

MX

developer's journal

volume 2 issue 8 2004 www.mxdj.com



THE LEADING MAGAZINE FOR MACROMEDIA MX DEVELOPERS & DESIGNERS



Building LEGO Company's Interactive CD-ROM with Director MX

VIRTUAL SHOWROOM

DREAMWEAVER
Valid Code

FLASH
The Evolution of Web Services and Flash

FREEHAND
Form vs Feature Part 2

COLDFUSION
Beginning OOP in AS 2.0





Valid Code
Dreamweaver MX 2004
by justin kozuch



The Evolution of Web Services and Flash
A match made in http://heaven
by david voegeleer



Form vs Feature Part 2
Drawing methods
by ron rockwell



Beginning OOP in AS 2.0
How to duplicate MovieClips with classes
by john c. bland ii



LEGO Virtual Showroom
Practice makes perfect
by lisa del padre

7 WPS
A scalable approach
by erik larson

40 xile
Cartoon
by louis f. cuffari

58 vanguard
140 Decibels
by insomnia creations

18 Grab Hold of the Vine and Swing Over the Pitfalls
How to deliver interactive Flash sites efficiently
by ryan moore

38 Text on the Other Platform
Accent on ANSI
by james newton

48 Interactive Kiosks
Maximizing the capabilities
by darrel plant

58 ASCII, ANSI, Roman 1, and What's All That?
Earn your 'guru' nickname
by kerry thompson

CFDynamics

A Division of Konnections Inc.

WHERE COLDFUSION EXPERTS HOST.



RUN YOUR CFML:

Within BlueDragon Server,
or as a native J2EE web
application.

NOW AVAILABLE!

BlueDragon [6.1]



As one of the ColdFusion hosting community leaders, CFDynamics is constantly looking for ways to deliver new and cutting edge technologies to the ColdFusion community. We are excited to announce our premiere partnership with New Atlanta by offering BlueDragon hosting. We look forward to seeing how BlueDragon expands the possibilities of ColdFusion.

QUALITY CF HOSTING PLANS

- FREE SQL server access
- FREE account setup
- UNLIMITED email accounts
- GENEROUS disk space / data transfer
- 30 day MONEY-BACK GUARANTEE
- GREAT VALUE!

RELIABLE NETWORK

- 99.99% average uptime!
- State-of-the-art data center has complete redundancy in power, HVAC, fire suppression, bandwidth, and security
- 24 / 7 network monitoring

FANTASTIC SUPPORT SERVICES

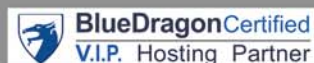
- Comprehensive online support
- Knowledgeable phone support
- We focus on your individual needs

**SIGN UP FOR A HOSTING
ACCOUNT TODAY!**

Use promo code: **MXDJ04H**
and receive a **FREE T-SHIRT!**



WWW.CFDYNAMICS.COM
866 - CFDYNAMICS
866-233-9626



Dreamweaver Website Development

MANY needs - ONE solution



MX Kollection for ColdFusion and PHP

Create powerful dynamic lists

- Sort, filter and page result sets
- Perform multiple deletes
- Easily create Master/Detail lists

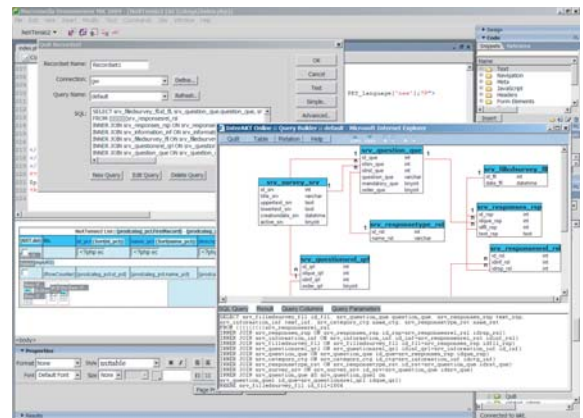
Build unified add/modify forms

- Client side and server side validation
- Insert into two tables
- Create form actions such as send mail or delete related record
- File/Image upload and resize

Create recordsets visually

- Build complex queries across multiple tables quickly

... and many more



The MX Kollection is a suite of extensions designed to **change the way you create dynamic web applications** with Dreamweaver MX.

ColdFusion and **PHP** developers will find our product enabling them to visually develop **e-Commerce, CMS, CRM, Backend** and other web solutions with increased efficiency, quality and capability.

Our customers think of it as **the next level in Dreamweaver MX visual software development.**

Download the demo and see the features and benefits of our extensions:

<http://www.interaktonline.com/>

MX Kollection - Professional web tools



Group Publisher Jeremy Geelan
Art Director Louis F. Cuffari

EDITORIAL BOARD
Dreamweaver Editor
Dave McFarland
Flash Editor
Jesse Warden
Fireworks Editor
Kleanthis Economou
FreeHand Editor
Louis F. Cuffari
Ron Rockwell
ColdFusion Editor
Robert Diamond
Director Editor
James Newton

INTERNATIONAL ADVISORY BOARD
Jens Christian Brynildsen **Norway**,
David Hurrows **UK**, Joshua Davis **USA**,
Jon Gay **USA**, Craig Goodman **USA**,
Phillip Kerman **USA**, Danny Mavromatis **USA**,
Colin Moock **Canada**, Jesse Nieminen **USA**,
Gary Rosenzweig **USA**, John Tidwell **USA**

EDITORIAL
Executive Editors
Gail Schultz, 201 802-3043
gail@sys-con.com
Jamie Matusow, 201 802-3042
jamie@sys-con.com

Editors
Nancy Valentine, 201 802-3044
nancy@sys-con.com
Jennifer Van Winckel, 201 802-3052
jennifer@sys-con.com

Assistant Editor
Torrey Gaver, 201 802-3041
torrey@sys-con.com

Technical Editors
James Newton • Sarge Sargent

To submit a proposal for an article, go to
<http://grids.sys-con.com/proposal>.

Subscriptions
E-mail: subscribe@sys-con.com
U.S. Toll Free: 888 303-5282
International: 201 802-3012
Fax: 201 782-9600
Cover Price U.S. \$5.99
U.S. \$29.99 (12 issues/1 year)
Canada/Mexico: \$49.99/year
International: \$59.99/year
Credit Card, U.S. Banks or Money Orders
Back Issues: \$12/each

Editorial and Advertising Offices
Postmaster: Send all address changes to:
SYS-CON Media
135 Chestnut Ridge Rd.
Montvale, NJ 07645

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ

List Rental Information
Kevin Collopy: 845 731-2684,
kevin.collopy@edithroman.com,
Frank Cipolla: 845 731-3832,
frank.cipolla@epostdirect.com

Promotional Reprints
Kristin Kuhnle, 201 802-3026
kristin@sys-con.com

Copyright © 2004
by SYS-CON Publications, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system, without written permission.

MX Developer's Journal (ISSN#1546-2242) is published monthly (12 times a year) by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

SYS-CON Media and SYS-CON Publications, Inc., reserve the right to revise, republish, and authorize its readers to use the articles submitted for publication. MX and MX-based marks are trademarks or registered trademarks of Macromedia, in the United States and other countries. SYS-CON Publications, Inc., is independent of Macromedia. All brand and product names used on these pages are trade names, service marks or trademarks of their respective companies.

MX
developer's journal

Introducing...

The Macromedia Web Publishing System
by erik larson

On most Web sites there is a barrier between the people who want to update and publish content to the site and the Web site itself. Content eventually becomes out of date, improvements remain undone, and new information languishes unpublished – waiting for precious design time or development budget before going live. For corporate Web sites, large budgets are required to fund the work; for intranets or smaller sites, this often means doing the work when there's time – which means not doing it at all.

The Macromedia Web Publishing System (WPS) solves this intractable problem by offering a scalable approach that works for large and small organizations alike – a solution that provides everything an organization needs to build, manage, and publish to Web sites. The WPS meets the needs of Web developers, IT managers, and business professionals, empowering users to easily publish to the Web within a centrally controlled, standards-based environment.

The WPS includes three key elements: Macromedia Studio MX 2004 with Flash Professional for Web developers; the new Macromedia Contribute 3 with FlashPaper 2 for nontechnical Web site publishing; and the new, server-based Contribute Publishing Services for central administration by Web and IT managers. Perhaps most importantly, with a starting price of less than \$2,500 to empower 10 users plus one developer, the WPS provides this at an affordable price, allowing IT purchasers to buy only what they need and begin using it immediately.

Eliminated Bottlenecks

Organizations have been challenged by the Web content management problem since the first Web site was published. Many still rely on Webmasters and Web teams to maintain the site, creating a bottleneck that frustrates and delays business managers and bothers the Web team with never-ending maintenance duties.

Some organizations attempt to teach nontechnical managers how to use Web site-building products like Microsoft FrontPage, but those tools require significant technical training to be used effectively and do not offer critical administrative and HTML code controls needed to manage users and protect the design and functionality of the Web site.

Web content management systems often promise the nirvana of central control and broad-based empowerment, but the majority of those systems fail to deliver because of their technical complexity, expensive up-front licensing costs, and daunting implementation. Even successful, custom-built, dynamic systems suffer from two major shortcomings: they usually do a poor job of flexibly handling unstructured content and they do not give content contributors the intuitive, true visual editing and rich content creation experience they expect.

The WPS is a radical alternative to traditional solutions: it's easy to use, it's affordable, and it works. This new solution distributes Web publishing capabilities throughout an organization and across enterprises so that business managers can make changes themselves or delegate updates to others.

The affordable price and flexible architecture make it a perfect stand-alone solution for intranets and small to midsized business Web sites. It can also be used to complement existing systems and fill in their gaps. For example, it can be used to manage unstructured content that does not lend itself to dynamic systems, or used as an intuitive front end for editing content stored in file-based content management systems, such as Interwoven TeamSite.

Integrated with Existing Macromedia Products

The WPS builds on extensive efforts by Macromedia to integrate its products deeply and create a comprehensive solu-

tion to today's Web site problems. Integration points range from Dreamweaver templates that protect page integrity when used by Contribute users to Fireworks integration that facilitates powerful image editing in both Dreamweaver and Contribute. The results are heavily leveraged in the WPS, making Web-site publishing straightforward and approachable. This also allows over 2 million Web professionals already familiar with Dreamweaver and Studio to use the WPS to collaborate effectively with the nontechnical business professionals who contribute content to the sites they build and manage.

Many of the core innovations of the WPS lie in the new Contribute 3 release, which truly takes the product to the next level and allows for unprecedented scalability, affordability, and ease of use. Improvements include a new approval and review system, dramatically enhanced performance, full support for CSS and CSS-P, an integrated image editor, and completely re-architected and enhanced administrative controls.

Contribute 3 includes new customization and extensibility features that are valuable for large deployments. With Contribute 3 it is possible to extend Contribute using the same APIs available for Dreamweaver extensions and create customized installers suitable for remote deployment across an enterprise. This means you can create your own version of Contribute with branded elements or entirely new custom features.

The new Contribute Publishing Services server component included with the WPS allows Web and IT managers to manage users, roles, and Web-site editing permissions from a central location. This lightweight application is easily installed on Windows, Linux, and Unix servers; or easily deployed to standard J2EE application environments. It enables administrators to centrally manage access to Web sites, integrate with enterprise systems using LDAP and Active Directory user directories, and track publishing activities across large numbers of Web sites and publishers.

Contribute Publishing Services tracking takes advantage of a new Web services-based notification feature in

Contribute 3 that allows partners to create server-side extensions. All editing and publishing activities taken by the Contribute client can be published via SOAP to a server and then processed as needed to drive page deployments, send notifications, generate reports, or trigger any other required server-side actions.

The new release of Contribute also integrates Macromedia FlashPaper 2, a dramatic update to this exciting document-sharing technology included on both the Mac OS and Windows. The printer-driver technology behind FlashPaper 2 generates enhanced Flash documents (SWF files) with full text search and selection, bookmarking, and hyperlinking. The Flash document user interface includes an extended ActionScript API. FlashPaper 2 integrates directly with Microsoft Office and, on Windows, the FlashPaper 2 printer driver can even generate Adobe PDF files.

New Business Opportunities

The WPS offers Web agencies a range of new business opportunities. The system encourages clients to engage in Web projects to revamp existing sites or build out new extranets and intranets. Its affordable price makes projects possible for more customers than ever – from small organizations to departmental intranets. Used in conjunction with custom-built dynamic systems, it maintains site sections that are easily treated as page-based or static content and focuses dynamic efforts on transactional applications and catalog-type database content.

By focusing first on the needs of nontechnical individuals, the Macromedia Web Publishing System unlocks the power of Web publishing for everyone within an organization who needs to communicate information to others without putting Web-site integrity at risk. Organizations can communicate internally through intranets or externally through corporate Web sites, government agencies can publish policies and procedures to their constituencies, and educational institutions can publish campus-wide e-learning and academic Web sites.

Try out a preview version of the solution at www.macromedia.com/go/wps. 

SYS-CON MEDIA
President & CEO
 Fuat Kircaali, 201 802-3001
fuat@sys-con.com
Vice President, Business Development
 Grisha Davida, 201 802-3004
grisha@sys-con.com
Group Publisher
 Jeremy Geelan, 201 802-3040
jeremy@sys-con.com

ADVERTISING
Senior Vice President, Sales & Marketing
 Carmen Gonzalez, 201 802-3021
carmen@sys-con.com
Vice President, Sales & Marketing
 Miles Silverman, 201 802-3029
miles@sys-con.com
Advertising Sales Director
 Robyn Forma, 201 802-3022
robyn@sys-con.com
Advertising Sales Manager
 Megan Mussa, 201 802-3023
megan@sys-con.com
Associate Sales Managers
 Kristin Kuhnle, 201 802-3026
kristin@sys-con.com
 Beth Jones, 201 802-3028
beth@sys-con.com
 Dorothy Gil, 201 802-3024
dorothy@sys-con.com

PRODUCTION
Production Consultant
 Jim Morgan, 201 802-3033
jim@sys-con.com
Lead Designer
 Louis F. Cuffari, 201 802-3035
louis@sys-con.com
Art Director
 Alex Botero, 201 802-3031
alex@sys-con.com
Associate Art Director
 Richard Silverberg, 201 802-3036
richards@sys-con.com
Assistant Art Director
 Tami Beatty, 201 802-3038
tami@sys-con.com

SYS-CON.COM
Vice President, Information Systems
 Robert Diamond, 201 802-3051
robert@sys-con.com
Web Designers
 Stephen Kilmurray, 201 802-3053
stephen@sys-con.com
 Matthew Pollotta, 201 802-3054
matthew@sys-con.com
Online Editor
 Martin Wezdecki, 201 802-3045
martin@sys-con.com

ACCOUNTING
Financial Analyst
 Joan LaRose, 201 802-3081
joan@sys-con.com
Accounts Payable
 Betty White, 201 802-3002
betty@sys-con.com
Accounts Receivable
 Shannon Rymysza, 201 802-3082
shannon@sys-con.com

EVENTS
President, SYS-CON Events
 Grisha Davida, 201 802-3004
grisha@sys-con.com

CUSTOMER RELATIONS
Circulation Service Coordinators
 Shelia Dickerson, 201 802-3082
shelia@sys-con.com
 Edna Earle Russell, 201 802-3081
edna@sys-con.com
 Linda Lipton, 201 802-3012
linda@sys-con.com
JDJ Store Manager
 Brundila Staropoli, 201 802-3000
bruni@sys-con.com

Erik Larson is director of product management for the Macromedia Web Publishing System. A Macromedian since early 2000, Erik has focused on new product development in the areas of user experience, information architecture, and content management.
elarson@macromedia.com

Complete source code and asset management in Dreamweaver MX—now possible with Surround SCM.

Dreamweaver users know a beautiful Web-based product is only skin deep. Underneath, it's a tangle of hundreds or thousands of ever changing source files. Without a good development process and strong tools, bad things happen. Surround SCM can help.

Surround SCM lets you...

Track multiple versions of your source files and easily compare and merge source code changes.

Check out files for exclusive use or work in private workspaces when collaborating on a team.

Automatically notify team members of changes to source files—push changes through your organization.

View complete audit trails of which files changed, why, and by whom.

Associate source code changes with feature requests, defects or change requests (requires additional purchase of TestTrack Pro).

Remotely access your source code repository from Dreamweaver MX.

Surround SCM adds flexible source code and digital asset control, powerful version control, and secure remote file access to Dreamweaver MX. Whether you are a team of one or one hundred, Surround SCM makes it easier to manage your source files, letting you focus on creating beautiful Web-based products.

Features:

Complete source code and digital asset control with private workspaces, automatic merging, role-based security and more.

IDE integration with Dreamweaver MX, JBuilder, Visual Studio, and other leading Web development tools.

Fast and secure remote access to your source files—work from anywhere.

Advanced branching and email notifications put you in complete control of your process.

External application triggers let you integrate Surround SCM into your Web site and product development processes.

Support for comprehensive issue management with TestTrack Pro—link changes to change requests, bug reports, feature requests and more.

Scalable and reliable cross-platform, client/server solution supports Windows, Linux, Solaris, and Mac OS X.

Want to learn more?



Download Seapine's just-released white paper, **Change Management and Dreamweaver**, to discover tips, tricks and best practices for achieving full software configuration functionality. Visit www.seapine.com/whitepaper.php and enter code WM0604 to receive your copy today.

www.seapine.com
1-888-683-6456





validcode



In today's wild, wild Web, the catchphrases are usability, accessibility, standards-compliance, and valid code. Why is valid code so important? The sheer number of browsers out there that display the same code differently make it virtually impossible to have different versions of code for each browser. In this day and age, maintaining standardized and valid code is essential.

by justin kozuch

Dreamweaver MX 2004, being the top-of-the-heap software application it is, can help you code, but sometimes it gets a bit confused. Today, we are going to look at how to extend and use Dreamweaver MX 2004 to write standards-compliant, valid code.

Are you ready to get your valid code freak on? If so, keep reading!

Analyzing Your Workflow

The cool thing about Dreamweaver is that its users come from all walks of life, from beginners who've just installed the software, to hard-core code warriors who use its Server Behaviours to build complex applications or code their own applications from scratch.

Hand coders will love the Code View (View -> Code) because it facilitates typing code without having to use a third-party text editor. To the relief of many hand coders using WYSIWYG applications, Dreamweaver, unlike other applications, won't change your code at will.

Designers will love the Design View (View -> Design) because it allows them to see the end-result of the underlying code, and the Property Inspector (Window > Properties, Ctrl+F3) will allow them, to a degree, to tweak the code with-

out having to remember HTML or crack open their trusty HTML books.

As I mentioned previously, Dreamweaver will sometimes get confused and insert invalid or deprecated code. Poor Dreamweaver. However, have no fear, we will make it all better in a second. Let's start by configuring Dreamweaver for accessibility.

Setting It All Up

Open Dreamweaver MX 2004 and click on Edit -> Preferences (Ctrl+U). Select the Accessibility option from the Category list (see image I) and check all four boxes, labeled Form Objects, Frames, Media, and Images.

What happens when you insert one of those items? A dialog box will pop up (see images II through V) and prompt you to add information, such as ALT tags and the like. These dialog boxes almost literally force you to enter the proper required attributes for the XHTML doctype. Just don't hit the "Cancel" button. Otherwise you will have to hand code the information, or delete the element and start over. Not fun and very time-consuming!

Now we will configure Dreamweaver to format your code properly. Select Code Format from the Category list (see

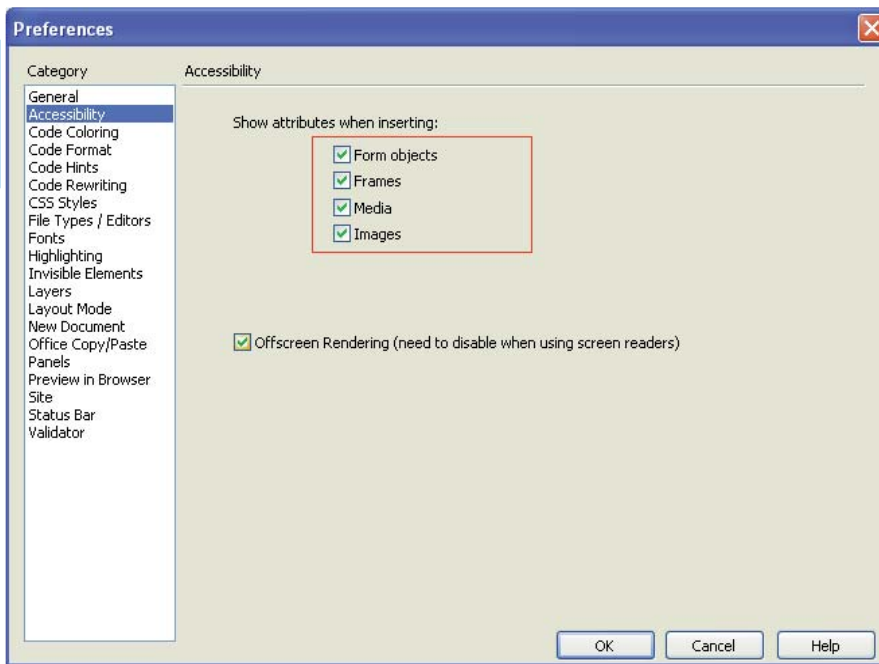


Image VI) and, in the resulting pane, select <lowercase> from the Default Tag Case drop-down menu. Finally, select lowercase="value" from the Default Attribute Case drop-down menu. For centering, select the radio button next to the words "Use DIV tag".

We choose <lowercase> from the Default Tag Case drop-down menu because XHTML doesn't allow for uppercase tags. The same rule applies for the Default Attribute Case. For an example of what happens if you use uppercase tags and attributes with the XHTML doctype, create a page with uppercase tags and attributes and try to validate the page (<http://validator.w3.org/>). Validation errors galore!

Also, don't forget to check off the box named "Make Document XHTML Compliant." Because the acceptable format for documents on the Web uses the XHTML doctype, we will configure Dreamweaver to automatically use the

XHTML doctype if we create a new file. Click on "New Document" in the Category list and, in the resulting pane, select UTF-8 from the Default Encoding drop-down menu.

Now, when you create a new page in Dreamweaver, you will see a few changes. The biggest of them is the new doctype.

```
Transitional: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-1-transitional.dtd">
```

The other two doctypes available for use with XHTML 1.0 are:

```
Strict: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-1-strict.dtd">
```

```
Frameset: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-1-frameset.dtd">
```

If you would like to use XHTML Strict doctype as opposed to XHTML Transitional, simply add it to the Default.html file located in C:\Program Files\Macromedia\Dreamweaver MX 2004\Configuration\Templates, or Macintosh HD > Applications > Macromedia Dreamweaver MX 2004 > Configuration > Templates, if you are

using a Macintosh Operating System. I will discuss how to do that later on in this article.

For more information about doctypes, read Holly Bergevin's excellent article "Rendering Mode and Doctype Switching" on CommunityMX at www.communitymx.com/content/article.cfm?page=2&cid=E2F258C46D285FEE.

Finally, the last thing on our list is to configure the Code Validation capabilities of Dreamweaver. Click on "Validator" in the Category list, and check the box next to "XHTML 1.0 Transitional".

And Dreamweaver is configured! The fun stuff really starts now, because we will go more in-depth and do some hacking (the good kind!).

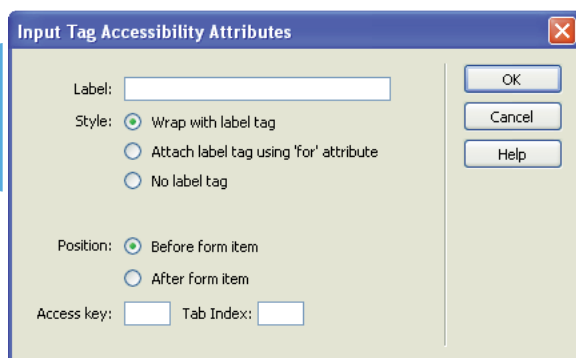
Doctypes and Charsets

Dreamweaver does a pretty good job of generating the code for an XHTML-compliant Web document:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-1-transitional.dtd">
<html
xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
</head>
<body>
</body>
</html>
```

However, there are still a few errors, and the namespace is incomplete. That's not a big deal; we will fix all that right now.

Locate the document called Default.html on your hard drive. If you are on the Windows operating system, this file is located in C:\Program Files\Macromedia\Dreamweaver MX 2004\Configuration\Templates. If you are using a Macintosh, you can find it in Macintosh HD > Applications > Macromedia Dreamweaver MX 2004 > Configuration > Templates. Danilo Celic has created an awesome extension that allows for the opening of



default document types. Check out www.communitymx.com/content/article.cfm?cid=01E77 for more information.

Make a backup of this file (ALWAYS make backups if you are editing any of Dreamweaver's main configuration files), and call it something like Default (backup).html.

Open the Default.html file in a text editor, and enter in the following code:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD
XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1
1-transitional.dtd">
<html
xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en" lang="en">
<head>
<title>Untitled Document</title>
</head>

<body>

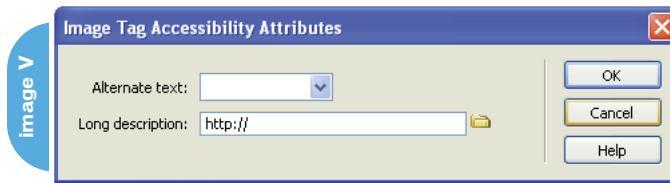
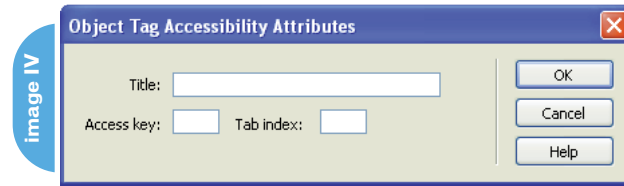
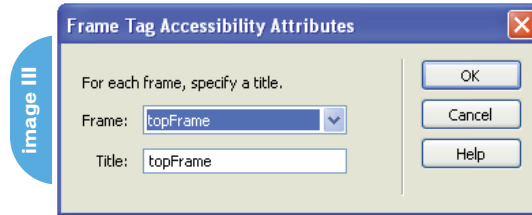
</body>
</html>
```

Note that in this example, the XML declaration is included. XHTML document authors are strongly encouraged to use XML declarations in all of their documents. However, an XML declaration like the one previously mentioned is not required in all XML documents. In fact, some browsers fail when confronted with an XML declaration.

The XML declaration is required when the character encoding of the document is other than the default UTF-8 or UTF-16 and no encoding was determined by a higher-level protocol.

Because we are declaring the document encoding type to be UTF-8, replace the code snippet with this one:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD
XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1
1-transitional.dtd">
<html
xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" con-
tent="text/html; charset=utf-8" />
<title>Untitled Document</title>
```



</head>

<body>
</body>

</html>

While you are here, you can set up links to any style sheets, insert basic meta tags, or do any other general maintenance.

The reason we use UTF-8 as the encoding type is that UTF-8 is the normal character encoding for any HTML file that contains text in two or more non-Latin scripts, but it can be used for any document.

Save the Default.html file, and close it.

Closing the Tags

A few months ago, I had a conversation with an industry colleague at a local user group event. He was speaking about how to get the software to do the work for you, as opposed to having to do all the hard work yourself. The interesting thing was, I realized just how right he was. Think about it. Why do all the hard work yourself, when the software can do it for you?

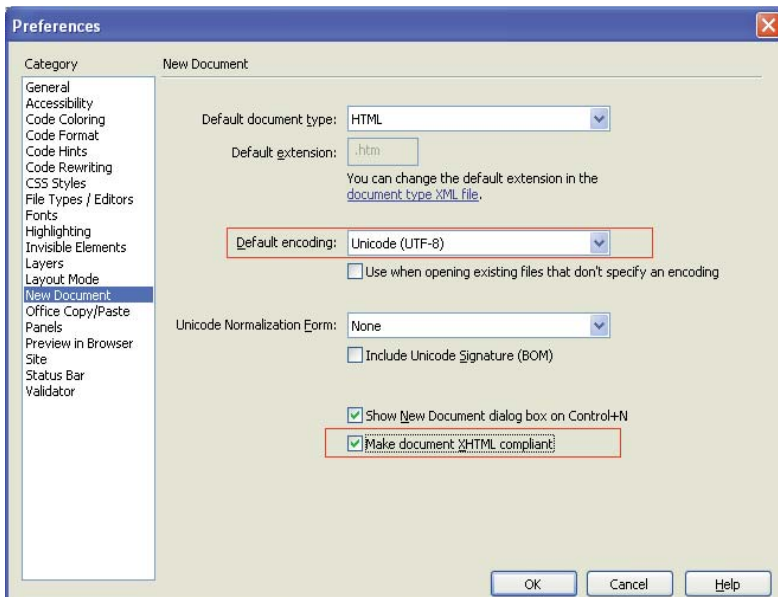
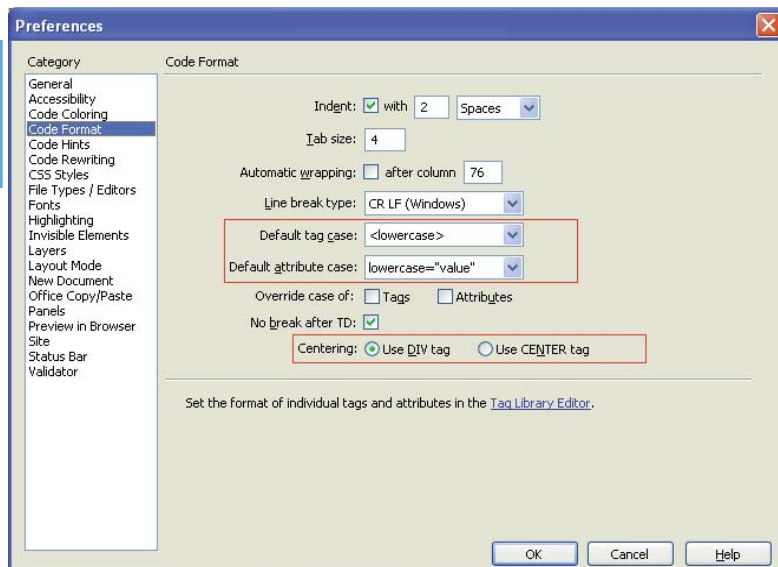
XHTML requires you to close non-closing ("empty") elements, including input tags, image tags,

horizontal rule tags, and break tags. (For more information, see XHTML Guidelines in the NYPL Online Style Guide at www.nypl.org/styleguide/xhtml/guidelines.html and W3C's HTML Compatibility Guidelines for XHTML 1.0, located at www.w3.org/TR/xhtml1/#guidelines.) Closing these pesky elements by hand can be one of the most time-consuming tasks in making the transition to XHTML.

Line Breaks

You can modify Dreamweaver to do the work for you. Let's start with the simple line break, an HTML element with which we are all familiar.

First, find the file called Linebreak.htm. The file is located in C:\Program Files\Macromedia\



Justin Kozuch is a writer, Web developer, and Team Macromedia member who takes pride in helping other Macromedia Dreamweaver users. His work is published weekly on *CommunityMX.com*. He's also the founder of *Dreaming in TO* (www.dreaminginto.com), a Macromedia Dreamweaver User Group located in Toronto, ON. A dynamic "junkie," Justin's passion lies in PHP/MySQL, organic design, and breadcrumb navigation. justin@dreaminginto.com

Dreamweaver MX 2004\Configuration\Objects\Characters, if you are using the Windows Operating System. If you are on the Macintosh Operating System, it is located in Macintosh HD > Applications > Macromedia Dreamweaver MX 2004 > Configuration.

Change line 18 of the `Linebreak.htm` to look like this:

```
return "<br />";
```

To test if your changes worked, create a new HTML document in Dreamweaver, and insert a line break. View the Code (View -> Code) to make sure it worked.

Images

We will also need to "close" the images. The process is similar to what you've just done with line breaks, but a bit more complicated.

Go to Configuration\Objects\Common and make a backup of the `Image.js` file. Open the file in a text editor (any one will suffice) and modify line 45 to look like this:

```
rtnStr= '
```

```
<menuitem name="_Italic" key="Cmd+I"
file="Menus/MM/Text_Style.htm"
argument="' em' "
id="DWMMenu_Text_Style_I" />
```

TIP: Do not open this `menus.xml` in Dreamweaver. Open it using a text editor. Also, make sure you don't use a keyboard shortcut sequence that is currently in use. `Cmd+Shift+h` is a pretty safe bet.



It's everybody's PDF™

Finally, a software company that offers affordable yet flexible PDF solutions to meet every customer's needs. Using activePDF™ to automate the PDF creation process eliminates the need for end-user intervention so your employees can concentrate on what they do best.

Licensed per server, activePDF solutions include a COM interface for easy integration with ColdFusion. Dynamically populate PDF forms with information from a database, convert ColdFusion web pages to PDF on the fly, dynamically print reports to PDF using CF and much more. Users can also merge, stitch, stamp, secure and linearize PDF, all at a fraction of the cost of comparable solutions. Download your free trial version today!



www.activePDF.com

MXDJ Section Editors

Dreamweaver

Dave McFarland

Author of Dreamweaver MX 2004: The Missing Manual, Dave can be relied upon to bring Dreamweaver MX to life for MXDJ readers with clarity, authority, and good humor.



Flash

Jesse Warden

A multimedia engineer and Flash developer, Jesse maintains a Flash blog at www.jessewarden.com and says, referring to the MX product range, that "Things are changing, opportunity is on the frontier, a paradigm shift is occurring for Web design, Web applications, et al."



Fireworks

Kleanthis Economou

A Web developer/software engineer since 1995, now specializing in .NET Framework solutions, Kleanthis is a contributing author of various Fireworks publications and is the technical editor of the Fireworks MX Bible. As an extension developer, he contributed two extensions to the latest release of Fireworks.



FreeHand

Louis F. Cuffari

Cofounder and art director of *Insomnia Creations* (www.insomniacreations.com), Louis has spent most of his life as a studio artist, including mediums from charcoal portraits to oil/acrylic on canvas. In addition to studio art, he has been involved in several motion picture projects in the facility of directing, screenwriting, and art direction. Louis's creative works expand extensively into graphic design, and he has expertise in both Web and print media. He is deputy art director for SYS-CON Media and the designer of MX Developer's Journal.



Ron Rockwell

Illustrator, designer, author, and Team Macromedia member, Ron Rockwell lives and works with his wife, Yvonne, in the Pocono Mountains of Pennsylvania. Ron is MXDJ's FreeHand editor and the author of FreeHand 10 f/x & Design, and coauthor of Studio MX Bible and the Digital Photography Bible. He has Web sites at www.nidus-corp.com and www.brainstormer.org.

ColdFusion

Robert Diamond

Vice president of information systems for SYS-CON Media and editor-in-chief of ColdFusion Developer's Journal, Robert was named one of the "Top thirty magazine industry executives under the age of 30" in Folio magazine's November 2000 issue. He holds a BS degree in information management and technology from the School of Information Studies at Syracuse University. www.robertdiamond.com



"A standards-compliant Web site is one step closer to accessibility compliance"

And that's it! You're done. Now Dreamweaver is creating standards-compliant code.

You might be asking yourself why all of this work was worth the effort. There are many reasons.

Accessibility

Because Web standards incorporate and support accessibility compliance, a standards-compliant Web site is one step closer to accessibility compliance. By making your site accessible to the millions of people affected by disabilities, you can increase your customer base. In many jurisdictions, accessible sites are mandated by law.

An improved user experience also tops the list of reasons. Non-standard code can make it next to impossible for people using less common platforms, devices, or user agents to access your site's content. If your site is standards-compliant, then your site is available to all Web users, who are your potential customers.

Faster Page Loads and Lower Bandwidth Costs

Studies have shown that using modern, standards-based design can

reduce the weight (also known as file size) of a site by 25% to 50%. Who can argue with that?

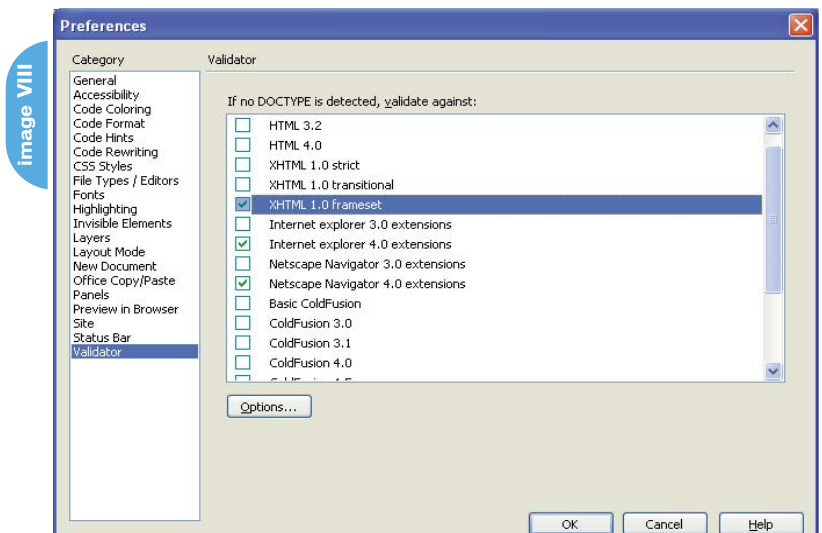
Device Independence

By employing the most recent standards for CSS, you can make it possible for your Web page content to be accessed by different browsers and devices – for example, the same Web site can be usable on both a cell phone and a fancy new computer.

There are many other reasons why you should use standards-compliant code; these are just a few of the more convincing arguments as to why you should adopt a standards-centric workflow.

Resources

- *What Every Web Site Owner Should Know About Standards: A Web Standards Primer:* <http://maccaws.org/kit/primer>
- *The Way Forward with Web Standards:* <http://maccaws.org/kit/way-forward>
- *The Web Standards Project:* www.webstandards.org
- *The Web Standards Group:* www.webstandardsgroup.org





HostMySite.com

Built for ColdFusion Pros

by ColdFusion Pros

plans from

\$8.95 / mo.

FREE Domain Name*

FREE Setup

FREE 2 Months

- 24 / 7 / 365 Phone Support
- 99.9+% Uptime
- Macromedia Alliance Partner
- "Full Control" Panel
- CFMX 6.1 or CF 5.0
- SQL Server 2000 or 7.0
- Custom Tags Welcome

Visit www.HostMySite.com/mxdj for:

2 Months Free

and FREE Setup on Any Hosting Plan*



**"When it comes to ColdFusion hosting,
HostMySite.com rules them all!"**

James Kennedy
mbateam.com

*offer applies to any annual shared hosting plan

call
today!

877•248•HOST
(4678)





Grab Hold of the Vine and Swing Over the Pitfalls

How to deliver interactive Flash sites efficiently

by ryan moore

The world of Flash development can be a dangerous place. Although Flash offers the most comprehensive set of tools for Web site user interface development, there are numerous pitfalls that can cause a project to go sour. Flash's complexity can result in "two-day" projects taking two weeks, database access becoming nearly impossible, and file sizes that are daunting. I'll talk about some of the methods I've used to overcome these types of problems in a site I've recently developed, www.getfreshfruit.com.

Get Your Fresh Cut Fruit Here

www.getfreshfruit.com is a project I worked on through my company, Balance Studios (www.balancestudios.com), and Lindsay, Stone & Briggs (www.lsb.com) for Chiquita Brands, Inc. (www.chiquita.com). The purpose of this Web site was the introduction and launch of Chiquita's newest product innovation, Fresh Cut Fruit. Targeted at females between the ages of 25 and 54 who are concerned about their health, love fruit, appreciate convenience, and are "on the go," the Fresh Cut Fruit site required a friendly, engaging, and interactive user interface. The site's back-end features included a searchable and updateable store locator as well as an e-mail contact form. This interactivity builds the customer loyalty desired by Chiquita.

Immediately, this seemed like a great candidate for Flash. Fun, interactive, and integrated user interfaces are certainly Flash's forte. Developing the site in Flash also had some challenges, such as keeping the file size to a minimum (large product images were part of the project requirements), and, more importantly, staying within our limited timeline for

site development. As I will explain, by maximizing the efficiency of routine Flash tasks, such as animations and data access, I was able to give www.freshcutfruit.com an intriguing front end in the appropriate time span.

What Happened to My Timeline?

Whether it's loading functions, button rollovers, or text effects, nearly every project using Flash involves some type of animation. The traditional method of animating in Flash is accomplished using the timeline and motion tweens with numerous keyframes and a small amount of ActionScript. For the Fresh Cut Fruit site, however, I chose to use a different method of animating: ActionScript-based MovieClip tweening.

Unlike traditional Flash animations, ActionScript-based tweens are accomplished without ever using the timeline. In fact, these tweens do not even require a MovieClip on the stage, just some clever coding. The lack of keyframes in ActionScript-based tweens make the Flash animations much easier to update as a project evolves. The developer is also given much more control over properties of the tweens, such as easing and the ability to react to the completion of the animation.

To accomplish ActionScript tweening for Fresh Cut Fruit, the "MovieClip Tweening" prototype by Ladislav Zigo was used. This Flash extension is available at <http://laco.wz.cz/tween/>. The extension provides the ActionScript classes necessary to accomplish ActionScript-based tweening without having to write any of the movement classes used to accomplish the tweens. The user-required documentation for this extension is extremely thorough, making the extension much easier to use than many of the other tweening-class alternatives.

The most essential method in this prototype is the tween function. Using just

```
instancename.tween(property, value, duration, tweenType, delay, callback);
```

properties of a MovieClip, such as position or alpha, can be tweened for a duration of time. The tweenType of the animation can be set to a variety of options, ranging from an elastic 'bounce' to a simple linear tween to produce highly controllable animations with great "feel."

In the Fresh Cut Fruit project, these tweening classes were used to create the navigation buttons as well as the loading functions' fruit animations. For example, to create, place, and animate a series of top-level navigation buttons, the following ActionScript could be used:

```
Function PlaceButtons()
{
    // create button array
    var x;
    buts = [];
    buts[0] = {label:"6oz Cup", data:"6oz.swf"};
    buts[1] = {label:"12oz Cup", data:"12oz.swf"};
    buts[2] = {label:"24oz Bowl", data:"24oz.swf"};
    buts[2] = {label:"32oz Tray", data:"32oz.swf"};
    buts[3] = {label:"64oz Tray", data:"64oz.swf"};
    // attach the buttons, apply the label, and create the tween
    for (x=0; x<buts.length; x++)
    {
        var nn = "nb" + x;
        attachMovie("navbut", nn, 100+x,
        {_x:50, _y:50+(x*20), _alpha:0,
        myLabel:buts[x].label,
        myMovie:buts[x].data});
        this[nn].tween(["_x", "_alpha"], [55,
```



```
100], 1, "easeoutelastic", x*.2,
null);
}
}
```

Initially in this script, an array is declared, which holds the data used to create the buttons, followed by a *for* statement that loops through the array, places the buttons, and sets some properties, which gets them movin'. Let's break down the last line in the *for* statement, the implementation of the ActionScript-based tweening:

```
this[nn].tween(["_x", "_alpha"], [55,
100],
```

This segment declares that we will be tweening the *_x* and *_alpha* properties of the newly attached MovieClips from their initial values of 50 and 0 (which we determined in the previous line), to 55 and 100, respectively.

```
1, "easeoutelastic",
```

The "1" sets the period of the tween to one second. Note that the tweening prototypes are based on time instead of frames. "easeoutelastic" is the easing equation used on the MovieClips. Instead of tweening straight to the point, they've got a bit of "elastic bounce," just for effect (thank Robert Penner, www.robertpenner.com, for these great functions).

```
x*.2, null);
```

The next property is the amount of time each MovieClip will wait before starting its tween. In this case, we don't want each button to start at the same time, but to "flow in" from top to bottom. The last property, *callback function*, is set to null, but could be directed to a function that would execute when the buttons complete their motion.

By using ActionScript-based tweening in this project, the navigation menu was created without a MovieClip on the stage or a tween on the timeline. Because of this, the navigation menu and the "feel" of the buttons could be quickly updated, allowing much less development time on the entire site while also preparing the site for future updates or unforeseen revisions (adding and remov-

ing pages DOES happen!), simply by changing just a couple lines of code.

Searching for Stores? Use FlashOrb

One of the primary challenges in any Flash-based project is transferring database stored information and server-side logic to the Flash front end efficiently. A number of options exist for Flash-to-server communication, such as Web service components, the XML class, the LoadVars class, and the FlashOrb. Because development time was a major factor in this project, Web services and the FlashOrb were my two most likely choices. Due to its speed, low overhead, and ease of use, the FlashOrb turned out to be my selection.

FlashOrb, as seen in previous issues of MXDJ, is a technology built by the Midnight Coders (www.flashorb.com), based on Flash Remoting, which facilitates client-side Flash to server-side .NET or Java communication. The FlashOrb, as compared to Macromedia Web service components, is faster and more lightweight. In recent benchmarks, the FlashOrb has proven to be up to three times faster than SOAP-based communication when passing complex data types.

When considering overhead, both in file size and client-side memory allocation, Web service components also fall dramatically short of the FlashOrb. Web

service components single-handedly add nearly 39KB to a Flash movie's file size, whereas FlashOrb components add only about 9KB. Web service components also appear to have significant memory allocation issues, whereas the FlashOrb uses an extremely small amount of client-side memory.

In the Fresh Cut Fruit project, a searchable store database was required, as well as a database-driven contact form. To accomplish this on the server side, I selected Microsoft SQL Server and C# .NET. The SQL server contains tables that house Fresh Cut Fruit's store database as well as its contact information. The back end consists of a .NET class, FreshFruit.BgFiles. I won't go into great .NET detail, but here are a few of the methods used for the site:

```
public store[] GetNearestStores(int
zipcode)
```

This method is used in the "Store Locator" and takes a zipcode from the Flash client and, using a custom zipcode distance class and SQL queries, returns an array of "store" structs to Flash. This passing of complex data types would consume many resources using Web services, and would involve intricate parsing using the Flash XML class, but is no problem using FlashOrb.





```
public string[]
GetStateStores(string state)
```

The `GetStateStores` method receives a two-letter state abbreviation from Flash and returns an array of store names that exist in the requested state. This method is used by the "Store Locator" page to display, for the user, a list of stores that carry the Fresh Cut Fruit product in their state.

```
public string SendMailTo(string
emailto, string emailfrom, string
title, string message, string pw);
```

`SendMailTo` is simply a server-side e-mail script (using the `OpenSmtp.Net` component) that sends an e-mail to a specified recipient with a specific title and message. This method is used in both the "Let's Chat" contact form and the "Send to a Friend" form.

Great, but what's the use of all these server-side methods if we have no easy

image 11



way of accessing them? Luckily, executing these methods using `FlashOrb` is as easy as the following, calling the `GetNearestStores` method:

```
// First, set up the Remoting
Connection:
var orbConn = NetServices.create
GatewayConnection
("remoting.aspx");
// Creating the Class Object
var service =
_root.orbConn.getService
("FreshFruit.BgFiles", this);
// Calling the Method
service.GetNearestStores
(Number(tbZipCode.text));
// Receiving the Result
function
GetNearestStores_Result(res) {
// Handle the results
// Results will be in the
form of a multi-dimensional
array
}
```

I won't go into great detail on the `FlashOrb` methods (see Joe Orbman's article in the May 2004 issue of *MXDJ*), but essentially, the .NET server-side classes can be called using only three lines of code and the responses require no parsing! Server-side access

functions from Flash couldn't be any quicker or easier to develop.

It is by using these asynchronous database calls that one of the great user-experience benefits of Flash is realized – user interactivity. When the user runs a search on the store database, he does not lose control of the interface or get a screen refresh. Instead, after a very short load, the user is presented the search results and taken to the next step on the same page. This progression causes much less confusion for the site user, and, in this case, makes the store data much more accessible and fun to find!

Although this is just a sampling, these methods contributed to making the www.getfreshfruit.com site a success, based on timeline, file size, and user interactivity. Macromedia Flash provides developers with a tremendous opportunity to merge user interaction with technology, and by using the right tools, these low-bandwidth, database-driven, and interactive Flash sites can be delivered efficiently. ∞

Ryan Moore is the lead software architect and a principal of Balance Studios Inc. (www.balancestudios.com) as well as epicsoft, Inc. (www.epicsoft.net) of Green Bay, WI. Ryan is a C# programmer and Certified Macromedia Flash Developer. More of Ryan's articles can be found at his personal weblog, www.rymoore.com.

"I was totally intimidated by Java, but I knew I had to learn it. Your class taught me what I honestly thought I couldn't be taught." - Sharon T



Java for ColdFusion Programmers?

Java for ColdFusion Programmers, the five-day Hal Helms, Inc. training class is designed for ColdFusion programmers; no previous Java experience is needed. You'll learn how to think in objects and program in Java.

For class information and registration, come to halhelms.com.

CommonSpot™

Efficient Content Management

fast. easy. affordable.



- 100 % browser-based
- Content object architecture
- Template driven pages
- 50+ standard elements
- Extensible via ColdFusion
- Content reuse
- Content scheduling
- Flexible workflow
- Granular security
- CSS support
- 508 compliance
- Personalization
- Replication
- Custom metadata
- Custom authentication
- Static site generation
- Multilanguage support

With CommonSpot Content Server you get it all. CommonSpot's exceptional blend of rapid deployment, ease of use, customization and scalability make it the leading ColdFusion content management solution.

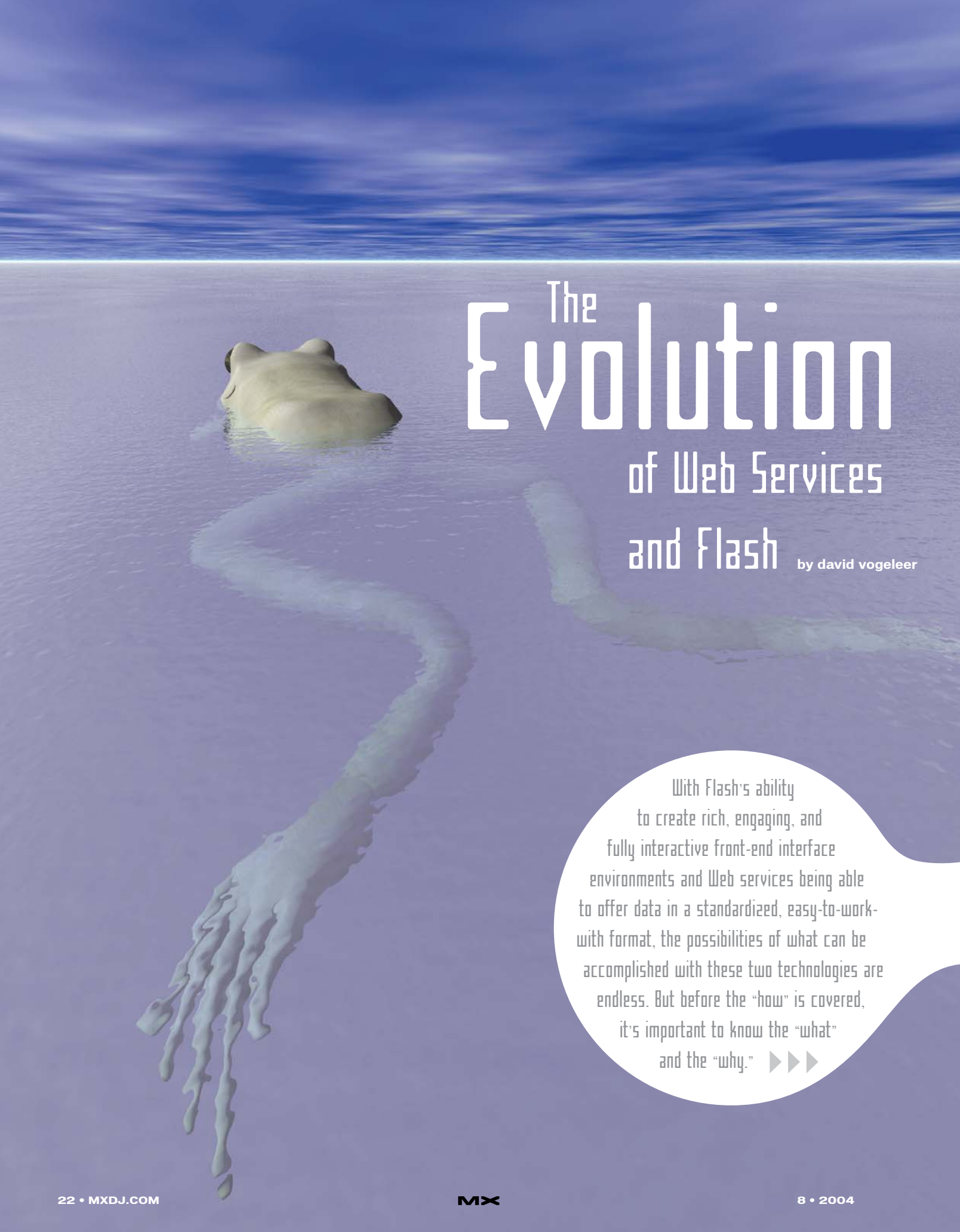
Our rich Web publishing framework empowers developers with out-of-the-box features such as template-driven pages, custom content objects, granular security, flexible workflow and powerful ColdFusion integration hooks (just to name a few), allowing you to rapidly build and efficiently deploy dynamic, high performance Web sites.

For larger implementations, CommonSpot scales efficiently, delivering enterprise-level capabilities like replication, static content generation, multi-site clustering, personalization and custom authentication, across a diverse set of platforms.

For the past six years, PaperThin has been a leader in the ColdFusion community, and CommonSpot has been the solution of choice for organizations of all sizes, including AFL-CIO, Boeing, Kaiser Family Foundation, Ohio University, PGA.com and hundreds of others. CommonSpot's sophisticated feature set and affordable pricing are an unbeatable combination.

Call us today at **800.940.3087** to schedule a live demonstration of your site running under CommonSpot, or visit www.paperthin.com to learn more.

Paper | Thin



The Evolution

of Web Services and Flash

by david vogeeler

With Flash's ability to create rich, engaging, and fully interactive front-end interface environments and Web services being able to offer data in a standardized, easy-to-work-with format, the possibilities of what can be accomplished with these two technologies are endless. But before the "how" is covered, it's important to know the "what" and the "why." ▶▶▶

What Is a Web Service?

A Web service is exactly what it says; it's a service being provided on the Web. So what services does a Web service provide?

A Web service's goal is to provide raw data in well-formed XML format to any application that makes a request to it. That may not make sense by itself, so here is exactly what a Web service does.

A Web service sits on a server, much like any server-side page, and when a request is made to it, the Web service will perform a desired task, and return data in the form of XML. XML is a W3C standard way of packaging data and is a language that nearly any application can read because it is, in fact, not a language at all, but a well-structured document containing raw data.

And Web services can provide data for such things as weather and stock information, and can even be used to translate text into other languages, as you will see in a later example.

So, that is what Web services are and what they do, but that doesn't explain why anyone should use them.

Why Use a Web Service?

Before Web services were so prevalent on the Web, data consumption was a tricky business with everyone having their own ideas of how data should be formatted and returned. If you wrote your own middleware to control how the data was being returned, it wasn't too bad to work with, but if someone else wrote it, you would have to do a lot of guesswork, and patch things together

until you got the results you wanted.

But with Web services, the format of the data is already known: XML. And because Web services are standard, the results will be consistent across nearly all Web services. All you need to know is what the data being returned is, and how to work with XML.

Working With XML Data in Flash

Before you jump right in and connect to Web services in Flash, you should understand how to walk through an XML document in Flash using the built-in XML object.

To use the XML object with Web services, you pass the load method of the XML object, the path to the Web service, followed by a slash, and then the name of the Web method being called. It looks a bit like this:

```
myXML.load("http://www.where-ever.com/myWebService.asmx/myWebMethod");
```

Enough of the academics; here is a short example using the XML object to consume a Web service that will display a new tip about XML every day:

1. Create a new Flash document with two layers: "actions" & "content".
2. Select the first frame of the content layer, drag a TextArea component onto the stage, give it an instance name of xmlTip_ta, and size it to 200x100.
3. Now switch to the actions layer and open up the Actions panel in the first frame and place these actions in it.

```
var myXML:XML = new XML();
myXML.ignoreWhite=true;
myXML.onLoad=function(){
    xmlTip_ta.text =
    this.firstChild.firstChild;
}
myXML.load("http://www.xmlme.com/WSDailyXml.asmx/GetXmlDailyFact");
```

What this code does is to first create the XML object into which we are going to load the Web service. We then turn on the ignoreWhite property so that when we parse the object it doesn't count white space as a node. After that, we create the callback for when data loads into the XML object. At that point, we set the text of our

TextArea component to the tip being returned. Finally, we load the Web service in using the GetXmlDailyFact Web method, which is part of the Web service we used.

Go ahead and test the movie, and you should see something like Image 1. Using the XML object is a good way to get started with Web services.

Although there was not much data being returned, we still had to parse the XML (firstChild.firstChild) because the data coming back was in raw XML format, and looked something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<string
xmlns="http://xmlme.com/WebServices">
    This is where the tip would be.
</string>
```

Even though that example worked, using the XML object isn't always the easiest way to work with Web services, especially if large sets of data are being returned.

But before we revisit this example, we need to go over three things that will make connecting to Web services quick and easy: data binding, the WebServiceConnector component, and the Web Services panel.

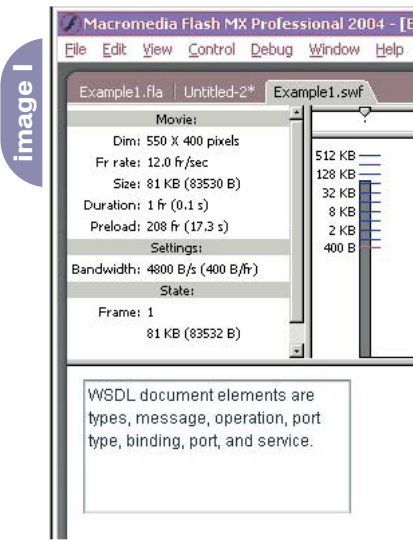
Data Binding

Data binding is a Flash 2004 Professional feature that allows seamless data integration between components. With data binding, you can link a property of component A to a property of component B, and when a component A event is triggered, (in this case, when the Web service sends back data), component B will automatically be updated without any ActionScript required.

In the case of the next few examples, component A will be represented by the WebServiceConnector component. Using data binding, when this component receives data back, it will automatically send the results to the other components to which it is bound. But before you can data bind it, here is a brief description of what the WebServiceConnector component is and what it does.

The WebServiceConnector Component

The WebServiceConnector component is a Flash 2004 Professional compo-



Web Services buttons. The former will add Web services and the latter will refresh all the information with regard to the current Web services in the panel.

The great thing about the Web Services panel is that it will keep track of all of your Web services. It will also display all of the Web methods for each service, with details such as the parameters that are required to be sent with their prospective data types as well as the results being returned. The Web Services panel finds this information by examining the WSDL (Web Service Description Language) that all Web services have. Here is the link to the WSDL with which we have already worked: www.xmlme.com/WSDailyXml.asmx?WSDL.

Now you have seen the WSDL; you should add it to the Web Services panel by pressing the Define Web Services button. The Define Web Services pop-up will appear. Then, use the Add Web Service button to create a new spot in which to put the link to the Web service. Place the full path to the WSDL and press OK. After that, the Web Services panel will load the WSDL in (if you are connected to the Internet) and all the Web methods with their specific details can be visible within the panel as seen in Image II. The Web Services panel is a great way to keep track of all of your Web services and their Web methods.

Another great thing about adding Web services to the Web Services panel is that the WebServiceConnector component will automatically be able to list the Web methods for a particular Web service in the parameters. Also, when you data bind components, you can bind them directly to the parameters of a Web method and even the results of the Web method.

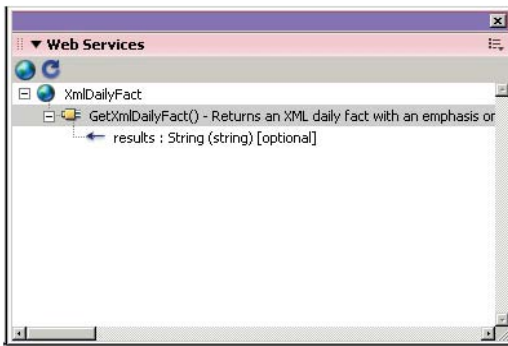
This example will use the same Web service we just used, but this time we will use the WebServiceConnector component and some data binding techniques:

1. Create a new Flash Document with two layers: "actions" & "content".
2. Select the first frame of the content layer and drag a TextArea component onto the stage, give it an instance name of xmlTip_ta, and size it to 200x100.

3. While still in that frame, drag an instance of the WebServiceConnector component onto the stage, give it an instance name of "myConnector", set the WSDLURL to "http://www.xmlme.com/WSDailyXml.asmx?WSDL" and set the operation parameter to "GetXmlDailyFact" (see Image III). Use the Properties panel to set the parameters of the WebServiceConnector component.
4. With the WebServiceConnector component still selected, open up the Component Inspector panel by going to Window/Development Panels/Component Inspector, select the Bindings tab, press the add binding button, and you will see two options (see Image IV):
 - params:Object
 - results:String
5. Select the results:String option and press OK. You will see a binding in the Component Inspector, but it isn't bound to anything yet, so now we have to bind the results of the Web service to the TextArea component.
6. While the Bindings tab is still up, and the newly created binding is still selected, select the Bound To field and a magnifying glass will appear at the end of the field. Select it and the Bound To pop-up will appear (or you can double-click the field itself).
7. In the Bound To panel, all of the components to which you can bind will be visible, but in this case, select the TextArea component, and the bindable properties will appear in the Schema location window, which is the text property. Select the text property and press OK (see Image V). Now the results of the Web service are bound to the TextArea component, so the final step is to tell the WebServiceConnector to go out and get the data.
8. Select the first frame in the actions layer, open the Actions panel (Window/Development Tool/Actions), and place this line of code in:

```
myConnector.trigger();
```

Now you can test the file, and providing you are connected to the Internet, a tip about XML should appear in the TextArea component.



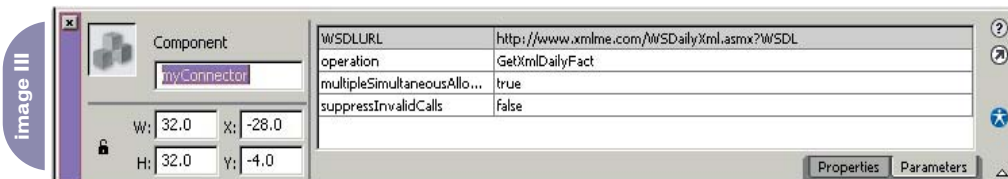
component designed to easily and quickly connect to Web services on the Web. To use it, simply drag it onto the stage, set the URL for the Web service, and call the trigger method on the component to make it connect to the Web service.

And you don't have to maintain a list somewhere on your computer to keep track of all of your Web services and how they work. You can store them right in Flash using the Web Services panel.

The Web Services Panel

With the release of Flash MX 2004, Macromedia included a very helpful panel in the mix, the Web Services panel. It's an easy way to keep track of all of the Web services with which you are working or experimenting. To open it, go to Window/Development Panels/Web Services.

You will see two buttons at the top, the Define Web Services and the Refresh



So far, all we have done is receive data from a Web service. The next step is to send some data to a Web service and collect the results.

For the next example, you will need to add this Web service's WSDL to the Web Service panel: www.aspxpressway.com/maincontent/webservices/piglatin.asmx?WSDL.

This Web service has a Web method called toPigLatin with one parameter, textToTranslate, which is a String data type. It takes the string and translates it into Pig Latin and then returns the translation as a string.

So, if you have a sentence just like this, it would read like this: So, if you avehay a entencesay ustjay ikelay isthay it oudlway eadray ikelay isthay.

Here is an example that will create a small application using the Pig Latin translator Web service:

1. Create a new Flash document with two layers: "actions" & "content".
2. In the first frame of the content layer, drag an instance of the TextArea component onto the stage, give it an instance name of "translate_ta", and change its dimensions to 250x130.
3. Now drag an instance of the Button component onto the stage under the Text Area component, give it an instance name of "translator_butn", and change its label parameter to "Translate".
4. Next, drag an instance of the WebServiceConnector component onto the stage, give it an instance name of "transCon", and set its WSDLURL parameter to the Pig Latin Web service: "http://www.aspxpressway.com/maincontent/webservices/piglatin.asmx?WSDL", then set the operation parameter to "textToTranslate".
5. With the WebServiceConnector still selected, open up the Component Inspector panel, select the Bindings tab, and press the Add binding button (see Image VI). The first binding will be the text that is being sent to the Web service, so select textToTranslate:String, and press OK (see Image VII).
6. With the new binding selected, double-click the Bound To field and in the Bound To panel, select the TextArea component, and hit OK.
7. Now create another binding, select results:String, and press OK.
8. Select the Bound To field again, and

again bind it to the TextArea component and press OK. What these bindings do is take data from the TextArea component, send it to the Web service, and then put the results back in the TextArea component. The last step is to put the trigger within the click event of the button.

9. Select the first frame of the actions layer and place this code in:

```
translator_butn.clickHandler=function(){
    transCon.trigger();
}
```

This code creates an event callback for the click event of the button. Because the information being sent and received from the Web service is bound to the TextArea component, everything else is done automatically.

You can test the movie now, by typing anything you want in the TextArea component, then translating it to Pig Latin.

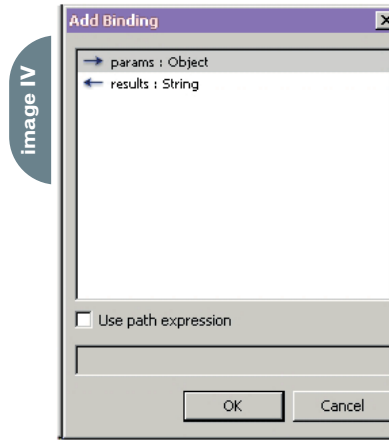
And if you still want to use XML instead of the WebServiceConnector component, remove that component and replace the code in the actions frame with this:

```
//create the XML object
var myXML:XML = new XML();
//ignore the white space
myXML.ignoreWhite = true;

//create the path to the Web service,
// and append the Web method,

// and parameter to the end
var thePath:String = "http://www.aspxpressway.com/maincontent/webservices/piglatin.asmx/toPigLatin?textToTranslate=";
//create event for the button
translator_butn.clickHandler=function(){
    //get the text
    var myString_str:String = translator_ta.text;
    //call the Web method
    myXML.load(thePath+myString_str);
}
//collect the results
myXML.onLoad=function(){
    //display the results

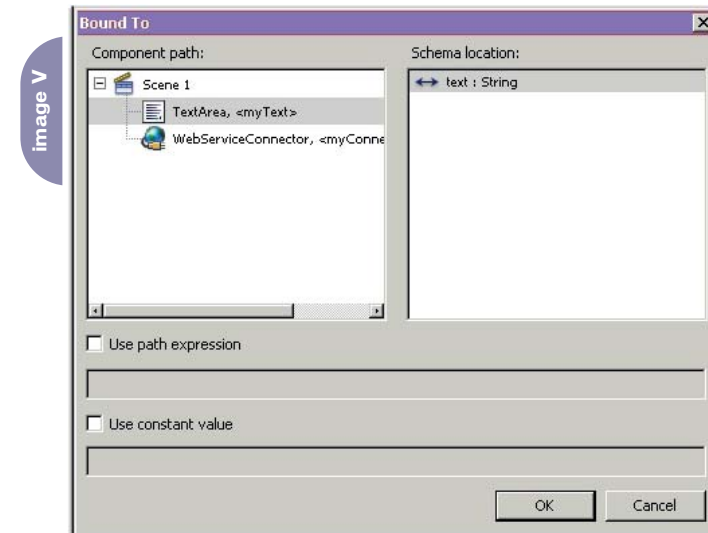
    translator_ta.text=this.firstChild.childNodes[0].nodeValue;
}
```



```
// and parameter to the end
var thePath:String = "http://www.aspxpressway.com/maincontent/webservices/piglatin.asmx/toPigLatin?textToTranslate=";
//create event for the button
translator_butn.clickHandler=function(){
    //get the text
    var myString_str:String = translator_ta.text;
    //call the Web method
    myXML.load(thePath+myString_str);
}
//collect the results
myXML.onLoad=function(){
    //display the results

    translator_ta.text=this.firstChild.childNodes[0].nodeValue;
}
```

This code creates the XML object and sets it to ignore white space. Then, a String variable is created to hold the full path to the Web service as well as the Web



David is the Chief Programming Officer at EMLlabs.com and writer for FlashMagazine.com. He has been working in Flash since version 4, contributed to 2 Flash books and is a certified Flash developer and instructor. missing-link@emllabs.com



method, and the parameter name for the Web method. Next, it creates the event callback for the button. Within the callback, a variable is created to hold the text from the TextArea component, which loads the XML with the main path, and then the variable holding the info string is used to translate. Finally, a callback event is created for when the results load into the XML object, and is displayed back into the TextArea component.

Summary

As you can see, tying into Web services with Flash is quick and easy using the WebServiceConnector component and data binding, or you can use the XML object for more hands-on control. You can also use Flash Remoting to create a more streamlined connection between Flash and the Web service. And

don't forget, whenever you want to start working with a new Web service, add it to the Web Service panel to keep track of it.

Here are a few links to collections of Web services, as well as a few popular ones I have used.

- WebserviceX.NET – Free Web Services Provider:
www.websvcx.net/WS/default.aspx
- XMethods: www.xmethods.com
- Application for Amazon Web Services Developer's Token:
<https://associates.amazon.com/exec/p/anama/associates/join/developer/application.html>
- Google Web API Service:
www.google.com/apis
- Macromedia XML News Aggregator (MXNA) Tools:
www.markme.com/mxna/tools.cfm

image VI

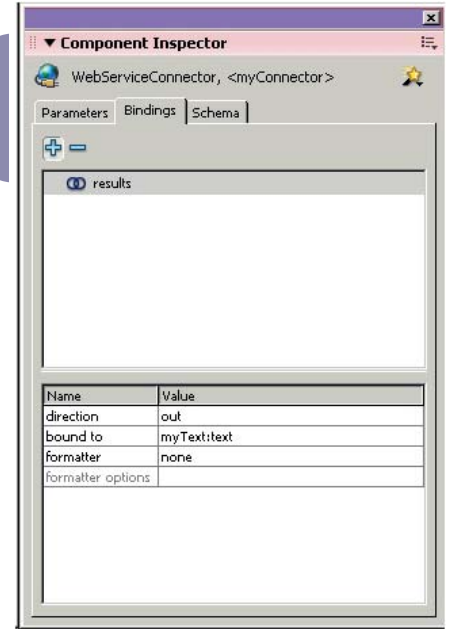
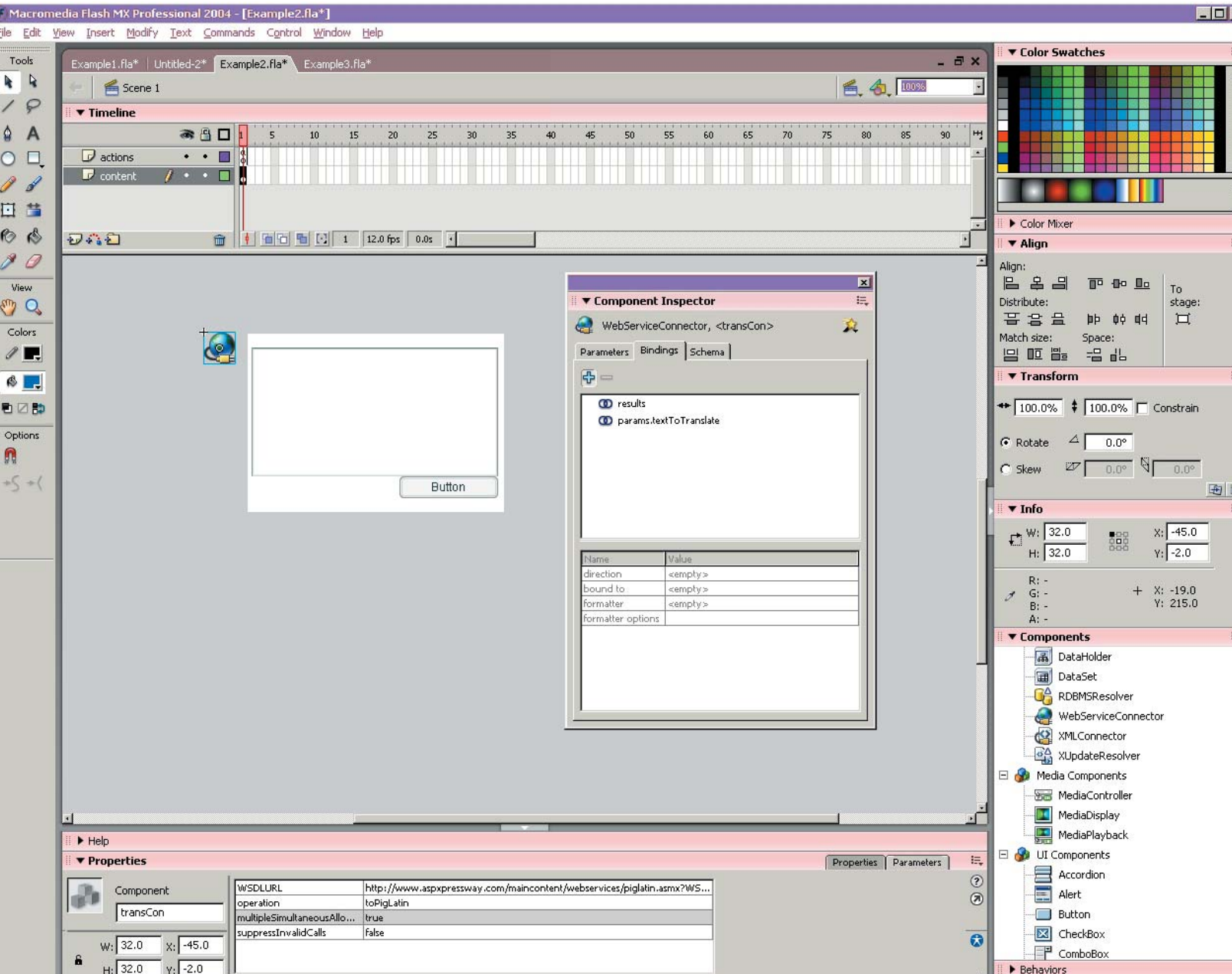


image VII



A new tool for MX professional developers and designers...



ADVERTISE

Contact: Robyn Forma
robyn@sys-con.com
(201) 802-3022
for details on rates
and programs

SUBSCRIBE

[www.sys-con.com/
mx/subscription.cfm](http://www.sys-con.com/mx/subscription.cfm)
1 (888) 303-5282

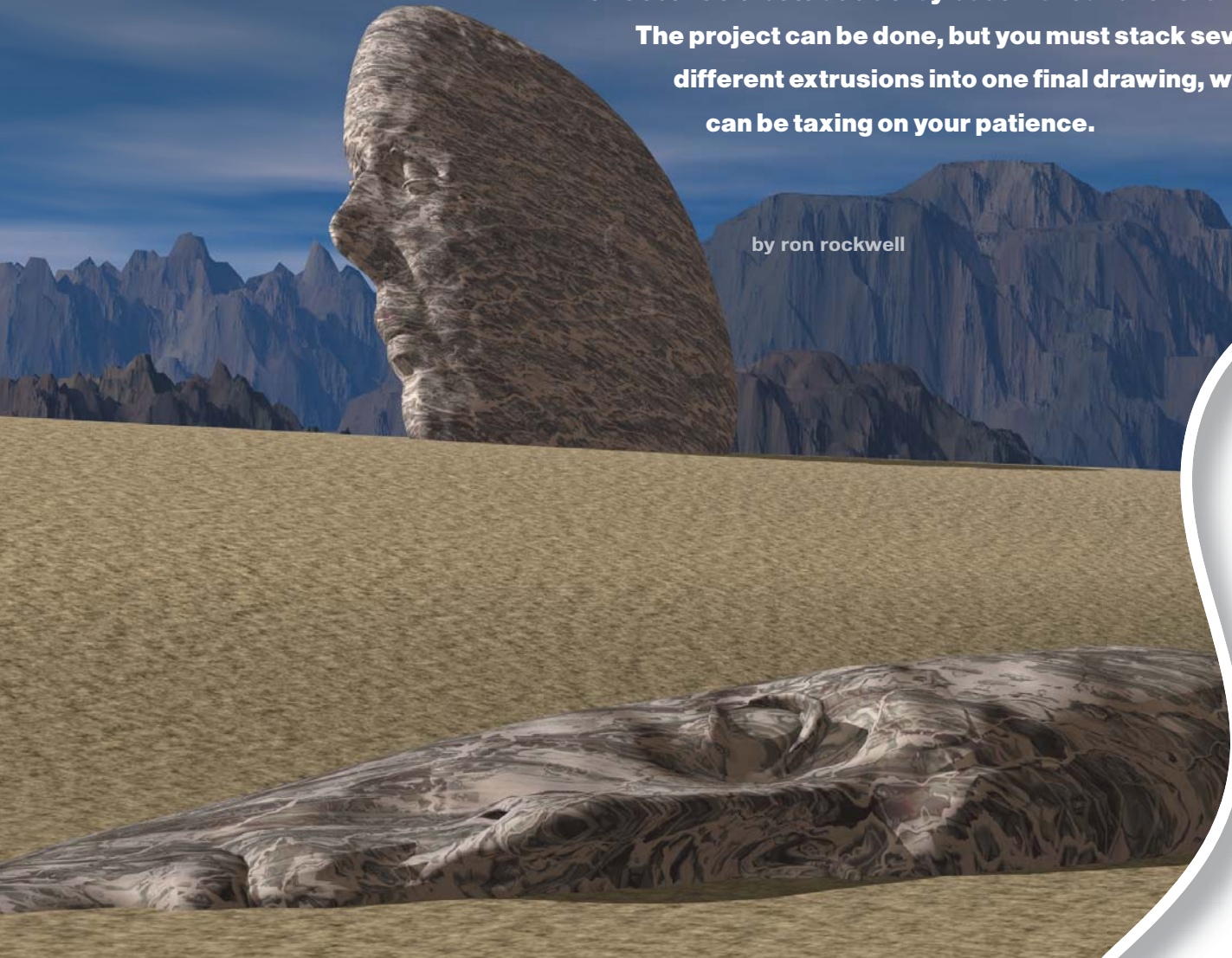


FORM



In the last issue of MXDJ (Vol. 2, issue 7), we discussed the Extrusion tool. This time we'll explore other drawing methods. Some modeling effects can be handled quickly in FreeHand with the Extrude tool, but gradient fills and creative blends from one shape to another are sometimes the best bet. The Extrude tool does a super job with text, but it has severe restrictions when it comes to compound shapes, such as the lumps and facets on ketchup or Coca-Cola bottles as they become round for the label. The project can be done, but you must stack several different extrusions into one final drawing, which can be taxing on your patience.

by ron rockwell



FEATURE

PART 2

the drawing to Flash or Fireworks for use on the Web, and then again, the end result might be in a printing project. I'll provide a few basic modeling techniques and explore FreeHand artwork as it impacts other programs and the printing process.

Blending Objects to Create Form

The ability to blend or morph one object into another has been with us for a long time in FreeHand. Just about every tutorial or lesson about blends will give an example of a star or other geometric shape being morphed into another shape or a letterform. That's all well and good for a demonstration, but hardly representative of what you can do with the tool. Frankly, I haven't seen the need for too many star-to-square blends in advertising, editorial, or Web illustrations.

A far more likely use of a blend is to blend from one shape in one location to a modified duplicate of the same in another location – for instance, a white star blending to a bright red star on a white background would make the star look as if it is zooming out of the background. FreeHand's ability to attach a blend to a path allows you to make that star zoom on an arc or zig-zag path.

However, another use of a blend is to add form to an object. Start by creating an object with a base color. Then create an area of highlight or shadow that blends into the base color. Inspect Image II to see how it all comes together. A simple ellipse was drawn for the sphere shape. Areas around the proposed highlights and shadows are drawn and given the base color. Smaller ellipses and shapes are made from clones of the larger shapes for the lightest highlights and

the darkest shadows. All that's left to do is to create highlight colors (usually white) and shadow colors (adding a complementary color or black to the base color). Remove all strokes and blend the shapes. The number of steps in the blend – and how visible the banding is – can be changed in the Object panel. If colors are similar, fewer steps will be created; dissimilar colors need more steps to create a gradient to the blend. The higher the number of steps in a blend, the more work the computer and printer have to do, so don't go more than 25 or 30 steps unless the rendering is extremely sensitive.

The circles and numbers on the balls aren't difficult at all. Set the number in a bold font and convert it to paths (Text > Convert to Paths). Center the number inside the circle and group them, then place the circle/number roughly where you want it to be on the ball – rotate the circle/number if you wish. Then use the Fisheye Lens Tool. Double-click the tool's icon (in the same group as the Extrusion and Perspective tools) to bring up its dialog box. Make adjustments for a 100% Convex distortion (see Image III), and drag an ellipse with the tool – over the existing ball, with the circle/number selected – just as if you were dragging another ball. When you release the Fisheye Lens Tool, the circle/number will be distorted. If the result is too large, do an Undo, resize the circle/number, and try again.

Name your colors for base and shadows. That way, you can clone a completed ball and use the Find & Replace function to create differently colored balls quickly. The stripe on the 11 ball is nothing more than a clone of the red ball, with a lighter base color (6% Cyan, 6%

Before extrusions found their way into FreeHand, we did it the old fashioned way by plotting points and paths and filling them with blends and gradient fills. To become proficient in realistic

2D rendering, you must learn to think differently about what you see. Once you see how highlights and shadows are formed on an object, it's a simple matter of recreating the effect with vector objects. An example of that style of thinking is shown in Image I, in which blends, gradients, and other vector effects were used.

Beyond rendering, FreeHand MX also gave us bitmap effects to aid with finishing touches. Like cayenne pepper, a little goes a long way, but definitely kicks the end result up a notch. Drop shadows, glows, transparency, and bevels certainly add a lot to a drawing, but careless application can render the job unprintable or amateurish.

Many factors can influence your choice of drawing methods. For instance, file size may be important to you in one job, while a crisp, sharp image is of utmost significance in your mind for another project. You may be exporting



Black). The shadow is a little darker. An ellipse was drawn for the top edge of the stripe, cloned, and moved down for the bottom of the stripe. The ellipses were split at their left and right apexes and connected/closed. That shape was used to crop the stripe out of the white ball. Note that the 8 ball is not simply black; it is 79C 73M 60Y 83K to be able to show reflections, shadows, and highlights.

Reflections are simple clones of adjacent balls that have been reduced (scaled down) and had the Fisheye Lens Tool applied. Multiple reflections on a given ball were done all at one time. The reflection of the handsome artist in the 8 ball is a jpeg that has been traced with the Trace tool (an 8-color trace), and fish-eyed as well. All of the reflections were

given a Transparency effect from the Object panel. That adds the color of the reflected image to the base, shadow, and highlight areas all at once.

Gradient Fills

As the name implies, a gradient fill is a fill that has a gradation of colors. To apply a gradient fill to a selected object, it's only necessary to click the Fill item in the Object panel or, if the object has no fill, click the Add Fill button at the top of the Object panel. The FreeHand default is a Basic fill, which means a solid color of your choosing. For a gradient fill, click the Fill Type drop-down menu and choose Gradient from the list. The Object panel changes its appearance to show all the adjustments you can make in the gradi-

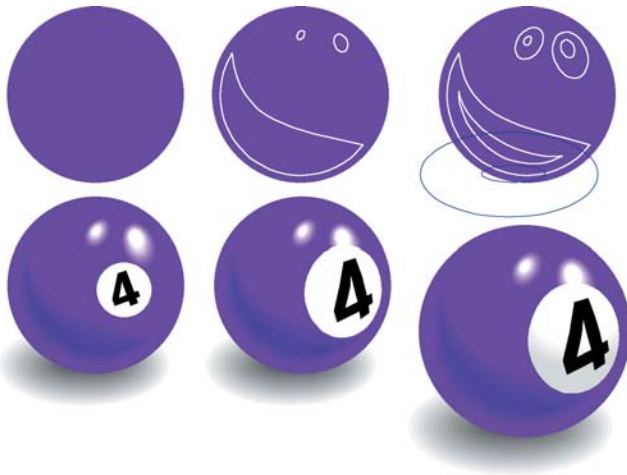
ent, as seen in Image IV.

Gradient types are:

- Linear: is the default and applies a smooth transition in color from one side of the fill to the other. It is the workhorse and most used gradient.
- Logarithmic: similar to linear, only instead of a straight-line transition, a logarithmic formula is applied that causes a more abrupt – or gradual – change in color, depending on how it's applied.
- Radial: gradates from the center to the outer edge of the fill. It's very useful for adding tone to spheres.
- Contour: similar to a radial gradient, except, instead of radiating in a circular pattern, the gradation works from the outer perimeter of the object and fol-

image I





lows that shape inward. It's good for a domed or smooth beveled appearance.

- Rectangle: applies a mirrored linear gradient along two axes.
- Conical gradients: creates cone modeling, but, if used creatively, can give even flat objects form and tone.

Image V compares the various gradients, color ramps, and the positions of the gradient handles in one example of each gradient. The Linear and Logarithmic gradients share the same color ramp – notice how differently the gradation is on each of the objects.

Each gradient has a single or double gradient handle. To see the gradient handles, you must first select the object, then click the Fill item in the Object panel. The Pointer tool or Subselect tool must be active in order to see or use the gradient handles. The central point of the gradient is marked by a small circle on one end of a dashed line, ending in a larger black square. The square may be moved in any direction and stretched to any length. The gradation will comply to the adjustment. By selecting the central point, you can move the gradient to any location.

Double gradient handles are independent from each other and can be maneuvered to develop many types of gradients. Colors are introduced to the ramp by dragging and dropping a color swatch from the Swatches panel, the Mixer/Tints panels, or a color selected with the Eyedropper tool. Move the color boxes left or right to adjust their position in the gradient. To remove a color, simply drag the color box off the ramp.

If you have a set of colors arranged in a gradient, you can drag the fill onto the Styles panel. Later, you can apply the style to a different object, then change the gradient type.

Beyond six different types of gradients and countless variations with color placement and control handles, there are four more options to fine-tune your gradient fill.

- Normal: provides the types of gradients as seen in Image V.
- Repeat: allows you to enter a number of iterations of the gradient. The gradient fill area will then be divided by your number, and equally-sized gradients will fill the area.
- Reflect: runs the gradient to its end – with the number of iterations you input – then repeats itself in the reverse direction.
- Auto Size: creates the gradient as usual, but doesn't provide control handles.

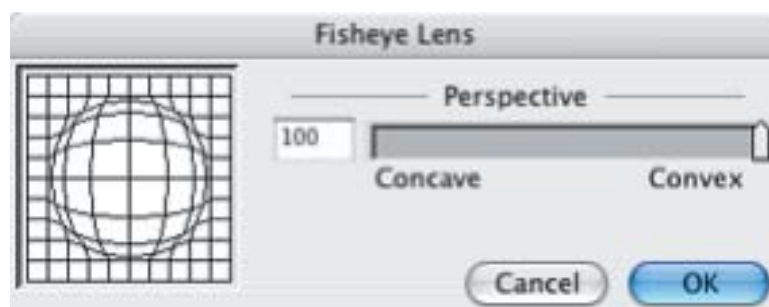
A great way to make a multicolored gradient is to make a simple gradient in one object, and have another, similar gradient in a different object with a different color scheme. Blending between the two

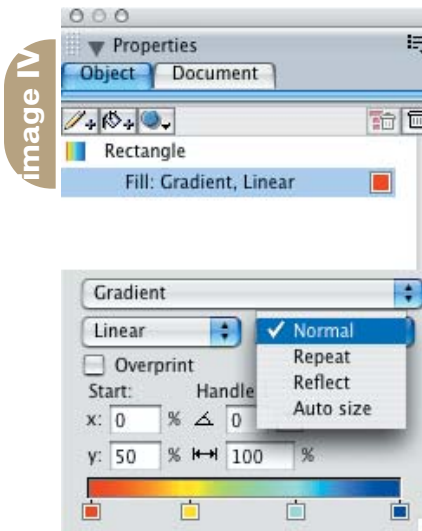
objects creates a blend of the gradients. As you know, blends can be attached to paths, but with a blend of gradients this technique quickly falls apart.

Xtra Effects

Several years ago, FreeHand introduced Xtras that would apply a shadow, embossing, blend, or smudge. These effects are shown in Image VI. They're really easy to use, but are not in the same league as the Bitmap Effects you'll find in the Object panel. These effects are pure vector so you don't have to worry about document resolution, thereby eliminating possible printing problems. With planning, the effects provide exactly the 3D effect you're after. Each of these Xtras has a button you can drag to your toolbar if you use them enough (Windows > Toolbars > Customize > Xtras). All but the Blend have dialog boxes that give you a few options.

- Emboss Xtra: gives you the choice of a hard or soft embossing. You can choose the offset and the amount of highlight or shadow, expressed in percentage of a tint of the object's color.
- Smudge Xtra: allows you to select the end fill and/or stroke colors. When you select this Xtra, the cursor changes to a pair of fingers. As you drag, a keyline preview shows where the smudge will stop. Several iterations of the original object are blended from the original colors to the colors you've input – which should be your background color in most instances.
- Shadow Xtra: gives you the choice of having a hard or soft edge, as well as a zoom effect. You can input the offset and ending colors or tints for each effect, as well as the size of the ultimate shadow.
- Blend Xtra: can also be achieved by choosing Blend from the Modify > Combine menu. This blend feature is





different from the Blend Tool in the main Toolbox, as you simply select two or more objects and click the Blend Xtra button to create the blend. With the Blend Tool, you select one object

and drag to select subsequent objects. If you create a blend with the Xtra, you can modify the connection points by clicking on the Blend Tool.

Bitmap Effects

In direct contrast to the Xtra effects above, bitmap effects for shadows, glows, embossing, and so on are found in the Object panel. However, there's a huge "but" in using them. First of all, the effects are not vector, they're raster or bitmap objects consisting of dots or pixels. Next, these bitmaps are RGB. That means they'll look fine on-screen, but in order to print they must be converted to CMYK. Any spot colors you're using in the object will also be converted to RGB, then reconverted to CMYK when printing. The issue isn't too complicated because the program takes care of the color space conversion, but it is your

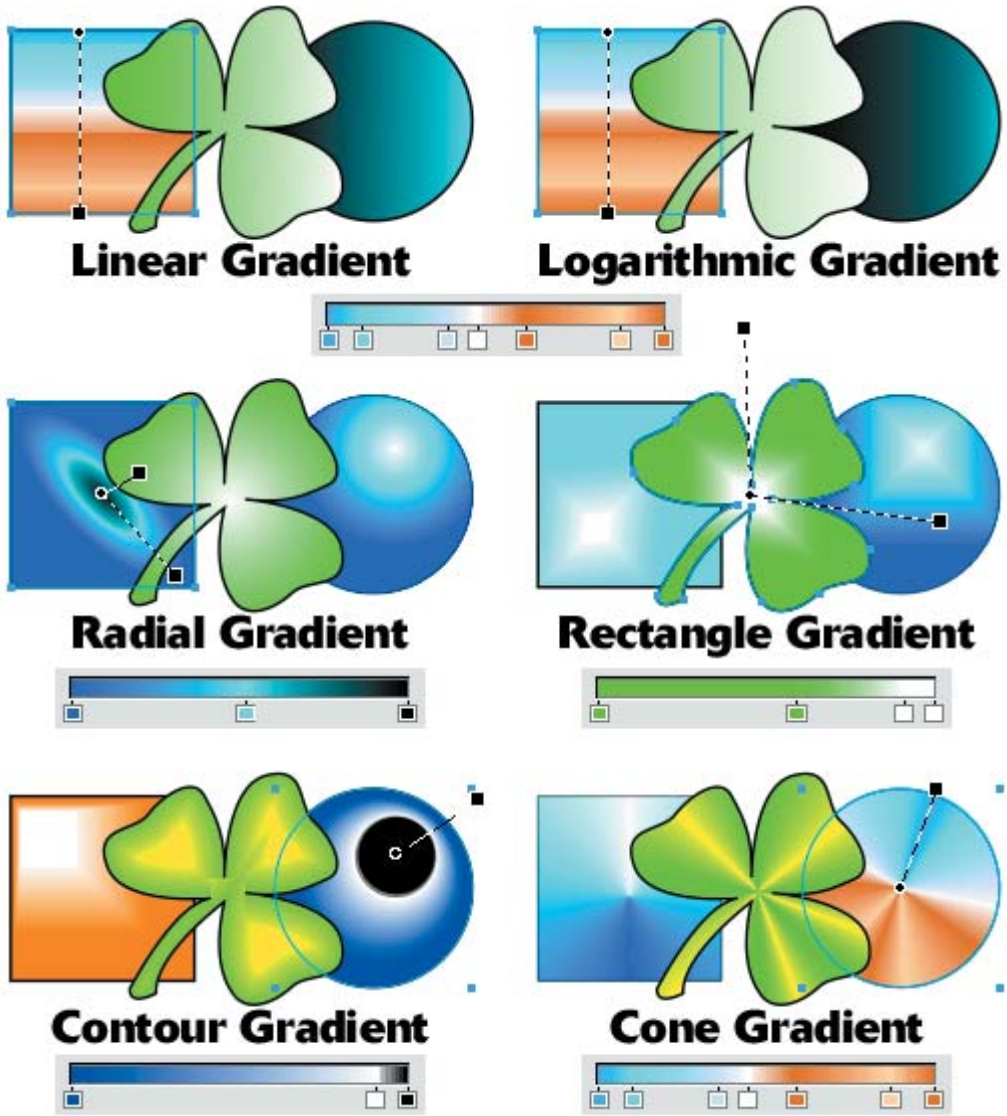
responsibility to assure that the document's resolution is set correctly. A vector document will print sharply at any output size. A bitmap will also print at any size, but its sharpness depends on the resolution that has been applied. It's extremely important if you're going to use bitmap effects that you change the resolution of your document before committing to print. That applies if you're just printing to your color laser or going to a commercial printer. It's not difficult. With the object selected, go to the Document panel and click on its options icon in the top-right corner of the panel. Drag down to Raster Effects Resolution and release the mouse. A dialog box will open set to the default of 72 ppi (pixels per inch). Click on the drop-down menu and change the resolution to 300 ppi. Now you're ready to print.

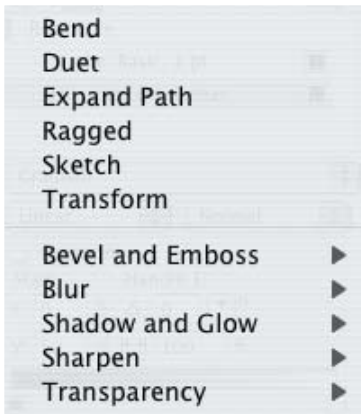
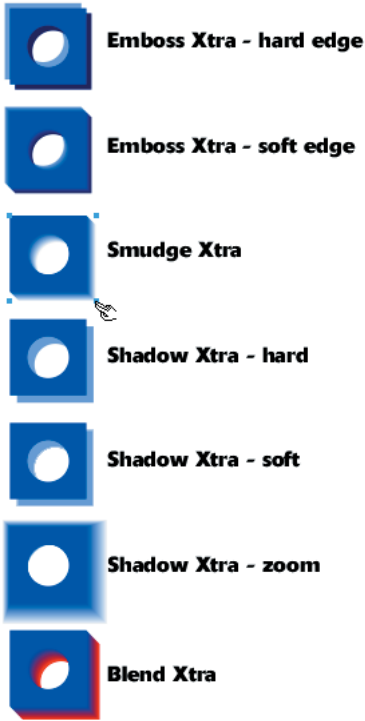
On the other hand, if you're taking the same document to the Web, set the Raster Effects Resolution to 72 ppi. You can go to File > Document Settings > Raster Effects Resolution and make the resolution changes for all raster objects in the document. This approach gives you the opportunity to select Optimal CMYK Rendering, which bypasses your color management settings. ALL of the colors in your document must be CMYK. You cannot have spot, or PMS, colors in the document – convert them to CMYK before selecting this option.

A further caveat is that when you do pop the resolution up, your screen redraw speed will slow to a crawl, especially if you have the Redraw Preferences set to a high quality. Again, it's not a big deal. Just work with the document set at 72 ppi. When you go to print, switch to 300 ppi. The effects will also impact your printing time. Oh yeah, prepare to wait if you have a lot of effects piled on top of each other. When you think about how the program has to interpret one effect on top of another, on top of another, it's a wonder that anything prints at all. And sometimes it doesn't. I've gotten carried away on a number of occasions and created a job that choked the printer. Then, it's a matter of picking and choosing the necessary effects and trashing the rest.

As a fairly quick solution to the problem – if it's a problem – you can drag a selection around the drawing and select

image V





Modify > Convert to Image. Be extremely conscious of what you're doing in this step, because whatever is selected will be converted into a TIFF file – a bitmap. No more vectors! A small dialog box allows you to choose the resolution and the amount of anti-aliasing you want. Make your selections, click the OK button, and you have a raster image instead of vectors. Depending on the complexity of the drawing, you'll have a shorter or longer wait for the conversion. It's important to know that you're basically getting a picture of what's on the screen within the bounding box of your selection. For instance, say you have a circular shape selected with text running up next to the text. If you Convert to Image, the portion of text that is within the circle's bounding box will be part of the new TIFF file.

Applying Effects

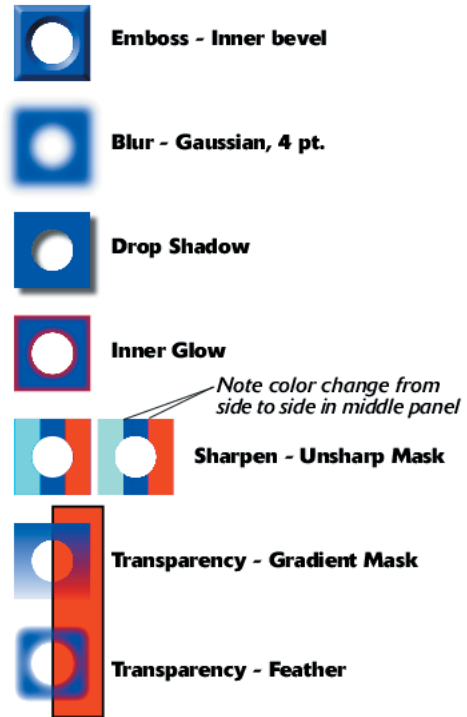
With all the scary stuff behind you, it's a simple process to add a bitmap effect to an object. Just select the vector or bitmap object, or text, and click the Add Effect button in the Object panel to access the drop-down menu. Choose the effect (shown in Image VII) you want and make adjustments to it in the Object panel. The panel configurations for the Emboss and Drop Shadow examples in Image VIII are shown in Image IX. The effect you want is in your head and it's up to you to change the various options to create that effect.

Bevel and Emboss can give you an infinite number of inner bevels, outer bevels, inset embossments, and raised embossments. It's a good effect to use for a quick button shape, although you'll be more excited by what Fireworks can do for you when it comes to Web buttons.

Blur gives your object a soft focus. There are two flavors: Basic, which makes the object fuzzy in a 1- to-10-pixel radius that you input; and Gaussian, which gives a foggy appearance. At higher levels, it can make an object appear as if it were a wisp of fog.

Shadow and Glow come in two flavors each. You have your standard Drop Shadow and an Inner Shadow. Both can be adjusted as to color, width, angle of displacement, and level of transparency. The glows are Inner and Glow (which is outer), and have the same adjustments as the shadows, except displacement occurs radially from the center of the object, following the object's outline. The Glows could be described as fuzzy inset paths, with negative numbers outlining an object and positive numbers creating an inline.

Sharpen and Unsharp Mask are usually seen in bitmap applications, but FreeHand uses the same techniques on vectors. The Sharpen effect increases contrast in a graphic, creating crisper edges between color changes. There are three types of Transparency: *Basic*, in which you control how opaque or transparent a graphic is (and therefore how much of an underlying object shows through); *Feather transparency*, which allows the center of an object to be solid and the edges to become transparent to the degree you set in the Object panel; and *Gradient Mask*, where transparency is determined using the same gradient fills

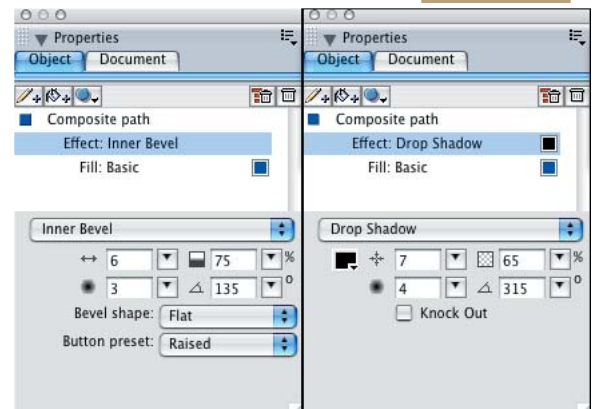


as you would use in a nontransparent object, except the object is more or less transparent.

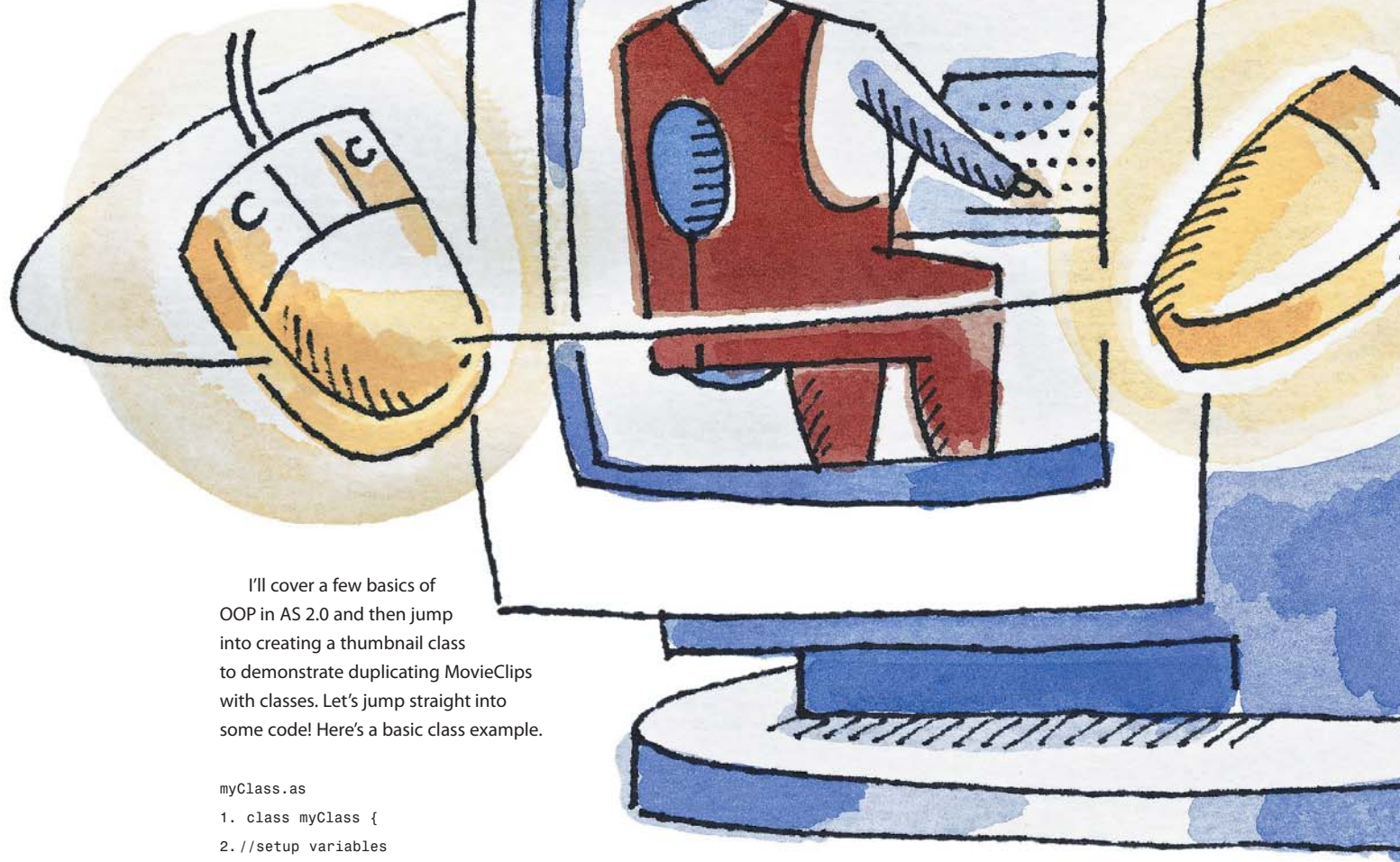
Using any of these effects can instill a high level of believability in your drawing. Just remember to keep it as simple as you can so you don't create an output problem.

Summary

In this second of two parts about creating form with FreeHand MX's tools, it's easy to see how simple it is to create depth and reality in a drawing. Drawing techniques can be simple blends, gradients, Xtras, or many other powerful tools and features in the program. You're only limited by your own imagination and willingness to explore and practice different approaches. ☁



tend to keep myself available to help people with code and Web projects on a daily basis. It seems to be quite fruitful. The majority of the questions tend to refer to Actionscript 2.0 and/or object-oriented programming (OOP) in Actionscript 2.0. It has become the inspiration for this article.



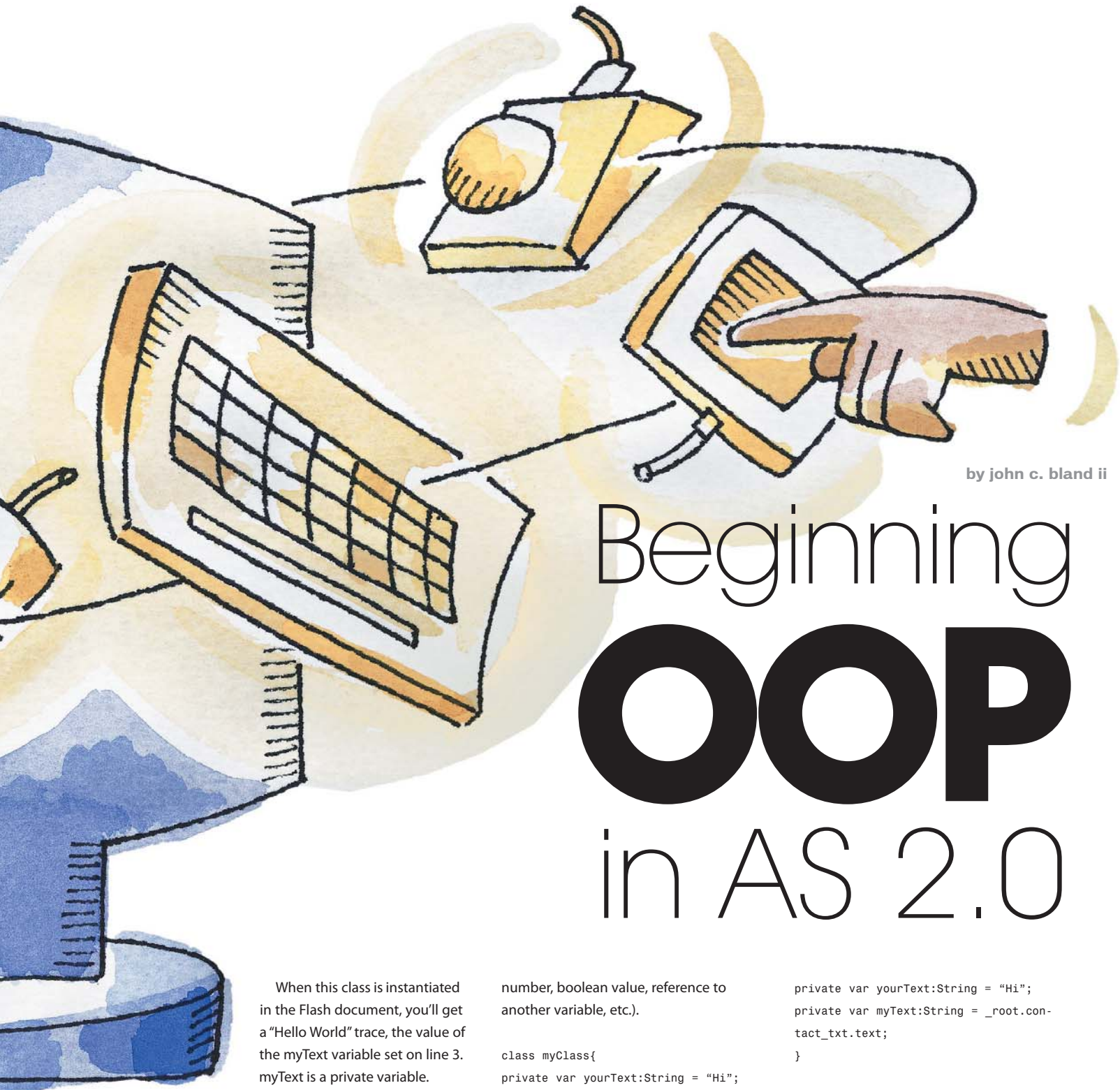
I'll cover a few basics of OOP in AS 2.0 and then jump into creating a thumbnail class to demonstrate duplicating MovieClips with classes. Let's jump straight into some code! Here's a basic class example.

```
myClass.as
1. class myClass {
2. //setup variables
3. private var myText:String = "Hello
world";
4.
5. //constructor function
6. function myClass(){
7. init();
8. }
9.
10. private function init(){
```

```
11. trace(myText);
12. }
13. }
```

Save the script as myClass.as. The file name must equal the name of the class. Jumping forward to line 6, you see the constructor function. A constructor func-

tion is called automatically when the class is instantiated. A constructor function isn't required; the compiler will create one automatically if one is not declared. I use constructors only when I need something automatic to happen. In this case, I'll have the constructor call an init() method, which handles the initial setup.



by john c. bland ii

Beginning **OOP** in AS 2.0

When this class is instantiated in the Flash document, you'll get a "Hello World" trace, the value of the myText variable set on line 3. myText is a private variable.

Private variables are only available by the class in which they are created. If the variable is needed outside of the class, you could set it to public. One important thing to remember when declaring class variables is that the initial value must be a constant. You don't have to set an initial value, but if you do, make sure it's a constant value (a text string,

number, boolean value, reference to another variable, etc.).

```
class myClass{  
private var yourText:String = "Hi";  
private var myText:String = yourText;  
}
```

This is fine because the variable yourText is set to the constant "Hi". The variable myText is set to yourText, a constant, so myText is a constant.

```
Class myClass{
```

```
private var yourText:String = "Hi";  
private var myText:String = _root.con-  
tact_txt.text;  
}
```

You will see in the following error: "A class' instance variables may only be initialized to compile-time constant expressions." yourText is fine but myText is now set to a textfield named contact_txt on the _root.

Let's integrate the example into a Flash document. It's very easy to integrate this class. Create an empty Flash document and put this on frame 1:



```
myClassExample.fla
1. import myClass;
2. var blah:myClass = new myClass();
```

Test the file and you'll see "Hello World" in the Output panel. As soon as the class is instantiated, the `init()` function traces the `myText` variable. This class isn't very useful, but it's a start. Let's move into a more in-depth example.

In the next example, we'll create a class to handle e-mail validation. There are many ways to do e-mail validation and I'm not proclaiming this as "the" way to do it. It's merely an example.

```
Validator.as
1. class Validator{
2. public function
email(email:String):Boolean{
3. var dot = 0;
4. var ext =
```



```
5. var at = email.indexOf("@",0);
6.
7. if(at > 0){
8. dot = email.indexOf(".", at+2);
9. ext = email.substr(dot+1,
email.length);
10. ext = ext.length;
```

```
11. }
12.
13. if((email.length == 0) || (at == 0
|| dot == 0) || (ext < 2)){
14. return false;
15. }else if((at > 0) && (dot > 0) &&
(ext > 1)){
16. return true;
17. }
18. }
19. }
```

```
validateEmail.fla
1. // frame 1
2. import Validator;
3. var blah:Validator = new
Validator();
4.
5.
trace(blah.email("john@jdevinc.com"));
//traces: true
6. trace(blah.email("john"));
//traces: false
```

`Validator.as` is a simple class with one public function: `email()`. The function accepts one string named `email` and returns a Boolean specified by `:Boolean` after the closing parenthesis in the method signature. Lines 3 through 5 are simple but crucial. Notice the word `var` before `dot`, `ext`, and `at`. `var` makes the variables local to the method. They're local – think of local as temporary – so they aren't a part of the class, meaning they can't be accessed outside of the `email()` method.

Without `var`, you'd have to specify `dot`, `ext`, and `at` as class variables (see `myText` variable in the example). This gave me a world of trouble when I first began developing AS 2.0 classes.

Everything after the variable declarations is a simple validation script. It checks for an "@" symbol, a period, and an extension (`com`, `org`, `net`, etc). Based on what it finds, it returns `true` or `false`.

Now you have an e-mail validation class. Test the `test.fla` file and you will see the results in the results pane.

Classes are also excellent to use when duplicating objects on the stage. Without using a class, you'd attach or duplicate movie or on the stage, set properties for each (could use a loop here), and set up certain methods (`onRelease`, `onPress`, etc). It's very sloppy and not very controllable from the standpoint of reusable code. Let's

explore how using classes can improve your workflow and help structure your application or site with a duplication example.

Photo galleries seem to be pretty common these days. We won't create a complete gallery. Our focus is on duplicating `MovieClips` that can be used to create a complete photo gallery. Pop into Flash and let's get started.

```
thumbnail.as
1. dynamic class thumbnail extends
MovieClip{
2. private var path:String =
"images/";
3. private var file:String;
4. private var info:String;
5. private var image_mc:MovieClip;
6.
7. function thumbnail(){
8. init();
9. }
10.
11. private function init():Void{
12. image_mc =
createEmptyMovieClip("img", 100);
13. this.info_txt.text = info;
14.
15. image_mc.loadMovie(path + file);
16. onRelease = release;
17. //onPress = release;
18. }
19.
20. private function release():Void{
21. trace(info);
22. }
23. }
```

This class file is pretty simple. The key notes here are `dynamic` and `this.info_txt`. Because the class is attached to a `MovieClip` in the Library, making the class `dynamic` allows you to access the objects inside the `MovieClip` without specifying them as class vars. You can think of this as putting the code directly inside the `MovieClip`. If I took `dynamic` off, I would have to target the `MovieClip` timeline (`_root`, `_level0`, etc.), then access `info_txt` – in this case, I would definitely utilize a class var.

The `path` variable holds the path of the image. This too can be set up as `dynamic`, in case you want to load images from multiple locations. For the sake of simplicity, we'll just load all images from the "images" folder. `file` is the actual filename and `info` is the text label for the image. Upon class

instantiation, `init()` is called in the `thumbnail()` constructor on line 8. `init()` does all the work. On line 12, we create an empty `MovieClip` and save a reference in the `image_mc` var. `image_mc`, which is only a variable. This is not an actual `MovieClip` – rather, it holds the reference to the `MovieClip`. Line 13 sets the text of `info_txt` textfield, which is already present in the `MovieClip`. We then load the image into the `MovieClip` referenced in `images_mc` variable and set the `onRelease` and `onPress` methods to the private function `release`. By setting the `onRelease` method to the `release` class method, we are basically saying “Call `release` whenever an `onRelease` happens.” I commented out `onPress`, line 17, to show you how multiple methods can call one function.

This class alone is capable of handling a full-fledged gallery. In fact, I used this class, with a few modifications, to resize the images on a photo gallery currently being used by a Fortune 200 company. Now let’s put this class to use.

Open a new Flash document and do the following:

1. Draw a 50x50 pixel square on the stage. This will serve as the image background.
2. Create a text field below the box, size it to the width of the box, and give it an instance name of `info_txt`.
3. Select the box and the text field.
4. Convert to a symbol by pressing F8 or on the menu select `Modify` then `Convert to Symbol`.
5. Name it `thumbnail`, give it the `MovieClip` behavior (default), and check `Export for Actionscript` (under the `Advance` options).
6. Make the Identifier `thumbnail` as well as the AS 2.0 Class to `thumbnail`. (This is the name of our class; if you name your class `myThumbnails` just change this value to `myThumbnails`.)
7. Click `OK`.
8. Delete the object from the stage. It is automatically added to your Library (press F11 or select `Window -> Library` from the menu).

Now we can get into the code for the gallery.

```
gallery.fla
1. var xVal:Number = 0;
```

```
2. var yVal:Number = 0;
4.
5. for(var i = 0; i < 21; i++){
6. attachMovie("thumbnail", "thumb" +
i, 1000-i, {_x: xVal, _y: yVal, info:
i+1, file: "pic" + (i+1) + ".jpg"});
7.
8. if(i != 0){
9. if(i%4){
10. xVal += 52;
11. }else{
12. xVal = 0;
13. yVal += 70;
14. }
15. }
16. }
```

The code is pretty short, but, by using the class, it is extensible. We start out by creating the `xVal` and `yVal` variables. These vars are critical to the layout. Next, we start looping from 0 to 20 (arbitrary number) on line 5. Line 6 attaches the `thumbnail` `MovieClip` to the stage, gives it the name “`thumb`” + `i` (where `i` is 0, 1, 2, 3 and so forth throughout the loop) sets the depth to 1000 (arbitrary number) minus `i`, and passes `MovieClip` and class properties in the `initObject`. The `initObject` is the fourth parameter encapsulated in brackets, see `MovieClip.attachMovie()` in the Flash Help documents.

When the movie is attached to the stage, the `thumbnail` class is instantiated because we set the AS 2.0 class linkage to the `thumbnail` class. What we need to concentrate on is the `initObject` parameter we pass: `{_x: xVal, _y: yVal, info: i+1, file: “pic” + (i+1) + “.jpg”}`. `_x` and `_y` correspond to the `MovieClips` `_x` and `_y` value. This is a common practice I use when using `attachMovie()`. `_x` and `_y` receive their values from `xVal` and `yVal`, respectively, which will change each loop iteration. `info` and `file` refer to the class vars we created in our `thumbnail` class. We set `info` to `i+1` to give us a number label, starting at 1 instead of 0, and `file` to “`pic`” + `(i+1)` + “.jpg” which will ultimately give us: `pic1.jpg`, `pic2.jpg`, etc. These values are saved in the class vars and utilized by the `init()` method.

Now the clip is on the stage and all values are set. The next part of the loop man-

ages the layout by changing the `xVal` and `yVal`, as necessary. The first thing we do is check to make sure `i` is not equal to 0. If `i` doesn’t equal 0, we’ll do a second check. If `i` modulo 4 is true, we increment `xVal` by 52. Modulo, from the Flash Help documents, calculates the remainder of expression1 divided by expression2. By using 52, the next `MovieClip` attached will move over one full thumbnail width and give it 2 pixels of column separation. If `i%4` is



John C. Bland
 He is CEO and chief developer for JDEV Inc.

(www.jdevinc.com), a Phoenix-based new media firm providing Internet consulting and development services for many companies nationwide. John’s strong suit is application functionality and he loves utilizing a combination of Flash, Flash Remoting, and ColdFusion to build rich Internet applications, central applications, and Web products. He credits much of his growing knowledge to the continued fellowship, support, and communication within the Flash and Multiple Users Group of Arizona and the Phoenix ColdFusion UserGroup. Look for new applications coming out around the first of the year from John and JDEV Inc. mxdj@jdevinc.com

false, we’ll start a new row by setting `xVal` back to 0 and incrementing `yVal` by 70 which gives 2 pixels of row separation.

That’s it! The loop will continue until `i` is no longer less than 20. You now have a gallery of thumbnails. Click on any one of the thumbnails and the clip’s `info` variable will be traced (see the `release` method in the class). Keep in mind this can be done several different ways and you can actually turn this into a powerful tool by adding dynamic data via XML, remoting, Web services, or any other data source. The beautiful thing here is, if you want to add an animation to the thumbnail, you simply script it in the class or edit the thumbnail in the library. Gotta love it! All examples in this article are available at www.jdevinc.com/tutorials.

Acknowledgements

Most of all I want to acknowledge God, my wife, my parents, and my brothers. Thank you for allowing me time to work! Mad Love! Big thanks to Michael Hagel, Jake Stutzman, Flash and Multimedia Users Group of Arizona (www.azflash.org), and Phoenix ColdFusion Usergroup (www.azcfug.com). ☺



Text on the Other Platform

Accent on ANSI
by james newton

Windows, Macintosh, and Unix have three different ways of referring to a line break. Windows uses two separate characters – Carriage Return followed by Line Feed – while Macintosh uses just Carriage Return and Unix uses just Line Feed. To complicate matters, the fonts generally available on the platforms are different, and accented characters are also coded differently. While Director does its best to help you cope with all of this diversity, there are times when you need to take matters into your own hands. This article will show you how.

If you need to create a cross-platform multi-user chat application, or if you need to export and import text files that can be used on either platform, then this article is for you. You'll be discovering:

- How much you can rely on Director's built-in font and character mapping

- How the various line break characters appear on "the other platform"
- How to convert files to the format used by the current platform, regardless of their origin
- How to create files that can be used by your application on either platform.

You can find a Director 8.0 cast containing a text conversion script and a wrapper script for the FileIO xtra at:

- **Windows:**
<http://nonlinear.openspark.com/tutorials/macpc.zip>
- **Macintosh:**
<http://nonlinear.openspark.com/tutorials/macpc.sit>

fontmap.txt

When you type text into a Director field or text member, Director knows the platform on which you are working.

When you copy your movie or external cast to the other platform, Director realizes that the platform has changed. Director then uses its external look-up table to convert fonts and accented characters accordingly.

This look-up table is a text file called fontmap.txt. Prior to Director MX 2004, it appeared in the same folder as the Director application itself. In Director MX 2004, you will find it in the Configuration folder. In Windows, the fontmap.txt file will open in NotePad. On a Macintosh, I recommend that you use SimpleText or a third-party application like BBEdit that allows you to retain the Carriage Return line breaks.

Mapping Fonts

The fontmap.txt file is divided into two parts. The first part allows you to define which Windows fonts to use in place of particular Macintosh fonts, and vice versa. As Image I shows, a number of common mappings are already included. You can adjust the size of the mapped fonts if you so desire.

Mapping High ANSI Characters

Computers don't know about characters. They don't understand anything more complex than zero or one. All of the characters that you can type into your computer are coded as a series of zeroes and ones. The binary number 11111111 corresponds to the decimal number 255 and for languages that use the Roman alphabet, or certain variations of it, 255 codes cover most eventualities, with the code 0 representing End Of File.

If you're working in English, you may only be concerned with the first 127 codes, which cover all the numbers, punctuation, and non-accented letters.

image I

```

Untitled: Scripts
Scripts:Movie Script 1:fontmap *
fontmap
1 Internal
27 : FONT MAPPINGS
28 :
29 : Font mappings specify which font and size substitutions to make when
30 : moving a movie from one platform to another.
31 :
32 : The format for font mapping definitions is:
33 :
34 : Platform:FontName => Platform:FontName [MAP NONE] [oldSize => newSize]
35 :
36 : Specifying MAP NONE turns off character mapping for this font.
37 : If you specify size mappings, they apply for THAT FONT ONLY.
38 :
39 : Here are some typical mappings for the standard Macintosh fonts:
40 :
41 :
42 Mac:Chicago => Win:System
43 Mac:Courier => Win:"Courier New"
44 Mac:Geneva => Win:"MS Sans Serif"
45 Mac:Helvetica => Win:Arial
46 Mac:Monaco => Win:Terminal
47 Mac:"New York" => Win:"MS Serif"
48 Mac:Symbol => Win:Symbol Map None
49 Mac:Times => Win:"Times New Roman" 14=>12 18=>14 24=>18 38=>24
50 Mac:Palatino => Win:"Times New Roman"
51 :
52 :
53 : Here are some typical mappings for the standard Windows fonts:
54 :
55 :
56 Win:Arial => Mac:Helvetica
57 Win:"Courier New" => Mac:Courier
58 Win:"Courier New" => Mac:Courier
59 Win:"MS Serif" => Mac:"New York"
60 Win:"MS Sans Serif" => Mac:Geneva
61 Win:Symbol => Mac:Symbol Map None
62 Win:System => Mac:Chicago
63 Win:Terminal => Mac:Monaco
64 Win:"Times New Roman" => Mac:"Times" 12=>14 14=>18 18=>24 24=>38
65 :
66 : Note: When mapping from Windows to Macintosh, Courier and Courier New
67 : map onto Courier. When coming back to Windows only Courier New
68 : will be used.

```




characters, as well as all the comments. If the movie uses only low ANSI characters and fonts that appear on both platforms, you can use an empty fontmap.txt file.

It's probably best to work on a copy of the original fontmap.txt. You can use the Property Inspector at the Movie tab to import a custom version of the file into your current movie (see Image V).

What fontmap.txt Cannot Do

The character mappings in the fontmap.txt file are applied only to fields and text members saved in Director casts. If you need to import or export text using the FileIO xtra, then no character mappings are applied: the text is read in exactly as it appears on the hard disk. The same is true if you need to send text from

“There are times when you need to take matters into your own hands”

one machine to another over a multiuser connection, as in a chat application.

Suppose you have a project where you need to read in external text that is likely to include high ANSI characters. How do you know if the file is in Windows or Macintosh format? How do you know whether or not the high ANSI characters need to be converted? As long as the text

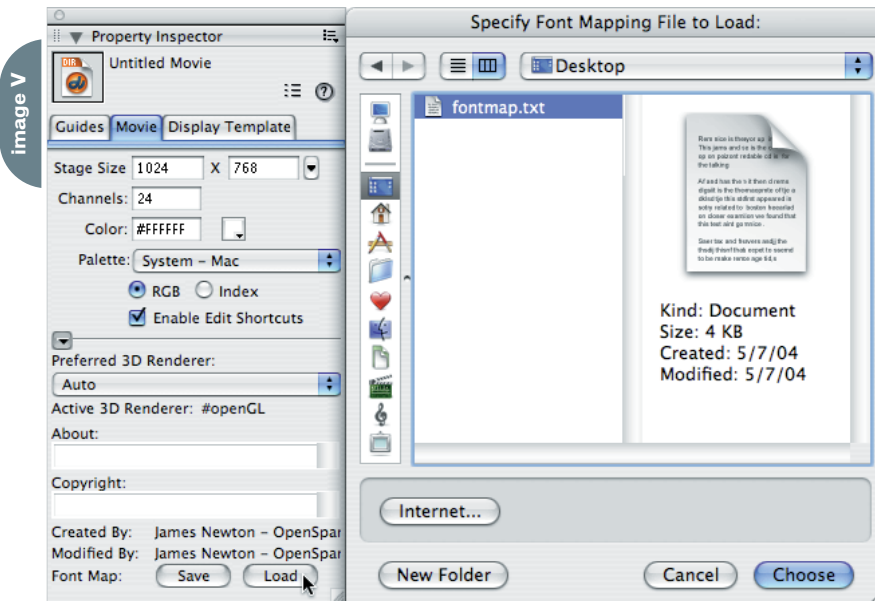
contains a line break, you should be able to make an educated guess.

Line Breaks

Director started life as a Macintosh application. As a result, it uses a single Carriage Return character – numToChar(13) – for line breaks on both Macintosh and Windows. If you write the contents of a field or text member to the hard disk using FileIO, then open the resulting file in NotePad, the line breaks will look distinctly odd, as Image VI shows.

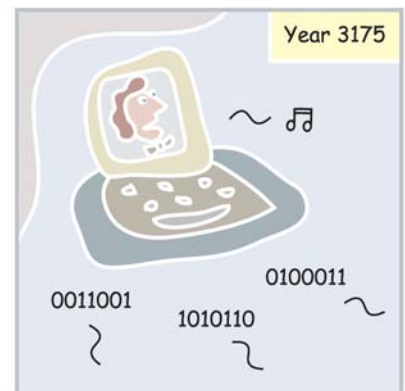
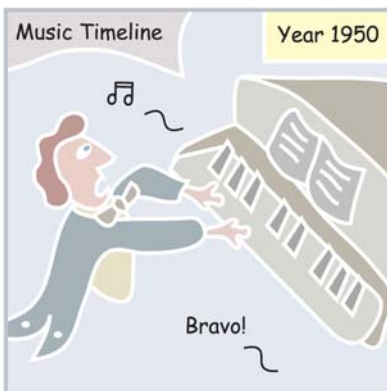
The solution is to replace all Director-style line breaks with Windows-style line breaks before you export. To do this, I use a generic ReplaceAll() handler (see code II). Image VI contains three lines that are commented out; if they were uncommented, the output would be in Macintosh format on a Macintosh and in Windows format in Windows.

The extra Line Feed character used by Windows may appear in different guises on a Macintosh, depending on the application that displays the text. In SimpleText, it appears as rectangles at the beginning of each new line. In



xile

written & illustrated by louis f. cuffari



Director, it appears as an extra blank line in both field and text members, though it may appear as a rectangle in the Cast window thumbnails, as you can see in Image VII.

Cross-Platform Text Files

So, if Windows-style line breaks look strange on a Macintosh, and the line breaks used by both Director and Macintosh misbehave on Windows, how can your application work cross-platform with text files?

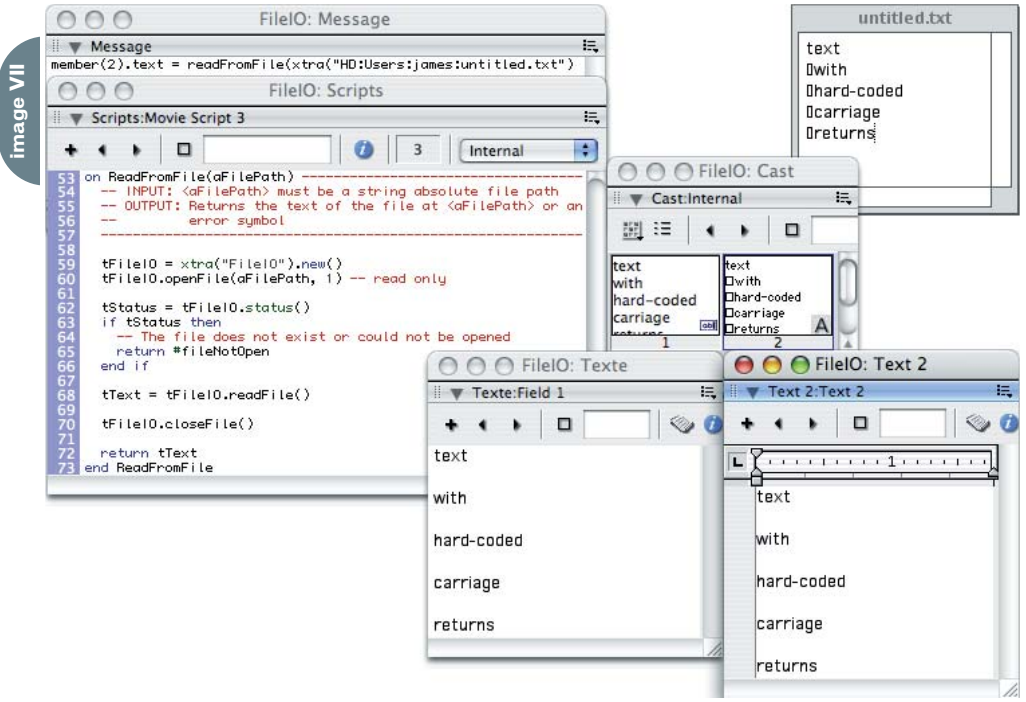
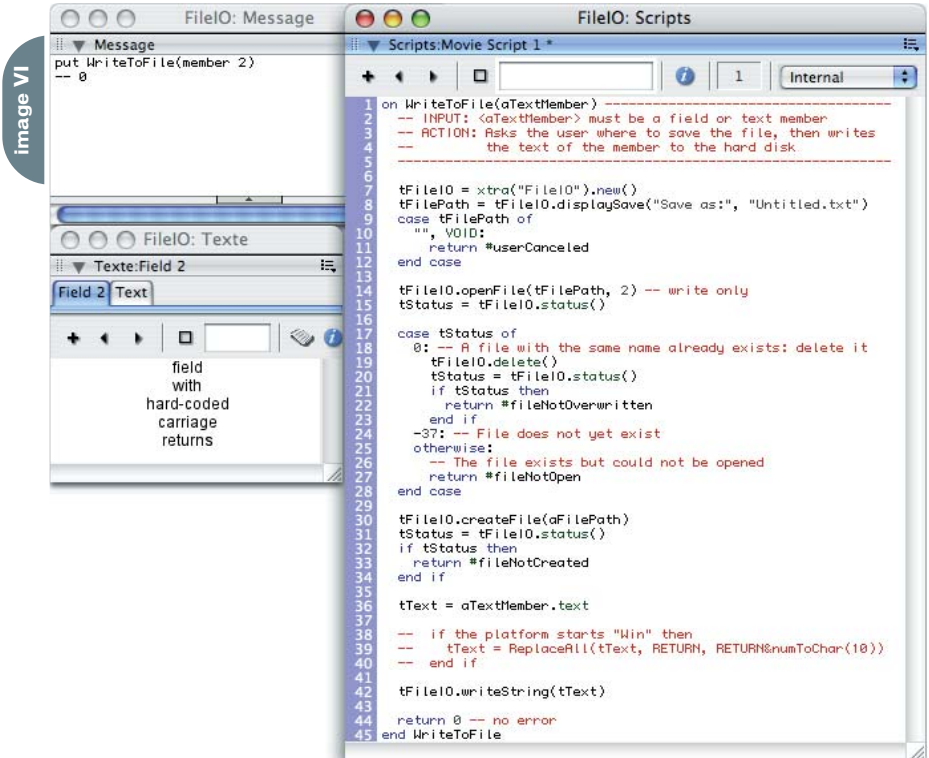
My solution is to export the files in Windows format and to check the format on import and make the necessary conversions. Checking the format means testing what line break character is used in the imported file, and using that to decide how the high ANSI characters have been encoded (see code III).

Suppose the file is in Windows format, and the application is running on Windows. All that needs to be done is to remove the extra Line Feed characters, so that Director can display it correctly.

If the application is running on a Macintosh, the high ANSI characters need to be converted as well. This means that the ANSI code for every character needs to be checked, and characters whose codes are greater than 127 need to be converted. To do this, I create a Director list that contains the same information as the character mapping data in the fontmap.txt file. You might like to compare the first few entries in the tFontMap list in code IV with those in Image III.

I use a similar conversion process to convert from Macintosh format to Windows format before I save a file to disk on a Macintosh. This means that the file is converted twice on a Macintosh (once on import and once on export), while all that happens in Windows is that Line Feed characters are added or deleted.

In fact, the complete handlers that you'll find in the downloadable cast can handle conversion from Mac, Unix, or Windows formats to either Windows or Mac. The cast also includes a script containing much more robust versions of the WriteToFile() and ReadFromFile() handlers mentioned previously. These are called FileWrite() and FileRead(). Code V shows handlers that you could use for importing and exporting text cleanly on both platforms.

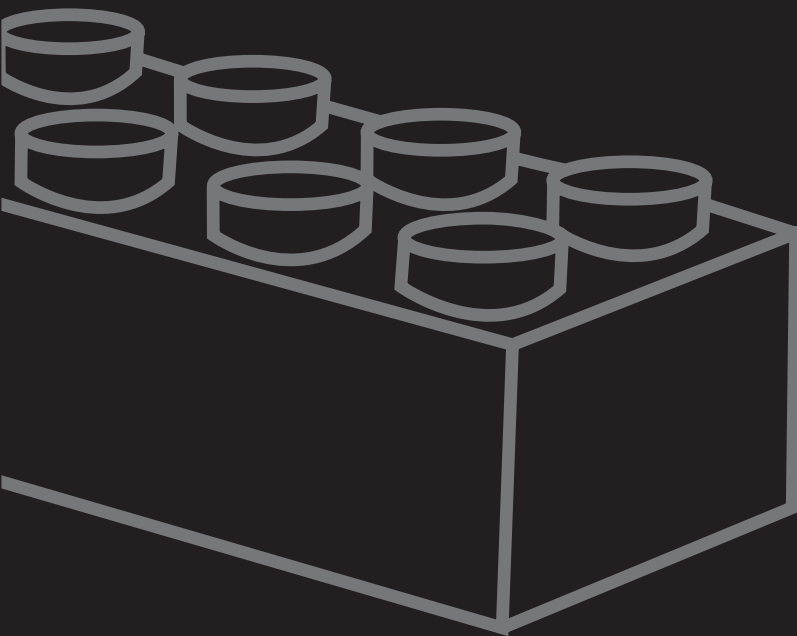


Conclusion

Director takes care of most of the platform differences between Windows and Macintosh, as far as fonts and text encoding are concerned. If you need to import text on the fly, you can mimic the techniques used by Director to convert to the appropriate format, as required.

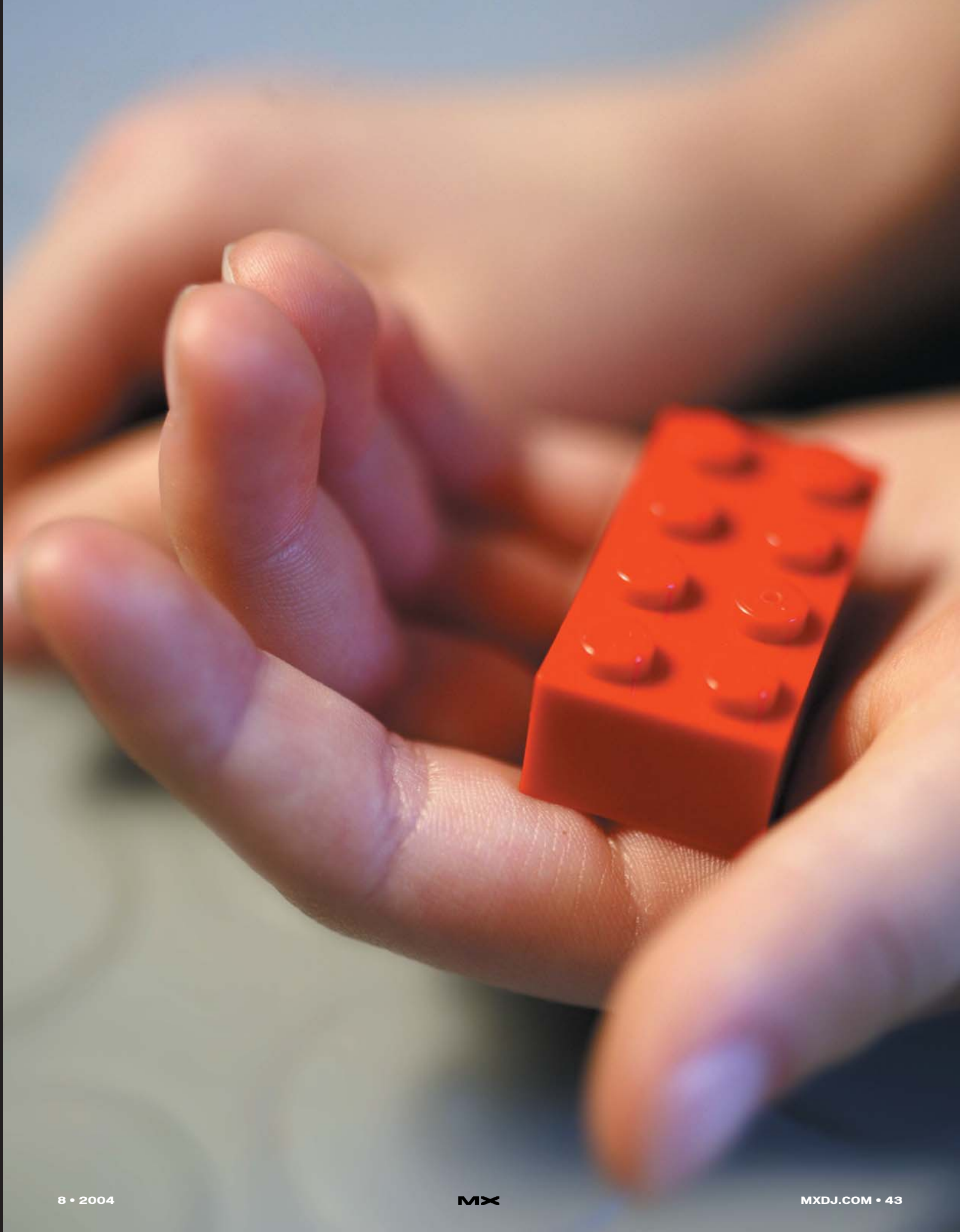


VIRTUAL SHOWROOM



Yes it's true — at some point, it was a challenge for me to swap an image on the stage. Hey, we all start somewhere. About a million challenges later, and voila — a decent Director developer!

by *lisa del padre*





Here is how it happened and some of what I've learned: I started using Director a couple of years ago when I was in college in Austria and made my first "baby steps" with it. Granted, I had (little) programming background. I created a very simple application for my niece, which let her solve calculations. Whenever she had one right, she could see a picture of herself. It wasn't anything fancy, but in developing this program, I learned the basics of Director and I gained confidence that I could do more.

After creating a few other small projects with Director, I thought I knew it all and took on a huge job during my internship. The project was called "Virtual Showroom" and the customer was LEGO, an obviously well-known toy manufacturing company.

The task at hand was to transform their paper-based retail catalog into an interactive CD-ROM-based application. The products had to be separated into categories and a menu had to be put in place to access individual categories. Also, we needed to create different product views, a thumbnail view with multiple products on one page, and a detailed product view with bigger images and more information. They also wanted to incorporate their product line video commercials.

Because my knowledge in Director was fairly basic, we had to create every

page "by hand." We took a product image, made it the correct size for both thumbnail and detailed views (we used PCT images, because this format allows for gradient alpha channels), and placed it on the Director stage. Now, multiply that by about 200 products. This doesn't seem so bad, right? Well, it soon turned out that this method wasn't very effective because product lines, images, and info constantly changed during the development process. Proofing was a nightmare. For the next version, we needed to create something more updateable and dynamic.

We are into version 5 of this project right now and it's fully updateable. It also includes product-ordering functionality, database access to client data, PowerPoint marketing presentations, videos, and lots more. I can't go into much detail about how we pulled it off, but I'll try to give you some ideas.

The heart of the program is a big list of property lists containing all of the product info. This list is dynamically generated on startup of the "Virtual Showroom" from a tab-delimited text file. As you may know, a file like this can be exported from many database-like applications, including Microsoft Excel. We chose to go with this format because the requirements for knowledge and software of the person editing the data are few. Another advantage is that products

come up in the exact order in which they are listed in the Excel file, so no numbering is needed. I have to say that this list is HUGE, but it does not bog down Director at all. Even working through the entire list to calculate totals only takes a split second.

This is a simplified version of what the list looks like:

```
dbList = [[#name: "product1", #price:
12.00], [#name: "product2", #price:
40.00]]
```

In order to access the price of product 2, I simply call this command:

```
Price = dbList [2].price
```

On to more secrets – how we are dealing with product images. Every thumbnail and detailed product view is generated dynamically. All images are available in an external cast. Using "Standard Import" when importing all images into this cast would have made the cast file size too big, so I only linked to the images. Even when linking to the images, the cast was somewhat big, because Director automatically creates thumbnail images for each imported image. To work around this problem, I wrote a routine that replaces the thumbnail image for each cast member with an image of one pixel black.

```
member(memberNum, imageCastNum).thumb-
nail = member("thumbnail").picture
```

The cast contains 800 linked images and is only about 500 kb!

For the dynamic product-page generation, I created template screens with transparent placeholder sprites for product images. Depending on the viewed product, I swapped out the placeholders with the actual product image from the image cast.

There's one last thing I want to mention here because it is a trick I've used over and over again, even for other projects. If you want to create an overlaying menu, video interface, or whatever, without using MIAW, you can try this:

Consider the fact that I have a project with some sort of content on the stage and a button that, when clicked, is supposed to display an overlaying menu.



First, I create a button behavior for the menu button that takes a screenshot of the stage and goes to the "menu" frame.

```
on mouseUp me
```

```
member ("screenshot").image = (the stage).image
go "menu"
```

```
end
```

The member "screenshot" can be any bitmap cast member. I would just import any image and name the member "screenshot." Of course, now I have to create a "menu" marker and put the "screenshot" member in the center of the stage. Create an exitFrame behavior so the playhead stays on the menu marker once it gets there. Try this and see what happens. You will see that there is a seamless transition from the actual content to the "menu" frame with the screenshot. Now you can go on to create whatever menu you would like on the "menu" frame. I hope the benefits of this process are clear: it certainly cleans up the timeline and simplifies the task of creating anything that needs to overlay the stage.

Obviously, there's a lot that I can't explain here. To make a long story short – I sweat, I swore, and I conquered.

Troubleshooting

Being able to troubleshoot is almost as important as being able to program with Director. It just so happens that there are ALWAYS last-minute problems with a project, and this is a rule with few exceptions.

As frustrating as it can be to troubleshoot, the bright side is that you'll learn a lot in trying to figure out the problem. And if everything you try fails,



there's always Macromedia Tech Support to help out. Anyway, here are some tips I can give after endless hours of troubleshooting.

Prepare Your Troubleshooting

There are a lot of ways to get to the bottom of a problem. However, making troubleshooting easier, because you know it's coming, can start at the beginning of a project.

Externalize Scripts

Whenever I start a project, one of the first things I do is create an external cast that is called "scripts.cst." Needless to say, all of my scripts are placed in this cast. Doing this brings a lot of advantages: Apart from being able to reuse scripts throughout the entire project, changes can be made very efficiently. It also saves time, because you don't need to create a

projector every time you make a change to your scripts. Also, if a customer calls with a problem, you can update your scripts cast and e-mail it to your customer (because there are only scripts in this cast it's also very small). The customer can then replace the file on his end and see the changes instantaneously.

It's also a good idea to create a stub projector. To do that, I create an empty director file and a "startMovie" script. The startMovie script looks like this:

```
On startMovie
initProject ()
end
```

I declare the initProject method on a script sheet in my external scripts cast. In this script, I initialize all necessary variables and tell the projector to go to the first movie. That way, I have all scripts "exter-

“BEING ABLE TO TROUBLESHOOT IS ALMOST AS IMPORTANT AS BEING ABLE PROGRAM WITH DIRECTOR”



nal” and I have control over almost everything without creating a new projector.

Display Full Error Text

Another preparation I make at the beginning of a project is to create an “.ini” file, which is called the same name as the projector. For example, if the project is called “test.exe” I would have a file called “test.ini” containing the following text:

```
[lingo]
DisplayFullLingoErrorText=1
```

This file ensures that the full error text is displayed when a script error occurs while running the projector. Instead of just saying “Script error” when there’s a problem, the message will give you more detailed information as to why it occurred, as you would see in the authoring environment. I find this to be very helpful.

Verify Xtra Versions

The last thing I want to mention here is about Xtras. Make sure all Xtras are the

correct version for the given Director version. If this isn’t the case, it can create all sorts of bad (but sometimes subtle) problems. A typical case of this would be when everything seems to work fine in the authoring environment, but not when you run the projector.

What to Do When a Customer Calls With a Problem

Just when you think you’re done with a project, there are those painful calls – the customer says that this or that isn’t working, the project crashes, it doesn’t run smoothly enough – and YOU have to take care of it FAST. Here is how you prepare your troubleshooting process:

Get All the Info You Can Get

It’s very important that the customer helps you in trying to narrow down the scenario that is creating the problem. If you know that the problem always occurs when you hit a specific button, you’re already halfway there. Have him/her send screenshots of any error messages (this is where the previously mentioned .ini file comes into play!).

Before you start going crazy trying to figure out the problem, make sure that the customer is running the project on an “allowed” system. Also, have him/her explain exactly which actions were taken before the problem occurred. For example, “First, I clicked on a product, then I opened the menu, etc.,” just make sure he or she isn’t trying to do something he or she isn’t supposed to be doing. For example, I had case where the customer was trying to run two instances of the projector at the same time, which would obviously create problems. If you think the problem might be due to a corrupt file, send another copy, or e-mail/FTP the file in question for the customer to download.

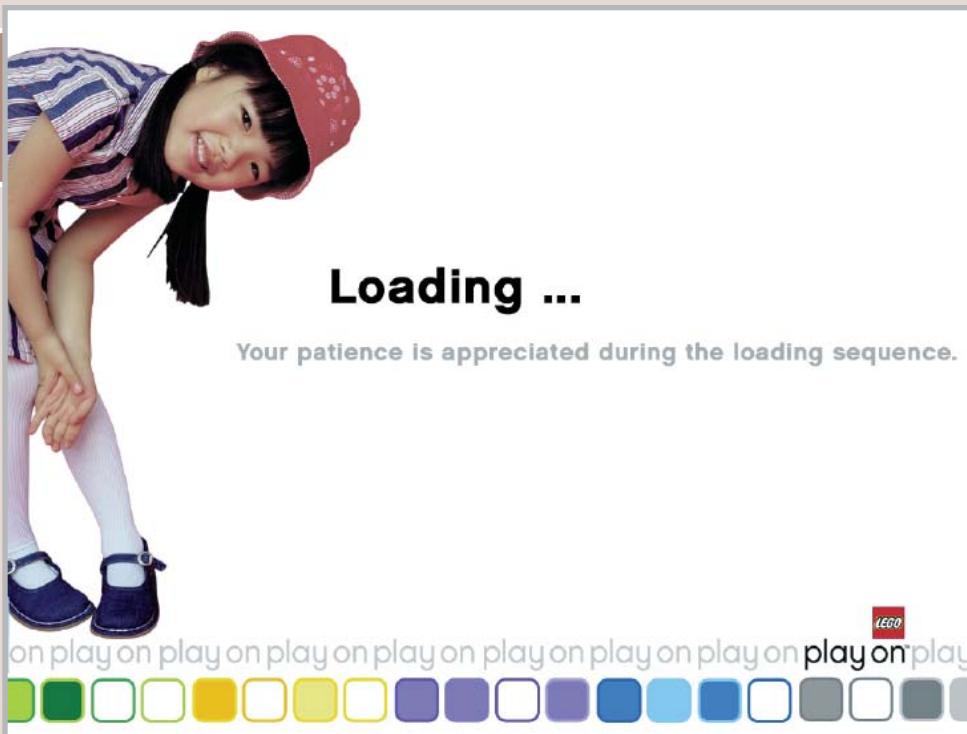
Verify the Problem

Once you have as much info as you can get, the challenge can begin. The first step you should take is to verify the problem on your end. If you have a system setup that’s similar to your client’s system, you should be able to recreate the problem. This will speed up the troubleshooting process tremendously because you’ll need less input from your client in testing the possible solutions.

image III



image IV



Use Your Resources

The second step I usually take is searching the Tech Support section on www.macromedia.com as well as their user forum. I have found answers and work-arounds for so many problems there because chances are, if you're having this problem, someone else had it before you and found a solution for it – don't re-invent the wheel, it just doesn't pay very well. Some people on these forums seem to know it all "Mark A. Boyed," whoever he is, must have posted thousands of answers. I want to thank everybody on the user forum for their valuable input to the Director community and for saving my a** over and over again.

Troubleshooting Methods

Whether or not you found answers, it's time to get your hands dirty. First thing: MAKE A BACKUP OF YOUR EXISTING VERSION and take the "Final" label off this one!! I didn't always do this, and sometimes I really regret it.

Be Creative!

If you found answers anywhere, implement and test them. If you didn't find answers anywhere, there are some things you can do before calling Tech Support. One very good trick I discovered recently is putting the following command in your "startMovie" or "initProject" script:

```
the debugPlaybackEnabled
```

This command brings up a message window during runtime and you can use it just like the one in the authoring environment. This is a tremendous help when problems only occur in projector mode or on certain operating systems. Even still, you'll have to be creative in finding the bug! Anything is possible – I have experi-

enced the oddest and most unusual problems. A command line that doesn't create problems in one spot could create problems in another for no obvious reason. Pay attention to memory usage in the task manager! Sometimes a problem causes the projector to keep eating up memory and never clear it out again, even when using unload commands. Certain script sheets can become corrupted, so copying and pasting the script into a new script sheet and deleting the old one may help – like I said, anything is possible.

Top-Down Method

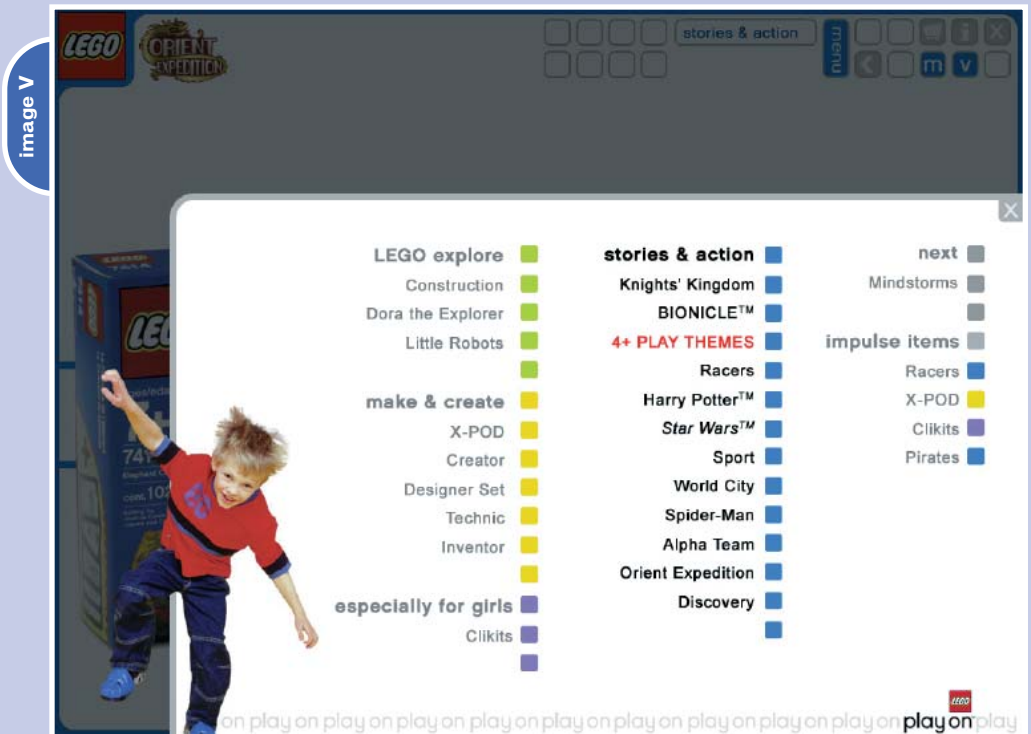
I usually start solving problems this way. What I mean by "top-down" is that you start taking stuff (script lines, images, etc.) out of the project and keep testing, testing, testing, until you don't experience the problem anymore. Of course, it's important to keep track of your changes. Whatever you changed last before the

project started working again is probably the root of the evil. If you can't take anything out, I would try finding different ways of implementing the particular parts of your project that seem to cause the problem. For instance, using an Xtra instead of built-in Director commands may help. Or maybe if using MIAW seems to be a problem, you can avoid using it using the "screenshot" method I described earlier.

Bottom-Up Method

Once you have an idea of what the problem is, it's often good to start with a blank project and recreate the problem on purpose with the simplest assets possible. At least you will verify that what you found out is true, and you can also start working on a work-around in a test bed instead of pulling apart the real project. Remember, work-arounds don't have to be elegant – as long as they work. ∞

Imported from Austria, Lisa has brought us her sophisticated sense of style and a great understanding of the medium. Adding clever functionality to sleek designs with an artistic flair and a unique perspective, Lisa has produced an heir and a lot of world-class multimedia. She is wicked smart.
lisa@delpadre.com



“REMEMBER, WORK-AROUNDS
DON'T HAVE TO BE ELEGANT
– AS LONG AS THEY WORK”



Interactive Kiosks

Maximizing the capabilities of Director
by darrel plant

most of my work is invisible. At least, it isn't seen by very many people. Much of it is business-to-business sales tools for high-end server components. Lately, a good portion of it has been in languages where the target audience is even smaller than the English-speaking market. The majority of the projects on which I work have a maximum audience of a few thousand people worldwide. The concept demos and analysis tools that have been my bread (except for the Atkins diet) and butter over the past few years may only be seen by less than a couple hundred people.

So, it was with some relish that I took on a project this past spring that had a much higher profile, despite a couple of potential drawbacks: a very short, very firm deadline and a rather limited budget. One of the reasons I was able to get the job done in the allotted time and keep it under cost (pretty much the same goal in this case), was Director's flexibility in handling various graphic and media elements, as well as its long-standing external file linkage capabilities.

Specifications

In September of 2003, Chris Williams, an interactive kiosk hardware specialist with whom I had worked on some other projects, put me in touch with the Oregon Zoo. The Zoo was investigating the possibility of creating a kiosk that incorporated a live, remote camera. The kiosk would need to do more than just show video – it would also display information about an animal and its habitat. The design staff was aware of Director – a couple of other kiosks in the Zoo had been created with it – but they weren't very familiar with it. They wanted to verify that the hardware they were considering for the remote cameras could be dis-

played in Director.

I was reasonably certain Director could take on the task. Several different video formats are supported by built-in and third-party Xtras (the Director components that extend its capabilities). My own personal preference for video within Director is Apple QuickTime. It's the best integrated and most flexible of the supported video formats – in my experience – and it was the first one I tested on the camera vendor's test URLs. I was able to create a Shockwave movie within a couple of minutes that displayed one of their streaming video samples. A simple field and button added to the movie by the time of the initial meeting at the Zoo made it possible to link to almost any QuickTime-compatible video stream. The meeting went well, but then things were quiet for seven months.

Seven months, it so happens, was approximately the amount of time it took to complete the Oregon Zoo's new Eagle Canyon Exhibit. The outdoor installation is part of the larger Great Northwest section of the Zoo, and contains habitat areas for beaver, otters, water birds, salmon, and – of course – eagles, which were nursed back to health from injuries and couldn't survive in the wild.

I had more or less forgotten about the project, figuring the Zoo had gone with another developer, when suddenly, in the first week of May, things got moving again. Eagle Canyon was scheduled to open May 28. Media for the project was still under development and would be in my hands mid-month. While the artwork for the interface was minimal – consisting of only a few buttons and informational screens – and was being provided by the Zoo's own Design Services department, the video from the United States Department of Agriculture's Forest Service Pacific Northwest Region was still

in the digitization process.

Additionally, there were now going to be two kiosks. The kiosks would be virtually identical in function, but one would feature eagles and the other would feature salmon. The eagle kiosk would link to a camera installed in a long-time nesting location in the southern Willamette Valley. The salmon kiosk's camera was already in place in the Columbia River, in spawning grounds near Bonneville Dam.

The kiosks would run continuously, with an attract loop showing full-screen video and inviting the user to interact with the screen. When activated, the kiosk menu would link to a screen featuring five video clips (different for each kiosk), an informational screen about the USDA Forest Service NatureWatch Program, and a credits screen. The initial design document actually required two versions of each kiosk, but because the view of the empty eagle nest and spawning grounds during the winter wasn't desirable, we were now up to four individual applications: two kiosks in two modes each (see Image I).

Planning

Right away, I started looking for ways to simplify the project implementation. You might call me lazy, but I like to think of myself as efficient. By the time I had the final go-ahead on the kiosks, we were less than two weeks away from the morning when the exhibit was supposed to open.

First, I decided that a single engine would drive all versions of the kiosks. Rather than create four separate applications, I was determined to use a single engine for both the eagle and salmon kiosks, with an easily modifiable initialization file that controlled the source for the live camera, so that the Zoo's staff could turn the camera on and off easily.

This was made much easier due to the similarities between the two kiosk versions. All of the identical content – including the operational logic – was put into the main Director movie. Using an external cast file for the unique elements meant that switching from the eagle kiosk to the salmon kiosk would require the modification of only a couple of file names. An external text file would hold the initialization information.

Cutting four applications down to a single application made the project a lot more manageable and much easier to debug when we got to the final stages.

Implementation

Most of the technical and physical details of the kiosks were decided before I was brought onto the project. The human interface is a Planar PT170M touchscreen, a 17" screen running at 1280x1024 pixel resolution (the video playback was specified for full screen at that size). The computers inside the kiosks are Pentium 4 Little PCs from Stealth Computer Corp. (measuring about 10"x6"x3"). The live cameras are from VBrick Systems and transmit a QuickTime-compatible RTSP (Real-Time Streaming Protocol) signal. The two cameras are hooked into the Zoo's internal virtual private network so outsiders can't hook into the signal. Final testing of the live camera had to be done on site.

Artwork Preparation

Prior to the installation of the Eagle Canyon Exhibit kiosks, the Zoo had two other computer-driven interactive kiosks (there are, of course, many other types of interactive exhibits at the Zoo). While the Design Services department was very familiar with preparing printed materials and outdoor signage, they weren't accustomed to digital preparation standards.

I received eight Adobe Illustrator files per version/configuration combination for screen art. Due to some effects and type translation problems, I couldn't open the files with FreeHand MX, so I worked on them with Illustrator 10.

Regrettably, all of the screen art was sized to the approximate physical dimensions of the screens (13.3"x10.6") and not to 1280x1024. The proportions are close, but not as exact as you'd like when shifting between screens with corresponding

elements. Simply exporting bitmap images from the original files wouldn't work either, because they come out about one-third too small (13.3" @ 72dpi is only 957 pixels). The positioning images I created were from scaled-up versions of the screen comps.

To create the bitmap elements used in the kiosk, I used a combination of direct bitmap manipulation in Fireworks and cropping of the screen comps in Director's Paint window. Because the scaled comps weren't sized to the exact pixel dimensions of the kiosk screen, I just opened the image in Fireworks and sized it there. The video captures used as buttons on the video selection screen were obtained by duplicating the placement screen capture, then cropping the image. The background was shared between both versions of the kiosk, so its bitmap was placed in the default Internal cast of the movie. The video buttons show scenes specific to the kiosk version; they're in the external media cast file.

The only other elements created outside of Director (apart from the video) were the round button elements, which were exported from Illustrator to Photoshop format, then converted to Director-friendly PNG format in Fireworks (Director can import Photoshop PSD files, but there is a difference in how the alpha channels are built for things like drop shadows that makes PNGs preferable).

Graphics of the text for the credits and information screen information were exported in the same way and placed on the stage rather than recreating the elements inside Director from individual text and graphic elements.

The Rotis San Serif font was used throughout. All of the original artwork had been developed on a Macintosh, but the delivery platform was a Windows XP computer. When building the Director movie, I embedded the font on my own Mac so that I could use the embedded version throughout production. Type associated with buttons and messages was created using text cast members (although I used Director MX 2004 for this project, I didn't use the new Flash component cast member types, preferring to rely on that with which I was more experienced).

Initialization

One of the key items to be addressed was a simple way for the Design Services

(or Information Technology) department to modify the kiosk's live camera function. They needed to be able to turn it on and off, and to change the URL to which the kiosk connected if there was a modification in the camera setup. I decided to make it as easy as I possibly could.

In a different situation, I probably would have used some sort of formal data structure like XML for the initialization file, but due to the simplicity of the initialization data, I just used a text file delimited by line feeds as the easiest to understand. There are two lines in the file: the first contains information on the camera URL, the second contains file and timing information about the video clips.

The initialization file (called settings.txt) looks like this:

```
rtsp://128.0.0.1/video1
[#path: "salmon.mov", #clips:
[[#start: 360, #end: 4140], [#start:
4560, #end: 6900], [#start: 7420,
#end: 10200], [#start: 10740, #end:
12960], [#start: 13380, #end: 16080]]]
```

The initialization of the kiosk happens right in the startMovie handler:

```
vfile = new (xtra "fileio")
vfile.openfile (the moviepath &
"settings.txt", 1)
vsettings = vfile.readFile()
put line 1 of vsettings into field
"liveurl"
put line 2 of vsettings into field
"cannedInfo"
vfile.closeFile ()
gInfo = value (field "cannedInfo")
```

Nothing fancy. Create a FileIO Xtra instance, open the file for reading, read the entire file into a variable, then pop the two lines of the text variable into separate field cast members and close the file. The cannedInfo field is converted from pure string data into a property list and stored in the gInfo global variable.

The Dispatcher

The heart of the kiosk is its ability to do things on its own: to change what it displays during the attract phase, to return to the attract screen when people walk away, etc. Even in this very simple kiosk, there were a number of variations I had to keep in mind when determining



what, when, and how to get the kiosk to its next stage.

Fortunately, Lingo's timeout command provides a great method for handling these types of automated instructions. Director's timeouts let you create a named timeout object, give it a cycle time, and execute an action when its cycle comes up. I called this section of the movie script the dispatcher.

As an example, during the attract phase of the kiosk, the message shown in the blue band at the bottom of the screen cycles between an invitation to click on the screen and an attribution of the video to the Forest Service. As is the case with so many tasks in Director, this could be done several different ways. I chose to use handlers called go1A and go1B (1A and 1B corresponded to the original design document's screen designations) and a timeout in each along these lines:

```
timeout ("default").new (12000,  
#go1B)
```

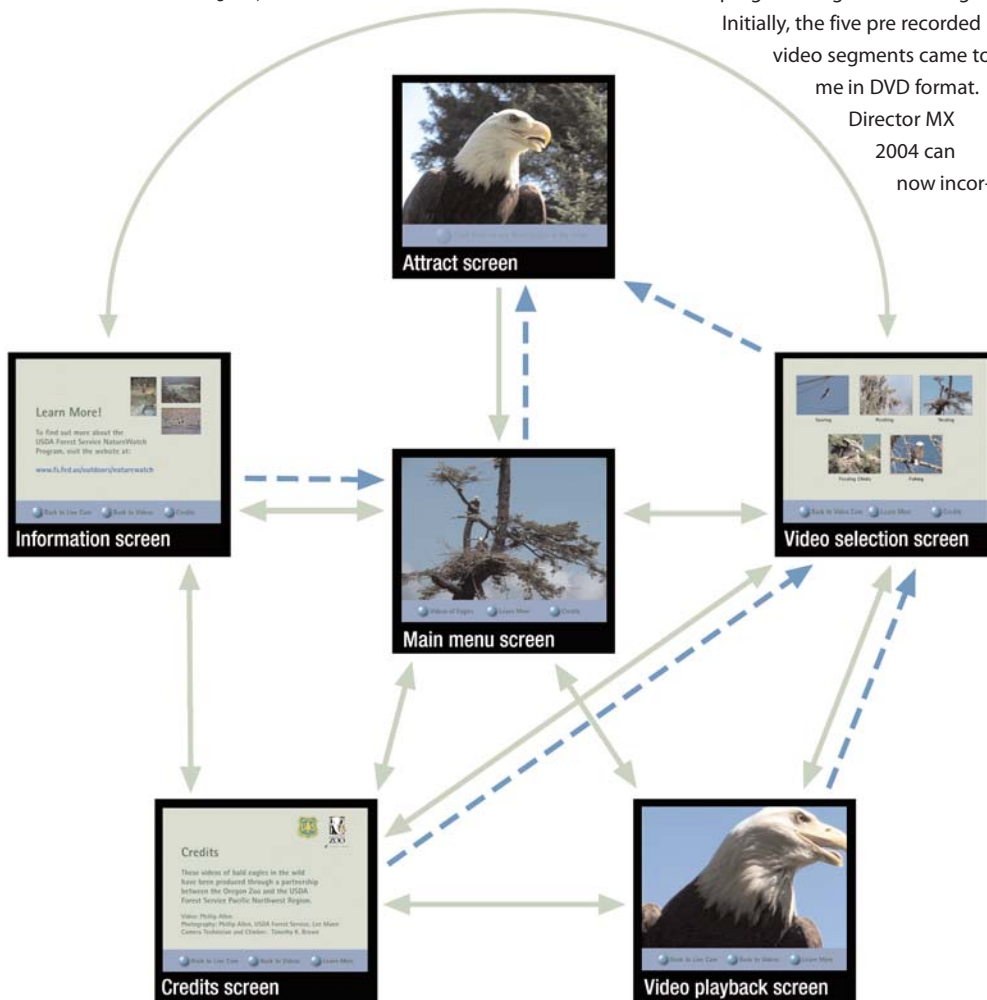
This command sets (or resets) a time-out object called default with a 12-second cycle. When the cycle expires, the go1B movie script handler is executed (the timeout command in go1B is identical except that its action is #go1A).

The movie script for the kiosk has seven such handlers, no more than six lines in length (although most call a video subroutine discussed later). There are two non-video screens for which the dispatch handler script consists of nothing more than resetting the default timeout object, deleting a video reconnect timeout, and moving the playback head with a go command.

Video Setup

If it hadn't been for the video, this project wouldn't have been much more than buttons moving between marker labels in the Score, with some timeouts thrown in for good measure. The video added a layer of challenge that affected both the programming and the design.

Initially, the five pre recorded video segments came to me in DVD format. Director MX 2004 can now incor-



porate DVD video. I could have used it as it was, running the video from the hard drive of the computers (as they didn't have internal DVD drives). DVD video can only be displayed in direct-to-stage (DTS) mode, however, and a couple of the screen designs had graphics overlaying the video, which would preclude overlays. For that reason, I had the video shop provide me with QuickTime versions of the eagle and salmon videos, because QuickTime has the option of running in non-DTS mode with other sprites appearing on top of the video. Although that affects playback quality, my tests with the samples didn't find it to be a problem. This decision also simplified my video management tasks. Instead of switching between DVD and QuickTime, the kiosk application would use only QuickTime.

A single QuickTime cast member is used in the kiosk, sized to 1280x884 pixels (the area of the screen above the blue menu band). All digital video cast members in Director are linked; the kiosk's QuickTime cast member would be alternately linked to the streaming video from the live camera and the file for the canned video on the hard drive.

Video Handling

The canned video segments were the simple part of the video handling. There are two conditions under which the video segments appear. The first is when one of the buttons is pressed on the video selection screen. The other is in the kiosk's attract loop. If the first line of the settings file has only a line return and no URL, the kiosk automatically chooses a random canned video clip to display instead (this is the "winter" mode for the kiosk).

The settings file tells the kiosk application movie which digital video file to use for the canned videos. The video shop that handled the creation of the QuickTime files only gave me a single file for each of the kiosk versions. Rather than take the time to split them up, I used Director's great control of QuickTime media to play back selected portions of the single video where appropriate.

The second line of the settings file contains a property list with path and clips properties. The path is just the file name for the canned video movie. The clips property is a linear list with five

The bible for <CF_Developers>



ADVERTISE

Contact: Robyn Forma
robyn@sys-con.com
(201) 802-3022
for details on rates
and programs

SUBSCRIBE

www.sys-con.com/
cfdj/subscription.cfm
1 (888) 303-5282



COLD FUSION Developer's Journal





items, each of which is another property list, with start and end properties. The five canned video clips are defined by these values, which are in ticks (measurements of 1/60th of a second) from the beginning of the digital video file. The order of the clips determines which button plays each section of the video. In the example of the settings file above, the first button (top left) would play from 6 seconds (360 ticks) to 69 seconds (4140 ticks) into the movie.

The video selection buttons have a simple behavior on them that identifies the button with an integer (from 1 to 5).

```
global gClip

property pVideoID

on getPropertyDescriptionList
    return [#pVideoID: [#comment: "Video ID", #format: #integer, #default: 1, #range: [1, 2, 3, 4, 5]]]
end

on mouseDown me
    sendAllSprites (#setVideoHighlight, me.spriteNum)
    gClip = pVideoID
    go4
end
```

When the touchscreen detects that one of the buttons has been activated, the button for selected video becomes highlighted in red, the gClip global variable is set to the identifier for the selected button, and the go4 dispatch handler is called.

```
on go4
    member ("livevideo").filename = gInfo.path
    timeout ("default").forget ()
    timeout ("reconnect").forget ()
    go "4"
end
```

To play back the canned video segment, the filename property of the QuickTime cast member is set to the appropriate video file. When the video segment is finished playing, the application returns to the video selection screen, so the default timeout isn't needed. The reconnect timeout object is used for the live camera, but is also unnecessary in

this section of the application. The go command takes the movie to a position in the score with a marker labeled 4 (again, to correspond with the design document), not frame 4.

The section of the Score that plays back the canned videos contains a QuickTime video sprite with my Canned Video behavior:

```
global gInfo, gClip

property pSprite

on beginSprite me
    pSprite = sprite (me.spriteNum)
    pSprite.member.video = true
end

on enterFrame me
    if pSprite.movietime < gInfo.clips[gClip].start then
        pSprite.movietime = gInfo.clips[gClip].start
    if pSprite.movietime > gInfo.clips[gClip].end then
        go3B
    end if
end
```

The beginSprite handler ensures that the video playback of the sprite's cast member is enabled. The enterFrame handler is simplicity itself. The sprite's movieTime property is checked against the start property of the clip selected when the button was pressed (and gClip was set). If the movieTime is too low (which it will be when the sprite is instantiated), the movie is brought up to the starting point. When the movieTime reaches the clip's end value, the go3B dispatch handler will reset things and return the movie to the video selection screen.

Dealing with the attract loop is a bit more difficult because of the live camera. The attract video shows up on three screens and should appear seamless as those screens are navigated. Two of the screens are the different phases of the unactivated kiosk, when nobody has touched the screen. The third screen is a menu screen showing the attract video and three menu options, which is the first thing users see when they touch the screen in attract mode.

Each of the dispatch handlers for those screens uses the setupVideo han-

dlers to link to the proper video source:

```
on setupVideo
    vPath = field "liveURL"
    if vPath = "" then
        vPath = the moviePath & gInfo.path
    end if
    if vPath <> member ("livevideo").filename then
        member ("livevideo").filename = vPath
    end if
end
```

If the liveURL field is empty (i.e., if the first line of the setting file is blank), then the vPath variable is set to the canned video path. Then, if the filename of the QuickTime cast member isn't already a match for vPath, it's reset. This keeps the video from being reset when it's not needed; it's also the mechanism that changes back to the live video stream when the movie returns from canned video playback.

Of course, this was complicated by the fact that the kiosk needed to operate with canned video when the live camera wasn't available. In the attract mode, the live camera would be on continuously. However, but the canned video clips that would show as an alternative needed to play back in some sort of random order. This could have been accomplished by creating different sections of the score to handle the live video and canned video tracks, but I felt that would be more trouble than the alternative, which was to create a system that automatically detected which mode was playing. This was easily checked by looking at the liveURL field and writing a behavior to accommodate both. The behavior is actually a bit too involved to go into in this article, but, essentially, it expands on the Canned Video behavior previously mentioned. In the portion of the behavior that handles the canned video, instead of jumping to the video selection screen, a new random clip is selected and the movieTime is immediately set.

I put together the essential elements of one version of the kiosk in about three days. Things worked pretty well using a live video test feed from vBrick's Web site and the QuickTime files. The whole thing needed to be tested in situ, however,

Advertising Index

with the actual video stream from the salmon beds (regrettably, the aerie was empty this year due to a road-killed eagle). The video was only accessible through the Zoo's network, so six days before the opening, I schlepped my laptop, a USB drive, and the latest copies of the application up to the Zoo for testing.

A couple of things were immediately obvious. For whatever reason, the live cameras took significantly longer to synchronize than the test streams did. It was taking ten or more seconds for the image to fully resolve. Also, while I hadn't encountered any problems running the test streams in non-DTS mode, the actual streams weren't working well that way, which meant some minor redesigns were needed. The latter wasn't a significant problem, because we'd discussed the possibility earlier in the process, and the client was aware that we might need to make some sort of change to avoid overlays. The resolution issue was more distressing because of the extreme delay it caused before an acceptable image was displayed. Further testing showed that it happened every time the sprite's image was initialized on the stage, whether or not separate cast members were used for the live and canned video feeds.

I'd love to say I found a great solution for the problem, but what we ended up doing was to slap a screen capture on the screen, turn off the video property for the QuickTime cast member so no picture was visible, and add a film loop that says "Connecting to Live Video" in the center of the screen for 15 seconds. Such are the constraints of time. Of course, all of this had to be turned off when the canned video was used as the attract loop.

Testing and fixing the video issue took almost as long as the initial phase of developing this project. But once everything was up and running the way we wanted it to, I took a copy home and slapped the text and graphics in place for the eagle version in just a couple of hours.

One glitch turned up after installation: when the link to the camera would be lost overnight. I added a reconnect timeout to handle that problem, kicking it off every few hours when the live camera was active.

Advertiser	URL	Phone	Page
ActivePDF	www.activePDF.com		15
CFDynamics	www.cfdynamics.com	866-233-9626	5
Electric Rain	www.swift3D.com/mx	888-613-1500	Cover III
Hal Helms	www.halhelms.com		20
HostMySite.com	www.hostmysite.com/mxdj	877-248-HOST	17
Interakt	www.interaktonline.com		6
Macromedia	www.macromedia.com/go/dwupdated		Cover II
Macromedia	www.macromedia.com/go/dwupdated		3
PaperThin	www.paperthin.com	800-940-3087	21
Seapine Software	www.seapine.com/whitepaper.php	888-683-6456	9
Macromedia	www.macromedia.com/go/max		Cover IV
Web Services Edge 2005 East	www.sys-con.com/edge	201-802-3004	57

Conclusion

The end result of this project was that, even though I received the artwork and canned video in the middle of the week before the exhibit opened, I had a working video kiosk application that ran in two versions. It had capabilities for text control of video streaming, video playback, and even clip timing control, all of which were up and running on the Sunday before a Friday opening.

Both of the computers used for the kiosks have the same data on them. The salmon kiosk can be switched to an eagle kiosk by renaming the external cast files and swapping a text file. The most time-consuming part of setting the application up on another computer, should something happen to the one in the kiosk, is duplicating the large video files (the eagle movie is 1.6 GB). The rest of the data for the kiosk is less than 25 MB, so updating the engine is a snap. Once the Design Group is done with their next project, I'll show them how to change the graphics in Director. Meanwhile, on opening day, I was inside the eagle enclosure,

holding one of the monitors up out of the water as it was being installed, because I was on hand to make sure things ran smoothly.

Director is responsible for getting this project done on time. Its media handling is rugged enough that I can expect it to do what I tell it to, particularly in a controlled environment like a kiosk. The ability to combine video from hard drive and streaming sources by simply changing a cast member property, the font portability in both playback and authoring, external position-based cast swapping, even bitmap editing within the application, all combined to bring this project to fruition.

The fact that there are many ways to accomplish the same task was also critical: it gave me the options I needed to structure the movie with a combination of simple movie scripts and behaviors. Simplicity is much more reliable than complexity. Some people count their programming accomplishments in the number of lines of code that they've written. Call me lazy, but I count mine in the number of lines of code that I didn't need to write. ☺

Darrel Plant is the author of several out-of-print books on Director and Flash. He's the publisher at Moshofsky/Plant Creative Services (www.moshplant.com), a Portland, Oregon-based contract developer. Darrel likes to think that he specializes in animation programming. He has worked on projects for Intel, Second Story Interactive, and Macromedia, among others. dplant@moshplant.com



ASCII, ANSI, Roman 1, and What's All That?

Earn your 'guru' nickname
by kerry thompson

“Yo, guru!” I looked up to see the guy a couple cubicles over prairie-dogging, trying to get my attention. I think they call me “guru” because I’m too old to be called “dude,” not from the feeling of awe I think I am due. “What’s the ASCII code for the German u-umlaut?” I was tempted to say “none,” which would have been, technically, the correct answer. Instead, I opted to be nice for once, and told him “Lower case? 252, decimal.”

I could have spent time explaining the difference between ASCII, ANSI, ISO 8859, and the Apple Standard Roman character set. Perhaps I could have thrown in some tidbits about so-called double-byte characters used in Chinese, Japanese, and Korean. I love to see people’s eyes glaze over.

Instead, with a little prompting, I decided to put that information into this article, in the hope that it will reach some people who can actually use it.

The Basics

You probably already know this, but we have to start someplace. Letters, numbers, punctuation, ideograms, syllabaries – the building blocks of written language – are represented in a computer by numeric codes. Pretty much everybody agrees that, when the computer sees a code of 101 (decimal), it represents a lowercase “e”. We don’t all agree on what 252 represents, and therein lies the rub.

ASCII vs ANSI

We commonly refer to character encoding as a letter’s “ASCII value,” when we really mean “ANSI value.” A lot of the time that’s sufficient, but in fact the ASCII standard is pretty much obsolete.

ASCII (American Standard Code for Information Interchange) is a 7-bit stan-

dard that has been around since the late 1950s (its current incarnation dates from 1968). It defines 128 different characters, which is more than enough for English: upper- and lowercase letters, punctuation, numerals, control codes (remember control-c?), and nonprinting codes such as tab, return, and backspace.

Over the past couple of decades though, computing has become worldwide, and the old English-centric ASCII system just wasn’t up to handling German, French, Spanish, and Portuguese, not to mention Turkish, Arabic, and Chinese. There are national variants of ASCII, which enjoyed a brief popularity in the 1980s. However, an 8-bit encoding system was developed by the American National Standard Institute (ANSI) in the late 1980s, and it included all the characters needed for most Western European languages. Microsoft’s adoption of the ANSI standard for Windows 3.1 gave it the momentum to become the dominant standard. Today it is the de facto standard for Western European languages.

You may have noticed that I haven’t mentioned the word “Macintosh” yet. That’s because this article is going to be (appallingly, for some) Windows-centric. That’s not because of any prejudice against the Mac; I have been using Macintosh computers for years, the Apple II before that, and the Apple before that. I love the Mac, and use both Mac and Windows computers every day.

I decided to concentrate on Windows because space for this article is limited, and I had to choose one or the other. At the end of the day, it doesn’t matter a whole lot, because the concepts apply equally to both platforms. The details differ – Macintosh encodes the extended character set (mostly accented and special characters) somewhat differently

from Windows, but the basics are the same. There is a wealth of information on the Web, especially on Apple’s Web site, to fill in the missing details.

What About Turkey, Greece, and Russia?

Computing has spread well beyond the U.S. and Western Europe, so a need quickly developed for a standard for encoding other languages. The International Standards Organization (ISO) provided the answer in its ISO 8859 standard.

I know what you’re probably thinking. There are hundreds of written languages in the world, from Afrikaans to Vietnamese. If they’re all single-byte languages, how can they be covered by a single standard? The answer is that ISO 8859 is really several standards – 15 now, and growing.

The most commonly used ISO 8859 standard is ISO 8859-1, or Latin 1. It covers most Western European languages, and then some: Albanian, Basque, Catalan, Danish, Dutch (partial), English, Faeroese, Finnish (partial), French (partial), German, Icelandic, Irish, Italian, Norwegian, Portuguese, Rhaeto-Romanic, Scottish, Spanish, Kurdish, Swedish, Afrikaans, and Swahili.

Following is a summary of the other ISO 8859 standards:

- **ISO 8859-2 (Latin-2 or Central European):** Central and Eastern European languages that use a Roman alphabet, including Polish, Czech, Slovak, Slovenian, and Hungarian.
- **ISO 8859-3 (Latin-3 or South European):** Turkish, Maltese, and Esperanto; largely superseded by ISO 8859-9 for Turkish and Unicode for Esperanto.
- **ISO 8859-4 (Latin-4 or North European):** Estonian, Latvian,

Журнал Forbes опубликовал список перспективных (если использовать терминологию Михаил Борисовича) бизнесменов российского разлива.

image 1

- Lithuanian, Greenlandic, and Sami.
- ISO 8859-5 (Cyrillic):** Covers most Eastern European languages that use a Cyrillic alphabet, including Russian, Ukrainian, and Belarusian.
- ISO 8859-6 (Arabic):** Covers the most common Arabic glyphs, although not nearly all of them.
- ISO 8859-7 (Greek):** Modern Greek
- ISO 8859-8 (Hebrew):** The modern Hebrew alphabet as used in Israel.
- ISO 8859-9 (Latin-5 or Turkish):** Largely the same as ISO 8859-1, replacing the rarely used Icelandic letters with Turkish ones.
- ISO 8859-10 (Latin-6 or Nordic):** A rearrangement of Latin-4. Considered more useful for Nordic languages.
- ISO 8859-11 (Thai):** Not a complete representation of Thai, but covers most of the glyphs.
- ISO 8859-12:** Was supposed to be Latin-7 and cover Celtic, but this draft was rejected.
- ISO 8859-13 (Latin-7 or Baltic Rim):** Added some glyphs for Baltic languages that were missing from Latin-4 and Latin-6.
- ISO 8859-14 (Latin-8 or Celtic):** Mostly a rearrangement of the ISO-8859-12 draft. Covers Celtic languages like Gaelic and the Breton language.
- ISO 8859-15 (Latin-9):** A revision of 8859-1 that removes some little-used symbols, replacing them with the Euro symbol € and the letters Æ, œ, and Ÿ, which completes the coverage of French and Finnish.
- ISO 8859-16 (Latin-10 or South-Eastern European):** In development intended for Albanian, Croatian, Hungarian, Italian, Polish, Romanian and Slovenian, but also Finnish, French, German and Irish Gaelic.

Using Other Character Sets: Code pages

English Windows is designed – are you ready? – for English. That doesn't mean it doesn't support other languages, though. It just means we have to work a little harder to utilize them.

Output is fairly easy, especially for Latin-1 languages (remember, that's most of the Western European languages, including English). Here is some text copied from a German French Horn manufacturer's Web site, www.ricco-

kuehn.de/index.htm:

Werbeproschüren vieler Handwerksbetriebe im Musikinstrumentenbau beginnen mit der Darstellung einer langen Firmengeschichte.

I simply used the same Times New Roman font I've been using for English. Since it is an ANSI font, it has all the German characters we need, including "ü". It would have worked with French, Italian, Portuguese, or any other ISO 8859-1 language.

But what about Russian? Could I go to the Pravda Web page and copy something from a Russian-language Forbes article? Check it out in Image 1.

I have no idea what that means, but apparently I can use Cyrillic text on English Windows. I can even use my dependable Times New Roman font. So what's going on here? Didn't we say that Russian used a different encoding, ISO 8859-5? It does. We were fortunate, because the version of Times New Roman I have on my system supports code page 1251, where the Cyrillic alphabet lives, in addition to Latin-1's code page 1252.

The term "code page" can be confusing because it is really just another way of referring to the character encoding. Essentially, code page 1251 and Cyrillic encoding are synonymous. You could imagine a font as a stack, something like this:

Code Page	ISO 8859
1250	8859-2 (Central Europe)
1251	8859-5 (Cyrillic)
1252	8859-1 (Latin 1)
1253	8859-7 (Greek)
1254	8859-9 (Turkish)
1255	8859-8 (Hebrew)
1256	8859-6 (Arabic)
1257	8859-4 (Baltic)
1258	VISCII (Vietnamese)
874	8859-11 (Thai)

Font publishers often include several national character sets within a single

font – it's simpler if you can just install Arial, and have Roman 1 on an English system, Cyrillic on a Russian system, and Greek on a Greek system.

Not all programs support code pages the same way though. Macromedia's Director, for example, has rather spotty code page support. It will display Russian on an English system – but dependably only if you embed a Russian font in your movie. Also, sending text information between Flash, which is Unicode-enabled, and Director, which is not, needs special care. Communications between Director and Flash are further complicated by the fact that, although both are cross-platform Macintosh and Windows, the two systems encode high-ANSI characters differently.

Asian and Double-Byte Languages

You may have noticed that we have so far neglected a lot of languages, especially Asian languages. Chinese, Japanese, and Korean are special cases that we will cover shortly, but what about Tibetan, Lao, and other Asian languages?

A few years ago I was a contributor to the Radio Free Asia Web Site, which had text in Vietnamese, Khmer, Laotian, Burmese, and Tibetan, none of which were (or are) covered by ISO 8859. These are all single-byte languages, and codifying them remains a work in progress. Although you can find Tibetan fonts, and by and large they agree on code usage, creation of a standard is hampered partly by low per-capita computer usage, and partly because scholars don't agree on some of the basics of the written language, like standardized spelling.

Currently, those languages are likely to be covered by Unicode, a standard that has been developing for over two decades. Unicode is outside the scope of this article, but it is basically a double-byte standard designed to offer standardized encodings for virtually every written human language (and purportedly for some nonhuman languages like Klingon, which, for some reason, has been excluded).



ed from the Unicode specification).

Chinese (simplified and traditional), Japanese, and Korean, often referred to as CCJK, have too many characters for any single-byte encoding, so they are commonly referred to as “double-byte” languages. Actually, that is a misnomer, because they are a mixture of single- and double-byte codes. Most encodings of those languages have ASCII or ANSI characters as single-byte characters, and the rest of the characters are double-byte. They should more properly be called “multibyte” languages, but nobody but a purist (like me) will be bothered if you call them double-byte.

To understand multibyte encodings, it helps to understand a little about the languages. We'll begin with Chinese, because the concepts carry over to the other multibyte languages. Also, I know more about Chinese than either of the others. I deny allegations that I chose Chinese because my wife is Chinese.

Chinese

Chinese, as most people know, uses ideograms – characters that represent a concept such as day, person, happy, or spoon, but have no inherent phonetic characteristics.

Actually, ideograms are not too hard a concept to grasp, because we use them in virtually every Western language. Consider this: 42. Those two characters represent a concept that means the same to you whether you speak English, Chinese, or Arabic, which happens to be where they originated. What's more, each of the two numerals can stand on its own, with a somewhat different meaning.

There is nothing inherently phonetic about them: you can pronounce them forty-two, zwei und vierzig, or si-shi er, and they convey exactly the same meaning. Now, extend that concept to encompass an entire human language, and you understand ideograms.

Clearly, there are more than 256 concepts in any human language. That's why Chinese can't use a single-byte encoding. Nobody really knows how many Chinese characters there are, but you need to know about 3,000 to read a newspaper, and a scholar may know 10,000 characters. With two bytes, we can represent over 65,000 characters – sufficient for Chinese.

No Chinese font can reasonably expect to contain every possible character a Chinese writer will need, so Chinese encoding systems specify locations for custom characters. People's names are often written with unique characters; occasionally new characters are invented for new concepts; and there are some characters that are rarely used, like the 15th-century word for county magistrate in Yunnan province.

The situation is further complicated by the fact that there are two methods of writing Chinese, alluded to above – simplified and traditional. Traditional characters, or “complicated-body characters” as they are called in Chinese, are used in Hong Kong, Taiwan, and by most overseas Chinese. Simplified characters, popularized by Mao Ze-Dong's push to bring literacy to the masses, are widely used in mainland China, and in Singapore. Increasingly, they are used overseas as a new generation of Chinese speakers emigrates.

Simplified and traditional characters have nothing to do with pronunciation. If you read Chinese characters, you can read Chinese, regardless of which Chinese language you speak – Mandarin, Cantonese, Shanghaiese, Chongqing-hua, or any of the dozens, and probably hundreds, of distinct Chinese languages and dialects. If you think about our Western ideograms (numerals), you will realize we do the same thing. $2^8 = 256$ means the same to everybody, no matter what language they speak.

Traditional and simplified characters do, however, have an impact on encoding. The Big-5 encoding system was originally developed for traditional Chinese, and is the basis of Traditional Chinese Windows as used in Taiwan. GB encoding was developed for simplified characters, and is the basis of Simplified Chinese Windows as used on the mainland. Nowadays, there are Big-5 encoded simplified Chinese fonts and GB-encoded traditional character fonts, but the basic distinction remains.

Some Chinese systems allow you to convert from traditional to simplified characters automatically. That's relatively straightforward – you can reliably map traditional characters to simplified characters. However, because of the way Chinese was simplified, you often can't

automatically convert simplified to traditional.

Chinese characters were simplified in two ways. Some characters are written with a reduced number of strokes – for example, the “speech” character was simplified from seven strokes to two. Other characters were simply combined. For example, the Mandarin words for “empress” and “behind” are pronounced exactly the same, so one of the traditional characters was dropped, and both are represented by one character in simplified Chinese.

Japanese

Japanese uses one of the most complex writing systems on earth, with no fewer than four character sets: kanji, hiragana, katakana, and romaji, all of which can be freely mixed in Japanese text.

Kanji requires little explanation: the characters are Chinese. In fact, the very word “Kanji” means, literally, “Chinese Character.” Japanese uses about 6,000 Chinese characters – 2,000 in everyday life – and by and large they have the same, or similar, meanings as they do in Chinese. In fact, they usually have a “Chinese” pronunciation in addition to one or more Japanese pronunciations. For example, the character for mountain, 山, can be read with the “Japanese” pronunciation “yama,” or “san,” similar to the Chinese pronunciation “shan.”

Hiragana and katakana, collectively known as kana, are phonetic writing systems. However, they are not alphabetic in the way Westerners are used to. They don't represent vowels and consonants. Rather, each kana character represents one syllable. For example, “a,” “mi,” or “to.” There are ways of modifying some of the characters – For example, making “sa” into “za,” or “ki” into “kyo,” but all sounds in the Japanese language can be represented by one of the 46 hiragana or katakana characters.

Hiragana and katakana characters are phonetic duplicates of each other. That is, for every hiragana character, there is a corresponding katakana character, pronounced exactly the same. With some exceptions, hiragana is used for native Japanese words, and katakana is used for “loan words” – words imported from other languages, usually English – of

which there are many.

The other character set the Japanese language uses is romaji, or Roman characters. These are the same letters used in Western European languages – basically, the same letters you have been reading for the past fifteen minutes or so. Romaji are usually used to spell a foreign name or word.

There are three popular Japanese character encodings, all based on the “Japanese Industrial Standard,” or JIS. When people speak of JIS encoding, they are referring to the ISO standard 2022-JP, which is actually a 7-bit code sometimes used for transmitting characters.

More commonly used now is EUC, or Extended UNIX Coding, which is a multi-byte system utilizing 8-bit bytes, and is pretty much the standard on the Internet. The other system, Shift-JIS, was developed by Microsoft for Japanese versions of their OS and software, and is the encoding used for things like files and applications.

Korean

Korean text, or Hangul, is probably the easiest to understand. Hangul char-

acters, like Japanese kana, represent syllables. Unlike Japanese kana, however, Hangul characters are formed with strokes representing vowels and consonants, and combined into syllabic blocks called jamo. You should note that, even though the basic Hangul strokes represent individual sounds, they are not combined to form words, but syllables that are then combined to form words.

The Hangul system has been used in Korea for over 500 years, since its invention in 1443. Before that, the Korean language was written with Chinese characters. You will still find Chinese characters in Korean text, though it is becoming less common. The Korean Standard Hangul Coding Scheme for Communications (KS5601) includes nearly 5,000 Chinese characters in addition to over 2,300 Hangul characters.

Tying It All Together

Representing the world’s hundreds of languages on computers is a daunting and complex task. This article has covered the basics of a few languages, on a single system, Windows. We haven’t

addressed input or Unicode, either of which would need an entire article, or book, to be covered in any depth.

However, this is a start, and hopefully will provide a jumping-off point for you to explore the subject in more depth.

Then maybe your coworkers will call you guru – and mean it. ☺

Kerry Thompson is a Boston-area freelance developer specializing in multi-media and multilingual projects. He has been programming since 1981, working in languages such as Basic, FORTRAN, COBOL, 6502 and 8088 Assembler, C/C++, JavaScript, XSLT, and HTML. He first came to the Macromedia world in the early 90s with Director 4 and, a few years later, Flash. His interest in languages stems from his time living in Beijing, and subsequent stints developing multilingual software for companies such as Sony and Disney Interactive. He devotes much of his spare time to music, playing French Horn in several Boston area groups, including holding the principal spot in the Boston Civic Orchestra. alpha@cyberiantiger.biz



Web Services Edge 2005 East

International Web Services Conference & Expo

Tuesday, 2/15: Conference & Expo | Wednesday, 2/16: Conference & Expo | Thursday, 2/17: Conference & Expo



The Largest i-Technology Event of the Year!

Guaranteed Minimum Attendance 3,000 Delegates

February 15-17, 2005, Hynes Convention Center, Boston, MA

Announcing Web Services Edge 2005 Call for Papers Now Open!

Submit your proposal by August 15, 2004

Interested in exhibiting, sponsoring or advertising?

Becoming a Web Services Edge exhibitor or sponsor offers you a unique opportunity to best position your company’s message and visibility to a targeted audience of Web services professionals. Through a wide variety of pre-show, on-site, and post-show marketing programs and opportunities, Web Services Edge 2005 East will serve industry stalwarts as well as exciting start-up companies in reaching the world’s most mature internet-based regional market, the US Northeast Region, at Boston’s best-known IT business address, the Hynes Convention Center. For Exhibit & Sponsorship information please contact: Grisha Davida, 201-802-3004, e-mail: grisha@sys-con.com

Sponsored by:




Contact for Conference Information: Lin Goetz, 201-802-3045, lin@sys-con.com

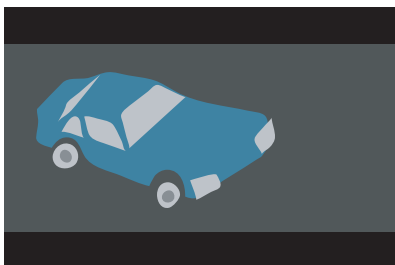
www.sys-con.com/edge PRODUCED BY SYS-CON EVENTS

140 Decibels

One Hundred and Forty Decibels is a short film about a depressed man whose anxiety escalates as he becomes increasingly more sensitive towards his environment, particularly noise. His midtown Manhattan apartment is both sanctuary and prison.

Louis F. Cuffari, art director of 140 Decibels, created animated introduction titles using Macromedia Flash. The animation spirals through scenes that produce loud noise in the city. The credits of the film dissolve in and out along with each illustration.

140 Decibels is an Insomnia Creations production, directed by Steve Blanco.
www.insomniacreations.com/online/140decibels/ 



140^{dB}

o n e h u n d r e d a n d f o r t y d e c i b e l s



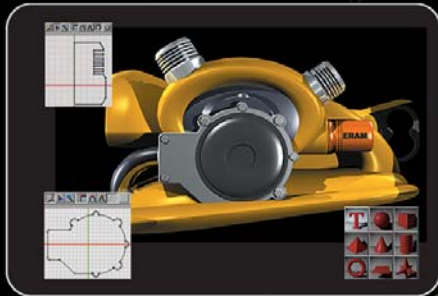
THE ULTIMATE 3D FLASH™ TOOL

There's a reason Swift 3D leads the industry in 3D vector and raster animation

Swift 3D was created specifically for Web designers, graphic artists and non-3D professionals. It provides a toolset and interface that allows anyone to quickly learn the basics of 3D modeling and animation while providing plenty of room to grow into a full set of advanced 3D tools. Swift 3D's vector and raster rendering capabilities have become the industry benchmark in quality, speed and versatility, making it the tool of choice for beginners and experts alike.

Swift 3D is power, quality and ease-of-use in a very affordable package

3D Made Easy



CONVERT existing artwork

Import vector files or use installed fonts: Instantly turn your AI or EPS artwork and logos into 3D objects or create 3D text using TrueType and PostScript fonts.

Use professionally designed models: Browse a built-in model gallery or import high-quality 3D content from the web in the 3DS or DXF format.

CREATE with familiar tools

Model with basic building blocks: Choose from a wide variety of 3D primitive shapes, modify them as needed, and quickly assemble your scene.

Draw 2D paths to create 3D objects: Harness your drawing skills by using a Bezier pen tool to transform 2D shapes into 3D models.

Use a frame-based timeline: Start with pre-built animations for quick results or tap into the full-featured Flash-based keyframe timeline.

GROW into advanced features

Polygonal Modeling: Use a powerful modeling system that's easy enough for anyone to master, yet supplies infinite versatility.

Cameras, Lighting and Textures: Employ cinematic camera and lighting techniques and surface your models with UV bitmap texturing.

UNRIVALED VECTOR AND RASTER OUTPUT



Vector Transparency: Adjust the opacity of objects to create transparencies that integrate perfectly with your 2D artwork.

Shadows: Cast shadows from any light source and add impressive visual effects to any rendering style.

• OUTLINES
• CARTOON FILL
• GRADIENTS
• VECTOR RENDERING

• PHOTO REALISTIC TEXTURING
• RASTER RENDERING

Texturing: Create and apply custom bitmap textures to give your objects an ultra-realistic look and feel.

Reflectivity: Use reflective materials on any object to create scenes with depth in both vector and raster format.

www.Swift3D.com/MX