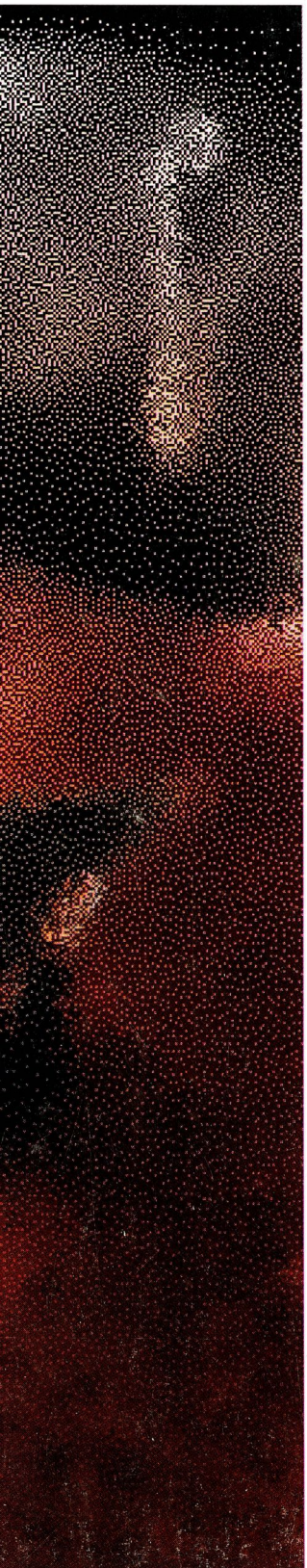
d e

The App

velop

le Technical Journal



MAKING THE MOST OF COLOR ON 1-BIT DEVICES

THE TEXTBOX YOU'VE ALWAYS WANTED

MAKING YOUR MACINTOSH SOUND LIKE AN ECHO BOX

SIMPLE TEXT WINDOWS VIA THE TERMINAL MANAGER

TRACKS: A NEW TOOL FOR DEBUGGING DRIVERS

USING THE PALETTE MANAGER OFF-SCREEN

BACKGROUND-ONLY APPLICATIONS IN SYSTEM 7

MACINTOSH Q & A

APPLE II Q & A

NEW FEATURE: KON & BAL'S PUZZLE PAGE

Issue 9 Winter 1992

EDITORIAL STAFF

Editor-in-Cheek *Caroline Rose*

Technical Buckstopper *Dave J*

Review Board *Pete "Luke" Alex*
Derossi, C. K. Haun, Larry K
Shebanow, Gregg Williams

Managing Editor *Monica Meff*

Assistant Managing Editor *An*

Contributing Editors *Lorraine*
Toni Haskell, Judy Helfand, I
Rilla Reynolds, Leslie Steere, C

Indexer *Ira Kleinberg*

Manager, Developer Support S
Communications *David Kr*

ART & PRODUCTI

Production Manager *Hartley L*

Art Director *Diane Wilcox*

Technical Illustration *Geoff Ma*
John Ryan

Formatting *Forbes Mill Press*

Printing *Wolfer Printing Compa*

Film Preparation *Aptos Post, In*

Production *PrePress Assembly*

Photography *Sharon Beals, Der*
Kathleen Siemont

Circulation Management *Dav*

ohnson
cander, Chris
Rosenstein, Andy

ert
a Wilczynski
Anderson,
Rebecca Pepper,
Carol Westberg

Systems and
athwohl

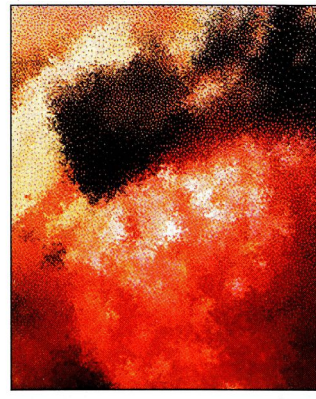
ON
Lesser

:Cormack,

any, Inc.
c.

nnis Hescox,

id Wilson



To create this cover, Hal Rucker and Cleo Huggins bought the nicest-looking fruit they could find, photographed it and scanned in a slide, manipulated the scan with Adobe Photoshop, and blended in a dithered version of it. Delicious!

develop, The Apple Technical Journal, is a quarterly publication of the Developer Support Systems and Communications group.

The *Developer CD Series* disc for February 1992 or later contains this issue and all back issues of *develop* along with the code that the articles describe. The contents of this disc, which includes other handy software and documentation, can also be found on AppleLink.

CAROLINE ROSE

Dear Readers,

Let's talk about *develop*: w
you can do for it. This jo
by reading on and giving

Originally, *develop* was th
accompanying CD, it wa
you, the developer, could
be compatible with futur
were written primarily by
Apple.

But other types of article
Most notable was the gro
which source code was n
by our readers, yet the ov
favorable. So we've move
possible. We still make eve
in future systems.

Recently we've had some
ideas, not code. Our curr
good Apple products, we
to see. There are some A
would like to know just w
Developer Technical Sup
let us at *develop* know dir

Regarding who writes th
Apple engineers, there's
We'd like to encourage a
fellow developers—some
showcase and release you
offer something those *oth*
the assurance of future co
prose shine so brilliantly

CAROLINE ROSE (AppleLink
writing computer documentatio
Jobs was barely a teen. When
moved in down the block from
as a writer and then a prograr
no notice—until they asked if s
what even then was known as
Around the time she completed
tome, Steve left Apple to form

/hat it is, what it might be, what it can do for you, and what urnal exists to meet your needs, so I hope you'll help us out us your two cents (if not your articles).

ought of as "heavily commented code": along with the s meant as a vehicle for providing well-explained code that plug into your application with the confidence that it would e system software. To ensure compatibility, articles and code / Apple engineers and heavily reviewed by other engineers at

s have been submitted, and some have made it into print. ound-breaking Threads article in Issue 6, the first article for ot provided. This lack of source code did not go unnoticed verall response to the Threads package was extremely ed from *always* providing source code to providing it *if at all* ery effort, however, to give you something that won't break

requests to publish articles that describe algorithms or ent feeling is that as long as an article can help you create 'll consider publishing it. Please let us know what you'd like pple engineers who are willing to contribute to *develop* but /hat developers want to see. We get a lot of input from pport about what you seem to need the most help with—but ectly, and we can try to make it happen faster.

e articles: we feel that as long as the code is reviewed by no need to rely solely on people at Apple for contributions. ll of you to think about what you'd like to share with your thing that would help them and also give you a way to r code in a way that wouldn't otherwise be possible. We *ner* journals don't: not only review by Apple engineers and ompatibility, but also an editorial process that will make your you'll need to wear shades. We'll assign an editor who will

---

: CROSE) has been on ever since Steve his company where she worked mmer, Caroline took she wanted to write *Inside Macintosh*. d that three-volume NeXT, and Caroline signed on to launch NeXT's Publications group. A year ago she returned to Apple to take on the fun-filled job of being *develop*'s editor in chief. For fun outside of work, Caroline dances up a storm, listens to music, plays with her cat and other friends, treks through the wilderness (in boots or on skis), swims like a maniac, reads fiction (not sci-fi!), studies Italian, does Tai Chi, and never stops exploring new ways to have fun.

help turn your raw material into a polished piece—or tread li
you need. We'll give your article that professional look and f
humor. So, if you're willing, please send me your ideas or ou
from there.

Back to the subject of your opinions about *develop*: Many of y
Associates and Partners have by now been formally surveyed
support-related materials, of which *develop* is only one shinin
like to hear from the rest of you, however informally. I can't
important your opinions are and how much they'll affect *dev*
express yourself! Tell us what's good or bad about this journa
delivery, or anything else. We're all ears.

Issue 8 ended with this trivia question: What word was used
describe the action of pressing a button on that first mouse?
of you have gotten as of this writing, is "bug." Maybe you'll
one: The original hardcover *Inside Macintosh* Volumes I-III h
Macintosh computers across its endpapers (those heavy shee
and end of hardcover books). What broke this pattern, and w

SUBSCRIPTION INFORMATION

Use the order form on the last page of this issue to subscribe to *develop*. Please address all subscription-related inquiries to *develop,* Apple Computer, Inc., P.O. Box 531, Mt. Morris, IL 61054 (or AppleLink DEV.SUBS).•

BACK ISSUES

For information abo
how to obtain them,
form on the last pag
are also on the *Dev*

ightly on it if that's all
eel without killing the
tlines, and we'll take it

ou who are Apple
on how you rate various
g example. We'd also
overemphasize how
elop's future. So please,
l's content, format,

instead of "click" to
The answer, which none
do better on this next
ad a running pattern of
ts at the very beginning
hy?

Caroline Rose
Editor

ut back issues of *develop* and
see the reverse of the order
e of this issue. Back issues
eloper CD Series disc.•

LETTERS

CURLING UP WITH D

Regarding your editorial
agree with you on liking
copy" to be able to curl u
trying to understand som
first time. I can always go
computer and try exampl
to lay back and put up m
under a quilt in bed is mo
let concepts sink in and d
own, to spring forth with

I like the idea of sending
separately in its own case
never had a mangled disc
the magazine itself that h
trip. I received Issue 8 wi
and hope that the disc is
The mailing label on the
half off, not torn, but det
Flapping in the breeze, s

Keep up the good work o
magazine. I look forward
time.

—Robert Redmond

Thanks for your letter. It's
heartening to hear from de
agree with me on this, but
difference. Your opinions do

The disc is now in a separa
not mailed under a separat
should have arrived wrapp
We'll send you the disc righ
about that.

—Caroline Rose

TEXT FORMATS GAL

There's a problem with t
Technical Notes which I
increasing frequency in A
electronic publications. T

PLEASE WRITE!

We welcome timely letters to t
especially from readers reacti
we publish in *develop*. Letters
addressed to Caroline Rose (c
develop-related questions, to D
Apple Computer, Inc., 20525
M/S 75-2B, Cupertino, CA 9
CROSE or JOHNSON.DK). A

EVELOP

in Issue 8: I
to have a "hard
ıp with when
ıething for the
o to the
es or ideas. But
y feet or nestle
ore relaxing to
levelop on their
clarity later.

the disc
—though I
problem. It is
as a rougher
thout the disc
not far behind.
back cover was
ached.
o to speak.

on the
to it each

not only
velopers who
it makes a
count.

te case, but it's
e label. They
ed cozily together.
t away. Sorry

ORE

he Macintosh
'm finding with
Apple's
The only word

processor I use is Nisus, with which I can read MS Word 3.0 and 4.0 files without buying a Microsoft product. This may be an unreasonable prejudice, but I bet it isn't uncommon.

But Nisus can't decipher fast-saved Word files. This means, I suspect, that the entire set of new Macintosh technical publications is unavailable to me. Worse, I fear that the next Developer CD is going to have lots of files with new, valuable, and (for me) hidden information.

I know Apple is serious about electronic distribution of technical documents. I'm sure fast-saving in Word is a great convenience to the authors, but surely using a format not widely readable defeats the purpose of the exercise. I don't object to standardizing on Word 3.0 or 4.0, so long as that format—and not Microsoft's convenience variant—is actually used.

Could you *please* ask your authors, when providing documents for publication, to use an accessible format?

—Fritz Anderson

Thank you for alerting us to this problem. It was a snafu on our part. None of the files should have been fast-saved in Word.

We know that having text documents in Word and MacWrite® represents a bias toward these products. Unfortunately our alternatives are limited and we'll probably have to continue using these products until the spring.

The good news is that we're working on a new text formatting tool. This tool will be available on the CD and will be able to open, search, and print text documents available on the CD. The dilemma of how

---

he editors,
ng to articles that
should be
r, if technical
Dave Johnson) at
Mariani Avenue,
5014 (AppleLink:
ll letters should

include your name and company name as well as your address and phone number. Letters may be excerpted or edited for clarity (or to make them say what we wish they did). •

to make every document available to every developer has been a topic of discussion for some time. We're hoping this will solve the problem.

Again, thank you for your input. Developer feedback is the fuel of change around here. Keep it coming.

—Sharon Flowers

NEW AND IMPROVED CD

I just received Issue 8 of *develop*, and was pleased to find that the developer's CD has improved. Is this new?

—Mike Caputo

Yes, starting with Issue 8 the CD is not just Developer Essentials, but the entire Developer CD Series disc (of which Developer Essentials is just a subset).

—Caroline Rose

SUBMITTING TO DEVELOP

First of all, I'd like to say that I'm a big fan of *develop*. The combination of excellent technical articles (with required humor) and a CD-ROM of other developer materials is unmatched. At least the flak surrounding the CD-ROM has finally calmed down in the Letters section. I've always liked the idea from the start even though I purchased a CD-ROM drive only last week.

I'm writing to find out if *develop* accepts articles from non-Apple employees. I haven't looked through the back issues to see if there were any, but none come to mind. If so, do you have a style guide for writing articles?

Keep up the gre

—Paul-Marcel S

Thank you for yo
develop*; it's alwa*
mail from a big f
letter that editors

We do indeed acce
employees (see thi
have a vast array
prospective author
introductory docu
form, a set of deta
and even a Micro
entering your art
format. We'll star
send the rest as yo

On the subject of
it may experience
Apple's dropping
Notes from the m
Associates and Pa
what developers t

—Caroline Rose

ART ILLEGIBL

Figure 1 from M
on Futures (Issu
printed under S
Macintosh II on
It's just as illegib
ideas?

The HyperCard
convenient for f
articles but is te
page at one time
windows are no
more, HyperCa
from a CD. And
fast nor intuitive

—Steve Tyler

at work!

St-Onge

ur kind words about
ys a pleasure to receive
m. Yours is the type of
in chief dream of.

pt articles from non-Apple
s issue's Editorial). We
of materials ready for
rs, including an
ment, a short submission
iled author's guidelines,
soft Word template for
icle in a develop*-like*
rt with the intro and then
u need it.

the CD-ROM controversy,
a revival as a result of
printed develop *and Tech*
onthly mailing to
rtners. We'd like to hear
hink about that.

E ON-LINE

Michael Gough's article
e 7) was illegible when
ystem 7 from a
a LaserWriter IINTX.
ole on the screen. Any

l® format on the CD is
lipping to pages and
rrible for seeing all of a
e, since HyperCard's
t resizable. Further-
rd is slow—especially
l searching is neither
e.

I looked into the problem an
a mistake was made when t
version of Issue 7 was create
EPS format is normally ope
application that interprets F
then saved as a PICT. This
followed, with the result tha
to PICT was only an approx
not very legible. This will be
on the CD.

FEELING LOST? SEE THE MACINTOS

To a Macintosh developer starting a new project, the range of equipment available can seem daunting. To an experienced engineer suffering the constant barrage of catalogs, technical brochures, and advertisements, it ca feel safest to hang on to familiar tools, whatever their shortcomings. But what's available? What systems or tools might help you get your project done? Developer University has released the Macintosh Development Tools Advisor to help answer these questions.

The Tools Advisor offers a broad array of information. A hypertext system, it tailors the data it presents to your particular interests and demands. The Advisor incorporates comprehensive technical data on over 80 programming tools—compilers and languages, debuggers and prototypers, CASE tools, and multimedi packages. It also includes essays on critical topics such as object-oriented programming, Apple events, and System 7. In preparing the Tools Advisor, Developer University collected a considerable body of catalog-styl information on products available.

But a catalog is rarely sufficient. It's not enough to read lists of capabilities as recorded by manufacturers. You need to know how the tools get used in actual projects. So the Tools Advisor provides a collection of stories by programmers who use the tools it describes. These stories provide a real feel for the product. They're sometimes critical, warning of potential hazards and

You can obtain a copy of Advisor through APDA. The c can also be found on the *Deve* disc. To use the Tools Advisor, Macintosh with System 6.0.5 HyperCard 2.0 or later, and a the CD-ROM version, you'll of a CD-ROM drive. •

d found out that
he electronic
d: Art that's in
ned in an
PostScript® and
process wasn't
t the conversion
cimation, and so
e fixed in Issue 7

Regarding the HyperCard format, a lot of people agree with you. We're working on an alternate viewing mechanism—but this mechanism may not apply to develop *for a while yet. Meanwhile, HyperCard's windows are in fact resizable. If you're not able to resize them, your memory partition for HyperCard is probably not large enough; try increasing it.*

—Caroline Rose

H DEVELOPMENT TOOLS ADVISOR

shortcomings of particular tools. They're also often inspiring in explaining how particular achievements were made. To help you find stories most appropriate to you, the Advisor lets you match a loose profile of your needs and wants to stories by developers with similar backgrounds and tasks.

To augment its profiles of programming tools, critical essays, and developers' stories, the Tools Advisor includes a glossary that describes exactly what technical and trade terms mean and what they imply to a development effort. Glossary entries and cross-references let you navigate the intricate terrain of technical information without losing sight of your particular interests.

Two versions of the Tools Advisor are available. The disk-based edition includes screen shots and comprehensive data on programming tools in a range of categories as well as technical details on the Macintosh and on Macintosh programming in general. The CD-ROM edition of the Tools Advisor adds demonstration versions of dozens of tools; for instance, you can take a multimedia tool for a test drive as you learn about animations and about other developers' experiences with that product.

We hope that with the Tools Advisor guiding you, you won't feel lost any more.

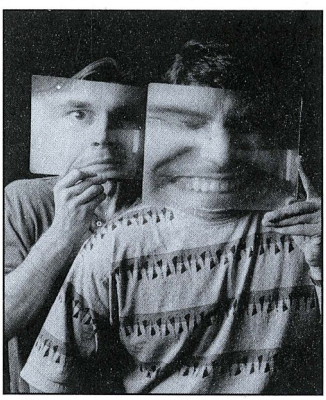
---

the Tools

disk-based version
eloper CD Series
you'll need a
or later,
hard disk. To use
course also need

MAKING THE MOST OF COLOR ON 1-BIT DEVICES

Macintosh developers fac
develop software for—m
with Color QuickDraw—
adequately on the lower-
when running on the hig
simple and elegant soluti
development and testing
outcome more satisfying,
process them for display



KONSTANTIN OTHMER
AND DANIEL LIPTON

Suppose you're writing a pro
it to work on all Macintosh c
with the original QuickDraw
support for bitmaps, thus sev
despair. In our continuing qu
with original QuickDraw, we
process them for display, albe
come up with a technique to
resolve single pixels, which re
accompanying sample code (
with you.

SAVING COLOR IMA

The key to saving color imag
QuickDraw is a transcript of
PICT created on one Macint
the version of system softwar
later than the version on the
Macintosh Plus you can draw

KONSTANTIN OTHMER has wa
photograph to appear in *Sports Illu*
long as he can remember. Unfortun
college was in the NCAA's Division
often overlooked by SI's editors, an
they've missed his virtuosity on the
Tahoe, Vail, and Red Lodge. So Ko
scale down his dream, setting his si
making the pages of *develop* instea

ed with the dilemma of which platform to
achines with the original QuickDraw or those
—can always choose to write code that runs
end machines and gives additional functionality
gher-end machines. While this sounds like a
ion, it generally requires a great deal of
effort. To make this effort easier and the
we offer techniques to save color images and
on 1-bit (black-and-white) devices.

gram that controls a 24-bit color scanner and you'd like
omputers. The problem you'll run into is that machines
(those based on the 68000 microprocessor) only have
erely crippling the potential of your scanner. But don't
est to add Color QuickDraw functionality to machines
've worked out techniques to save color images and
it in black and white, on the latter machines. We've also
address the problem of a laser printer's inability to
sults in distorted image output. This article and the
on the *Developer CD Series* disc) share these techniques

AGES

es is using pictures. Recall that a picture (or PICT) in
calls to routines that draw something—anything. A
osh can be displayed on any other Macintosh (provided
e on the machine doing the displaying is the same as or
machine that created the picture). For example, on a
a PICT containing an 8-bit image that was created on a

nted his
strated for as
ately, his
III, which is
d somehow
ski slopes at
n's had to
ghts on
d. Here he's

gotten to try on various alter egos. To come up with his latest persona, he spent a few late nights in a secret Apple lab with skilled pixel surgeon Jim Batson.

Macintosh II. With Syste
bit pixMaps on machines
displayed as 1-bit images

Creating a picture normal

1. Call OpenPicture
2. Perform the drawi
3. Call ClosePicture

The catch is that the only
those available on the Ma
this procedure on a machi
pixMaps into a picture, si
can't create an 8-bit PICT
But that's exactly what wo
PICT containing deep pi
With this ability, you coul
Color QuickDraw machi
machine with original Qu

To get around the limitati
called CreatePICT2 to m
application can display th
whether creating your ow
who directly modify priva
ease your mind, see "But

The parameters to Create
procedure stdBits. The di
not use a maskRgn.

The first thing the routin
allocate that amount of st
returning a NIL PicHand
to disk if the memory is n
Rather than writing out t
specified number of bytes
Essentially, you need an e

At this point the size of th
well the pixMap will com
picture frame. Next is the
with a header that has ver
ignore the header data. (

DANIEL LIPTON (a.k.a. "The
a two-and-a-half-year veteran o
Software Imaging Group, whe
the next generation of printing
Macintosh. When he's not thin
enjoys taking in a good flick, s
his iguana, "Iggy" (who's neve
Dan for the time she nearly fro
cargo compartment of a 747),

m 7, you can even display PICTs containing 16-bit and 32-
with original QuickDraw. (Of course, they will only be
there.)

lly requires three steps:

to begin picture recording.

ng commands you want to record.

to end picture recording.

drawing commands that can be recorded into a picture are
cintosh on which your application is running. Thus, using
ne with original QuickDraw provides no way to save color
nce there's no call to draw a pixMap. In other words, you
on a Macintosh Plus and see it in color on a Macintosh II.
uld make a developer's life easier—the ability to create a
xMap information on a machine without Color QuickDraw.
d capture a color image in its full glory for someone with a
ne to see, while still being able to display a 1-bit version on a
ickDraw.

ons of the normal procedure, we came up with a routine
anually create a PICT containing color information. Your
e picture using DrawPicture. Now, you may be wondering
n pictures is advisable. After all, Apple frowns on developers
te data structures, and isn't that what's going on here? To
Don't I Need a License to Do This?"

ePICT2 are similar to those for the QuickDraw bottleneck
fference is that CreatePICT2 returns a PicHandle and does

e does is calculate a worst-case memory scenario and
orage. If the memory isn't available, the routine aborts,
lle. You could easily extend this routine to spool the picture
ot available, but that's left as an exercise for you. (*Hint*:
ne data inline as is done here, call a function that saves a
in the picture. Have that routine write the data to disk.
quivalent to the putPicData bottleneck.)

ne picture is not known (since there's no way to know how
press) so we simply skip the picSize field and put out the
picHeader. CreatePICT2 creates version $02FF pictures,
sion $FFFF. This version of the header tells QuickDraw to
OpenCPicture, available originally in 32-Bit QuickDraw

PostScript Kid") is
f Apple's System
e he's working on
software for the
king backward, he
pending time with
r quite forgiven
ze to death in the
and writing zany

new lyrics to classic tunes (his "Working in the Print Shop Blues" is well known to his coworkers). Most of all, Dan enjoys building and flying model airplanes, and he's recently joined the competition circuit. In fact, when asked what he'd really like to do with his life, Dan replies:

```
sunny { { { { hours 8 { flying } for } rather_be }
    dayforall } } if •
```

BUT DON'T I NEED A LICENSE TO DO TH

The reason Apple doesn't want developers modifying data structures is that it makes it hard to change them in the future. For example, early Macintosh programs locked handles by manually setting the high bit of the handle rather than calling HLock. This caused numerous compatibility problems when the 32-bit-clean Memory Manager was introduced.

So what gives? What if Apple changes OpenPicture so that it creates a totally different data format—won't the manually created pictures break?

Calm down, because the answer is no. The difference between creating your own pictures and directly modifying other data structures is that Apple can't make the current picture data format obsolete without invalidating users' data that exists on disk. Just as you can still call DrawPicture on version 1 pictures and everything works, you will always be able to call DrawPicture on existing version 2 pictures, regardless of the format of pictures created in the future.

version 1.2 and in Color QuickDraw in System 7, still creates but the header version is now $FFFE and contains picture re

In addition, the bounds of the clipping region of the current picture. Without this, the default clipping region is wide ope QuickDraw have trouble drawing pictures with wide-open cl

Next we put out an opcode—either $98 (PackBitsRect) or $9 depending on whether the pixMap is indexed or direct. Then dstRect, and mode are put in the picture using the (are you re PutOutPixMapSrcRectDstRectAndMode routine. Finally, ei PutOutPackedDirectPixData or PutOutPackedIndexedPixD the pixel data.

There's an important difference between indexed and direct baseAddr field is skipped when putting out indexed pixMaps for direct pixMaps. This is done because machines without su pixMaps (opcode $9A) read a word from the picture, skip tha

IS?

One possible pitfall is that you might create a picture with ubtle compatibility risks that draws on the existing system oftware but breaks at some future date. To minimize the hances of such an occurrence, you should compare the ictures you generate with those that QuickDraw enerates in identical circumstances. You must be able to ccount for any and all differences.

Creating your own pixMaps (as our example code does) s definitely in the gray area between risky and outright lisastrous behavior, and you shouldn't do it. Then why vould an article written by two upstanding citizens do uch a thing? The answer is that the pixMaps used by this ode are kept private; they're never passed as arguments o a trap. We could just as easily have called them omething else, but pixMaps work for what we're doing, o we used them. If you want to pass a pixMap to a trap, ou can generate it using the NewPixMap call (not vailable on machines with original QuickDraw) or let ther parts of Color QuickDraw, like OpenCPort, enerate it.

s version $02FF pictures,
solution information.)

port are put in the
n, and some versions of
ipping regions.

A (DirectBitsRect),
the pixMap, srcRect,
eady for this?)
ther
ata is called to put out

ixMaps here. The
and is set to $000000FF
upport for direct
t many bytes, and

continue picture parsing.
baseAddr, the number of
ends the picture playback

An interesting fact buried
the value of packType. Al
to be unpacked. Thus, yo
the pixMap should get wh
Compression" (*develop* Iss
compression schemes use
packing schemes lose no i
in the lead article in *devel*
sacrifice some image qual

Anyway, these routines su
any pixMap with rowByte
bit direct pixMaps with r
support 16-bit pixMaps.

Finally, the end-of-pictur
actually used.

```
PicHandle CreatePICT2(
   short mode)
{
PicHandle    myPic;
short        myRowByte
short        *picPtr;
short        iii;
long         handleSiz

#define CLIPSIZE 12
#define PIXMAPRECSIZE
#define HEADERSIZE 40
#define MAXCOLORTABLES
#define OPCODEMISCSIZE
#define ENDOFPICTSIZE
#define PICSIZE PIXMAP
   ENDOFPICTSIZE + OPC

   myRowBytes = srcBit
/* Allocate worst-case
   myPic = (PicHandle)
      ((myRowBytes/12
      - srcBits->bounc
```

When such a machine encounters the $000000FF
bytes skipped is $0000 and the next opcode is $00FF, which
. A graceful exit from a tough situation.

l in the PutOutPixMapSrcRectDstRectAndMode routine is
l in-memory pixMaps (that aren't in a picture) are assumed
u can set the packType field to specify the type of packing
ıen put in a picture. "The Low-Down on Image
sue 6, page 43) gives details of the different pixMap
d by QuickDraw. Note that all of QuickDraw's existing
image quality. QuickTime (the new INIT described in detail
op Issue 7) adds many new packing methods, most of which
ity to achieve much higher compression.

ıpport only the default packing formats: 1 (or unpacked) for
es less than 8, 0 for all other indexed pixMaps, and 4 for 32-
owBytes greater than 8. Note that these routines do not

e opcode is put out and the handle is resized to the amount

```
PixMap *srcBits, Rect *srcRect, Rect *dstRect,


s;


e;


50

IZE 256*8+8
 2+8+8+2  /* opcode+srcRect+dstRect+mode */
2
RECSIZE + HEADERSIZE + MAXCOLORTABLESIZE + \
CODEMISCSIZE + CLIPSIZE

ts->rowBytes & 0x3fff;
 memory scenario using PackBits packing. */
 NewHandle(PICSIZE + (long)
7)+2+myRowBytes)*(long)(srcBits->bounds.bottom
ds.top));
```

```
    if(!myPic)
        return(0);

/* Skip picSize and put out picFrame (10 bytes). */
    picPtr = (short *) (((long)*myPic) + 2);
    *picPtr++ = dstRect->top;
    *picPtr++ = dstRect->left;
    *picPtr++ = dstRect->bottom;
    *picPtr++ = dstRect->right;

/* Put out header (30 bytes). This could be done from
    taken from an existing picture. */
    *picPtr++ = 0x11;       /* Version opcode. */
    *picPtr++ = 0x2ff;      /* Version number. */
    *picPtr++ = 0xC00;      /* Header opcode. */
    *picPtr++ = 0xFFFF;     /* Put out PICT header vers
    *picPtr++ = 0xFFFF;
/* The rest of the header is ignored--0 it out. */
    for(iii = 10; iii > 0; iii--)
        *picPtr++ = 0;      /* Write out 20 bytes of 0.

/* Put out current port's clipping region. */
    *picPtr++ = 0x01;       /* Clipping opcode. */
    *picPtr++ = 0x0A;       /* Clipping region only has
    *picPtr++ = (**thePort->clipRgn).rgnBBox.top;
    *picPtr++ = (**thePort->clipRgn).rgnBBox.left;
    *picPtr++ = (**thePort->clipRgn).rgnBBox.bottom;
    *picPtr++ = (**thePort->clipRgn).rgnBBox.right;

    HLock(myPic);
    if(srcBits->pixelType == RGBDirect)
    {          /* Must be 32-bits/pixel */
    /* Put out opcode $9A, DirectBitsRect. */
        *picPtr++ = 0x9A;
        *picPtr++ = 0;  /* BaseAddr for direct pixMaps
        *picPtr++ = 0xFF;
        PutOutPixMapSrcRectDstRectAndMode(srcBits, &pi
            dstRect, mode);
        if(PutOutPackedDirectPixData(srcBits, &picPtr)
            goto errorExit;     /* Nonzero indicates an
    }
    else
    {
    /* Put out opcode $98, PackBitsRect. */
        *picPtr++ = 0x98;
```

```
n a resource or

ion. */

 */

 bounds rectangle. */

 is 0x000000FF. */

cPtr, srcRect,

)
error. */
```

```
            PutOutPixMapSrcR
               dstRect, mode
            if(PutOutPackedI
               /* Nonzero in
               goto errorExi

         }
         HUnlock(myPic);

      /* All done! Put out e
         *picPtr++ = 0x00FF;

      /* Size handle down to
         handleSize = (long)
         SetHandleSize(myPic
         /* Write out pictur
         *((short *) *myPic)
         return(myPic);

      errorExit:
         DisposHandle(myPic)
         return(0);
      }
```

Just remember that it's no
The reason is that althou
it's not a persistent data st
application.

The subroutines the Crea
CreatePICT2 are on the

PROCESSING COL

The remainder of this art
(black-and-white) devices

There are many techniqu
color resources are limite
routines for determining
color space. For example,
Picture Utilities can tell y
CreatePICT2 routine jus
using the Picture Utilities

```
RectDstRectAndMode(srcBits, &picPtr, srcRect,
e);
IndexedPixData(srcBits, &picPtr))
ndicates an error. */
it;



nd-of-picture opcode, $00FF. */


 the amount actually used. */
 picPtr - (long) *myPic;
, handleSize);
re size. */
 = (short) handleSize;



;
```

ot advisable to pass a pixMap you create yourself to a trap. gh it's unlikely, the format of a pixMap could change (since tructure, as a picture is); this would then break your

atePICT2 routine calls as well as some sample code that uses *Developer CD Series* disc.

LOR IMAGES FOR DISPLAY

icle focuses on processing color images for display on 1-bit , both monitors and laser printers.

es for representing a full-color image on a monitor when d. The Picture Utilities Package (new in System 7) offers optimal colors to use when displaying a pixMap in a limited if you want to display a 32-bit image on an 8-bit monitor, ou the 256 best colors to use to display the image. The t described creates a picture that you can legally analyze s.

You can also use the techniques of thresholding and of dither
three varieties: error diffusion, ordered, and random. Ordere
as halftoning, is particularly useful for producing images to b
printer. We'll examine each of these techniques in turn.

USING A 50% THRESHOLD

The first technique that leaps to mind when one is faced with
picture on a 1-bit screen is to convert each color to a luminan
threshold value to determine whether or not to set the corres
out that green contributes the most to the luminance and blu
Red, green, and blue contribute approximately 30%, 59%, an
the luminance. Thus, our formula to convert an RGB value t

Luminance = (30*RED + 59*GREEN + 11*BLUE)/100

If the resulting luminance is 128 (50% of 256) or greater, the
otherwise it's set to black. This technique produces the result
gray gradations and a lovely picture of one of the authors. No
occurs at the source pixel resolution. Thus, even though the
produce Konenna is 300 dpi, the thresholded picture appears
contrast, the techniques of error-diffusion dithering and halft
following pages occur at the destination device resolution.

The results shown in Figure 1 are far from ideal. The gray gr
black rectangle beside a white rectangle, and the picture of K
is completely devoid of detail.

USING ERROR-DIFFUSION DITHERING

The major problem with the threshold algorithm is that a gre
thrown away. The luminance is calculated as a value between
information we use is whether it's 128 or greater.

An easy fix is to preserve the overall image lightness by maint
and then passing the error onto neighboring pixels. Both orig
QuickDraw have dithering algorithms built in for precisely th
true—while a dither flag cannot be passed explicitly to any or
a picture containing a color bit image created using dither mo
QuickDraw machine will dither when drawn with original Qu
calculated as

Error = Requested Intensity - Closest Available Intensity

For a black-and-white destination, the closest available intens
255 (white). The requested intensity is the luminance of the

ing, of which there are
d dithering, also known
e printed on a laser

displaying a color
ce and then use a
ponding pixel. It turns
e contributes the least.
d 11%, respectively, to
o a luminance becomes

pixel is set to white;
s shown in Figure 1 for
te that thresholding
utput device used to
to be 72 dpi. In
oning discussed on the

adations end up as a
onenna, while still cute,

at deal of information is
0 and 255, but the only

aining an error term
inal and Color
is purpose. (Yes, it's
iginal QuickDraw trap,
de on a Color
uickDraw.) The error is

ity is either 0 (black) or
urrent pixel plus some

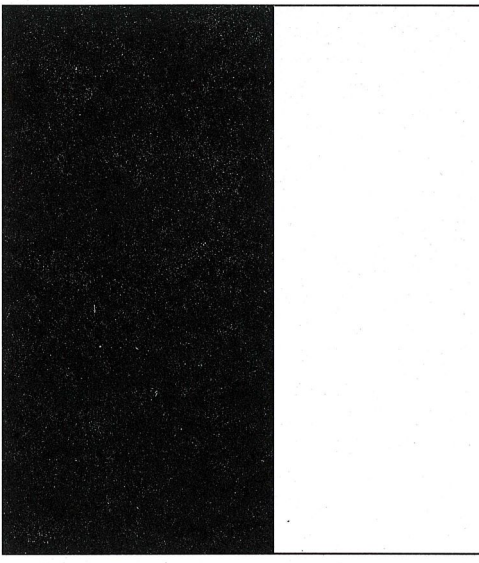


Figure 1
Gray Gradations and Kone

part of the error term of
among all surrounding pi
uses a shortcut. In origin
scan lines and to the left
technique, except it push
to the pixel immediately
QuickDraw at monitor r

This form of dithering is
each pixel is thresholded
across the image in some
produces very pleasing re
accurately rendering a sir
this; laser printers are no
laser printer, a different t

USING ORDERED DITHI
There are two kinds of la
printer (such as some of t
process) starts the image

enna Pictured Using 50% Threshold

surrounding pixels. Ideally, the error term is spread evenly ixels. But to maintain acceptable performance, QuickDraw al QuickDraw, the error term is pushed to the right on even on odd scan lines. Color QuickDraw uses the same es only half the error to the left or right, and the other half below. The result of using this technique in Color esolution for the two test images is shown in Figure 2.

normally referred to as error diffusion. That is to say that at 50%, but the error incurred in that process is distributed manner, thus minimizing information loss. Error diffusion esults when the device being drawn onto is capable of ngle dot at the image resolution. Monitors are quite good at t. If you want your application's output to look good on a echnique is called for.

ERING (HALFTONING)

ser printers: write-white and write-black. A write-white the high-end Linotronic printers that use a photographic out black and uses the laser to turn off pixels. A write-black

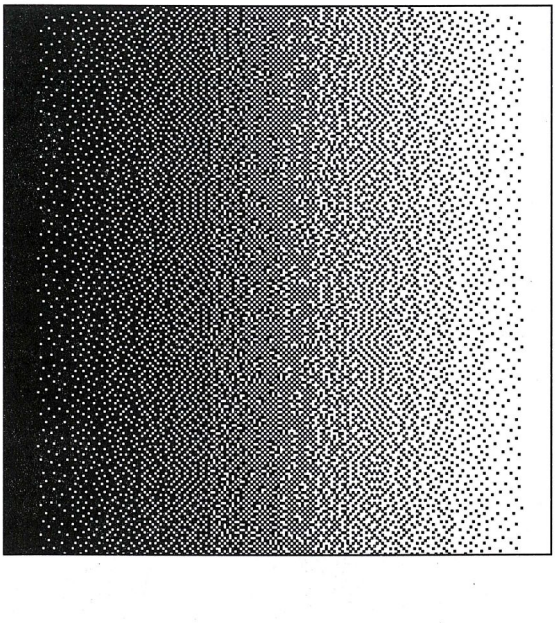


Figure 2
Gray Gradations and Konenna Dithered at Monitor Resolution

printer (such as Apple's LaserWriter) starts the image out whi
with the laser. Since the pixels are thought of as being square
round, neither process can accurately turn on or off single pix

Generally, the circle generated by the laser beam is slightly bi
the computer "sees" it, to guarantee that all space is covered (
effect of this with a write-black printer is that the black dots t
the individual pixels, causing any 1-bit image drawn at device
dark. The effect with a write-white printer is that the black d
than the individual pixels, causing any 1-bit image drawn at d
appear too light. If the area of the circle is 20% greater than t
percentage of unwanted toner, or error, for a single pixel is 20

Figure 3
A Laser's Idea of a Square Pixel

te and turns on pixels
and the laser beam is
els.

gger than the pixel as
see Figure 3). The
end to be bigger than
resolution to appear too
ots tend to be smaller
evice resolution to
he individual pixel, the
%.

Because the error is intro
two or more pixels are dr
reduced to the perimeter
single pixel is 20%, two p
error, and four pixels in a

Ordered dithering, or ha
clumping pixels. Pixels ar
other and the luminance
way that clumps of pixels
This allows us to minimi
order is determined by w
things get deep, so put or
following to use the samp

About the dither matri
gray or primary colors, w
effectively lower the devi
shades that we can print.
pixel on the page, we've l
now have 2^4 or 16 differe
pattern has anywhere fro
0 and 100%. In fact, for t
0%, 25%, 50%, 75%, an
the cell. The dither matr
represent the five possibl
matrixes using the algori
CD Series disc has a comm

If we construct a matrix v
use (2x2 for the describe
100, we can use this matr
in the pattern correspon
gray, we turn on all the d
or equal to 50. The posit
pattern, as shown in Figu

The dither matrix is used
threshold described earli
element has a value of 50
where (x, y) is the device
matrix.

It turns out that the spati
the dither matrix. For re

ɔduced only at the black/white boundaries, it's reduced when
awn next to each other. Then the percentage of error is
of the pixel group. So in the case where the error for a
ixels drawn next to each other would have only a 15.5%
square would have only a 10.25% error in the area covered.

lftoning, minimizes the dot-to-pixel error just described by
e turned on and off in a specific order in relation to each
of the source image. The order can be specified in such a
next to each other are turned on as the luminance decreases.
ze the effects of the laser printer's dot-to-pixel error. The
hat's known as a dither matrix. (*Warning:* From here on out,
ı your waders. You don't really need to understand all the
ɔle code we provide.)

x. With a dither matrix, to render intermediate shades of
e sacrifice spatial resolution for shading—that is, we
ce's dots-per-inch rating while increasing the number of
For example, if we use a 2x2 cell of 300-dpi dots for every
owered the spatial resolution of the device to 150 dpi but we
ent patterns to choose from for each one of the pixels. Each
m 0 to 4 of the 300-dpi dots blackened, or a density between
he 16 possible patterns there are only five possible densities:
d 100%, corresponding to 0, 1, 2, 3, and 4 dots blackened in
ix determines which five of the possible patterns to use to
e densities. It's left to you as an exercise to generate these
thm we provide below. (The sample code on the *Developer*
monly useful example.)

with the same dimensions as the dot cell that we're going to
d case) so that the matrix contains the values 25, 50, 75, and
ix to determine each of the five possible patterns. Each dot
ds to a position in the matrix. To generate a pattern for 50%
ots in the pattern with corresponding matrix values less than
ion of the values in the matrix determines the shape of the
ıre 4.

l to render an image in much the same way as the 50%
er. In fact, that process uses a 1x1 dither matrix whose single
%. The dither matrix is sampled with (x mod m, y mod n),
pixel location and (m, n) is the width and height of the dither

ial resolution of the device isn't really reduced by the size of
gions that are all black, for example, the resolution remains

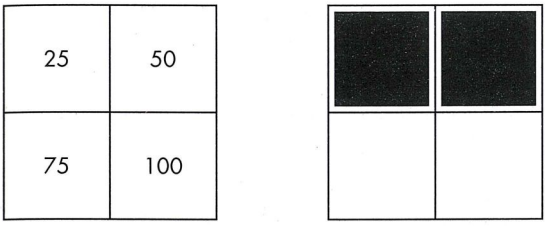


Figure 4
A 2x2 Dither Matrix

the device resolution. Each pixel in the device is still sampled
source image.

The basic algorithm for doing an ordered dither of an image
following:

For all device pixels x, y:

- s_x, s_y = transform(x, y) where transform maps device pi
coordinates to source pixel coordinates
- If sourceLuminance(s_x, s_y) > ditherMatrix[x mod m, y
device-dot(x, y) = black

The code on the *Developer CD Series* disc is an elaboration on

As stated before, the position of the various values in the dith
patterns that various luminances generate. A general way to s
use a spot function, as the PostScript interpreter does. If the
matrix is thought to be a continuous space whose domain is 0
directions, *spot-function*(x, y) will return some value that ultim
into a luminance threshold in the matrix. If the desired patter
from the center as the luminance decreases (known as a cluste
function(x, y) is simply the distance from (x, y) to the center of
dither matrix would be generated from the spot function as fo

```
for i = 1 to m
    x = i/m
    for j = 1 to n
        y = j/n
        matrix[i, j] = spot-function(x, y)
```

The result of this process is that the matrix contains the spot
we really want in the matrix are threshold values for the lumi

back to a pixel in the

onto a page becomes the

ixel

mod n],

this basic algorithm.

er matrix determines the
pecify this order is to
rectangle of the dither
–1 in the x and y
ately can be converted
n is a dot that grows
ered-dot halftone), *spot-*
the cell (0.5, 0.5). The
llows:

function's results. What
nance. The spot

function result is convert
dimensional array A, gen
from 1 to $m*n$. Then, rep
the desired threshold ma
code uses numbers that a
percentages assume that
The values can be modifi
relationship between ima

Ordered dithering is gen
is the number of cells (or
produced patterns are ori
example, the frequency (i
angle is 0°.

Because of the way our b
but not at 45° angles), it's
the dither matrix itself is
the dither matrix in such
to achieve the effect of b
requires us to "tile" an ar
enclosing a part of the ro
rotation, this rectangle m

Suppose we want to halft
and an angle of 45°. At 0°
possible shades of gray. H
approximate the desired
(0, 5) and (5, 0) by 45° an
(-4, 4). Since the magnitu
frequency achieved will b
angle is due to the need t

Here's the basic algorithm

1. The halftone cell
the vectors $(x_1, y_1$
2. A, the area of the
required dither m
vertical dimensio
$Q = GCD(x_2, x_1)$.
3. For every point i
we want to find it
repeated halftone
(See Figure 6.) C

The source of step 2 in the
"An Optimum Algorithm for H
for Displays and Hard Copies
Holladay, from the *Proceeding*
Information Display, Vol. 21, N

ed as follows: Treating the dither matrix as a one-
erate a sort vector V such that $A[V[i]]$ is sorted as i goes
lacing all of the values in A with $V[i] * 100/(m*n)$ will yield
rix, with each value being a percentage of luminance. (The
re more computer-friendly than percentages.) These
he device is capable of accurately rendering a single pixel.
ed by a gamma function to more accurately produce a linear
ge luminance and pixel density.

erally done at a specific angle and frequency. The frequency
dither matrixes) per inch and the angle refers to how the
ented with respect to the device grid. In the preceding
f printing on a 300-dpi device) is 150 cells per inch and the

rains work (our eyes tend to pick up patterns at 90° angles
desirable to orient these patterns at arbitrary angles. Since
never rotated with respect to the device, we must generate
a way that it contains enough repetitions of the rotated cell
eing rotated itself. In other words, because a square device
ea with 0° rectangles, we need to find a 0° rectangle
tated pattern that forms a repeatable tile. For some angles of
ay be much larger than the pattern itself.

one to a 300-dpi device at a frequency of 60 cells per inch
, the dither matrix would be 5x5 (300/60), yielding 26
lowever, as Figure 5 illustrates, we need an 8x8 matrix to
angle. These dimensions are found by rotating the vectors
d pinning them to integers, yielding the vectors (4, 4) and
ude of the vector (4, 4) is 4*sqrt(2), the actual halftone
e 300/(4*sqrt(2)), around 53. The error in frequency and
o pin the vectors to integer space.

n for computing the dither matrix:

is specified by the parallelogram composed of
) and (x_2, y_2) and based at (0, 0).

modified halftone cell, is $(x_1*y_2) - (x_2*y_1)$. For the
atrix, the horizontal dimension is A/P and the
n is A/Q, where $P = GCD(y_2, y_1)$ and

n the matrix, which is in (x, y) orthogonal space,
s relative position in the space of one of the
cells, defined by the vectors (x_1, y_1) and (x_2, y_2).
ll this point (u, v). The transformation is

---

above algorithm is
alftone Generation
" by Thomas M.
s of the Society for
No. 2, 1980.

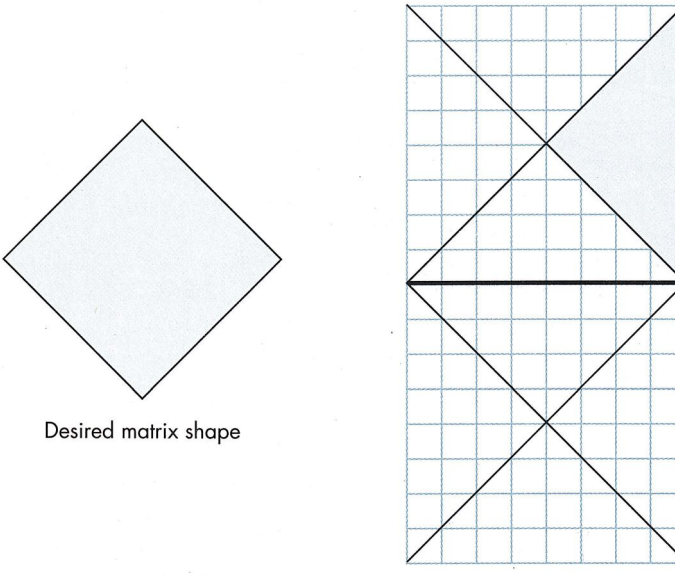


Figure 5
Approximating the Desired Angle

$u = A*x + B*y$, $v = C*x + D*y$. Since the point (x_2, y_2) in
is the point (1, 0) in halftone cell space and the point (
point (0, 1) in halftone cell space, the coefficients A, B
are found by solving the following simultaneous linea

$$
\begin{aligned}
A*x_1 + B*y_1 &= 0 \\
C*x_1 + D*y_1 &= 1 \\
A*x_2 + B*y_2 &= 1 \\
C*x_2 + D*y_2 &= 0
\end{aligned}
$$

We compute the dither matrix in the rotated case as follows:

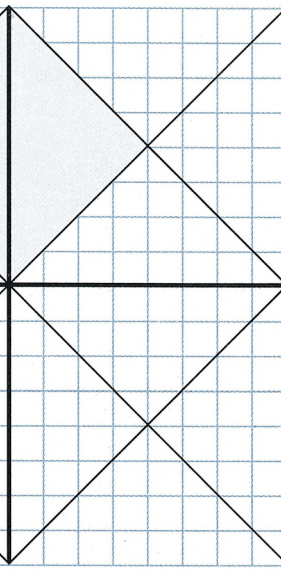
For each position in the matrix (i, j):

- Get (x, y) the center of the matrix point (i, j)

  $x = i + 0.5$
  $y = j + 0.5$

- Transform (x, y) to a point in halftone cell space (u, v)

  $u = A*x + B*y$
  $v = C*x + D*y$



x repeated four times

(*x*, *y*) space
(x_1, y_1) is the
C, and *D*
r equations:

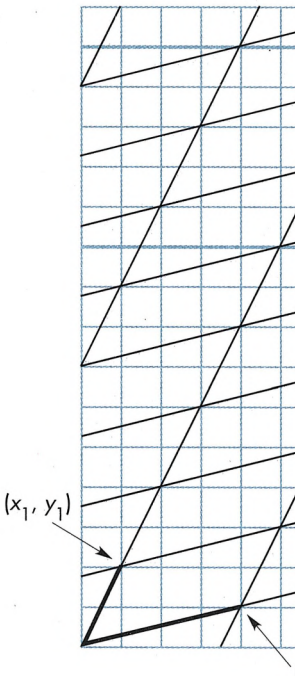


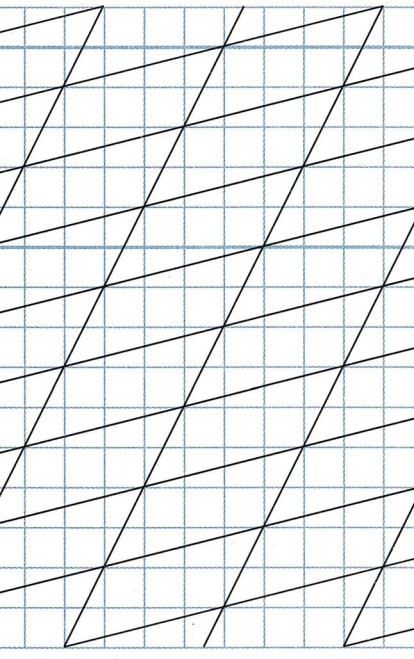
Figure 6
Transforming a Halftone Ce

u and v now expre
vectors. Therefore
position as if the p
(0, 0) cell.

- Z = *spot-function* (u
- Find the index of t
that $u = x$, $v = y$. If
table. (Note that t
allowable epsilon
error.)
- matrix[i, j] = index

Find the order of rec
(described earlier in
Reassign values of ma

Figure 7 shows our examp
luminances, filled in. A lu



(x_2, y_2)

$$A * x_1 + B * y_1 = 0$$
$$C * x_1 + D * y_1 = 1$$

$$A * x_2 + B * y_2 = 1$$
$$C * x_2 + D * y_2 = 0$$

$$A = \frac{y_1}{x_2 * y_1 - x_1 * y_2}$$

$$B = \frac{x_1}{-x_2 * y_1 + x_1 * y_2}$$

$$C = \frac{y_2}{-x_2 * y_1 + x_1 * y_2}$$

$$D = \frac{x_2}{x_2 * y_1 - x_1 * y_2}$$

ell

ss the point (x, y) as multiples of the two cell e, the fractional parts of u and v represent the particular halftone cell at the point (x, y) were the

- floor(u), v - floor(v))

the record (containing fields x, y, and Z) such the record doesn't exist, enter u, v, Z into the he equality between $[u, v]$ and $[x, y]$ requires an difference to account for fixed-point round-off

ords sorted by values of Z; store order in sort vector connection with converting the spot function result). atrix based upon sort vector.

ble matrix with values from 0 through 255, representing minance from an image with this range could be sampled

directly against the matrix. The values in this matrix are those used for a 300-dpi, 60-line-per-inch, 45° halftone. As in Figur repeated four times for the sake of clarity, with the 45° halfton position of any particular number in the matrix relative to the corresponds exactly to the relative position of that same numb 45° cells. Thus, the effect of having a rotated halftone cell is c unrotated dither matrix.

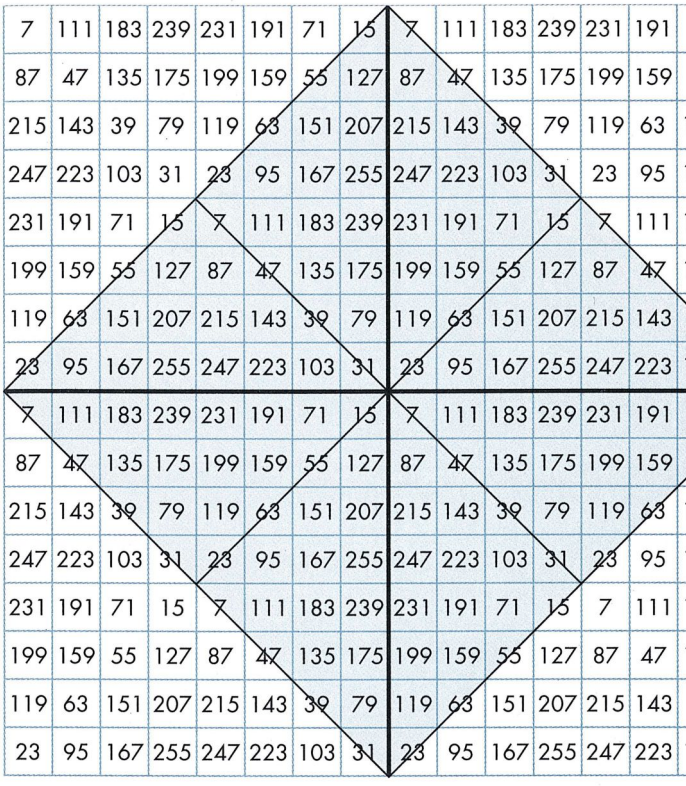
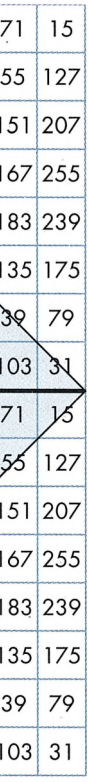


Figure 7
Our Example Matrix With Luminance Values Filled In

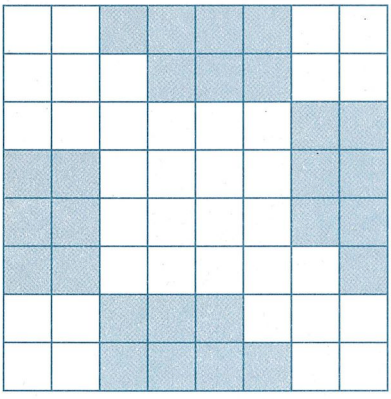
This particular example leads us to some other interesting po that QuickDraw patterns are 8x8 matrixes, just like our exam can halftone other QuickDraw primitives besides pixMaps wh non-PostScript device (provided that pattern stretching is dis bPatScale field in the print record to 0) and achieve a look sin PostScript device would give us.

e that would actually be
e 5, the matrix is
ne cells overlaid. The
45° cell it falls in
ber in any of the other
reated with an



ssibilities. It turns out
ble. This means that we
hen drawing to a 300-dpi
abled, by setting the
nilar to what a

Here's how. Suppose we w
from 0 to 255. We simply
correspond to the cells in
pattern (shown in Figure
primitive to get the halfto
aligned to the origin of th
will not generate undesira
nature of the clustered do
extent possible at the reso



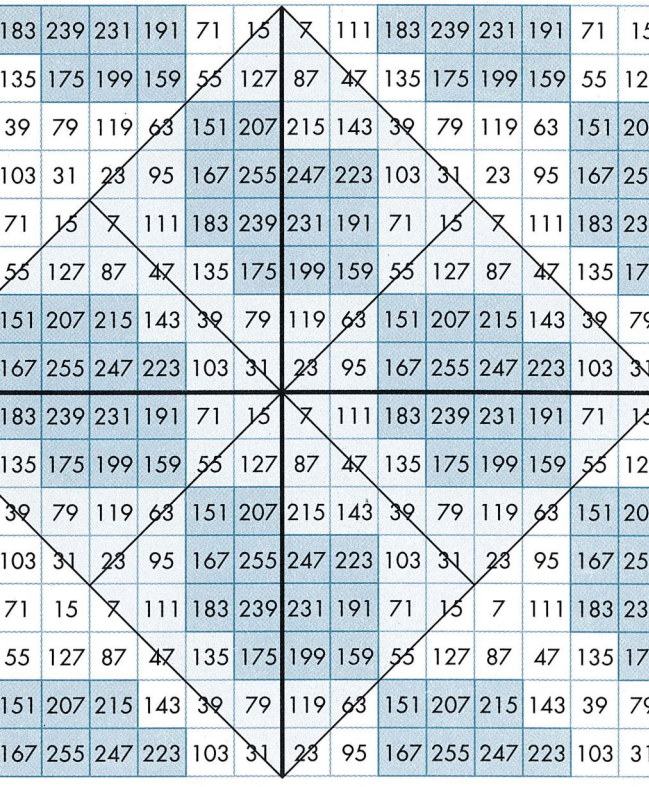
8x8 pattern

7	111
87	47
215	143
247	223
231	191
199	159
119	63
23	95
7	111
87	47
215	143
247	223
231	191
199	159
119	63
23	95

Figure 8
Pattern for an Image With a

Figure 9 shows the gray g
error-diffusion dithering
difference in print quality
"Printing: Ideal Versus Re

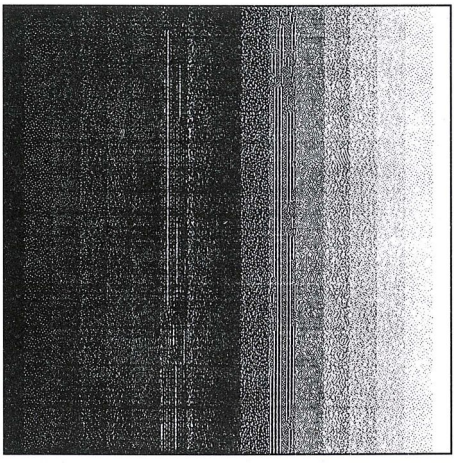
vant to paint a region with a luminance of 150 on the scale
create a QuickDraw pattern in which all of the 1 bits
the 8x8 matrix that are greater than or equal to 150. This
8) can then be used to paint any region or other QuickDraw
ne effect. Furthermore, because QuickDraw patterns are
e grafPort, separate objects drawn touching one another
ble seams, even when drawn with different shades. The
t pattern is such that gradations appear continuous to the
lution of the device.



Halftone dots created by repetition of 8x8 pattern

Luminance of 150

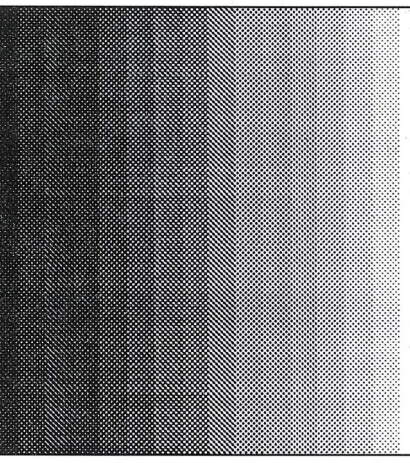
radations and Konenna printed on a laser printer using
compared with halftoning using the 8x8 matrix. The
is radical. For more commentary on this difference, see
eal."



Gray gradations dither

Konenna dither

Figure 9
Gray Gradations and Konenna Dithered and Halftoned at Laser R



Gray gradations halftone

Konenna halftone

rinter Resolution

PRINTING: IDEAL VERSUS REAL

We've already talked about the error introduced in printing by the fact that the laser beam is round while the pixel is square. Many other factors also can make the transfer of toner to paper deviate from the ideal. Sources of error include differences in inks, papers, printer drums, and even humidity. Additionally, a printer's behavior changes over time as the drum wears. Compensating for all these factors to achieve ideal images would require constant calibration and recalibration of the printer.

An error appears most pronounced in the final print when imaging directly at device resolution, as Figure 9 shows. Halftoning hides much of this error and produces reasonably uniform results among printers with varying degrees of error.

The tonal reproduction curves (known as TRC or gamma curves) shown in Figure 10 indicate the gray levels produced by the Apple LaserWriter when dithering and halftoning. Note that with dithering, the measured

About the code. And no
our sample code is pixel-b
Thus, the perfomance is s
optimized version of this
scan-line rather than a pix
only supports input pixMa
accept pixMaps of other d

The first routine we need
current pixel. The LUMV
255 using the 30%-59%-

```
long LUMVAL(Ptr pPixel
   {
      long      red, g

      if (pMap->pixelS
         red = (long)(

         green = (long
```

luminance of an image remains dark much longer than with halftoning as requested luminance increases, due to the error when each pixel is printed. Of particular interest is the point on the dither curve right at 50% luminance. The measured luminance is actually darker than when 44% luminance is requested. The reason is that with a 50% dither, every other pixel is drawn, maximizing the effect of the laser error.

While the TRC curve for the halftone print doesn't match the ideal curve, it's much closer to the ideal than is the dither curve. To get the halftone even closer to ideal, you could adjust the luminance calculation by the amount indicated by the halftone TRC to compensate. Indeed, most image-processing applications perform this TRC adjustment to compensate for the nonlinearities of the output device. See *Designing Cards and Drivers for the Macintosh Family,* Second Edition (Addison-Wesley, 1990) for more information about how gamma correction works on the Macintosh II family for monitors.

w, about the code. To illustrate the principle of dithering, ased—that is, the calculations are done on a pixel basis. luggish. A real-world commercial application would use an code. One way to do this is to make the routines work on a el basis. Also note that the routine that does the halftoning aps of 8 or 32 bits. It would be easy to extend the routine to epths.

is one that calculates the luminance given a pointer to the /AL routine returns a long luminance in the range of 0 to 11% formula described previously.

```
, PixMapPtr pMap)

reen, blue;

ize == 32) {
unsigned char)*(++pPixel);        /* Skip alpha,
                                     get red. */
))(unsigned char)*(++pPixel);     /* Get green. */
```

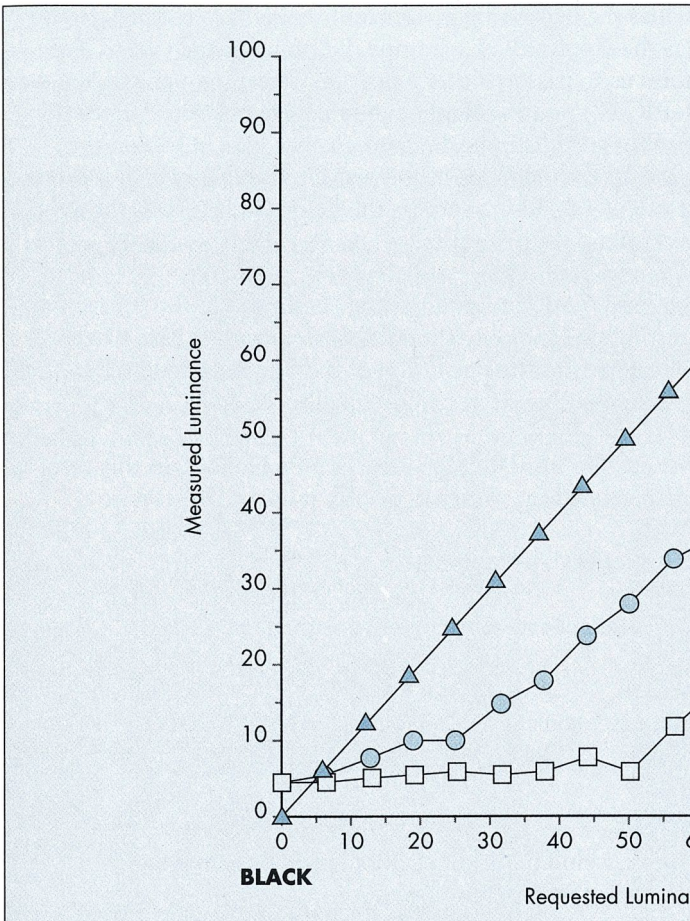
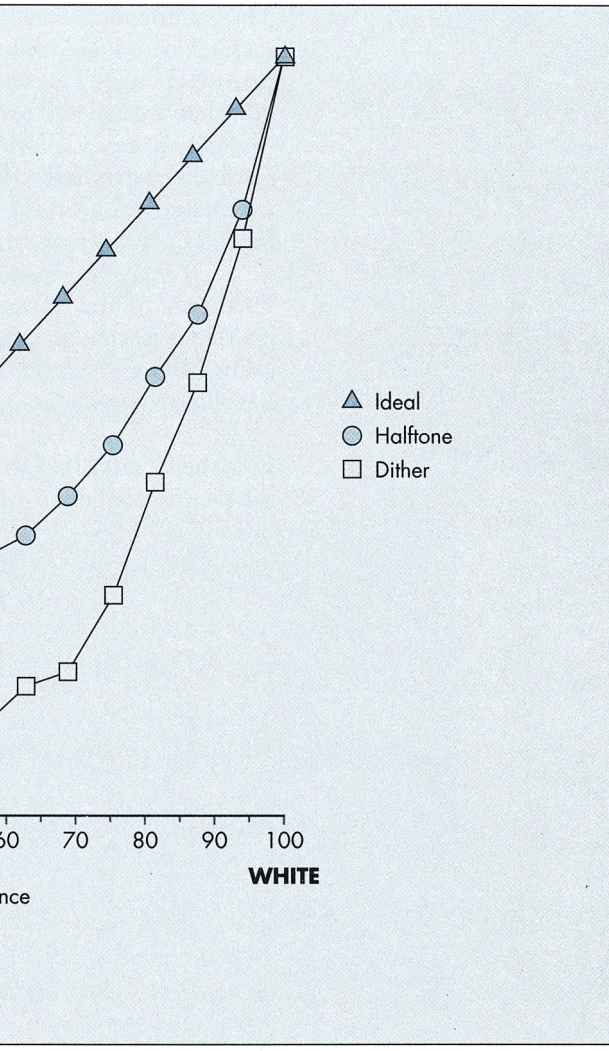


Figure 10
TRC Curves for the LaserWriter

```
        blue = (long)(unsigned char)*(++pPixel);
        return((30 * red + 59 * green + 11 * blue)/
    } else if (pMap->pixelSize == 8) {
        RGBColor*       theColor;
        theColor = &((*(pMap->pmTable))->ctTable[ (
            char)*pPixel ].rgb);
        return( (30 * (theColor->red >> 8) + 59 * (
            8) + 11 * (theColor->blue >> 8))/100);
    }  /* End if */
}  /* LUMVAL */
```




```
 /* Get blue. */
100);


unsigned

theColor->green >>
```

The routine that actually
than taking a PixMapPtr
PixMapHandle. This ena
did when we called Creat
example, from a GWorld
routine knows whether it
we created it) or if it mus
for the LockPixels and G

Furthermore, the Halfton
pixMap is 72 dpi (screen
(same hRes and vRes). Yo
Resolution parameter, bu

Like the CreatePICT2 ro
the picture contains a 1-b

The prototype for the Ha

```
PicHandle HalftonePixM
   short Resolution);
```

The source code for the
disc.

USING RANDOM DITHE

Random dithering is yet
discussed last, however, b

The method is simple. It'
earlier. The only differen
values are compared to a
that the probability of an
proportional to the lumin
point.

This method has three li
expensive operation that
except at very high resolu
bad reception on a black-
number generator that's

Ironically, this least frequ
physical process of photo
composed of pixels. How

does the halftoning is the HalftonePixMap routine. Rather as the CreatePICT2 routine did, this routine takes a bles us to pass in either a pixMap we create manually (as we ePICT2) or a PixMapHandle that QuickDraw creates (for). We must distinguish which one we pass in so that the can access the fields of the pixMap directly (which it can if t use QuickDraw to access the fields. This is relevant only etPixBaseAddr routines.

nePixMap routine assumes the resolution of the source resolution) and only supports devices with square pixels u can pass in the resolution of the destination device in the t it must be greater than or equal to 72 dpi.

outine, HalftonePixMap returns a PicHandle. In this case, it/pixel pixMap. You can display it using DrawPicture.

lftonePixMap routine is

```
ap(PixMapHandle hSource, Boolean qdPixMap,
```

complete routine can be found on the *Developer CD Series*

ERING

nother kind of dither useful for drawing images. It's ecause of its inherent limitations.

s much the same as the 50% threshold method described ce is that instead of being compared to 50%, the luminance random number between 0 and 100%. The effect of this is y dot in the device image being turned on is directly ance of the pixel in the source image at the corresponding

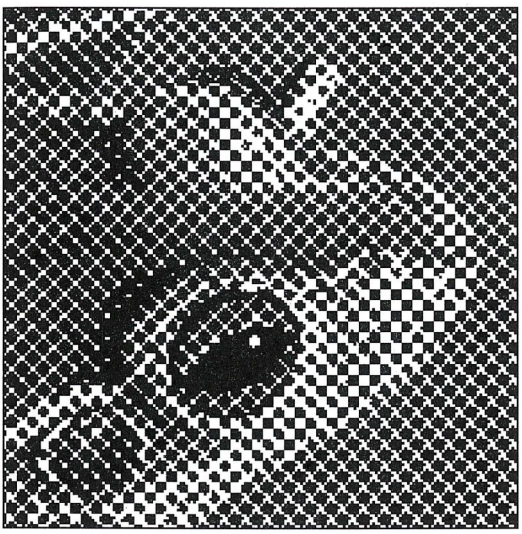
nitations. First, calculating a random number is an we would not want to do for every device pixel. Second, tions, images dithered in this manner appear very noisy, like and-white TV. And third, this method requires a random ery good at producing a uniform distribution.

ently used method of dithering most accurately models the graphy. Photographic film is like laser printing in that it's ever, the pixels are grains of silver rather than toner.

Additionally, there are tens of thousands of grains per inch rat
per inch we're used to with laser printers. The lower the ASA
higher the grain density.

The place on a film where a photon strikes one of these silver
the film is developed (which is why you get negatives). Since p
really small, the likelihood of a single photon striking one of t
very low. However, the brighter the light, the more photons t
probability of striking one of those silver grains increases in p
luminance. Thus, we see how random dithering simulates pho

Figure 11 shows the image of a frog's head produced using ha
matrix as compared with using a 72-dpi random dither. You ca
dithered image looks like a really grainy photograph.



Halftoned with 8x8 at 72 dpi

Figure 11
Frog's Head, Halftoned and Randomly Dithered

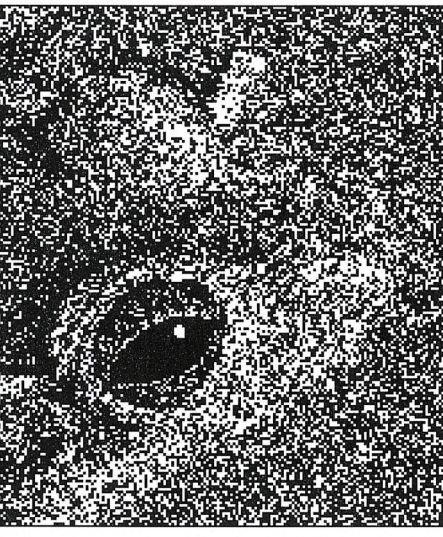
HASTA LA VISTA, BABY

This article has addressed several issues. First, the problem of
on machines with original QuickDraw was overcome by show
manually create a PICT, which can then be rendered by callin

her than the 300 dots
rating of the film, the

grains turns black when
photons are really, really,
he grains of silver is
here are; so the
roportion to the
otography.

lftoning with an 8x8
an see that the randomly



Randomly dithered at 72 dpi

saving deep pixMaps
ing you how to
g DrawPicture. Such a

PICT can be exported by
QuickDraw machine.

Second, several solutions
black-and-white devices v
50% threshold or error-d
way to get around the pro
Random dithering has pr
producing color images o

Thanks to these techniqu
need not be limited to Co
necessary code is small (a
very high. Now get to wo

WANT TO READ

If you'd like to delve mo
display, check out the fo

- "An Optimum Algori
by Thomas M. Hollac
Vol. 21, No. 2, 198

- *Digital Halftoning* by
Ph.D. thesis done at
extremely thorough c
ways.

- *Fundamentals of Inte*
(Addison-Wesley, 19
thorough as Ulichney

And then, of course, the

- *Programming with Q*
Othmer (Addison-We
the Macintosh.

- *Debugging Macintos*
Straus (Addison-Wes
software, including i

THANKS TO OUR TECHNI
Sean Parent, Forrest Tanaka, D

an application so that it can be viewed in color on a Color

to the problem of displaying and printing color images on
vere discussed. Images can be displayed on screen using a
iffusion dithering. Ordered dithering (halftoning) provides a
oblem of the laser printer's inability to resolve single pixels.
actical limitations but represents yet another alternative for
n black-and-white devices.

es, the market for applications that deal with color images
olor QuickDraw machines and PostScript printers. The
nd already written for you) and the gain in functionality is
rk on those applications!

MORE?

re deeply into the mysteries of processing color images for
llowing:

thm for Halftone Generation for Displays and Hard Copies"
day, in the *Proceedings of the Society for Information Display,*
0.

Robert Ulichney (MIT Press, 1987). This book, based on a
MIT, is devoted entirely to discussing halftoning algorithms; it's
nd includes many example images halftoned in different

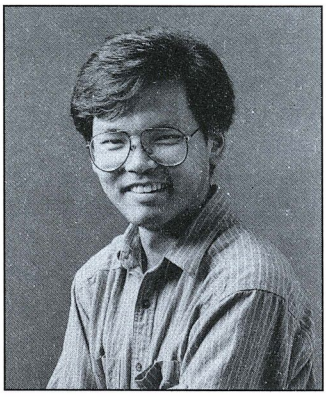
ractive Computer Graphics by J. D. Foley and A. Van Dam
82). The standard text on computer graphics. Not nearly as
, but has a solid discussion of the basics.

two books all Macintosh programmers should own:

uickDraw by Dave Surovell, Frederick Hall, and Konstantin
sley, 1992). Everything you need to know about graphics on

h Software with MacsBug by Konstantin Othmer and Jim
ley, 1991). Everything you need for debugging Macintosh
n-depth discussions of a number of the Macintosh managers.

CAL REVIEWERS
Dave Williams[•]



FORREST TANAKA

GRAPHICS HINTS FROM FORREST

USING THE PALETTE MANAGER OFF-SCREEN

Most people who've done any graphics programming on the Macintosh are aware of the Palette Manager, because it's the documented way to control the on-screen color environment, and perhaps because my cohorts and I in Developer Technical Support keep going on about how right the world would be if everyone used it. In an effort to follow the rules as best they can, some people have taken the Palette Manager so much to heart that they use it not only with windows, but with off-screen cGrafPorts as well—something that isn't heard about very much. Some of these people have concluded that all the features of the Palette Manager apply just as well to off-screen cGrafPorts as they do to windows. Logical enough, right?

Well, that's the kick; whether this is logical or not, the truth is that only a small part of the Palette Manager works with off-screen cGrafPorts. Specifically, the pmCourteous usage mode and the pmWhite and pmBlack usage-mode modifiers work fine when they're used in a palette that's attached to an off-screen cGrafPort, but the pmTolerant, pmAnimated, and pmExplicit usage modes do not. In this column, I'll describe how you can take advantage of the Palette Manager features that work off-screen and how you can simulate the features that don't work.

The pmCourteous usage mode seems pretty useless to a lot of people because it has no effect on the current

---

FORREST TANAKA has been playing Developer Technical Support as one of the graphics support people for slightly more than two years. "It amazes me still," he says, "that the more you learn about the Macintosh graphics tools, the farther off total understanding seems to be." Outside of DTS, he likes to ride his bike, and uses it to commute the three blocks to his office ("Hey, it's faster than driving the three blocks!"), and he likes to try getting his radio-controlled car to act as if it's actually controlled.

olor environment. But in general, making a palette full f pmCourteous colors is a lot better than hard-coding RGBColors into your code. Instead of hard-coding olors, make a palette of courteous colors—as many ntries as you need colors—and save it as a 'pltt' esource. When your application runs, call SetPalette o attach this palette to your off-screen cGrafPort. When you need to use a color while drawing into this GrafPort, pass the desired color's palette index to PmForeColor or PmBackColor, and then draw. This is etter than hard-coding colors because you or a oftware localizer can easily change the colors by hanging the 'pltt' resource—no code changes are ecessary.

The pmWhite and pmBlack usage-mode modifiers are ew with System 7; they let you specify whether you vant a particular palette entry to map to white or black n a black-and-white graphics environment. By default, olors whose average color-component value is larger han 32767 are mapped to white and other colors are napped to black. (If you use RGBForeColor, Color QuickDraw also checks to see whether your specified olor is different from your background color but maps o your background color; if so, Color QuickDraw uses he complement of the color you specified so that your drawing is visible over the background.) By specifying hat a palette entry is pmCourteous + pmBlack or pmCourteous + pmWhite, you can control which olors map to black and to white when there aren't enough colors available. This applies to palettes ttached to off-screen cGrafPorts as well as to palettes ttached to windows.

Those are the Palette Manager features that do work off-screen. Now I'll talk about the features that don't nd what you can do to get the same effect.

The pmExplicit usage mode is handy when you want to draw using a pixel value without knowing or caring what color that pixel value represents. With this mode you can easily show the colors in a screen's color table, nd you can also draw into a pixel image with a specific value even though you specify the color for that value lsewhere.

PRINT HINTS FROM LUKE & ZZ is in hibernation.•

When you have a palette that's attached to an off-screen cGrafPort, pmExplicit colors are interpreted as pmCourteous colors. Instead of using a palette, you should convert your pixel value to an RGBColor and use this as the foreground or background color. Set the current GDevice to your off-screen GDevice so that the color environment is set; then pass your pixel value to Index2Color, which is documented on page 141 of *Inside Macintosh* Volume V. Index2Color converts your pixel value to the corresponding RGBColor, which you can pass to RGBForeColor or RGBBackColor, and then you can draw. The result is that your pixel value is drawn into the destination pixel image.

Both the pmAnimated and pmTolerant usage modes are used to modify the color environment, and both are interpreted as pmCourteous when they're in a palette that's attached to an off-screen cGrafPort. The most important difference between the two usage modes is in the style of color-table arbitration that they do—pmTolerant gives the front window the colors it needs, while pmAnimated additionally makes sure that nothing outside the front window is drawn in its colors. Color-table arbitration doesn't apply off screen, so the pmAnimated and pmTolerant usage modes can be unified into "I want to change my off-screen colors."

Changing the colors in an off-screen color environment means changing its color table; the most straightforward way to do this is to modify the contents of the color table directly. That is, get your off-screen color table's handle and then directly assign new values to the rgb fields in its CSpecArray. You could also assign a whole new color table to the off-screen environment by assigning the new one to the pmTable field of the off-screen pixMap. Either way, you have to

For more details about changing or replacing off-screen color tables, see the October 1991 version of Macintosh Technical Note #120, "Principia Off-Screen Graphics Environments."•

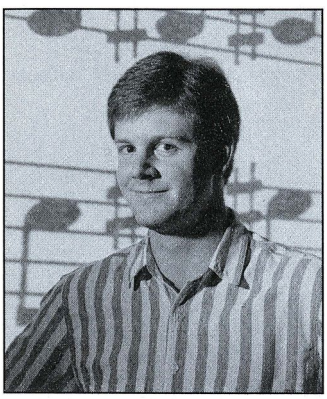
tell Color QuickDraw what you've done by updating the changed color table's ctSeed field. The next time you draw into your off-screen graphics environment, Color QuickDraw detects your change by comparing the ctSeed of your changed color table against the iTabSeed of the current GDevice's inverse table, and it rebuilds the inverse table according to the changed color table. You can update the ctSeed field by assigning to it the return value of GetCTSeed, which is documented on page 143 of *Inside Macintosh* Volume V. If the 32-Bit QuickDraw extensions are available, you can update a color table's ctSeed simply by passing the color table to CTabChanged, documented on page 17-26 of *Inside Macintosh* Volume VI.

If you have a GWorld and you want to replace its color table, you should call UpdateGWorld, passing it a new color table. UpdateGWorld makes sure that all the cached parts of a GWorld are properly updated, which is tough to do any other way. If you don't pass any flags to UpdateGWorld, it's within its rights to destroy your existing GWorld's image. But if you pass the clipPix or stretchPix flag, UpdateGWorld is obligated to keep your existing image, and it tries to reproduce the existing image in the new colors as best it can.

To wrap up, you can use the Palette Manager with off-screen graphics environments, but you'll only be able to use the pmCourteous usage mode and the pmWhite and pmBlack usage-mode modifiers. But that's not to cast aspersions on these features, because they can be very handy for both on-screen and off-screen drawing. The pmExplicit, pmTolerant, and pmAnimated usage modes don't work for off-screen drawing, but there are easy ways to simulate those features without the Palette Manager and without risking future compatibility.

THE TEXTBOX YOU'VE ALWAYS WANTED

NeoTextBox is an alterna
NeoTextBox provides full
TrueType features while
three routines that compr
bytes yet offer a 40% per
cases.



BRYAN K. ("BEAKER") RESSLER

In the deepest, darkest corner
there's an extremely useful rou

```
pascal void TextBox(void *
```

Given a rectangle and some te
drawing in the font, style, and

Anyone who's tried to word w
Perhaps that's why TextBox ta
creates a new TERec with TE
TESetText to create a tempor
TextBox. TextBox then calls T
TEDispose to dispose of the T
drawing, TextBox avoids doing
of overhead.

Despite its pass-the-buck impl
advantages. Perhaps most imp
systems like Japanese and Arab
Another handy side effect is th
DrawText, and can therefore b
designed specifically for drawi
function well.

BRYAN K. RESSLER, or "Beaker"
known at Apple, is one of our twiste
engineers who seems to be convince
anything is possible on a Macintosh,
already been done, it can be done b
his BSCS from the University of Calif
and wrote commercial MIDI applicat
coming to Apple. Beaker wrote man
programs used for testing TrueType f

tive to the TextEdit utility routine TextBox. -justification capability and the option to use retaining all the advantages of TextBox. The ise NeoTextBox compile to fewer than 900 formance increase over TextBox in common

of the TextEdit chapter in *Inside Macintosh* Volume I, tine called TextBox.

```
text, long length, Rect *box, short just)
```

xt, TextBox word wraps the text inside the rectangle, size specified in the current grafPort.

rap text knows that it's not as easy as it first appears. kes the approach it does: to perform its task, TextBox New, sets up the rectangles in the record, and calls ary handle to a copy of the text you provided to EUpdate to wrap and draw the text, and finally TERec. By calling TextEdit to do the text wrapping and g any hard work. Unfortunately, it also incurs quite a bit

lementation, TextBox's use of TextEdit has several ortant, TextBox works correctly with non-Roman script pic without the need for any extra programming. hat updates in TextEdit degenerate into calls to be recorded into QuickDraw pictures. TextBox was ng static text items in dialog boxes and performs this

as he's d software ed that and if it's better. He got ornia, Irvine, tions before y of the onts. When

he's not on a coding frenzy, he writes *noncommercial* MIDI applications, tries to have a life, and keeps a consistent blood-caffeine level so high you need scientific notation to express it.•

So TextBox is great—if y
better performance. You
want full justification (ins
use whizzy TrueType call
drawing mode. You can't
useful when you're drawi
don't like that 32K text li
single call to TextBox. An
height it used, and where
something below the text
TextBox.

Well, this is your lucky d

ENTER NEOTEXTB

NeoTextBox is the TextB
NeoTextBox is on the av
considerably more flexibl

- NeoTextBox allow
default (same beh
adjusts for charact
or descent line; or
- NeoTextBox prov
justification.
- NeoTextBox neve
erase or, if you wi
- NeoTextBox retur
- NeoTextBox can r
position of the las
draw the text.

NeoTextBox gives you al
TextBox. It is completely
(just like TextEdit). It's ea
it's easy to get TextBox-li

Let's take a look at the pa

```
short NeoTextBox(unsig
   Rect *wrapBox, sho
   short *lhUsed)
```

ou're drawing dialog boxes. But you want more. You want
want more flexibility. You want to control line height. You
stead of only left, center, and right alignment). You want to
s when they're available. You want to control the text
stand the way TextBox always erases (and therefore isn't too
ng to printers—it slows printing way down). Yeah, and you
mitation either. You want to word wrap *War and Peace* in a
nd you'd like some useful information back, too, like the line
the last line of text was drawn, so that you can draw
. And, of course, you want to retain the advantages of

ay.

BOX

ox you've always wanted (and didn't even have to ask for).
erage 33% faster than an equivalent call to TextBox. Plus, it's
le:

- vs a line height specification. You can ask for the
  avior as TextBox); use variable line height, which
  ters that extend beyond the font's standard ascent
  specify a line height in points.
- ides left, center, and right alignment and full

r erases the rectangle it's drawing into. It lets you
sh, draw a colored background.

rns the total number of lines in the wrapped text.

return, via VAR parameters, the vertical pen
t line of text and the line height that was used to

l this extra functionality, yet retains the advantages of
language independent and uses the Script Manager heavily
sy to call, and if you don't want all the spiffy new features,
ke behavior with a free performance increase.

rameters for NeoTextBox.

```
gned char *theText, unsigned long textLen,
rt align, short lhCode, short *endY,
```

The first two parameters, theText and textLen, are analogous
length parameters: they specify the text to be wrapped. Note t
Pascal string—it's a pointer to the first printable character.

The third and fourth parameters, wrapBox (box in TextBox) a
back to NeoTextBox's ancestor. Just as in TextBox, wrapBox sp
within which you're wrapping text, and the align parameter sp
In addition to the standard TextEdit alignments teFlushLeft, t
teFlushRight (see "Text Alignment Constants for System 7"),
defined—ntbJustFull. It performs full justification in whatever
for the current script.

The fifth parameter, lhCode, specifies how the line height is d
the default line height is derived via a call to GetFontInfo. Th
behavior as TextBox. If lhCode is less than 0, the line height is
determining which characters in the text that's being drawn ex
and below the baseline (see "SetPreserveGlyph With TrueTyp
lhCode is greater than 0, the value of lhCode itself specifies th
instance, you can draw 12-point text in 16-point lines.

The last two parameters, endY and lhUsed, are reference para
to retrieve the vertical position of the last line of text and the l
used to draw the text, respectively. The endY parameter can b

TEXT ALIGNMENT CONSTANTS FOR SYS

Before System 7, there was a conflict between the names of the
constants and their actual behavior. To help make applications
Roman scripts, teJustLeft was interpreted as the default text align
the current script rather than forcing text to be aligned on the le
example, on a Hebrew system, a TextBox call with a just param
actually use the default justification for Hebrew, which is teJustR

To overcome this conflict, new constants were introduced in Sys
Table 1.

Table 1
Text Alignment Constants

New Constant	Old Constant	Value	Meaning
teFlushLeft	teForceLeft	–2	Align text on t
teFlushRight	teJustRight	–1	Align text on t
teFlushDefault	teJustLeft	0	Use conventio
teCenter	teJustCenter	1	Center text for

to TextBox's text and
hat theText isn't a

nd align, also hearken
ecifies the rectangle
ecifies the alignment.
eCenter, and
a new alignment is
manner is appropriate

erived. If lhCode is 0,
is gives the same
derived by
tend the most above
e Fonts"). Finally, if
e line height. For

meters that allow you
ine height that was
e very useful if you

TEM 7

text alignment
compatible with non-
ment appropriate for
ft as specified. For
eter of teJustLeft would
ight.

tem 7, as shown in

he left for all scripts
he right for all scripts
nal alignment for script
all scripts

SETPRESERVEGLYPH WITH TRUETYPE

Before TrueType, all characters in all fonts fit beneath the font's ascent line and above the descent line, like the default characters shown in Figure 1. Bitmapped fonts were drawn so that diacriticals, like the angstrom over th *A* in Ångström, would fit beneath the ascent line. To do this, the letterform had to be distorted. With the advent o TrueType, this "feature" can be controlled, because TrueType fonts carry outline data that's true to the origina design (hence the name TrueType).

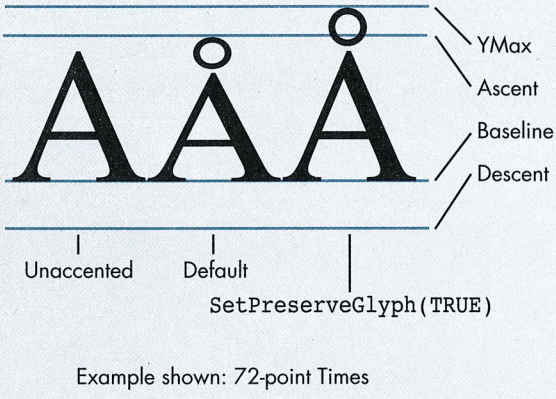


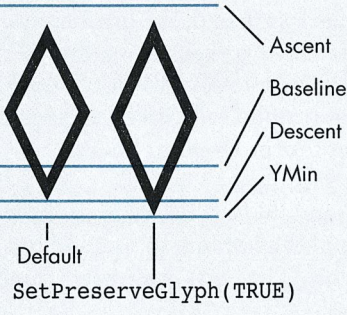
Figure 1
How SetPreserveGlyph Affects Line Height

intend to draw anything
text was drawn. To find
negative lhCode, use the
two parameters if you do

NeoTextBox returns the
off because they were be
overflowed wrapBox by
wrapBox.bottom. If you
the height of wrapBox by

E FONTS

Since most applications expect characters to fit beneath the ascent line and above the descent line, QuickDraw transforms characters in TrueType fonts to force them within those bounds. To override this transformation and preserve the original glyph shape, use the Font Manager call SetPreserveGlyph(TRUE). After this call, TrueType fonts will be drawn as shown to the right in Figure 1. Preserving the glyph shape makes it possible to take advantage of NeoTextBox's variable line height feature.



xample shown: 72-point Symbol

below the text, since it tells you exactly where the last line of out what the actual derived line height was if you used a lhUsed parameter. Pass nil for either or both of these last on't want this extra information.

total number of lines in the text. That includes lines clipped low the bottom of wrapBox. You can tell whether the text whether the value returned in endY is greater than want to know how many lines fit in wrapBox, simply divide y the value returned in lhUsed.

REQUIREMENTS

NeoTextBox uses some advanced Script Manager routines tha
System 6 or later. NeoTextBox assumes they're available, so m
program checks that it's running on System 6 or later via a Ge
call.

NeoTextBox requires one global variable, a Boolean named gH
be set to TRUE if the TrueType trap ($A854) is available, or F
development environment provides glue for Gestalt, you can u
to set up gHasTrueType:

```
#define kTrueTypeTrap  0x54   /* The TrueType trap numb
#define kUnimplTrap    0x9f   /* The "unimplemented" tr
long   gResponse;

if (Gestalt(gestaltFontMgrAttr,&gResponse) == noErr)
   gHasTrueType = BitTst(&gResponse,31-gestaltOutlineF
else {
   gHasTrueType = (NGetTrapAddress(kTrueTypeTrap,Tool
      NGetTrapAddress(kUnimplTrap,ToolTrap));
}
```

THE BASIC ALGORITHM

NeoTextBox does a lot. But, in order to appease the programm
avoid work, we allow the Script Manager to do the hard parts.
do full justification in Arabic?) In short, here's how NeoTextB

1. It saves the current grafPort's clipping region and clips
   we're drawing into.
2. It calculates the appropriate line height with the functi
   NTBLineHeight.
3. It calls the Script Manager routine StyledLineBreak to
   line-break point in the input text.
4. It draws each line with the function NTBDraw.
5. It advances the pen down one line.
6. When there's no more text, it restores the clipping regi
   returns the appropriate values.

It sounds simple, doesn't it? That's because StyledLineBreak d
knows how to find word breaks in whatever script we're using
smart, too. For instance, in English, it knows that it's OK to b
if necessary. It uses rules that are provided by the installed scri
takes the appropriate actions. Let's take a closer look at the co

t are available only in
ake sure your main
stalt or SysEnvirons

HasTrueType. It should
FALSE if not. If your
ıse the following lines

```
ber */
rap number */


Fonts);

Trap) !=
```

ner's natural desire to
(Do you know how to
ox gets its job done:

to the box

on

find each

ion and

loes all the work. It
. StyledLineBreak is
reak a hyphenated word
pt systems, so it always
de.

THE NEOTEXTBO

The source code for Neo
start in the NeoTextBox
we come upon them.

Here's the NeoTextBox c

```
short NeoTextBox(unsi
   Rect *wrapBox, sho
   short *lhUsed)
{
   RgnHandle
   StyledLineBreakCod
   Fixed
   Fixed
   short
   long
   unsigned short
   short
   unsigned short
   long
   unsigned char
   unsigned char
```

Many of these variables a
detail later. The most im
contains the line break c
lineBytes, which are retu
the current vertical pen l

GET READY

NeoTextBox, like TextBo
it's safest to save the clip
width of wrapBox, becau
which is used in calls to S
appropriate text alignme

```
GetClip((oldClip = Ne
ClipRect(wrapBox);
boxWidth = wrapBox->r
fixedMax = Long2Fix((
if (align == teFlushD
   align = GetSysJust
```

X FUNCTION

TextBox that's shown here is written in MPW C 3.2. We'll function and break out to a couple of utility functions when

declaration and local variables:

```
gned char *theText, unsigned long textLen,
rt align, short lhCode, short *endY,


 oldClip;          /* Saved clipping region */
e breakCode;       /* From StyledLineBreak */
 fixedMax;         /* boxWidth in fixed point */
 wrapWid;          /* Width to wrap within */
 boxWidth;         /* Width of box */
 lineBytes;        /* Number of bytes in one line */
 lineHeight;       /* Calculated line height */
 curY;             /* Current vert pen location */
 lineCount;        /* Number of lines we've drawn */
 textRemaining;    /* Number of bytes of text left */
 *lineStart;       /* Pointer to start of a line */
 *textEnd;         /* Pointer to end of input text */
```

re used in the call to StyledLineBreak, which is explained in portant variables to know about here are breakCode, which ode returned by each call to StyledLineBreak; lineStart and rned by StyledLineBreak to specify a single line; and curY, ocation.

x, clips to wrapBox. Since this is a general-purpose routine, ping region, then restore it at the end. We calculate the se it's used a lot, and convert it to fixed point as fixedMax, StyledLineBreak as a VAR parameter. Also, we retrieve the nt if the user has requested default alignment.

```
wRgn()));

ight - wrapBox->left;
long)boxWidth);
efault)
();
```

DETERMINE THE LINE HEIGHT

Now we need to determine the appropriate line height. NeoT
NTBLineHeight to perform this function, passing the text po
the wrap rectangle, the caller-specified line height code, and t
current vertical pen location. NTBLineHeight calculates and
and calculates the correct starting pen location. Here's the NT

```
unsigned short NTBLineHeight(unsigned char *theText,
   unsigned long textLen, Rect *wrapBox, short lhCode
{
   short              asc, desc;
   FontInfo           fInfo;
   Point              frac;
   unsigned short     lineHeight;

   GetFontInfo(&fInfo);
   if (lhCode < 0) {
      /* lhCode < 0 means "variable line height", so
      /* TrueType font use OutlineMetrics, otherwise
      frac.h = frac.v = 1;
      if (gHasTrueType && IsOutline(frac, frac)) {
         OutlineMetrics((short)textLen, theText, frac
            &desc, nil, nil, nil);
         lineHeight = MAXOF(fInfo.ascent, asc)
            + MAXOF(fInfo.descent, -desc) + fInfo.lea
         *startY = wrapBox->top + MAXOF(fInfo.ascent,
            + fInfo.leading;
      } else {
         /* Punt to "default" if we can't use TrueTyp
         lineHeight = fInfo.ascent + fInfo.descent +
         *startY = wrapBox->top + fInfo.ascent + fInf
      }
   } else if (lhCode == 0) {
      /* lhCode == 0 means "default line height." */
      lineHeight = fInfo.ascent + fInfo.descent + fIn
      *startY = wrapBox->top + fInfo.ascent + fInfo.l
   } else {
      /* lhCode > 0 means "use this line height" so w
      lineHeight = lhCode;
      *startY = wrapBox->top + lhCode + fInfo.leading
   }
   return(lineHeight);
}
```

extBox calls
inter, the text length,
ne address of curY, the
returns the line height
TBLineHeight function:

```
, short *startY)
```

```
if it's a */
use default. */
```

```
, frac, &asc,
```

```
ding;
 asc)
```

```
e. */
fInfo.leading;
o.leading;
```

```
fo.leading;
eading;
```

```
e trust 'em. */
```

```
;
```

Remember, there are thr

- Variable line heigl
  first. If the TrueT
  TrueType font, O
  height (see "Desc
  variety of informa
  and the lowest des
  asc and desc. Ther
  ascent or asc, and
  TrueType isn't ava
  font, we punt to th
- If lhCode is 0, the
  sum of the ascent,
  GetFontInfo call.
- Finally, if lhCode
  line height. In this
  line height.

Each of the three line hei
on the line height and wr

Back in NeoTextBox, we
lineHeight and curY:

```
lineHeight = NTBLineHe
lineCount = 0;
lineStart = theText;
textEnd = theText + te
textRemaining = textLe
```

DESCENT INTO

Descent is the amount o
baseline. When you ca
number of points below
industry it's more comm

In an attempt to be mor
its descent values as ne
note the sign of descen

ee possible line height codes:

nt (specified by an lhCode less than 0) is handled
ype trap is available and this particular font is a
utlineMetrics is called to determine the line
ent Into Hell"). OutlineMetrics can return a
tion, but we really only want the highest ascent
scent, which are returned in the local variables
n we choose whichever is higher, the default
whichever is lower, the default descent or desc. If
ilable or the particular font isn't a TrueType
ne default line height.

default line height is used. This is defined as the
descent, and line gap (leading) derived by a

is greater than 0, the caller is providing a specific
s case, NTBLineHeight returns lhCode as the

ght calculation methods also figures the correct startY based
apBox->top.

call NTBLineHeight to set up our local variables

```
eight(theText, textLen, wrapBox, lhCode, &curY);

extLen;
en;
```

HELL

of space that should be allocated for a font below the text
ll GetFontInfo, the value returned for descent is a positive
the baseline. Although this is convenient, in the typographic
on to represent descent values as *negative* numbers.

e typographically useful, TrueType's OutlineMetrics call returns
gative numbers. So, to avoid a descent into hell, remember to
t values when mixing calls to GetFontInfo and OutlineMetrics.

Here we also set up some other local variables. The variable li
number of lines we've drawn. The pointer lineStart points to t
current line, which initially is the beginning of the text. The va
pointer to just beyond the end of the input text and is used for
all used up. Finally, the variable textRemaining keeps track of h
input text remain to be processed.

THE BREAK-DRAW LOOP

Now NeoTextBox is ready to break lines and draw the text. Th
the following do-while loop:

```
do {
   lineBytes = 1;
   wrapWid = fixedMax;

   breakCode = StyledLineBreak(lineStart, textRemainir
      textRemaining, 0, &wrapWid, &lineBytes);

   NTBDraw(breakCode, lineStart, lineBytes, wrapBox, a
      boxWidth);

   curY += lineHeight;
   lineStart += lineBytes;
   textRemaining -= lineBytes;
   lineCount++;
} while (lineStart < textEnd);
```

If this looks simple, that's because it is. Anyone who's tried to v
knows that it's a difficult task. Making the algorithm compatib
systems complicates the matter even more. Fortunately, we ha
which in this case makes our lives a *lot* easier.

The workhorse: StyledLineBreak. First we set lineBytes to
StyledLineBreak that this is the first "script run" on this line. S
script run, we always reset lineBytes at the top of the loop. Als
be fixedMax (which was previously initialized to the fixed-poin
rectangle). WrapWid tells StyledLineBreak the width within v
and returns how much of the line is left (if any) after wrapping
reset it at the top of the loop each time).

Now we call StyledLineBreak. We provide a pointer to the be
this line, the number of bytes of text remaining, the wrap widt
of a variable where StyledLineBreak puts the number of bytes
StyledLineBreak does the hard work of finding word boundari
widths, and handling special cases, all in an internationally con

neCount records the
he beginning of the
ariable textEnd is a
testing when the text is
now many bytes of

is task is performed by

```
ng, 0,

align, curY,
```

write code to wrap text
le with different script
ve the Script Manager,

1, signaling to
Since we have only one
o, we reset wrapWid to
t width of the wrap
which to wrap the text
(that's why we have to

ginning of the text for
h, and the address
in this line.
es, adding up character
npatible way.

After StyledLineBreak re
beginning at lineStart, a
circumstances of the line

The head and in fr
English writer that
point is therefore c
letters in a time wh
the problem to an
The Shannon Text
wayultramegasupe
docious sentence.

Figure 2
Line Break Codes

Usually, StyledLineBrea
word boundary. The bre
was too long to fit on a s
StyledLineBreak returns
given width. These line

Draw the text with NT
line, NeoTextBox calls N
the line of text, the lengt
current vertical pen loca
NTBDraw:

```
#define kReturnChar

void NTBDraw(StyledLi
   long lineBytes, Re
   short boxWidth)
{
   unsigned long   bl
   short           sl


   blackLen = Visible
```

eturns, lineBytes tells us the length of the current line
nd breakCode has a line break code that tells us the
break, as shown in Figure 2.

ontal attack on an the character of this nother method for the en whom ever told unexpected.↵ is a strange, ercalafragilisticexpiala

Break Code
smBreakWord
smBreakWord
smBreakWord
smBreakWord
smBreakWord
smBreakWord
smBreakChar
smBreakOverflow

k returns smBreakWord, indicating that it broke the line on a
ak code smBreakChar says that it encountered a word that
ingle line and was forced to break in the middle of a word.
smBreakOverflow if you run out of text before filling the
break codes help determine how to draw the text.

TBDraw. After StyledLineBreak figures the length of the
NTBDraw to draw the line. NeoTextBox passes a pointer to
h of the line in bytes, the wrap rectangle, the alignment, the
tion, and the width of the wrap rectangle. Let's take a look at

```
 0x0d

neBreakCode breakCode, unsigned char *lineStart,
ct *wrapBox, short align, short curY,


ackLen;    /* Length of non-white characters */
op;        /* Number of pixels of slop for */
           /* full justification */

Length(lineStart, lineBytes);
```

```
    if (align == ntbJustFull) {
        slop = boxWidth - TextWidth(lineStart, 0, blacl
        MoveTo(wrapBox->left, curY);
        if (breakCode == smBreakOverflow ||
                *(lineStart + (lineBytes - 1)) == kRetur
            align = GetSysJust();
        else DrawJust(lineStart, blackLen, slop);
    }
    switch(align) {
        case teFlushLeft:
        case teFlushDefault:
            MoveTo(wrapBox->left, curY);
            break;
        case teFlushRight:
            MoveTo(wrapBox->right - TextWidth(lineStart
                blackLen), curY);
            break;
        case teCenter:
            MoveTo(wrapBox->left + (boxWidth - TextWidt
                blackLen)) / 2, curY);
            break;
    }
    if (align != ntbJustFull)
        DrawText(lineStart, 0, lineBytes);
}
```

NTBDraw's job is to move the pen and draw the text as indic
parameter, align, and the line break code, breakCode. NTBD
visible length of the line with a call to the Script Manager rou
This excludes white-space characters at the end of the line. W
space characters? Well, that depends on the script. VisibleLe
characters are visible and which are not for the current script
appropriate length in bytes, which is stored in the local variab

When align is ntbJustFull, we need to determine whether the
carriage return character ($0D) at the end, because a line witl
example, the last line in a paragraph) should always be drawn
alignment, rather than fully justified.

Looking back at the break codes for different types of lines sh
that the line that ends with the carriage return (denoted grap
illustration) returns a line break code of smBreakWord, wher
return smBreakOverflow. As you can see, StyledLineBreak ex
when a line is the last line of a paragraph. Therefore, every li
smBreakWord must be checked for a carriage return.

```
kLen);

nChar)

, 0,

h(lineStart, 0,
```

ated by the alignment
raw first calculates the
utine VisibleLength.
What exactly are white-
ngth knows which
, and returns an
ble blackLen.

e current line has a
n a carriage return (for
with the default system

nown in Figure 2, notice
hically in the
e you might expect it to
xpects the caller to know
ne whose break code is

NTBDraw looks at the las
Since the carriage return
guaranteed never to occu
having to test whether the
proceed directly to the las

We now know whether th
calculate the amount of w
Manager routine DrawJus
this script. (In Arabic, for
differently than for Roma
override the align parame

For the left, right, and cen
appropriately, and a Draw
blackLen) helps correctly
and full justification.

Update the variables. A
variables and loop around

```
    curY += lineHeight;
    lineStart += lineBy
    textRemaining -= li
    lineCount++;
} while (lineStart < te
```

First, we add lineHeight t
pointer to the beginning
current line. TextRemaini
current line, and lineCou
of the text, the whole brea

RETURN SOME VALUES
Now that NeoTextBox ha
some useful values to the

```
    if (endY)
        *endY = curY - l
    if (lhUsed)
        *lhUsed = lineHe
```

NeoTextBox returns these
get TextBox-like behavior
don't want a return value,

st byte in the line it's drawing to see if it's a carriage return. character ($0D) falls into the control-code range, it's as the low byte of a two-byte character. This frees us from last character in the line is two-byte and allows us to t byte.

e current line has a carriage return or not. If not, we hite-space slop remaining in the line, then call the Script st to draw the text fully justified—whatever that means for instance, full justification is performed completely n text.) If the current line *does* end in a carriage return, we ter with the default system alignment and fall through.

nter alignments, the switch statement moves the pen Text call is made to draw the text. The visible length (in calculate the pen position for right and center alignment

After NTBDraw returns, we need to update a bunch of local again.

```
tes;
neBytes;

extEnd);
```

o curY, setting us up for the next line. LineStart, the of a line, gets updated to the character after the end of the ng gets reduced by the number of bytes consumed by the nt gets incremented. If lineStart still hasn't run off the end k-draw process is repeated.

s done such a fine job wrapping the text, it's time to return caller.

```
ineHeight;

ight;
```

e values only if the caller wants them. This makes it easy to from NeoTextBox without having to do any work: if you just pass nil instead of providing the address of a variable.

CLEAN UP AND WE'RE DONE

The only thing left to do is a little cleanup, and we're outa he

```
    SetClip(oldClip);
    DisposeRgn(oldClip);

    return(lineCount);
}
```

We restore the clipping region, dispose of our saved region, a

CALLS TO NEOTEXTBOX

One of the best features of NeoTextBox is that you can easily
you're currently making to TextBox. If that's all you want to d
occurrence that looks like this

```
TextBox(textPtr, textLen, &wrapBox, justify);
```

with this

```
{
    EraseRect(&wrapBox);
    NeoTextBox(textPtr, textLen, &wrapBox, justify, 0,
}
```

To use NeoTextBox in place of TextBox, you pass 0 for lhCod
and nil for endY and lhUsed, and ignore the return value. If y
your program and just do the substitution above, every NeoT
the average 33% faster than the old TextBox call. If you use T
mean a real performance increase.

You can use NeoTextBox in more ways than just as direct subs
performance. It does, after all, have whizzy new features that
take a look at a more sophisticated call to NeoTextBox that us
features:

```
short UseNTB(void)
{
    Rect        wrapBox;
    RGBColor    ltBlue;
    Handle      textHdl;
    long        textLen;
    short       numLines = 0;
    short       endY, lineHt;
```

re.

nd return lineCount.

substitute it for calls
o, replace every

```
 nil, nil);
```

e (default line height)
ou add NeoTextBox to
extBox call will be on
extBox a lot, that can

stitution to improve
TextBox never had. Let's
es some of its unique

```
    /* Set up our RGBCo
    SetRect(&wrapBox, 1
    ltBlue.red = 39321;
    ltBlue.green = 5242
    ltBlue.blue = 65535

    /* Paint the backgr
    PenNormal();
    RGBForeColor(&ltBlu
    PaintRect(&wrapBox)
    ForeColor(blackColo
    TextFont(helvetica)
    TextFace(0); TextMo

    /* Retrieve some te
    textHdl = GetResour
    if (textHdl) {
        textLen = GetHar
        /* Be sure to lo
        HLock(textHdl);

        /* Wrap text and
        numLines = NeoTe
            18, &endY, &
        HUnlock(textHdl

        /* Beep if text
        if (endY > wrapB
            SysBeep(1);

        /* Prove we know
        MoveTo(wrapBox.l
        Line(20, 0);
    }
    return(numLines);
}
```

This sample function dra
from a TEXT resource i
The text is fully justified
overflows the box, a beep
subsequent text might be

Here's an example using

```
olor and wrapBox. */
10, 10, 110, 110);

28;
5;

round, then set up the port text parameters. */

ue);
;
or);
; TextSize(12);
ode(srcOr);

ext for us to draw. */
rce('TEXT', 128);

ndleSize(textHdl);
ock the handle. NeoTextBox can move memory! */


d set numLines, endY, and lineHt. */
extBox(*textHdl, textLen, &wrapBox, ntbJustFull,
lineHt);
);

 overflows wrapBox. */
Box.bottom)


w where the text ended by drawing a line. */
left, endY + lineHt);
```

ws a 100-by-100-pixel box in light blue, then wraps text
nto the rectangle, ORing the text over the blue background.
12-point Helvetica®, with 18-point line spacing. If the text
sounds. A small line is drawn at the baseline where
drawn.

NeoTextBox with variable line height and TrueType fonts:

```
void UseVariableLineHeight(Rect *wrapBox, short align
{
    Boolean   oldPreferred, oldPreserve;
    Handle    textHdl;
    long      textLen;

    if (gHasTrueType) {
        oldPreferred = GetOutlinePreferred();
        oldPreserve = GetPreserveGlyph();
        SetOutlinePreferred(TRUE);
        SetPreserveGlyph(TRUE);
    }

    textHdl = GetResource('TEXT', 128);
    textLen = GetHandleSize(textHdl);
    HLock(textHdl);
    NeoTextBox(*textHdl, textLen, wrapBox, align, -1,
    HUnlock(textHdl);

    if (gHasTrueType) {
        SetOutlinePreferred(oldPreferred);
        SetPreserveGlyph(oldPreserve);
    }
}
```

Notice that we save the current settings of the Font Manager' PreserveGlyph flags. This allows us to be transparent to the c OutlinePreferred to TRUE, we are ensured of using TrueTyp bitmapped fonts are available. By setting PreserveGlyph to T accurate glyph shapes and measurements (see "SetPreserveGl Fonts" on page 34). Calling NeoTextBox with -1 as its lhCod use variable line height, which results in the difference shown

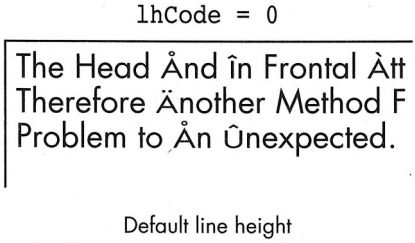


Figure 3
Using Variable Line Height

)

```
nil, nil);
```

s OutlinePreferred and
aller. By setting
e fonts, even if
RUE, we get the
yph With TrueType
e parameter causes it to
in Figure 3.

```
hCode = -1
```

Å Ånd Î̂n Frontal Àtt
Ä̈nother Method F
o Ån Ûnexpected.

riable line height

LIMITATIONS AN

NeoTextBox is a nice alte
could benefit from impro
limitations and adding us

32K TEXT SIZE LIMIT

All you *War and Peace* fan
32K text limitation that
only 32K of text in one c
arises from the OutlineM
can only handle 32K of t
implements variable line
NeoTextBox call (knock

DON'T FORGET TO ERA

Perhaps this isn't really a
NeoTextBox call. You ne
as shown earlier in the se

SCREEN-ONLY OPTIMIZ

If you know you'll be usi
won't be using it to draw
you don't care about the
wrap rectangle intersects
return.

If you don't need the retu
break-draw loop termina

SPECIAL ONE-LINE CAS

In Macintosh computers
worthwhile addition to N
boxWidth, simply use Dr
wrapping code. TextBox
items, which are often or

DON'T DRAW OFF THE

It might make NeoTextB
wrapBox->bottom + linel
total number of lines), bu
clipped.

D POSSIBLE ENHANCEMENTS

rnative to TextBox, but it has its limitations and areas that vement. Following are some suggestions for overcoming the eful features.

s out there need to do a little work. NeoTextBox shares the TextBox has, but not for the same reason. TextBox can wrap all because it uses TextEdit. In NeoTextBox, the limitation letrics call, which is used in deriving variable line height and ext. Heavy-duty Tolstoy types could remove the code that height and subsequently word wrap most novels in a single yourselves out).

ASE

limitation, but you can't simply replace a TextBox call with a ed to call EraseRect explicitly if you want TextBox behavior, ction "Calls to NeoTextBox."

ZATIONS

ng NeoTextBox only for screen applications (that is, you into a printer port), you can make a few optimizations. If return values, you can use RectInRgn to check whether the with the current port's visRgn; if it doesn't, you can simply

urn value giving the number of total lines, you can make the te when curY exceeds wrapBox->bottom + lineHeight.

SE

with 256K ROMs, TextBox has a feature that might be a NeoTextBox. If the TextWidth of the input text is less than awText to draw the text and don't bother with any of the has this feature because it's used for dialog box statText ne line.

END OF WRAPBOX

ox faster if NTBDraw isn't called when curY is greater than Height. You'd still have to wrap all the text (to determine the ut you wouldn't be drawing text that you know will be

MAKE SAVING/RESTORING THE CLIPPING REGION OPTI

It might be useful to be able to set up some complex clipping
NeoTextBox wrap as usual but clip its text to whatever the cli
invocation. You could add a Boolean swapClip parameter to c

STYLED NEOTEXTBOX

With considerable effort, NeoTextBox could be extended to h
multiscript text. Since StyledLineBreak, the workhorse of Ne
be used with styled text, such an enhancement is possible.

CONCLUSION

Once you start using NeoTextBox, you'll find it ending up in
tests on a Macintosh IIfx running System 7, NeoTextBox was
faster than TextBox, 33% faster on the average. Performance
font, screen depth, and the ratio of wrapping to drawing. For
on an 8-bit screen, NeoTextBox is 40% faster than TextBox.
reason to use it. Plus, it has features you can't get out of TextE

Perhaps the moral of this article is if you don't like some featu
OS go ahead and write your own. But you'll be doing yoursel
a lot more compatible in the future—if you can find lower-lev
OS facilities to aid you in your task, rather than recoding the

So go ahead and whip NeoTextBox into your application. Enj
performance and new features. And if there's something you c
there and change it. Make NeoTextBox the TextBox *you've* alv

THANKS TO OUR TECHNICAL REVIEWERS

Sue Bartalo, John Harvey, Joe Ternasky•

ONAL

region and have
pping region is set to at
ontrol this.

handle styled and
oTextBox, is designed to

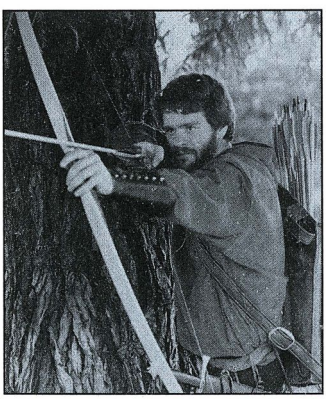
all your applications. In
between 25% and 50%
varies depending on
left-aligned Geneva text
That alone is a good
Box at all.

ure of the Toolbox or
f a favor—and you'll be
vel system, Toolbox, or
entire feature yourself.

oy the improved
don't like, go right in
ways wanted!

MAKING YOUR MACINTOSH SOUND LIKE AN ECHO BOX

Happy notes for soun
on the Developer CD
and record sounds at
record into one buffe
between the buffers.
out elements of this c



RICH COLLYER

We all know that the Ma
clever programming you
2BufRecordToBufCmd i
application (sans interfac
time that you're playing
purpose is to educate yo
definitive road to becomi

In addition to the main r
routines and a completio
code out of this article.

CONSTANT COM

Before I get into the sam
in the application.

GETTING A HANDLE O

The kMilliSecondsOfSo
sound the application sh
number of milliseconds,
used to calculate the size
sound effect you're after,
400,000 or so. If you set
time for the completion
high end of the range, o

RICH COLLYER is just your r
year Developer Technical Sup
often heard screaming at his
soothing accompaniment of B
KOME radio, he's honed his
fine point dodging (and castin
arrows at Apple, and he actu
degree from Cal Poly with a s
computational fluid dynamics.

d buffs: As you'll see from the sample code provided
Series disc, you can make your Macintosh play
the same time, simply by using double buffering to
while playing a second buffer, and then flipping
If you want to take things a few steps further, pull
ode and tailor them to suit your own acoustic needs.

cintosh is a sound machine, so to speak, but with a little can turn it into an echo box as well. The sample ncluded on the *Developer CD Series* disc is just a small e) that demonstrates one way to record sounds at the same them. There are other ways to achieve the same goal, but my about the Sound Manager, not to lead you down the ng your own recording studio.

outine, 2BufRecordToBufCmd includes various setup n routine. For easy reading, I've left out any unnecessary

MENTS

ple code itself, here are a few of the constants you'll run into

N IT

und constant is used to declare how many milliseconds of ould record before it starts to play back. The smaller the the more quickly the sound is played back. This constant is of the 'snd ' buffer handles (just the data). Depending on the kMilliSecondsOfSound can range from 50 milliseconds to it below 50, you risk problems: there may not be enough routine to finish executing before it's called again. On the ly the application's available memory limits the size. The

---

un-of-the-mill three-
port veteran: He's
computer to the
lazy and Bob on
archery skills to a
g) the slings and
ally admits to a
pecialty in
We let you in on

his outdoor adventures last time he wrote for us and he claims most of his indoor adventures aren't appropriate *develop* material, but we have it on good authority that he lives with carnivorous animals, if that's any clue. He's also a confirmed laserdisc and CD addict; he keeps promising to start a recovery program for those of us with the same affliction just as soon as he finishes writing that next sample . . .•

smaller the value, of course, the faster the buffers fill up and pl
an echo effect you'll get. A millisecond value of 1000 provides
between record and echo, which I've found is good for general
experiment to find the effect you like. (Beware of feedback, bo
and from anyone who's in close enough proximity to "enjoy" t
secondhand.)

YOUR HEAD SIZE, AND OTHER #DEFINES

The next three constants (kBaseHeaderSize, kSynthSize, and k
parse the sound header buffers in the routine FindHeaderSize.
the number of bytes at the top of all 'snd ' headers that aren't r
application itself. While the number of bytes isn't really of inte
parse the header in order to find the part of the sound header
bufferCmd. How much you parse off the top is determined by
header and the type of file; for the purposes of this code, howe
concerned with are the 'snd ' resources. The second constant,
of one 'snth'. In the calculations of the header, I find out how
and multiply that number by kSynthSize. The last constant, k(
one command, which is used in the same way as kSynthSize. (
derived from *Inside Macintosh* Volume VI, page 22-20.)

ERROR CHECKING WITH EXITWITHMESSAGE

2BufRecordToBufCmd includes error checking, but only as a
commercialization of the product. If the present code detects a
ExitWithMessage routine, which displays a dialog box that tel
where the error occurred and what the error was. Closing this
application, at which point you have to start over again. Note
ExitWithMessage at interrupt time could be fatal, since it uses
move memory. For errors that could occur at interrupt time, I
instead.

USING THE SOUND INPUT DRIVER

Use of the sound input driver is fairly well documented in *Insi*
VI, Chapter 22 (pages 22-58 through 22-68 and 22-92 throug
little overview of what 2BufRecordToBufCmd does at this poi
why. When you use sound input calls at the low level (not usin
SndRecordToFile), you need to open the sound input driver.
just opens the driver, which the user selects via the sound cdev

```
gError = SPBOpenDevice (kDefaultDriver, siWritePermiss
```

To open the driver, you call SPBOpenDevice and pass in a cou
parameters. The first parameter is a driver name. It doesn't re
name of the driver is; it simply needs to be the user-selected d

ay back, and the faster
a one-second delay
use. You'll want to
th from your machine
he experimentation

xCmdSize) are used to
. kBaseHeaderSize is
needed in the
erest here, you need to
that you'll pass to the
the format of the
ver, all you need to be
kSynthSize, is the size
many 'snth's there are,
CmdSize, is the size of
These equations are

placeholder for future
an error, it calls the
ls you more or less
dialog box quits the
that calling
routines that might
DebugStr is used

de Macintosh Volume
h 22-99), but here's a
nt in the routine, and
g SndRecord or
This section of the code
.

```
sion, &gSoundRefNum);
```

uple of simple
ally matter what the
river, so the code passes

GESTALT YOUR

You do need to check tw
First of all, your machin
trying to record sounds
sound input capabilities
Gestalt Manager will gi

Second, your hardware
for sound input and one
the returned feature vari

The following code sho
variable. (I didn't use th

```
err = Gestalt (gest
if (!err) {
    if (feature & (1
        //This Macint
    if (feature & (1
        //This Macint
    if (feature & (1
        //This Macint
    if (feature & (1
        //This Macint
    if (feature & (1
        //This Macint
    }
```

in nil (which is what kDe
tells the driver you'd like
enable the application to
gSoundRefNum. This pa
specific questions about t
sure that nothing went w
ExitWithMessage, and th

```
gError = SPBSetDeviceI
```

Continuous recording is
Quadra 700 and 900 that
input levels to their norm

MACHINE

vo rather critical sound attributes for 2BufRecordToBufCmd.
e must have a sound input driver. There's very little point in
if the sample is being run on a machine that doesn't have
. Checking bit 5 of the returned feature variable with the
ve you this handy bit of information.

needs to support stereo sound, since you need one channel
for sound output. Check for this attribute by checking bit 0 of
able.

ws how you can test all of the bits returned in the feature
is code in my sample.)

```
altSoundAttr, &feature);

 << gestaltStereoCapability))
osh Supports Stereo (test bit 0)
 << gestaltStereoMixing))
osh Supports Stereo Mixing (test bit 1)
 << gestaltSoundIOMgrPresent))
osh Has the New Sound Manager (test bit 3)
 << gestaltBuiltInSoundInput))
osh Has Built-in Sound Input (test bit 4)
 << gestaltHasSoundInputDevice))
osh Supports Sound Input (test bit 5)
```

faultDriver translates into). The constant siWritePermission
read/write permission to the sound input driver. This will
actually use the recording calls. The last parameter is the
rameter is needed later in the sample so that you can ask
he driver that's open. The error checking is just to make
rong; if something did go wrong, the code goes to
en the sample quits.

```
Info (gSoundRefNum, siContinuous, (Ptr) &contOnOff);
```

activated here to avoid a "feature" of the new Macintosh
gives you a slowly increasing ramp of the sound
al levels each time you call SPBRecord. The result in

2BufRecordToBufCmd is a pause and gradual increase in the s
buffers as the buffers are being played. Continuous recording
only on the first buffer, where it's almost unnoticeable.

BUILDING 'SND ' BUFFERS

Now that the sound input driver is open, the code can get the
build the 'snd ' buffers. As its name implies, 2BufRecordToBu
The reason is sound (no pun intended): The code basically use
method to record and play the buffers. The code doesn't tell t
play the sound until the recording completion routine has bee
have to worry about playing a buffer before it has been filled w
code also does not restart the recording until the previous buf

INFORMATION, PLEASE

To build the sound headers, you need to get some information
driver about how the sound data will be recorded and stored.
the GetSoundDeviceInfo routine, which looks for information
(the number of samples per second at which the sound is reco
(the sample size of the sound being recorded—8 bits per samp
CompressionType (see "Putting on the Squeeze"), the Numbe
of sound input channels, normally 1), and the DeviceBufferInf
internal buffers).

This code (minus the error checking) extracts these values fro
driver.

```
gError = SPBGetDeviceInfo (gSoundRefNum, siSampleRate,
   (Ptr) &gSampleRate);

gError = SPBGetDeviceInfo (gSoundRefNum, siSampleSize,
   (Ptr) &gSampleSize);

gError = SPBGetDeviceInfo (gSoundRefNum, siCompressio
   (Ptr) &gCompression);

gError = SPBGetDeviceInfo (gSoundRefNum, siNumberChan
   (Ptr) &gNumberOfChannels);

gError = SPBGetDeviceInfo (gSoundRefNum, siDeviceBuffe
   (Ptr) &gInternalBuffer);

value = kMilliSecondsOfSound;
gError = SPBMillisecondsToBytes (gSoundRefNum, &value
gSampleAreaSize = (value / gInternalBuffer) * gInterna
```

sound volume between
gives you this ramp

information it needs to
fCmd uses two buffers.
es a double-buffer
ne machine to start to
n called, so you don't
vith recorded data. The
fer has started to play.

from the sound input
That's the function of
n about the SampleRate
rded), the SampleSize
le is normal), the
erChannels (the number
fo (the size of the

m the sound input

```
,

,

nType,

nels,

erInfo,


);
alBuffer;
```

PUTTING ON THE SQUEEZE

If you want to use compression for 2BufRecordToBufCmd, keep in mind that the Sound Manager basically supports three types of sound compression: none at all, which is what I'm using, and MAC3 and MAC6, which are Mace compression types for 3:1 and 6:1 compression, respectively.

If you set the compression, the sound data is compressed after the interrupt routine is called (if you have one) and

Opening the sound input siSampleRate, siSampleS siDeviceBufferInfo are c the SPBGetDeviceInfo c pointer to a global variab the requested informatio

The last bit of work the c headers is to convert the buffer. To do this, the ro down the resulting value to bypass a bug connecte sound input device, whic buffer is not a multiple o size not only avoids this p record data into your buf

Now the code has the inf space, I've made a short r has to do is call this routi appropriate data.

IT'S A SETUP

The first line of code in t Memory Manager to allo data buffer and an estima checking (see the code its header. Setting up the so call, SetupSndHeader, to SetupSndHeader only on

before the Sound Manager internal buffers are moved to the application's sound buffers.

You have a couple of options for playing back a compressed sound. Either the bufferCmd or SndPlay will decompress the sounds on the fly. If you need to decompress a sound yourself, you'll want to call the Sound Manager routine Exp1to3 or Exp1to6 (depending on the compression you were using).

driver gives you the gSoundRefNum. The values ize, siCompressionType, siNumberChannels, and onstants defined in the SoundInput.h file; these constants tell all what information you want. The last parameter is a le. The SPBGetDeviceInfo call uses this parameter to return n.

ode needs to do before it's ready to start building the 'snd ' constant kMilliSecondsOfSound to the sample size of the utine needs to call SPBMillisecondsToBytes and then round to a multiple of the size of the internal sound buffer. This is d with the continuous recording feature of Apple's built-in n will collect garbage rather than audio data if the recording f the device's internal buffer. Creating a buffer of the right problem, but also enables the input device to more efficiently fer.

formation it needs to build the sound buffers. To save code outine that builds the buffers and their headers. All the code ne for each of the buffers it needs and pass in the

he SetupSounds routine is fairly obvious. It simply calls the cate the requested handles, based on the known size of the ted maximum size for the header, and does some error elf). Then, if the handle is good, the routine builds the 'snd ' und buffer requires building the header by making a simple the Sound Manager. There's a small problem with calling ce, however: When you call it, you don't know how big the

sound header is, so you just give the call the buffer, along with
size. When the call returns with the header built, one of the va
one that's the number of bytes in the sample—will be wrong. (
correct, but the data in the header will not be.) To correct this
your recording is complete and then put the correct number o
header, at which time you'll know how much data there is to p
misinformation in the header won't affect your recording, only

Once the header's built, the code resets the size of the handle,
(to avoid fragmentation of the heap), and locks it down. It's im
the handles in this way; otherwise the Sound Manager will mo
working with out from under itself.

```
*bufferHandle = NewHandle (gSampleAreaSize + kEstimate

gError = SetupSndHeader (*bufferHandle, gNumberOfChan
   gSampleSize, gCompression, kMiddleC, 0, headerSize

SetHandleSize (*bufferHandle, (Size) *headerSize + gS
MoveHHi (*bufferHandle);
HLock (*bufferHandle);
```

TELLING IT WHERE TO GO

The next part of the program allocates and initializes a sound
gRecordStruct. This structure tells the sound input call how t
wants it to do.

The first instruction is obvious: it simply creates a new pointe
structure can be stored.

```
gRecordStruct = (SPBPtr) NewPtr (sizeof (SPB));
```

The recording call will need to know where it can find the op
so next it needs the reference number to the driver (gSoundR
subsequent three lines of code inform the recording call how
to record into. Here, you could either give the call a count val
milliseconds are available for recording, or give it the size of t
this code, it's easiest to just make the bufferLength the same a
the milliseconds value. The code then tells the recording call
data as it's recorded.

```
gRecordStruct->inRefNum = gSoundRefNum;
gRecordStruct->count = gSampleAreaSize;
gRecordStruct->milliseconds = 0;
```

a 0 value for the buffer
lues in the header—the
The header size will be
, you simply wait until
f bytes directly into the
lay back. The
the playback.

moves the handle high
portant to lock down
ve the sound buffers it's

```
edHeaderSize);

nels, gSampleRate,
);

ampleAreaSize);
```

input parameter block,
o do what the code

r into which the

en sound input driver,
efNum). The
much buffer space it has
ue, tell it how many
he sound buffer. For
s the count and ignore
where to put the sound

```
gRecordStruct->bufferL
gRecordStruct->bufferP
gRecordStruct->complet
gRecordStruct->interru
gRecordStruct->userLon
gRecordStruct->error =
gRecordStruct->unused1
```

The recording call also n
the call is done asynchron
this routine later on.) You
driver periodically to see
the routine SPBGetReco
recording was finished, y
been filled. For this code,
recording is done because
likely you are to prevent

The userLong field is a g
you'll need in order to ha
completion routine. As y
doesn't need an interrupt
using the unused1 field.

You'd need to use an inte
before compression, or b
Interruptions").

TIME TO CHANNEL

Just before the code jump
generally is not a big deal
use no interpolation.

ROUTINE INTERRUPTIONS

The interrupt routine gives you a chance to manipulate the sound data before any sound compression is done. For some of the operations that you may want to carry out inside the interrupt routine, you'll need access to the A5 world of the application, which is why I stored 2BufRecordToBufCmd's A5 value in the userLong field of gRecordStruct.

```
ength = gSampleAreaSize;
tr = (Ptr) ((*bufferHandle) + gHeaderLength);
ionRoutine = (ProcPtr) MyRecComp;
ptRoutine = nil;
g = SetCurrentA5();
 0;
 = 0;
```

eeds to know what to do when it's finished recording. Since nously, it needs a completion routine. (I'll talk more about *could* leave out the completion routine and just poll the if it's finished recording. To do that, you'd repeatedly call rdStatus, and when the status routine informed you that ou'd restart the recording and play the buffer that had just however, it's better to know as soon as possible when the e the more quickly you can restart the recording, the more pauses between recordings.

ood place to store 2BufRecordToBufCmd's A5 value, which ve access to the application's global variables from the ou can see, the rest of the fields are set to 0. The code routine. There's also no point in passing an error back or

rrupt routine if you wanted to change the recorded sound efore the completion routine was called (see "Routine

s into the main loop, it needs to open a sound channel. This , but for 2BufRecordToBufCmd, I initialized the channel to

For more information about sound interrupt routines, take a look at *Inside Macintosh* Volume VI, page 22–63.

Warning: Don't try to accomplish too much in an interrupt routine. In general, you'll want interrupts to be minimal, and possibly written in assembly language, to avoid unnecessary compiler-generated code.

```
gError = SndNewChannel (&gChannel, sampledSynth, init
```

Interpolation causes clicks between the sound buffers when th
back, which can be a rather annoying addition to your record
you're going for that samba beat).

JUST FOR THE RECORD

To start recording, all the code needs to do now is call the lov
routine, pass in gRecordStruct, and tell it that it wants the rec
asynchronously.

```
gError = SPBRecord (gRecordStruct, true);
```

LOOP THE LOOP

The main loop of this code is a simple while loop that waits u
pressed or an error occurs in the recording, at which time the

```
/* main loop of the app */
while (!Button() || (gRecordStruct->error < noErr));
```

ROUTINE COMPLETION

You don't want a completion routine to do much, generally, s
time and keeps your system locked up while it's running. The
completion routine, one of which has four parts to itself.

The first part of the completion routine sets its A5 value to b
value of the application. This gives you access to the applicati
from the completion routine.

```
storeA5 = SetA5 (inParamPtr->userLong);
```

If the completion routine weren't broken into two parts here,
optimization scheme would cause a problem at this point: acc
would be pointed to in an address register as an offset of A5 b
to set A5 to your application's A5 value, and you'd get garbag
Therefore, it's necessary to restore your A5 value (part 1 of th
and then call the secondary completion routine to actually do

Before the routine does any work, it needs to make sure that
problems with the recording. If there were errors, the code d
completion routine without doing anything.

```
if (gRecordStruct->error < 0)
   return;
```

```
NoInterp, nil);
```

ney're played back to
ing (unless, of course,

y-level recording
cording to occur

ntil the mouse button is
application quits.

ince it's run at interrupt
re are three parts to this

e the same as the A5
on's global variables

the MPW C compiler
ess to global arrays
efore you had a chance
e information.
ne completion routine)
all the work.

there have not been any
rops out of the

Next the routine prepares
correcting the header's le
gRecordStruct, which no

```
header = (SoundHeaderP
   gHeaderSize);
header->length = gReco
```

Once the header's been fi
routine to play the sound.

```
PlayBuffer (gBufferHan
```

The last part of the real c
recording. To do this, the
rebuild gRecordStruct to
XOR on the variable gW
include setting the correc
bufferLength is correct. C
SPBRecord to restart the

```
#define NextBuffer(x)

gWhichRecordBuffer = N
gRecordStruct->bufferP
   gDataStart);
gRecordStruct->millise
gRecordStruct->count =
gRecordStruct->bufferL

err = SPBRecord (gReco
```

The last piece of the com
routine started.

```
storeA5 = SetA5 (store
```

PLAY TIME

The code in the PlayBuff
set up the command para
know what channel to pla
sound structure by telling
SndDoCommand along v
SndDoCommand then pl
call, false, basically tells th

s the header of the buffer, which has just been filled, by
ngth field. This field needs to be set to the count field of
w contains the actual number of bytes recorded.

```
tr) (*(gBufferHandle[gWhichRecordBuffer]) +

rdStruct->count;
```

xed, the code just sends the buffer handle off to the play
(See "Play Time" for a full explanation of the play routine.)

```
dle[gWhichRecordBuffer]);
```

ompletion routine prepares gRecordStruct to start the next
code needs to select the correct buffer to record to and
reflect any changes. The macro NextBuffer performs an
hichRecordBuffer to make it either 1 or 0. The changes
t buffer to record to and checking to see that the
Once the structure is reset, the code makes the next call to
recording.

```
(x ^= 1)

extBuffer (gWhichRecordBuffer);
tr = (*(gBufferHandle[gWhichRecordBuffer]) +

conds = 0;
 gSampleAreaSize;
ength = gSampleAreaSize;

rdStruct, true);
```

pletion routine resets A5 to what its value was when the

```
A5);
```

er routine is very simple Sound Manager code. All it does is
meters and call SndDoCommand. The routine needs to
y into and what buffer to play, so the code sets up the local
it which buffer to play, and sends that local structure to
vith the necessary channel information (gChannel).
ays the sound. The last parameter in the SndDoCommand
ne Sound Manager to always insert the command in the

channel's queue: if the queue is full, SndDoCommand will wai
insert the command before returning.

```
localSndCmd.cmd = bufferCmd;
localSndCmd.param1 = 0;
localSndCmd.param2 = (long) ((*bufferHandle) + gHeade
gError = SndDoCommand (gChannel, &localSndCmd, false)
```

If you wanted to send the sounds to a different machine to be
simply replace the code in the the PlayBuffer routine with IP
Toolbox calls telling a second machine to play the buffers.

CLEANING UP AFTER THE SHOW

Once the code finds the mouse button down or discovers that
the recording and exits the main loop, there's only one last thi
first part of cleaning up is to close the sound input driver. Bef
driver, you need to make sure it's not in use; the routine SPBS
recording.

```
gError = SPBStopRecording (gSoundRefNum);
SPBCloseDevice (gSoundRefNum);
```

Next you need to dispose of the handles and pointers you've b
sending them on their way, however, you have to make sure th
allocated, so the code checks to see whether or not the handle

```
for (index = 0; index < kNumberOfBuffers; ++index)
   DisposeHandle (gBufferHandle[index]);
DisposePtr ((Ptr) gRecordStruct);
```

Last but not least, the code disposes of the sound channel for
quitNow flag clears the sound queue before the channel is clo

```
gError = SndDisposeChannel (gChannel, true);
```

COMPOSE YOURSELF

So now you know a little bit more about doing basic sound in
fielded many questions about clicks, pauses between buffers, a
resolved and built into 2BufRecordToBufCmd. The specific t
here may not apply to what you're interested in doing right n
the sound input driver or are interested in continuous recordi
may be useful to you in some other application. You've heard
you like and leave the rest"? Sound advice (so to speak).

THANKS TO OUR TECHNICAL REVIEWERS

Neil Day, Kip Olson, and Jim Reekes, who burned the midnight oil ripping this code to shreds and putting it back together again. •

it until there's space to

```
rSize);
;
```

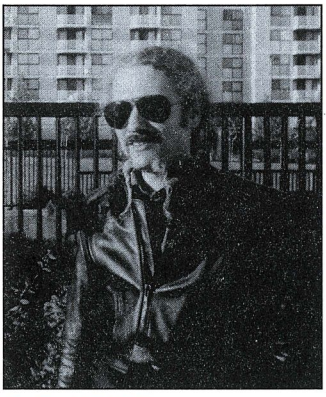
played, you could
C or Communications

an error occurred in
ng to do: clean up. The
ore you can close the
topRecording stops the

been using. Before
hat they have been
s and pointer are nil.

you. Setting the
sed.

put at a low level. I've
nd so on, which I've
echniques I've outlined
ow, but if you're using
ng, parts of this sample
the saying "take what



BE OUR GUEST

BACKGROUND-ONLY APPLICATIONS IN SYSTEM 7

C. K. HAUN

One of the least heralded new features of System 7, but nonetheless a very important one, is full support for faceless background applications (FBAs). An FBA is a full-fledged application that's invisible to the user. It has its own event loop, and it receives time and some events like any other application, but it doesn't have a menu bar, windows, dialogs, or other graphic components. An FBA is a normal file of type 'APPL'.

FBAs are, by a stretch of the imagination, similar to UNIX® daemons. The purpose of an FBA is to provide services to other applications or to monitor the system. For instance, an application that periodically checks your hard drive for files that haven't been backed up lately is a perfect candidate for FBA status. Thus, an FBA can be a silent partner to your application, INIT, cdev, desk accessory, or driver.

An FBA is the best way to provide certain services. For example, an FBA paired with a desk accessory can enable the DA to send Apple events, something a DA cannot usually do. (See the AECDEV/AEDAEMON sample in the snippets provided with the DTS Sample Code on the *Developer CD Series* disc.) An FBA can replace an INIT that patches traps to get time and provides services, or it can replace a driver that depended on periodic run messages to operate. Converting to an FBA not only frees you from having to patch to get the time you need, but also gives you a fully supported and documented interface and design.

C. K. HAUN has been programming Apple computers since 1979, writing commercial education, utility, and game applications for the Apple II, IIGS, and Macintosh, with some occasional dark forays into the Big Blue world. (It paid the rent.) He currently works in Developer Technical Support and focuses mainly on Apple events and the Edition Manager. Besides working to provide the best possible support to developers, he's been trying to organize the Silicon Valley chapter of Heck's Moofers, a motorcycle club devoted to the precept that computer nerds on