen, the touch sensor will report that it is being touched at a point between your pen point and your hand's heel, or at the pen point, or maybe at the point where your hand is touching. In short, the touch probably won't be reported where you would like.

Knowing that touching is somewhat inaccurate, you can design the elements in your package to minimize the problem. You can space controls widely, making them large enough to see and to touch easily. You can also make sure that missing one control and hitting one right next to it won't have permanent, disastrous effects.

One little window; lots of little controls. The row of numbers and letters chooses *what* you want to adjust; the arrows make the adjustment; the two buttons finish the whole business. Arranging the controls in three rows helps keep them comprehensible.

Making controls easy to distinguish and easy to touch

The *set time* window in the datebook contains a number of controls spaced quite close together. Some precautions have been taken in

it so that, even with 12 touchable controls in only about six square inches, users will often hit what they want to hit. Although they usually don't realize it, users are far less likely to hit the wrong control if they perceive the difference between it and the controls around it rapidly and unequivocally. For this reason, making controls easy to distinguish is an important way to make them easier to touch accurately.

diversity

The controls in the datebook's *set time* window are different shapes and arrayed differently on the screen. They are different sizes, their sizes reflecting to some degree their relative importance. Although every control in this window responds to the very same gesture—a simple touch—some of them are long and thin, some compact and triangular, and some of them rely on words to be understood.

transience

Some controls only appear when they are appropriate. The *a.m.* and *p.m.* controls don't show up when Magic Cap is set to display all times in 24-hour format.

The pen and the finger

Use wide hit areas. Because small objects can be hard to hit, you might opt to extend the hit area of objects beyond their visual boundaries. You can touch the desk that appears in the bottom-left corner of the screen by touching anywhere within the control bar from the left edge of the screen to 70 pixels from the screen's edge—well beyond the end of the image.

Keep hit-test areas wide. No matter what the size of your objects, try to follow these guidelines in hit testing:

Hit area pixels	**Use**
24 wide 28 tall	Very limited use, not for objects that users have to hit
28 wide 24 tall	Minimum for common use
32 wide 32 tall	Recommended size
larger	Even better!

Single-stroke gestures

The Magic Cap interface is designed to require virtually no training. Users don't ordinarily have to learn a lot of deft gestures to work the device. Nonetheless, a few simple gestures make Magic Cap much more powerful and make similar actions easier for the software to distinguish.

There are currently two simple gestures in the software. *Touch* by pushing against the screen and then releasing, without sliding your finger. *Slide* by holding your finger down on an object and then sliding your finger across the screen.

These gestures are each a single stroke. You never lift up half-way through. They're also both very simple. The distinction between them is based on the amount of motion.

Distinguish gestures generously and don't require a complex gesture to achieve some end. Gestures can be difficult to learn. Time-based gestures are especially dangerous. They make people feel rushed and uncomfortable.

Touch to activate; slide to move

Most objects in the Magic Cap universe should respond to a touch by performing their primary function. Consider, for example, a sound stamp on a message. Touch the stamp to play its sound, press on the stamp and slide to move it around.

Beware of sliding objects

Beginning users are reluctant to slide objects on the screen. Some beginning users won't even think of sliding an object. Experienced users, especially computer users, are more willing to slide things. In requiring users to slide objects that form part of your package, consider this disparity and reserve sliding for advanced operations. In much of Magic Cap, sliding is a part of construction and re-arranging objects, but not necessary for using basic features.

Sliding in Magic Cap has a substantial practical disadvantage: it is difficult to slide an object to exactly the position you want and unless you wear gloves and live in a cleanroom, touching a Magic Cap device's screen is liable to leave unsightly, glare inducing, opaque smudges. One solution to the smudge problem is to manipulate objects using a stylus or pen. But many users want to touch the screen directly with their hands or don't have a stylus ready all the time.

A better solution, therefore, is to limit the degree to which your software *depends* on sliding things around for everyday operations. If a user can do nearly everything without ever sliding an object or leaving a smudge, that user is likely to be pleased with your software's responsiveness and how clean it leaves the screen.

You might use sliding for a few basic manipulations and more advanced operations. You might also choose to require sliding objects in situations in which your users are likely to have already picked up a pen or a stylus.

- *Sliding is useful, but beginning users often don't know how to do it, it is difficult to slide objects perfectly accurately, and it can make a greasy mess of the screen. Use simple touching for basic operations.*

- *When your interface does require sliding, consider teaching users how to do it, perhaps using a tutorial.*

A tutorial on sliding forms one of the steps in the *Getting Started* lesson about creating a message.

Slide and drop

To make two objects work together, slide one of them to where the other appears on the screen and drop it in. The coupons in the software are the most obvious expression of Magic Cap slide and drop. Pick a coupon, press your finger down on it, and slide it by sliding your finger across the screen. You can now drop the coupon into an object by sliding the coupon over the target until the target is highlighted, then letting go.

Coupons act on the object they're dropped into. When you drop a coupon into another object, the other object takes on whatever attribute or quality the coupon specifies.

Sliding an object and dropping it into a second is one way to alter the second object or to change the state of the first.

Doors, snapshots, and some arrows transport objects dropped into them. There are a few objects that do something with objects you drop in, rather than having what you drop in do something to them. Most of these slide-and-drop targets are navigational tools like doors that lead to the next room and snapshots that lead to other places. The trash truck, folders, and tote bag also take things that you drop into them. Many objects accept coupons that are dropped into them.

But if Magic Cap limited slide-and-drop to construction operations and to shortcuts, it would miss out on easily understood features: dropping a message into the out box sends it. Dropping a document into the file cabinet files it.

As you design your package, you might find many occasions to use sliding, and sliding and dropping. If they are shortcuts for basic operations, then sliding might well make sense. But if you are requiring beginning users to learn to slide objects in order to use your software, you might well be excluding potential customers.

One-handed use

option key is never required

Magic Cap is primarily software for devices that you carry with you and use in many different conditions. Unlike a desktop computer, whose mouse and keyboard demand at least two hands, Magic Cap can't really ask for more than one of the user's hands, since people will use their Magic Cap in the car, on the subway, while skydiving. The touch screen only reports a touch from one location at a time. There is only one button used while operating the software—the *option* key—and it is not required for any basic activity.

When the user is asking for information and the device is displaying it, don't require the user to hold something down on the screen. Pop-up menus on computers generally require that the mouse button be held down for the menu to be displayed. Since the equivalent on Magic Cap to holding a computer mouse button down is holding your finger in place on the screen, it's safe to assume that most of the screen is obscured by the user's hand. So, Magic Cap's menus let the user first tap to bring up the menu, then look to see what's there, then tap or slide along the menu to select one of the choices.

The user's hand usually obscures a large part of the screen while touching an object, so avoid requiring your users to hold something down while reading text or looking at objects.

Sliding to the edge of the screen

Because a bezel surrounds the edge of the screen of Magic Cap portable communicators, it is very easy for users to slide things accurately into the corners. The edge acts as a visual and tactile guide. As a package developer, you might find it useful to lay out along the screen's edge those important targets for sliding in your software.

Sound

Brief, simple sounds offer an important channel for feedback. Sounds associated consistently with certain actions help people learn those actions. The *swallow* sound gets played when one object accepts another. Try sliding a color coupon into another object to hear the noodle-slurping swallow sound.

Use system sounds the way the system does. You can listen to all of the system sounds in the sound control panel. Tap any of the panel's sound proxies to hear the sound. You'll leave your users happiest if your software plays system sounds under the same conditions that cause them to be played elsewhere. Use the swallow sound when one object swallows another, the honk sound when the user tries to do something wrong.

Small sounds are good

Brief sounds save memory. Even compressed, sounds can cost your package a kilobyte or so per second. If your package is too large, it will be more expensive to distribute and more difficult to sell. Editing sounds and using them sparingly is one of the easiest ways to keep down the size of your software. You can also use system sounds instead of your own.

aural consistency

Use your own sounds consistently. You can make your sounds complement the appearance of your software by doggedly maintaining a consistent use for each sound. A romantic correspondence package for Magic Cap might make an excited, slightly suggestive sound every time a message comes in from the person whose address card is marked with the *sweetheart* stamp.

sound is sometimes off

Sound will sometimes be turned off. If your software *depends* on its users' hearing sounds, many people won't be able to use your software. Sometimes people will bring their Magic Cap devices into

meetings or into library reading rooms, where they have to turn the device sound all the way off. Some users will just find the sounds objectionable and leave sound off all the time.

No matter what perfect sounds you've associated with different events in your software, some people will never want to hear them. It might be appropriate for your software to depend on sounds in a game or in a music-making package. But if you are writing software to help people keep minutes in business meetings, think hard before you depend on any sound—yours or the system's.

Sounds make good instant feedback. You can use sounds to let users know exactly when they've touched something. Sometimes doing the work to draw an object or to draw the changes made to the screen takes a noticeable amount of processing time. Since the screen can't change until drawing is done, you can use sound to fill the gap in time and thus let the user know that Magic Cap is responding to the action. The keyboard uses this technique, playing a sound straightaway, before it has even decided what key was pressed. As a result, the keyboard seems responsive.

Text as input

text that must be typed

Sometimes it's impossible for Magic Cap to work properly unless the user actually types a piece of text. In addressing messages, for example, Magic Cap requires either that it has received an name-card for that person from the communications service or that the user create a new name card by typing out the addressee's name. The keyboard, while daunting to many beginning users, needs to appear only in this special case.

Many name cards will come to you from a communications service; any time you receive a message from someone, you also receive their public name card.

You can help make text an effective kind of input by helping the user to work with the keyboard. If your customers will be typing a lot of numbers, you can create a 12-key keypad just for numbers.

In the course of a process that requires users to type, you can make the keyboard appear automatically at the correct moment and you can set it to be ready to type capital letters and numbers when the user is likely to need to type capital letters and numbers.

Handwriting as input

Clearly the content of a message doesn't need to be machine readable. Handwriting is not just suitable, it's better for the kind of short, personal messages that Magic Cap favors, since handwriting carries, even in small chunks, emotion and personality that typing cannot.

text that can be handwritten

Allow scribbles wherever you can, particularly when the information is designed only for human eyes. Some users will still choose to type because they like typing. But many users will opt for the personal, direct quality of scribbles.

Size of text

Text should be big enough to read comfortably.

Characters whose capital letters are less than about 10 pixels high are too difficult to read on the Magic Cap display. Users with good eyes can make them out; most people find them something to squint at.

letters at least 10 pixels high

In the current scheme, text should be less than 12-point[1] only in a pinch. On current LCDs, text is printed in gray on a slightly different gray background, so type has to be larger on Magic Cap than in a printed book to be legible. If you find that you must put text

[1]Point sizes are relative measures within one style. The size of characters that make up printed text is usually given in *points*, each ⁄ /¶™ inch. But the characters in a 12-point font are not necessarily 12 points high. In fact, it is likely that only one character, the italic letter *f*, even comes close. When type was made of lead, all of the blocks in a font were the same height so that the blocks could be lined up together and held in a frame. The lead blocks in a 12-point font were each 12 points tall. The characters on them were therefore no more than 12 points tall.

against a gray background, try using a bold typeface it to increase its readability.

Because the Macintosh is such a deceptive environment for laying out Magic Cap text, trust your instincts rather than your eyes if you are working on a Macintosh. If you're lucky, your instincts will keep saying "bigger, bigger; maybe it will be legible." There is a special mode in Magic Cap Simulator to make the screen approximate the grays of a LCD.

Icras's engineers follow specific guidelines when placing text or places to type text on the screen. These guidelines are on the next page.

Many beginning users feel compelled to read all text that software designers place on the screen. These guidelines are intended to make sure there is no text in Magic Cap that users with average vision will have trouble reading.

Fonts in Magic Cap

Font name	Use	Sample
Sign 8 Fat Caps 10	very limited use, only for labels that could not otherwise fit. *Never* for text that users type or edit.	Antelopes ANTELOPES
Book bold 10	legends on buttons	
Sign 10 Book 10	for labels in tight spaces; generally only for text that is meant to be read, not for user-edited text	Antelopes Antelopes
Sign 12 Book 12 Book bold 12	general-use fonts for • labels • running text, such as blocks of text in a book or a message • text that the user might create or edit	Antelopes Antelopes
Jot 12	handwriting font for user data Avoid using Jot for words that a user didn't type, such as messages from your software to the user.	*Antelopes*
Sign 14 Book 14	For titles of windows, important labels, to call attention	Antelopes Antelopes
Sign 18 Book 18	usually for the most important piece of information on the screen, such as the name on a card in the name card collection	Antelopes Antelopes

Length of Text

Text in Magic Cap software generally comes in short chunks, since there is little room on the screen and the low-contrast display doesn't make for great reading.

Where possible, keep a continuous flow of text down to what can be seen at once on Magic Cap screen. Scrolling is possible but it is not necessarily easy, especially for users who have never had to contend with the scrolling text on a computer screen. Once one portion of the text has scrolled off the screen, the user has to scroll it back into place to reread it.

informational text

Throughout Magic Cap, text tells about the objects on the screen. Instructions in the information window, the information listed in the catalogs of information services, and even the names of objects displayed as their labels are kept brief. They have to provide some guidance but they have to do it efficiently.

The important text is that supplied by users and information providers as they use Magic Cap and communications services. Minimizing the amount of text that is *not* message or service content leaves that much more room on the screen, in device memory, and in users' heads for what interests people.

Naming

Any object in Magic Cap software can have a name, and the name of any viewable object can be displayed as a label. When you name an object, the name will work best if it describes the object's function or purpose. Buttons often have names and images. Windows and scenes have visible names.

changing names

Names are easy to change while you are developing software, and finding the right name might lead you through a few changes. In trying to name the Magic Hat, Icras went through a series of names: *Browser*, *Magic Window*, *Directory*, and finally *Magic Hat*, sometimes called the *stamper*. Each of the names had something in particular that appealed to us and to our users. *Browser* describes what the Magic Hat is for: browsing through a collection of Magic Cap objects. *Magic Window* describes the form that the object takes on the screen: a window. It's magic because it dispenses

objects at a touch. *Directory* is a generic term for catalogs of things and places and recalls directories for computer users. *Magic Hat* was especially appealing since the one window on the screen can dispense a practically unlimited number of objects—seemingly much more than it should be able to hold. Finally, the contents of the basic gadget are limited to stamps and a few components that look like stamps. To most Magic Cap users, the *stamper* and the *stamps window* are more familiar than *Magic Hat*.

Spelling, capitalization, punctuation

When naming objects, keep in mind that their names might appear on the screen.

spelling

Misspelled words can be embarrassing. The *American Heritage Dictionary*, 3d edition (Houghton Mifflin, 1992) is a good source for spelling help, especially as it includes capitalization and hyphenation, as well as common alternate spellings.

variants

Magic Cap 1.0 is designed for the North American market and U.S. spellings are recommended, as they are familiar to most of the people in that region. If you choose to use other spellings—like Canadian *enrol* and *levelling* or overseas-English *tyre*[2]—you will give your software a decidedly foreign feel.

hyphenation (not)

In general, hyphenation is nothing but trouble in software written for Magic Cap 1.0. Without built-in hyphenation algorithms, there is always a chance that a piece of text will get re-wrapped. When this happens, explicit hyphens turn into errors.

Magic Cap uses traditional English capitalization, rather than the title-style capitalization used in some computer software. In general, if you are in doubt, use lower-case letters. They will take less space. The capitalization rules in the *Chicago Manual of Style* are consistent and workable, especially in running text. The labels of objects are lower case. Place names, as seen at the top of the screen, and window names, as seen in the windows' title bars, are capitalized sentence style: *Name cards.* However, when a term like *name cards* appears in running text, is should have small letters. Of

[2]You might need this word if you are writing car-servicing software. Otherwise, you can probably avoid this word altogether.

course, proper nouns and trade names should be capitalized as their possessors would like them.

Except for proper nouns, use all lower case. A switch should be labeled inversion*, an info window should talk about the* drawing room.

Capitalization in Magic Cap software goes as follows:

Capitalization

Occasion	Style	Example
place names at top-left and top-right of screen	sentence style	Drawing room
place names and window names amid sentence	common noun	About the drawing room
window names when displayed at top right of window	sentence style	Tote bag
proper nouns	title style	General Magic
book titles	title style	Sending and Receiving
table headings	each heading sentence style	Items received When

Label positions

The position of labels is not an entirely straightforward issue. In a system where users have to touch objects, it might seem sensible to put labels above the objects. That way, the label can remain visible even when the user's finger is touching the object.

However, computer users who have experience with Macintosh, Windows, X, and other systems like the Canon Cat, expect

icons—small pictures on the screen—to bear their textual labels beneath.

Furthermore, when a group of small pictures is lined up, most Western and East Asian eyes expect the bottoms of the pictures to line up, just as the bottom of text lines up on a page. Text labels are relatively uniform in height, so the labels will line up if they are beneath a set of pictures whose bottoms line up. This line of text seems easy for people to scan quickly.

Modern typographers and publishers usually put captions underneath the pictures the captions describe. In Magic Cap, many designers find that these benefits outweigh the costs of putting labels underneath the small pictures on the screen. Magic Cap software usually labels objects underneath.

Putting labels in different locations differentiates the objects that bear these labels. There are some cases where the core Magic Cap software puts labels above or alongside small pictures. In the Magic Hat, category boxes represent whole categories of objects. The category box looks like one of the objects in the category with a dotted-line border around it and a label at the top left of the border. If the category label were centered underneath the representative object, it would look more like a label for the object itself than a label for the dotted-line border, which stands for the category.

Positioning objects' labels

Position	Kinds of objects	Example
underneath	image on buttons messages on the desk	
to the side	switches that switch back and forth checkboxes	
centered in the object itself	buttons books (titles)	
in the top left corner of the object	scenes windows certain menus category boxes	
above	objects on a shelf	

In creating forms for people to fill out, you often need to identify an area in which people write. Icras labels fields on the form much as category boxes are labeled, along the border in the top left corner.

Switches arranged together in a column also benefit from having their labels alongside. A label between two switches might go with either. By putting the label alongside its switch, the developer makes it obvious just what is being labeled.

Category boxes have their labels in the top-left corner to make clear that the category—and not the particular coupon shown, like the *send to back* coupon here—is what's being labelled.

User tests have shown that labels are persuasive. Icras' tests on first-time device users have made it clear that, if an object has a label,

the user will rely on that label at least as much as on an icon for semantic content. Labels can thus be especially assuring.

Buttons along the right side of the Magic Cap screen show their labels. These labels have pictures so they're attractive and easy to recognize; the labels complement and even explain the pictures. Pictures, especially for abstract concepts, are not always as explicit as words. In general your text fields, buttons, and other objects will be best understood by users if you provide each of the elements with a carefully chosen label.

Text fields can also have their names near the top left corner, in imitation of some of the clearest paper forms. (Not Federal tax forms.)

Don't count on your users to figure out a rebus or the semantics of an icon. Most small drawings are ambiguous, and a label removes the ambiguity.

8

Packages

Software developers make Magic Cap *packages*—collections of software components that users can treat as a piece, that they can buy or barter for, and that they can use together with the Magic Cap system and with other packages. As independent objects, each of the objects within a package can stand on its own in some way. Many packages consist of a set of closely connected objects—buttons and the features they control, boxes and the stamps they contain, cards and the scribbles on them.

Some packages are like complete applications for personal computers—they consist of a coherent, closely knit collection of objects, directed toward achieving a particular task or range of tasks.

Other packages are less integral. They are collections of lists, books, stamps, or other objects. Often, such packages have a unifying theme—one package might contains the addresses of congressional representatives along with maps of their districts, another might hold recipes in a book and stationery to use to send recipes to other Magic Cap users.

Downtown

Packages focused on communication with the outside world are represented as buildings along the street downtown. These packages aren't appropriate for the Hallway, which is focused on activity going on *inside* the Magic Cap communicator. These service-centered packages include elements to communicate directly with a service, with a service through a Telescript cloud, or that need to represent objects or services that exist somewhere besides in Magic Cap.

Place buildings downtown to represent services outside of the communicator. The building contains the on-communicator manifestation of the off-communicator service, including controls, mechanisms for describing preferences, instructions, subscription forms, and a description of the service.

Zoning and design

The left end of the street downtown is zoned as residential; this is where the user's house always sits. The rest of the street is for the user to arrange. Most users leave the buildings arranged on the street in the order in which the buildings appear. When a package installs a building, it installs it at the right end of downtown. By installing buildings in this way, downtown becomes a history of the new services to which the user gained access, starting from the basic Magic Cap features inside the house and running rightward to the newest service, at the right end of downtown.

Directories placed next to the house and at the right end of downtown ensure that, even as many buildings are positioned along the street, the user has quick access to all the buildings. These directories are updated automatically to include all the buildings along the street. Giving a building a descriptive name—*Acme Travel*, *Northern Gravel Mail Order*—makes the directory easy for the user to understand and use.

A directory of all the buildings downtown appears next to the house, making it simple for users to navigate among many buildings.

The building that represents a service can make an essential visual statement about that service. The design of a building on Magic Cap's downtown, like the design of a building in the real world, plays a large role in forming an image of the service offered in the mind of the customer.

Buildings that take up less than half the screen width are particularly effective.

When a building is added to those downtown, its name is automatically added to the directory.

Buildings are most effective, usable, and visible when smaller than 200 pixels square. Larger buildings that take up a whole block on the street discourage users from going past and robbing the building of its physical context.

A distinctive, narrow storefront is likely to be left close to home by the user. One building doesn't usually abut the next, so its shape—while simplest if rectangular—doesn't need to mate perfectly with its neighbors' shapes.

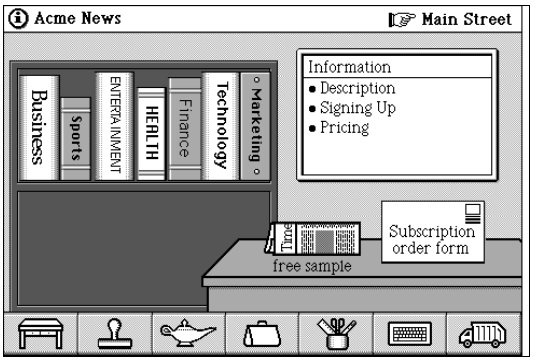
This storefront shows off some desirable characteristics of buildings on Main Street:

- the small image can be detailed and still not consume a prohibitively large amount of memory
- the building clearly spells out what service it represents
- the building is narrow enough that it doesn't take up a lot of Main Street frontage. As a result, users can scroll past easily when looking for a building beyond this one
- only the façade of the building needs to be drawn

Complicated, large building images often consume more memory than small, simple images. Building images, which are often distributed through telephone, radio, or other communication channels, are under particular pressure to remain small. Sophisticated, beautiful buildings are of no use if they are so large that they take too long for potential users to download.

As is the case with all images designed for first-generation Magic Cap, 2-bit deep grayscale images are most appropriate for buildings. Black-and-white images will lack the visual subtlety of grayscale images, and images with greater color depth will take more time to send.[1] Furthermore, color buildings are reduced to 2-bit deep grayscale on first-generation Magic Cap communicators and that reduction process is likely to make the buildings unattractive.

As with all other Magic Cap graphics, it is essential to look at buildings' images on the LCD screen for which they're intended.



The interior of many buildings is simply a room. This news service's interior scene includes objects for subscribing, for refining the service's pro?le of the user, for learning about the service, and for handling an ongoing subscription.

Navigation inside a building

Many buildings consist of only one room. That room is sufficient to contain all the controls and objects that the user needs to take advantage of a service. However, some services are sufficiently diversified, extensible, or complicated to require a larger, more open-ended structure. Like the house, which contains a hallway with doors opening off of it, buildings along the street downtown

[1] As a result, likely more money as well. As time is money, the one statement about time should have sufficed without this note.

can contain hallways. These hallways are most appealing to users if they look different from the hallway in the house, with different design and different furniture. The hallway within the building that represents a news service could look like the nerve center of a television station, with rooms leading off it devoted to various domains—subscription and delivery, newsgathering, editorial.

Using hallways and rooms in a way coordinated with their use along the street downtown in the hallway makes them easy for users to understand.

Maintaining the corridor-room metaphor throughout a service's on-communicator software helps users to understand the package readily. Doors generally open onto Magic Cap scenes that look like rooms. Minicards should zoom up as documents that fill the screen.

Hallways increase the size and perceived complexity of a software package. They should be used only if they make a product or product range better as a whole, even given the increased complexity.

Except for the house, buildings generally have their names printed as part of their façades. To make the sign fancy, buildings use two images superimposed on one another: one image is the building itself, the second image is used for the building's name.

The *Electronics* logogram is used here as the logo for the storefront of the Electronics building.

9

Graphics and Appearance

How your software looks determines how people feel about it. Appearance affects first encounter more than any other aspect of design. Carefully designed images will attract and interest new customers; sensible, meaningful images help keep customers interested in using your software. This chapter looks at some of the more important aspects of Magic Cap aesthetics.

Perspective and comparative size

The objects on and around the desk are drawn with a suggestion of depth. They aren't just ?at images or icons. The desk surface looks like it is horizontal and the datebook, phone, and other objects sit on top. But notice that they aren't made to look like a photograph or a perspective drawing. You can move the phone to the opposite side of the desk and it still looks three-dimensional. Moreover, the cards sticking out from the in box don't need to appear tiny just because they're farther away than the card in the center of the desk.

Effectiveness, not blind uniformity, can guide your depiction of perspective and the images you choose. It is far better that you disappoint a few users who only like vanishing-point perspective than that all the people who use your product ?nd themselves saying "what is that?"

In searching for examples in perspective to follow, look to medieval European mosaics and tapestries, not to Western painting after Tintoretto.

This informal, adaptable kind of perspective lets simple images evoke a sense of depth. It is not an innovation of Magic Cap and in many ways it is simpler than vanishing-point perspective, which might require a stamp to distort its image as the user slides it from one side of the desk to another.

As with any two-dimensional representation, the images displayed distort three-dimensional reality. All pictures—from *trompe l'œil* painting to high-resolution photographs to abstracted subway maps—depend on their viewers' training to be understood. Video games, cel animation, and computer software all often use an old-fashioned kind of perspective to display information.

Consider a hypothetical software package that lets telecommunications workers trace software bugs by displaying a telephone network on the screen. If a desk telephone is to ?t in a square 1 centimeter on a side, a properly scaled carpenter ant will suffer the distinct graphical disadvantage of occupying only a pixel or two. A larger ant, fully equipped with legs and antennae, might disturb people. Nonetheless, even though an ant might appear the size of a desk telephone on the Magic Cap screen, nearly every user will be able to tell them apart and can deal with them for what they are, rather than for their respective sizes.

Magic Cap perspective and the relative sizes of objects are designed to make software easier to understand: the objects on the screen reflect and represent *real-world objects, rather than depicting them precisely.*

Magic Cap uses suggestions of perspective particularly to make rooms appear three dimensional. The desk has its edges drawn toward some mythical vanishing point. No other objects in the Office are drawn with the same vanishing point. But this simple, limited use of perspective gives the desk scene a three-dimensional appearance.

Shadows

The light source in the Magic Cap world is generally somewhere above and to the left of the user's left ear. This means that objects

cast their shadows *down* and *to the right* on the screen. Shadows show that an object is held *above* those behind it. Windows, for example, are drawn with shadows to set them off from the scene underneath. When objects are moved, they are drawn with a shadow to show that they have been picked up from their positions and can be moved above other objects.

Themes

Shadows in the Magic cap universe are cast down and to the right.

Icras follows a few themes, or general principles, in drawing different kinds of objects.

For example, most touchable objects in Magic Cap are partly or substantially white. Simple themes like this one are not to be followed blindly. This theme is intended to make it clear which of the objects on the screen will respond when touched, not to dictate that all such objects must look the same.

Finally, when you design icons, keep in mind that intricate, heavily ornamented graphics can turn into incomprehensible splotches and blotches on the liquid crystal display. Simpler graphics are more readily understood.

In one simple theme, Magic Cap objects that invite touching have substantial areas of white and have thickish two-pixel borders.

Making controls look touchable

Icras has found that users are most ready to touch controls that look like they can be pressed. The buttons along the control bar, the buttons down the right side of the screen, and even the *x* that appears in the top right corner of windows are made to look like they stick up from the surface of the screen. This three-dimensionality encourages users to see these objects as controls.

This doesn't mean, of course, that all controls have to look three dimensional. Checkboxes, for example, look very much as if they were printed ?at on a paper page. But their shape is familiar, something that people expect to tick off.

Some kinds of objects don't usually look touchable. In user testing, Icras found that many people are reluctant to touch plain text on the screen. By putting a label on the place where the text appears, and then a button around the label, Icras encouraged users to actuate the text fields in attribute steps by touching.

Putting a button around the label for this field encourages people to go ahead and touch the thing. They need to touch here to change it, and text doesn't look to some people like it is to be touched.

Shades of gray

The displays on first-generation Magic Cap communicators offer four different shades of gray. However, in general these four values (lightnesses) aren't spaced linearly, and adjacent colors in the scheme are very similar.

dithering

Although there are only four shades of gray on the screen, areas of intermediate colors can be created by *dithering.* In Magic Cap dithering means making a checkerboard pattern of two different colors, one pixel one color, the next pixel the other color. Dithered color coupons can be found alongside the other color coupons in the Magic Hat.

Because dithering depends on the user's brain effectively blending the two components of the dither and making the user see an intermediate shade, the technique only works properly when there are enough dots that the colors can blend together. That is, dithering is only effective for creating intermediate area-fill colors. Dithering is especially useful for giving a wide background a particular intermediate color. Patterned borders are sometimes similar to dither patterns; the visual effect of a narrow strip of alternating colors is very different from the dither effect described here.

text and dithered backgrounds

Text on a dithered background is usually very hard to read; Icras only uses bold text on a dithered background. Look carefully at any object you place on a dithered background to make sure that the object's edge isn't the same color as either of the colors in the dithered pattern. As with any text displayed on a gray background, if you need to display text against a dither pattern, use large, bold type to offset the obscuring effect of the background.

In fact, the only text that seems particularly legible on first-generation Magic Cap communicators is black on white or, less pleasing to the eye, white on black. Black text on gray can be almost indistinguishable. If you need to put text against a gray background, use a bold typeface.

Anti-aliasing

Good computer artists make judicious use of *anti-aliasing* to smooth curves and shapes on the screen. Anti-aliasing means using intermediate shades for certain pixels around a pixelated edge so that the edge looks smoother. Techniques very similar to computer display anti-aliasing are used in video and in pointillist painting.

anti-aliasing

If you're up to it, try anti-aliasing images that appear often. Your corporate logo, shapely buttons, and even large header text might benefit from good anti-aliasing, provided the image is anti-aliased for the particular background on which it will appear. (The rabbit in the Magic Cap logo looks good anti-aliased when it appears on a white background; a little strange when it appears on a gray background.) Not all of the curved elements in Icras's software are anti-aliased.

Anti-aliasing text has bred considerable controversy. Some researchers have performed studies that show readers read faster and understand better when all on-screen text is anti-aliased. Some typographers have looked at anti-aliased text and deemed it unacceptable. In general, Magic Cap developers don't anti-alias small text. If you choose to try, please let us know whether or not it pleases your users.

A stolid corporate logo, plain (left) and anti-aliased

Beware of relying on the cathode-ray tube display of Magic Cap software for judging contrast. Macintosh screen black is blacker than first-generation Magic Cap communicator black and Macintosh white is whiter.

10

Other books

There are many resources worth consulting to learn about designing user interfaces. These are a few of the best books in design and wording, ones that are readable by anyone. They differ in their focus, but all share a common interest in conveying information and meaning to an *audience*. How that audience reacts and feels is your responsibility in making a user interface. All of these books are written for general audiences and provide seasoned designers with a complement to more technical works.

Edward O. Tufte. 1986. *The Visual Display of Quantitative Information.* The Graphics Press.
Edward O. Tufte. 1990. *Envisioning Information.* The Graphics Press.

In spite of their rather pedestrian names, these are pretty, popular books that seem to be on coffee tables in the houses of many smart people. They show how information—especially scientific, experiential, numeric, and non-visual information—can be represented efficiently and so that people understand. Tufte, a professor at Yale University, has an eye for both the beautiful and the beastly. *The Visual Display of Quantitative Information* treats especially well what Tufte calls *chartjunk*—the information-free, obfuscatory lines that crowd many designs, including many computer user-interface designs. For someone designing Magic Cap user interfaces, these two books show how to take a design and get rid of all but the essentials, and how to divide the information you want to present into manageably, comprehensible pieces. *Envisioning Information* shows in great detail how time and processes can be displayed through multiple, simple images.

E. Fowler. 1983. *Modern English Usage.* 2nd edition. Oxford University Press.

If you've ever gone beyond wondering *should a comma go here* to *why should I use a comma here*, this book will interest you. A collection of short essays on topics like *jargon* and *preposition at end*, this

is probably the best reference work on writing clearly. It is also very funny.

Scott McCloud. 1993. *Understanding Comics.* HarperPerennial.

Understanding Comics is a detailed review of the fundamentals of serial art, one of the applications of which is in comic books. McCloud is a professional comic-book author, but most of the material in the book—possibly excepting one about color—applies directly to Magic Cap user interface design. Among the techniques traced are abstraction and representation, how time and sequences are represented in art, and ways of conveying moods and emotions. The book reveals hundreds of ways to present information and instructions in limited space. It is also very fun to read.

Donald Norman. 1990. *The Design of Everyday Things.* Doubleday.

Donald Norman's book runs through some common objects and describes of the things that make certain industrial and interface designs good or bad. One of the messages of this book is that bad designs sometimes succeed commercially, but they always do so at the expense of their users. Norman focuses on all sorts of engineering, from mechanical elegance to extraordinary gaffes in computer user interfaces. Although the author is sometimes criticized for oversimplifying and for belaboring obvious points, this book is notable for cutting to the quick in the relationship between a bad user interface and the poor, suffering user. For the Magic Cap designer, this book demonstrates the tension between simplicity and usability better than any other.

Index