SYS-CON

GAMES: ESCAPE FROM REALITY WITH FLASH P. 26

2.7

MX developer's journal

THE LEADING MAGAZINE FOR MACROMEDIA MX DEVELOPERS & DESIGNERS

volume 2 issue 7 2004 www.mxdj.com

Flash Virtuosos

the best in web design

DREAMWEAVER
The Power of Your Descendants

FREEHAND
Form vs Feature

COLDFUSION
An OO Approach to 'War'

DIRECTOR
Best Practices in Kiosk Design

$9.99US $9.99CAN

MX developer's journal
july 2004

7 Web Standards
The right thing to do
by jeffrey veen

50 xile
Cartoon
by louis f. cuffari

58 vanguard
The Meatrix
by free range graphics

10 The Power of Your Descendants
Go forth and style with CSS
by stephanie sullivan

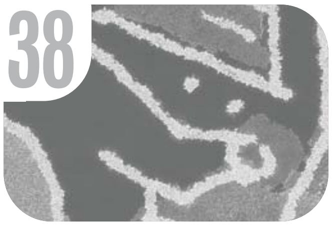
18 Flash Virtuosos
American Design Awards reward innovative design potential

14 Zoomin' on the Web
Add new dimensions to your applications
by john williams

22 CSV in Flash MX 2004
Make your classes easy and flexible
by danny patterson

30 Form vs Feature
The new Extrude tool in FreeHand MX
by ron rockwell

26 Creating Games in Macromedia Flash MX 2004
Escape from reality
by glen rhodes


38 An OO Approach to 'War'
Child's play for professional developers
by hal helms

52 Best Practices in Kiosk Design
There's no such thing as 'foolish consistency' here
by roy crisman

44 Best Kept Secret in Town
Application development with Macromedia Director
by jason macdonald & paul-catalin oros

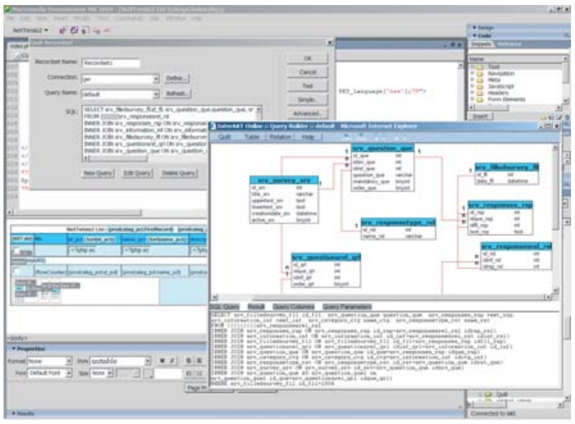
Dreamweaver Website Development

MANY needs - ONE solution

MX Kollection
Professional Dreamweaver Extensions
macromedia DREAMWEAVER MX 2004

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

Interakt
macromedia ALLIANCE PARTNER

Seapine Software™
changing the world of software development

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

Surround SCM Source Control
TestTrack Pro Defect Management
QA Wizard Automated Testing

macromedia ALLIANCE PARTNER

software development 13th annual product excellence award JOLT

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

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.

Web Standards

The right thing to do

by jeffrey veen

I've been involved in the Web standards community almost as long as I've been working on the Web, and I've long felt that designing to W3C recommendations is the right thing to do. It's easy to evangelize standards like XHTML and CSS, but when it came time to put my money where my mouth is for the redesign of my company's Web site, adaptivepath.com, my partners and I had a very frank discussion about whether the effort – and it would be a lot of effort – was really going to be worth it.

We had to ask ourselves, how important is standardization to an individual business like ours? Do Web standards give organizations a return on investment? Does the transition to XHTML and CSS make financial sense? The answer to those questions is yes.

Web standards help you:

- Speed development
- Simplify maintenance and increase opportunity
- Open up site accessibility
- Reduce bandwidth costs
- Improve user experience

Speed Development

So many of our clients have spent numerous hours building multiple versions of their sites, attempting to present a perfect design for as many users as possible who may be viewing their sites on any variety of browsers and devices. For every new design, there is an increase in development, QA, and maintenance time. We wanted to spend less time managing the site and still ensure that it rendered properly on available browsers. With over 95% of our audience now visiting our site with standards-compliant browsers, we knew it was the perfect time to make the switch. Using Web standards, we were able to achieve these goals by developing only one set of HTML pages and one stylesheet.

Adhering to Web standards forces you to error check your code to ensure it complies with the level of standards you've selected. Simply declaring which version of HTML (or, for that matter, XML) you're using will let you validate your pages against those specifications. Running your pages through a validator like the one available in Dreamweaver MX 2004 or *x* (online tool of choice) can save you an enormous amount of QA time. Before you even open the pages in your target browsers, you can run the validator and see exactly where your errors are. This reduces the time developers spend on QA and gives your site incredible consistency between browsers. While current browsers still have rendering bugs, they are far less severe than they were five years ago.

Simplify Maintenance, Increase Opportunity

For years, the standards community has been extolling the virtues of keeping visual design separate from content, but logically linked to each page. This clean separation makes it much easier for you to develop and maintain your pages for two main reasons. First, it makes the pages easier to update because of your content presentation such as type style, color, size, background colors and images, and even positioning of content blocks. Second, the separation of content and design corresponds to most teams' distinctions between design and editorial work, allowing the teams to work in parallel rather than sequentially. A nice by-product is code simplicity. This structure requires you to depart from coding pages with HTML in favor of XHTML. Your files become ridiculously simple because most XHTML pages are little more than a collection of semantically rich <div> and <p> tags, with a pointer to a powerful CSS file.

Recently, we hosted a CSS file for a client on our development server while they began production on content and back-end systems. As we continued to iterate the design, we were able to simply edit the file without having to integrate with their versioning and release system. By working in parallel, we dramatically reduced the time-to-market.

Using this structure also lets you immediately react to visitor feedback. Once you site is in production, you'll likely decide to change some aspect of the design – the size of your headlines, for example. If you had embedded the style information in each content page, it would take hours or even weeks to adjust the styles. But by using CSS standards, you can change thousands of pages by modifying a few lines of code in one file.

Speeding development and maintenance is a competitive and financial advantage. Shorter development times not only reduce costs, but free resources sooner, thereby increasing opportunity.

Open Up Access Options

Clean code pays even more dividends. Browsers that don't offer compliant CSS implementations can now simply skip the style. In other words, semantic XHTML markup can be rendered in any browser – including nontraditional clients like mobile phones, PDAs, voice interfaces and screen readers, and anything else that supports the most basic tag set.

A standards-compliant site that is coded for simplicity solves problems with mobile access, Section 508 accessibility, and past-version browser compatibility.

So you get all that and it's easier to develop and maintain? Indeed. You can even eliminate some hard costs in the process.

Reduce Bandwidth Costs

When we stripped away the fonts, tables, and little images used as design elements on our home page, we reduced the size of the code from 20.9K to 9.2K. Now, this may not seem like a lot, but it would aggregate to quite a bit if our site generated heavy traffic.

Our 56% reduction in bandwidth usage is hardly relevant to a site that gets a few thousand page views a day, but large commercial sites get that much traffic in a minute or two. The most popular sites often get tens of millions of page views a day.

Saving 30K to 40K from each page view – plus a cached stylesheet that never needs to be downloaded again – can save you thousands of dollars per month. Ever see an IT guy get excited about a new design? You will.

Improve User Experience

Cold, hard cash is easy to quantify, but there are additional benefits to slimming down code. It's no secret that a faster, more lively site will nearly always translate to a better overall user experience.

Huge interfaces squeezed through plodding modem connections have been a plague since the Web's inception. The increasing dominance of broadband has only helped a bit. A hotel phone line plugged into a business traveler's laptop may be the only tenuous link you've got to a new customer. Adopting clean, standardized code gives users a shortcut to accomplishing their goals at your site.

Justifying the Switch

These aren't formulas for determining the ROI of migrating to standards, but they are some pretty good financial justifications. "It's what all the cool sites are doing" shouldn't be your only point when arguing for a switch to XHTML and CSS.

The economic benefits of standardization are tangible: faster development, simplified maintenance, reduced bandwidth costs, improved user experience, expanded access and opportunity, and, of course, the assurance that Web users, regardless of platform, browser, or OS, can view content freely and consistently.... So what are you waiting for?

"By using CSS standards, you can change thousands of pages by modifying a few lines of code in one file"

Jeffrey Veen is a founding partner of Adaptive Path, a premier user experience consulting company. He launched HotWired.com in 1994, and is author of The Art & Science of Web Design *and* HotWired Style.
jeff@adaptivepath.com

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
Lin Goetz, 201 802-3045
lin@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 Rymsza, 201 802-3082
shannon@sys-con.com

EVENTS
President, SYS-CON Events
Grisha Davida, 201 802-3004
grisha@sys-con.com
Conference Manager
Lin Goetz, 201 802-3045
lin@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

BlueDragon [6.1]

RELEASED!

RUN YOUR CFML:

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

On the platform, web server, and operating system of your choice.

New Atlanta COMMUNICATIONS

Get BlueDragon hosting:

CFDynamics is now offering BlueDragon hosting!
Find out more by visiting: www.cfdynamics.com/bluedragon

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. Come join CFDynamics and New Atlanta in expanding the ColdFusion horizon!

POWERFUL 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

CFDynamics
A Division of Konnections Inc.

macromedia ALLIANCE PARTNER

BlueDragon Certified V.I.P. Hosting Partner

SIGN UP FOR A HOSTING ACCONT TODAY!

Use promo code: MXDJ04F and receive a FREE T-SHIRT!

WWW.CFDYNAMICS.COM

866 - CFDYNAMICS
866-233-962-6427

the power of your descendants

by stephanie sullivan

if you've decided this is the year you'll really get a handle on CSS, one of the first things you want to learn to do is harness its power – and avoid "classitis." Understanding the document tree – the structure of the document and the relationships between the elements – is the first step in writing highly efficient and compact CSS.

A mistake that many of us make when learning to use CSS is to put classes on most everything in the document to provide the specific styling desired. A bad case of classitis is hardly better than putting font tags everywhere. Yes, you're beginning to separate structure from presentation, but it's hardly elegant and efficient. Let's begin this exercise by looking at a simple hierarchy in the body area of an X/HTML document. First we'll look at the code, then the document tree (see Code I).

The hierarchy of the document tree can be demonstrated very much like a family tree. The example in Code I could appear graphically in the manner shown in Image I.

Notice that the hierarchy begins with the body element – much like your great grandparents might begin a portion of your family tree. From the body element descend the two divs – *#content* and *#side*. From #content descend the *h1*, *p*, and *ul* elements and it continues the same way throughout the document from there. The beauty of understanding the ancestral tree of your document is the ability it gives you to take advantage of the relationships that exist between elements in writing your CSS selectors.

Instead of giving the various elements different classes, you can leave your X/HTML portion clean and simple and write a descendant selector. (Don't be confused. Descendant selectors were called contextual selectors in CSS1.)

A descendant selector is simply a list of other selectors, in order of their appearance in the document tree, separated by spaces. The selectors used to create the descendant selector can be any of the following types:

1. ***Type selectors:*** A type selector is simply the redefinition of an element in your document. For example, if you write a CSS selector called *ul*, the attributes you write for that selector will apply to every *ul* element in your document. These selectors are the first thing I write when beginning to code a new site.
2. ***ID selectors:*** An ID can be given to any element in your document. To write succinct, clean CSS, I generally apply my IDs to containers on the page, whether they're divs or tables. An ID can only be applied once on a page. (See my article, Persistent Page Indicator, at http://nemesis1.f2o.org/aarchive?id=9 for an example of practical uses for an ID indicating the down state of a menu button. Especially if you enjoy using Dreamweaver templates.)
3. ***Class selectors:*** A class selector can be applied numerous times in a document. I generally use a class after I've redefined my elements (type selectors), defined my page area containers (IDs), and created all descendant selectors. Anything that falls outside those areas receives a class if needed.

Using our original X/HTML, let's look at an example of this. Since this article is about descendant selectors, I won't go into detail about the selectors I'm writing except to point out what kind they are. As mentioned previously, I always start with my type selectors. In this small document, I will only include the *body* type selector. I've given it the very basics (example files, if you'd like to work along, as well as the links used throughout this article are available from www.violetsky.net/mxdev/descend). Next, I move to the ID selectors for my containers. This will include *#side* and *#content*. I define their placement using those selectors. Now it's time for the descendants.

If you're following along and viewing the exercise in a browser, you likely noticed in the text of Code I that certain sentences are to be in color. Of course if you're following this exercise in a browser, it's obvious that at this point, they're all black. Many people would wrap those sentences in a <span> and give them a class. But there's no need for all that extra code. We've got the power of the descendants at our beck and call. Notice that in the first paragraph of the *#content* div, there's a sentence wrapped in a *strong* element. We'll use a descendant selector to color it orange. The descendant selector will begin with the highest level descendant and move in to the element we're styling (each separated by a space). Notice in the graphical example above, the strong element we want to give the orange color to descends from the p element. You could write the following selector:

```
p strong {
 color: #C30;
}
```

If you view this in your browser, you'll see why this won't give us the results we want on this page. All *strong* elements descending from *p* elements are now orange. Since that combination exists in

the *#side* div, that sentence is now orange as well, an undesired result. We need to be more specific in our descendant. Here's the descendant selector that will work in our page:

```
#content p strong {
 color: #C30;
}
```

Due to the specificity in the previous example, the *strong* element in the *#side* div is now black again. We could have written a simple type selector for the *h1* element. I chose instead to put the font sizing into the selectors for the div containers and thus, due to specificity which we'll discuss here later, I need to create the *h1* selector as a descendant of the *#content* div. It was written as *#content h1* so that it would override the sizing placed on the #content div.

Let's look at the other example in the unordered list. Using the principle we just learned, take a guess at the selector that could be written before you read on.

```
#content ul strong {
 color: #036;
}
```

Stephanie Sullivan is a Web developer, partner at CommunityMX (www.communitymx.com), owner of VioletSky Design (www.violet-sky.net), and contributing author of Dreamweaver MX 2004 Magic. *info@violetsky.net*

You could have also written – *#content ul li strong* to be even more specific. Either will work. Why would you want to be more specific? Glad you asked. Many times you'll hear that if you want one selector to override another, simply put it later in the CSS cascade. That's not completely true. The W3C has set up a formula for the calculation of a selector's specificity (www.w3.org/TR/REC-CSS2/cascade.html#specificity). In a nutshell, the ID selectors have the highest value, followed by classes and pseudo-classes and last/least are the type selectors. You can read the guidelines at the above URI learn how to calculate them, but keeping the above principles in mind will likely guide you through the more simple combinations.

To experiment with the specificity idea, create both of the above selectors – *#content ul li strong* and *#content ul strong*. Create them in that order so that the less specific selector is second. Give them two different colors. Notice that the more specific selector, even though it is first in the cascade, still overrides the less specific selector.

The principles of creating selectors that I've just discussed should help you really lighten up your CSS documents. For more examples of writing efficient CSS, see John Gallant and Holly Bergevin's free article, Writing Efficient CSS, at Community M (www.communitymx.com/abstract.cfm?cid=90F55). Now let's move on to a real world example of the power of descendant selectors – a site map.

image I

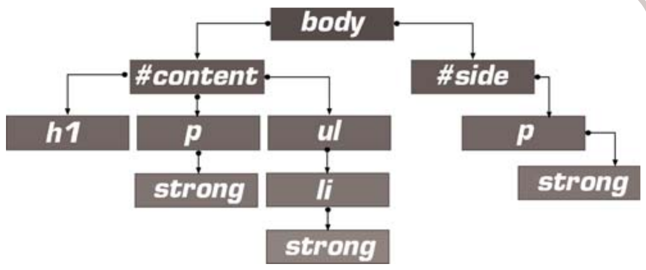
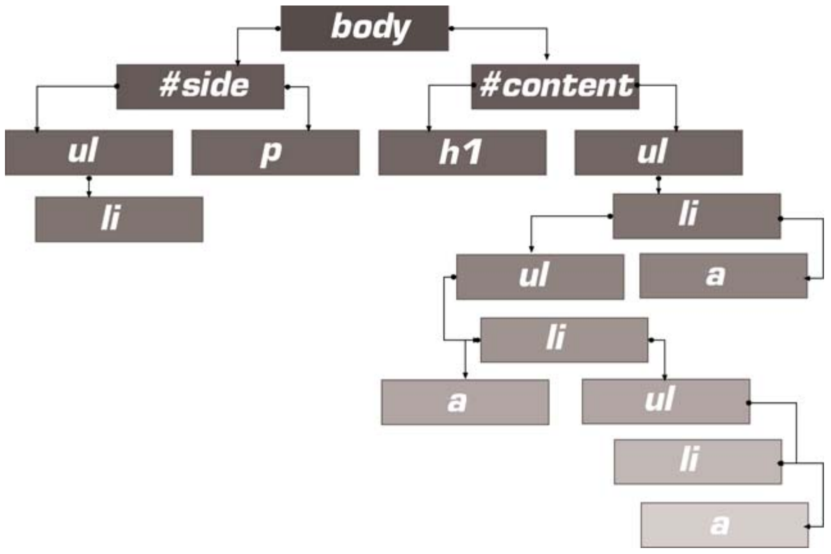


image II



Creating a Site Map Using Descendant Selectors

I've found that descendant selectors can be a simple and elegant solution for creating a site map. Using nested unordered lists and descendant selectors, you can make a very structured, easy-to-read map of any site under your control – no matter how complex. Your visitors will thank you for the nice hierarchal picture. And you'll be happy with the lack of having to class, or style inline, the whole hierarchical mess in your X/HTML document. It's all in the CSS, baby.

First, we'll take a look at the code (if you're working along with he example files this code is in the sitemap.htm file). This example uses three site sublevels. It can be used with as many levels as you need. Simply nest an unordered list for each one (see Code II).

Notice that Code II simply has three levels of nested unordered lists. And also notice that all the lists in the main content area are links (as any self-respecting site map would be). I've included an unordered list in the *#side* div as well so that you can see how handy more specific descendant selectors can be.

Now let's visualize the document tree for the above code (see Image II).

It's pretty straightforward, isn't it? You can see the way one list descends from the next. Now let's look at what we can do with the CSS. I'm going to keep the properties in the selectors pretty simple here. But play around in your own CSS document using background images and custom bullets. You can get some interesting looks. (If at any time in the following portion of the exercise you get confused by the descendant selectors, avail yourself of the SelectORacle – http://gallery.theopalgroup.com/selectoracle/ – he knows all.)

We'll leave the basic selectors from the previous page (only removing the two unnecessary descendant selectors – *#content p strong* and *#content ul strong*) and continue adding on to our CSS. Visually, what I want to do with the site map is alternate between a square and a circle bullet. I also want the bullets to be orange, but I want links to start with a brighter blue and, at each level, move to a darker hue. It's just a simple visual cue, in addition to the natural indentation, to let people visualize the difference between the levels. Last, I'll remove the underlines on the links but have them roll over to orange with underlines. Let's get started.

The first thing I want to establish is that all *ul* on my page should be orange. Remember, this will style all *ul*, not just the ones in the *#content* div. Something to remember about lists that contain links – like in the case of site maps – even though you'll give the links their own color, the bullet itself will be the color you've given to the list, not the link. Remember, the link is contained in the list. Since I want my bullet to be the complementary orange to the link's blues, I assign that to the *ul*:

```
ul {
 color: #C30;
}
```

Now I'll remove all the underlines from the links using a descendant selector. This one will take care of all the lists no matter how nested they are. Descendants do not care how far down the document tree they are. They just know that they descend and thus they obey (if only our children were that way):

```
ul a {
 text-decoration: none;
}
```

In order to have every other level alternate the square bullets with the round ones, I'm going to group selectors. If you haven't tried it, grouping is yet another way to cut down on your CSS document weight. If you have selectors with identical properties, simply place one after the next separated by a comma and a space (the space is imperative for it to work properly). Let's look at an example:

```
ul li, ul ul ul li {
 list-style: square;
}
```

Notice that we gave only the list items from the first- and third-level unordered lists the square list style. Let's give the second level lists a round disc style:

```
ul ul li {
 list-style: disc;
}
```

Next, we need to give each level its own link color as I described previously. I'm also going to give the primary level a heavier, bold styling. This means I've got to set any lists that follow back to their normal weight (see Code III).

As we discussed, descendants don't care where they come in the document tree. So once we changed the font weight back to normal, the next level follows suit. Last, we need to set the hover, active, and focus styles of our links (for further information about the accessibility reasons for these link styles as well as more information about descendants and styling and positioning with CSS, see my tutorial at Community MX – From Design to Completion: Case Study One – www.communitymx.com/abstract.cfm?cid=A5BE27AD9A15909B). Since I want all the rollover styles to be the same, I have to set them only once:

```
ul li a:hover, ul li a:active, ul li
a:focus {
 color: #C30;
 text-decoration: underline;
}
```

Save and upload the page. You should have three levels of blue links with orange bullets. The side bar list should be orange. The side bar list could be styled as buttons if you wish, using descendant selectors that begin with *#side*. There would be no conflict. See how powerful it is? In this example, the only extra information stored in the actual X/HTML portion of the document is two little IDs. Not a single class was harmed in the writing of this article. As always, go forth and style!

code I

```
<body>
<div id="content">
  <h1>This is the heading</h1>
  <p>This is the paragraph text.
<strong>This strong sentence should
be orange.</strong> And this one
should not.</p>
  <ul>
    <li>This is a regular list
item.</li>
    <li><strong>This strong list item
should be blue.</strong></li>
    <li>This is a regular list
item.</li>
  </ul>
</div>
<div id="side">
  <p>This is the side content.
<strong>This strong sentence should
be bold with no color.</strong> And
this sentence is normal again. </p>
</div>
</body>
```

code II

```
<body>
<div id="content">
  <h1>Site Map </h1>
  <ul>
    <li><a
href="javascript:;">Primary Level
Link 1</a>
      <ul>
        <li><a
href="javascript:;">Secondary level
link 1</a>
          <ul>
            <li><a
href="javascript:;">Tertiary level
link 1</a></li>
            <li><a
href="javascript:;">Tertiary level
link 2 </a></li>
          </ul>
        </li>
        <li><a
href="javascript:;">Secondary level
link 2</a> </li>
      </ul>
    </li>
    <li><a
href="javascript:;">Primary Level
Link 2</a>
      <ul>
        <li><a
href="javascript:;">Secondary level
link 1</a></li>
      </ul>
    </li>
    <li><a
href="javascript:;">Primary Level
Link 3 </a>
      <ul>
        <li><a
href="javascript:;">Secondary level
link 1</a></li>
        <li><a
href="javascript:;">Secondary level
link 2</a>
          <ul>
            <li><a
href="javascript:;">Tertiary level
link 1</a></li>
          </ul>
        </li>
      </ul>
    </li>
    <li><a
href="javascript:;">Primary Level
Link 4 </a></li>
  </ul>
</div>
<div id="side">
  <ul>
    <li>Side Item 1</li>
    <li>Side Item 2</li>
    <li>Side Item 3 </li>
  </ul>
  <p> This is side content in para-
graph form. And of course, if you
chose, you could go on and on till
half of forever.</p>
</div>
</body>
```

code III

```
ul li a {
 color: #005FBE;
 font-weight: bold;
}
ul ul li a {
 color: #004C97;
 font-weight: normal;
}
ul ul ul li a {
 color: #036;
}
```

Zoomin' on the Web

Add new dimensions to your applications

by john williams

If Flash is the death of HTML, Zoomify may be the death of JPGs, GIFs, and PNGs. While this sounds like marketing hoopla, zooms on the Web may change how designers and developers display images. Ultimately, this will enable quick, interactive access to high-resolution imagery on the Web, CD-ROM, and handheld devices creating richer, more informative, and more entertaining user experiences just by improving the way images are delivered.

Basically, Zoomify is a set of Flash MX 2004 APIs. Combine those with a "zoomified" image and you've got an SWF that allows users to pan and zoom. Creating an image is as simple as opening the Zoomifyer droplet. This .exe goes to work copying an image many times at different resolution levels, breaking it up into thousands of JPGs. When combined, the JPGs run resolution levels from thumbnail all the way up to the full resolution image, broken up into smaller tiles. This pyramidal tiling concept is popular with Scalado and Viewpoint, other companies with products in the Web zoom niche.

Although any image can be dropped onto the droplet for conversion, results are the most dramatic with images of large dimensions. The only size consideration, perhaps, is the 2GB restriction of Adobe Photoshop. Zoomify has no size restrictions. I have converted an image up to 1.5GB quickly and with no problems. The best part is the resulting folder of images was not even one-tenth that of the original image's size while maintaining high-quality imagery. The images should be the original image, not compressed or lossless. BMPs seem to work the best, but the program will work with TIFs, JPGs, and GIFs. The droplet will not convert PNGs.

The easiest route to creating a Zoomify image is to drag and drop the image to be converted into a folder of JPGs; make an HTML embed statement with parameters, such as initial view, zoom level, show/hide navigator window, show/hide toolbar, and other parameters; drop that folder of JPEGs, the HTML file, and a small SWF onto your Web server; and you're good to go. This set of files is available in the Zoomifyer 3 Extension folder, in Designer Tools/Simple Web Page. One strength of this scheme is that a single URL and single SWF can be used to deliver many images. With a URL that includes a question mark, parameters can be passed as a FlashVars parameter. For example, http://www.mySite.com/myDirectory/zoomifyURLDrivenWebPage.htm?zoomifyImagePath=http://www.mySite.com/content/myImageFolder/&zoomifyX=0.0&zoomifyY=0.0&zoomifyZoom=-1&zoomifyToolbar=1&zoomifyNavWindow=1.

The image path is specified; initial x, y, and zoom are set; and the toolbar and nav window are set to display. Many more parameters are available allowing basic functionality without the need to create separate zooms.

Zoomifyer for Flash MX extends this simple process into a workflow for creating high-impact Web sites. When installed, Zoomify components are added to the Flash Components panel and become available to be dragged and dropped into any movie.

Using an Image with Zoomify

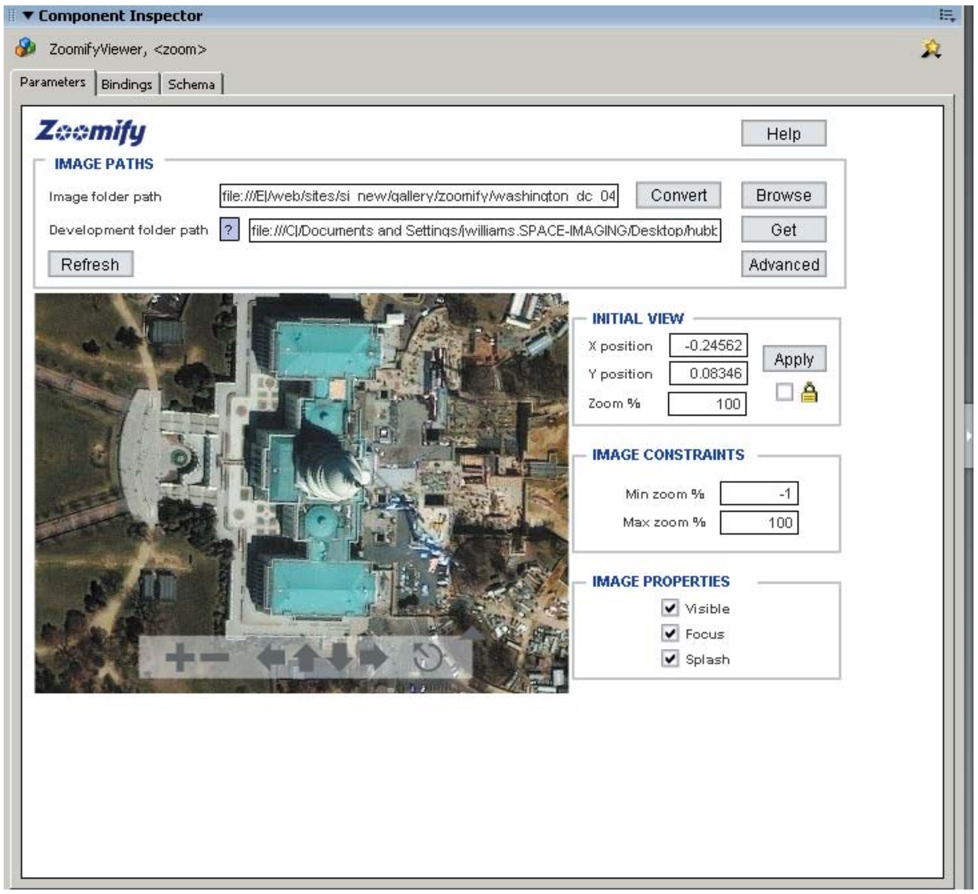
To use an image with Zoomify, create a development folder and place your folder of images into that folder. Start a new Flash project and save it to the development folder. Drag the ZoomifyViewer component onto the stage. Select the component and give it an instance name. Click the Launch Component Parameters Panel button and then browse for your image folder. *Note:* You may also choose to convert the image right here, by clicking the "Convert" button in the Component inspector. Click "Continue" once you've found the image folder. The image should appear in the window.

The development folder has been set relative to your Flash movie. You may find it useful to move the images to your Web server and type in the absolute path to the folder. Either way, your image should appear in the component inspector window so you may set other parameters if you wish.

Press Ctrl+Enter to test your movie. Pretty slick, but you can't move around the image.

Close the test window. Next, drag the ToolbarStandard component and drop it onto the stage. Placing it at the bottom-center of the image seems to be an intuitive place to get users to interact with it. One more step: we need to point it to the correct instance. In the Properties panel, select Target ZoomifyViewer and type in the instance name you gave the viewer. Test your movie and you'll be able to

"results are the most dramatic with images of large dimensions" image I



▼ Component Inspector

ZoomifyViewer, <zoom>

Parameters | Bindings | Schema

Zoomify

IMAGE PATHS

Image folder path: file:///E|/web/sites/si new/gallery/zoomify/washington dc 04

Development folder path: file:///C|/Documents and Settings/williams.SPACE-IMAGING/Desktop/hubk

INITIAL VIEW

X position: -0.24562

Y position: 0.08346

Zoom %: 100

IMAGE CONSTRAINTS

Min zoom %: -1

Max zoom %: 100

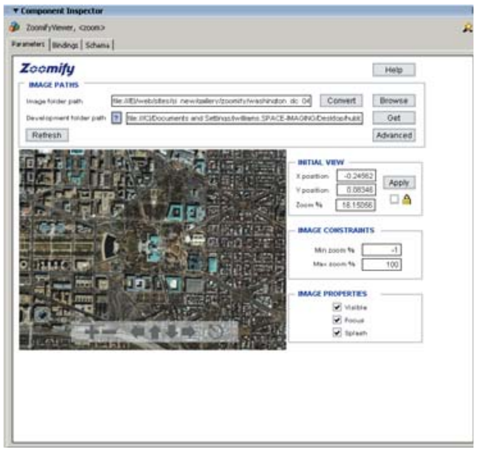
IMAGE PROPERTIES

✓ Visible

✓ Focus

✓ Splash

image II



INITIAL VIEW

X position: -0.24562

Y position: 0.08346

Zoom %: 18.15066

IMAGE CONSTRAINTS

Min zoom %: -1

Max zoom %: 100

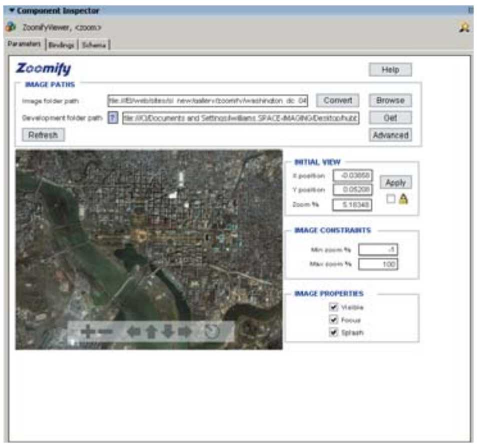
IMAGE PROPERTIES

✓ Visible

✓ Focus

✓ Splash

image III



INITIAL VIEW

X position: -0.03858

Y position: 0.05208

Zoom %: 5.18348

IMAGE CONSTRAINTS

Min zoom %: -1

Max zoom %: 100

IMAGE PROPERTIES

✓ Visible

✓ Focus

✓ Splash

zoom, pan, and check out the clarity of the image. You can also set "Show Slider" in the properties panel to "false" so the "slider" arrow is not visible. Add a bit of ActionScript and you can create a button to toggle the Toolbar on and off. Test the movie. We've just gone interactive. Images I – III show different zooms of the U.S. Capitol.

One thing to note about the StandardToolbar component: it's just an object. Open it in the Library and you'll notice you can tear it apart and re-skin the buttons, easily matching the look and feel of the application you're building.

Another incredibly useful component is NavWindow. Drag it onto the stage; upper-left is a pretty good location for this. Again, target the instance of the viewer. The NavWindow provides an overview of the image with a red box indicating the image boundaries in the main viewer. Clicking and dragging the red box provides a quick way to navigate a large image. You may resize the window to match your overview image by changing width and height in the component properties window. *Note:* the image that is used for the overview is a perfect image for use as a thumbnail. In your Web page, point your image link to your imagesFolder/TileGroup0/0-0-0.jpg.

You may notice other components there. ZoomScale provides a measuring component to your movie. This feature is used widely in conjunction with medical imagery, forensic images, and high-resolution satellite imagery – any image where scale might be important. Drag it onto the stage to get a feel for how it works. Again, target the instance name of the viewer. You should know the units of your image at the highest resolution. You may change pixels per unit, units per image, source magnification, and ruler units from yoctometers to yottameters. Test your movie and you should see this the scale update as you zoom in and out of the movie.

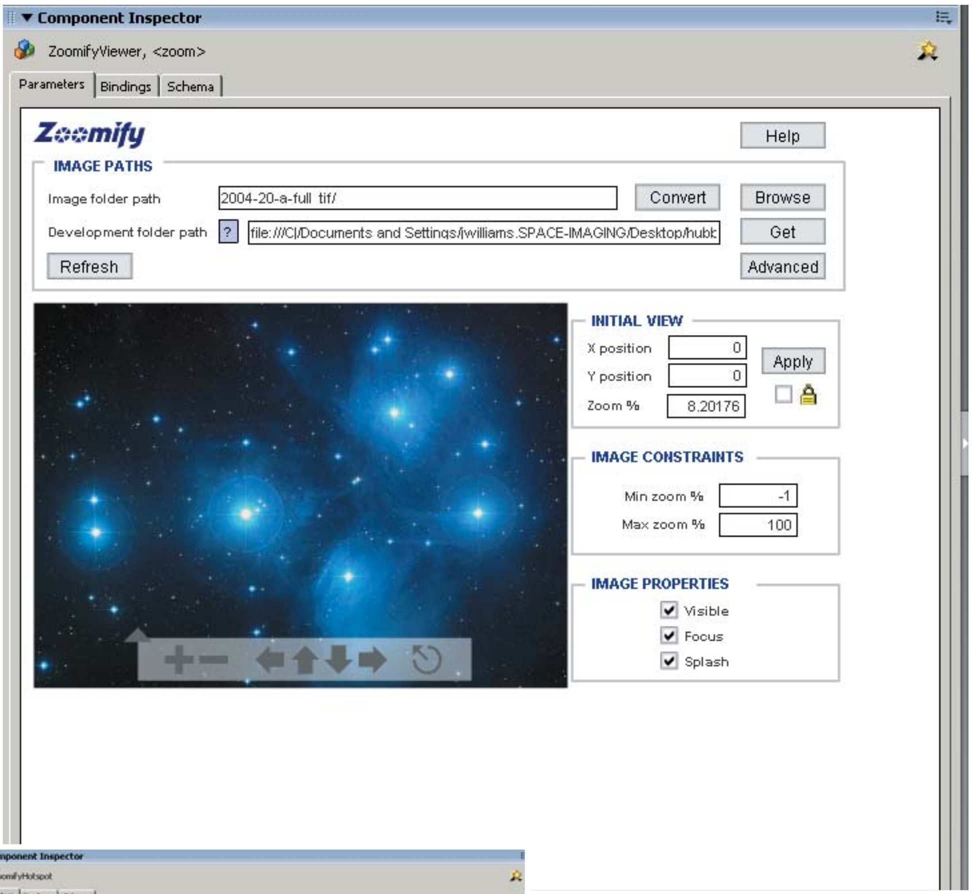
Other components in the toolbox include a hotspot creator and slideshow. Both are easy to use, but do require a bit of configuration. Drag the ZoomifyHotspot component onto your movie. The button that appears can be pressed to toggle the caption on and off by the user. Open Component Inspector for insight into all the configurations this tool has to offer. First target the Zoomify viewer instance you created. The Development folder path should be the folder where your project is located. Click "Get" to automatically retrieve the image path or folder of the viewer component. The image you've been using should appear in the window. Now it's time to set up some hotspots. Click "Add" and a box with text will appear. It may be easier to zoom into the area to which you want to add the hotspot, then click "add." Hotspot 0 should appear with that text, populating both "Edit selected name" and "Caption Text." Go ahead and edit the selected name and change caption text to whatever text you want to appear in the movie. You may use the nudge tools and zoom in/zoom out tools to position and size the hotspot text to your liking.

Test your movie. You can leave it at that or add a URL link that will take the user to that site when they click on the hotspot. You may change the text color and background color. Add as many hotspots as you wish in their appropriate locations. The only drawbacks to the text hotspots are lack of font control and the fact that you cannot create a transparent background. However, you can add GIFs or JPGs to the image as well as SWFs. Just type the path to the link to the image or SWF in the "Graphics/media URL." These can be linked to other URLs too. Images IV – VII show the Pleiades Hotspot created with Zoomify.

Zoomify Slideshow

Open a new project and save it to your working directory. Drag the "ZoomifyViewer" component onto the stage and name the instance "slideshow." This will become the window for our slideshow. Now drag the "ToolbarStandard" into the bottom-center of the image so users can interact with the image. Finally, drag "ZoomifySlideshow" onto the stage. Don't forget to target both the toolbar and the slideshow components to the main ZoomifyViewer instance you first dragged onto the stage. Click "Get" next to the "development folder" input area and the development folder should auto-populate. Now you can add views to your slideshow. You can target a zoom you've already created, or the inspector will allow you to convert an image while you work. Select the area of your initial zoom and the zoom level and the next step is to select the transition effect and the slide interval. Test your movie to view the slideshow.

image IV


▼ Component Inspector
ZoomifyViewer, <zoom>
Parameters | Bindings | Schema

Zoomify — Help

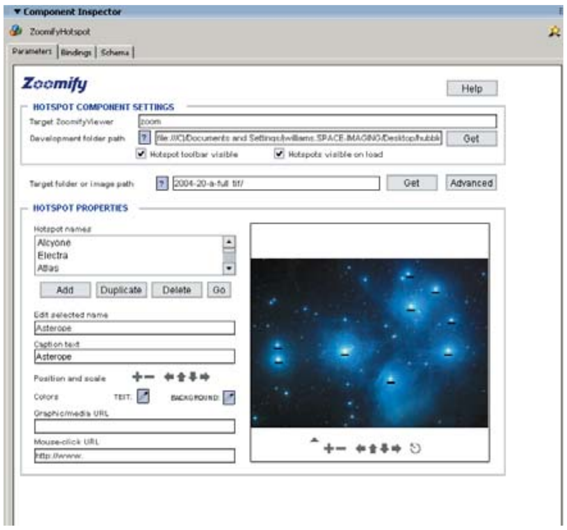
IMAGE PATHS
Image folder path: 2004-20-a-full tif/ — Convert — Browse
Development folder path: ? file:///C|/Documents and Settings/jwilliams.SPACE-IMAGING/Desktop/hubb — Get
Refresh — Advanced

INITIAL VIEW
X position: 0
Y position: 0
Zoom %: 8.20176
Apply

IMAGE CONSTRAINTS
Min zoom %: -1
Max zoom %: 100

IMAGE PROPERTIES
✓ Visible
✓ Focus
✓ Splash

image V


▼ Component Inspector
ZoomifyHotspot
Parameters | Bindings | Schema

Zoomify — Help

HOTSPOT COMPONENT SETTINGS
Target ZoomifyViewer: zoom
Development folder path: ? file:///C|/Documents and Settings/jwilliams.SPACE-IMAGING/Desktop/hubbl — Get
✓ Hotspot toolbar visible — ✓ Hotspots visible on load

Target folder or image path: ? 2004-20-a-full tif/ — Get — Advanced

HOTSPOT PROPERTIES
Hotspot names: Alcyone, Electra, Atlas
Add — Duplicate — Delete — Go
Edit selected name: Asterope
Caption text: Asterope
Position and scale
Colors — TEXT: — BACKGROUND:
Graphic/media URL:
Mouse-click URL: http://www.

Feature Combinations

Creating your own combinations of features is possible through a very powerful and well-documented set of APIs. One of the most approachable is zoomToView and is demonstrated in the examples included in the download as "Smooth Transition."

Let's see how this might work with some ActionScript. Create a new project and save it. Drag the ZoomifyViewer component onto the stage and name it zoom1. Drag "ToolbarStandard" component onto the stage and target the viewer instance, zoom1. Now that gives users the simple interactive interface. What if we wanted users to go to a specific spot and give them a nice ride while we're at it? Let's drag a button component onto the stage and attach the ActionScript shown below.

```
on(release)
  {
    _root.zoom1.zoomToView(0.042,
0.12, 200, 5000, 10,"");
  }
```

This is telling the movie that when the button is pressed, set the Zoomify movie instance to zoom1. The parameters indicate destination X, destination Y, zoom value, duration, interval rate, and callback. The interval rate is expressed in milliseconds. In this case, an update of position will occur every 10 milliseconds or 100 times per second. With the duration set to 5000, there will be 500 updates during the total transition.

Another easy yet powerful effect can be created by combining getX, getY, and getZoom to create a side-by-side zoom. Drag a ZoomifyViewer component onto the stage, name it zoom1 and select the zoom image you wish to display in that window. Drag the ToolbarStandard onto the stage and target zoom1. Drag another ZoomifyViewer onto the stage, name the instance zoom2 and select the image that should display in that window. Select the zoom2 instance, and attach the ActionScript shown below.

```
onClipEvent (enterFrame) {

_root.zoom2.setView((_root.zoom1.getX(
)), (_root.zoom1.getY()),
(_root.zoom1.getZoom()));
  _root.zoom2.updateView();
}
```

Test your movie and you'll see that as

you interact with the first zoom, the second movie updates instantaneously.

. . .

So is that it? We've barely scratched the surface. Check out these Web sites for some inspiration:

- www.spaceimaging.com/gallery/top10_2003/
- www.getty.edu/art/exhibitions/flemish/home.html
- www.lewisandclarkexhibit.org/
- www.moma.org/kikismith/
- http://128.250.125.178/
- www.markfennell.com/panoramas/
- imagearchive.compmed.ucdavis.edu/

With Zoomify's interaction with panoramas and 3D objects as well as integration with Flash's data grid components and XML, developers and designers can create new dimensions to more robust, dynamic, and versatile applications.

Note: Zoomify is offering a free copy of Zoomifyer for Flash v3.0 to MXDJ readers. Simply e-mail president@zoomify.com and mention MXDJ. No registration required, no strings.

John is a currently Web developer/designer with Space Imaging. While not drooling over amazing images of Earth or teaching HTML and Flash courses, he shovels snow and cuts wood in the foothills near Denver. jrwilliams@spaceimaging.com

image VI

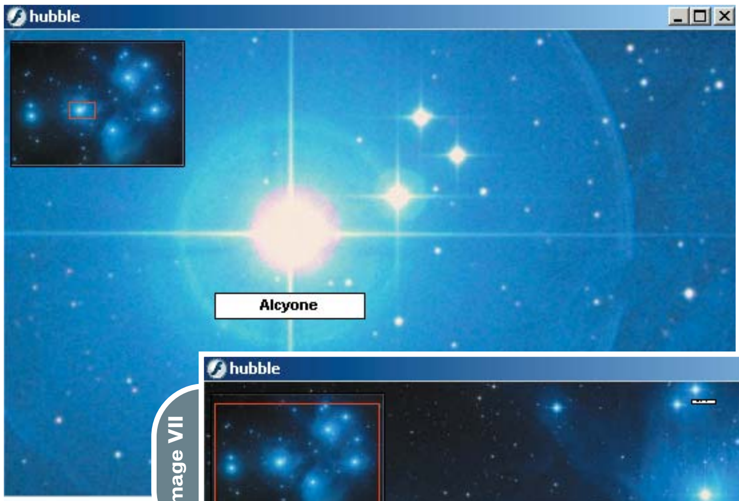
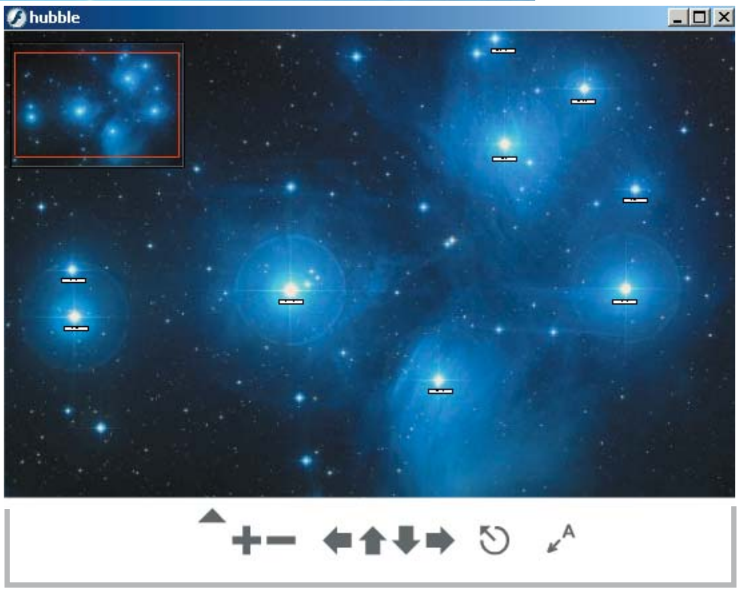


image VII



Introducing the new FuseTalk.

Flexible Security, 508 Compliance, Offline Capabilities, Reporting, Easy Integration and much more...

1-866-477-7542

CFDJ READERS' CHOICE AWARD

2003 CFDJ Readers' Choice Award Winner for Best eBusiness Software & Best Web Application

www.fusetalk.com

FuseTalk™

Discussion forum solutions that make web-based collaboration risk-free and easy.

Flash Virtuosos

the best in web design

What do you get when you combine a few hardcore design enthusiasts with the thousands of impressive portfolios and Web sites of artists around the world? The American Design Awards (ADA).

American Design Awards™

REWARDING INNOVATIVE DESIGN POTENTIAL

Established in February of 2000 in San Diego, California, the ADA has two contests for graphic and Web designers of all experience levels – a monthly contest and an annual contest. The founders of the ADA were overwhelmed by the abundance of talent they observed when reviewing portfolios of prospective employees and decided to find a way to help these designers obtain the much-needed recognition that would perhaps help them in securing a career in the field of visual arts.

Today, ADA has over 20,000 active members and participants, from not only the United States, but also Canada, Europe, the Middle East, Asia, Australia, and South America, who have found the confidence and support to carry out their dreams, with the backing of ADA.

The annual contest is open to all kinds of entries, from Web design, logo design, complete corporate identity packages, marketing material, posters, packaging art, illustrations, and more. Any form of graphic or Web design has a place in the contest.

The monthly contest, however, is all about the Web, and is open only to actual Web sites. Between 1,200 and 1,500 sites are submitted each month and the top 5% are presented an award. There is no cost to enter the monthly competition.

Flash and the ADA

It stands to reason that since the monthly contest is open only to Web sites, you'll see a lot of Flash sites among the highest scorers. ADA is not a "Flash Web site" contest, but according to Kevin Javid of ADA, more than half of the sites scoring 90 or higher in the ADA competition use Flash to some extent. Although the team at ADA does specifically look for Flash in the design, according to Javid, "using it in a tasteful and creative manner certainly helps propel the score up by 5 to 10 points."

For Daniel J. Ferkul of Ravalink Corp. (January 2004 winner), Flash provided the freedom to envision a site and create it, without worrying about technology limitations. "Flash allowed us to create a site that was not static, but interactive...it gave us the opportunity to inhibit limitations in our vision and creation...time was our only limitation." In fact, Ravalink's site was done entirely in a few weeks by only one of the talented creatives. Even with an award-winning site, the creative minds at Ravalink won't stop there and are already looking forward to their next redesign. "As with anything we create, halfway through we are saying, 'Man, I wish we could start over!' Oh, the way the creative mind works...we are never happy!" says Ferkul.

A key feature that attracts these artists to Flash is the integration of video in Flash, which can be just as much fun for the artist during the creative process as it is for the audience. Just ask Lee Walters, aka Lee Bones of lee.bones.com (January 2004 winner) about his first experience with video and Flash. "I went to a fabric store and purchased several yards of green fabric. The color of the fabric was very similar to the green fabric used for special effects (green screens) in TV and movies. I then hung the fabric up in my apartment, set up the video camera and lights, turned on the record player, and performed every stupid dance move I could think of. This was the first time I had tried anything with video and Flash together so it proved to be very fun!"

The use of Flash creates an extraordinary experience for Web site visitors and makes it pretty easy for designers and developers to ensure that users will see the sites as intended. "Since Flash is cross-platform, as long as the user has the latest Flash plug-in, we don't have to worry about browser compatibility issues like alignment and fonts. This allows us to explore new interface designs and layouts," says Amie LaRocque of Sumo Creative Services (March 2004 winner). And with the Flash player installed on more than 97% of Internet-enabled desktops, it's a good bet that users will see the site just as the artist envisioned.

From a development standpoint, Flash is a winner as well. Quentin Fountain of omegaDawn grafix (April 2004 winner) says, "We wanted to develop a site that was extremely user friendly, easy on the eyes, simple yet still exciting to use. And from an internal development standpoint, we wanted something that would be easier to develop and maintain than our previous site. With the use of Flash and its ability to work with PHP, XML, and MySQL, this was extremely easy." And it's only going to get better with the enhancements introduced in Flash MX 2004. "We're seeing a great deal of sophisticated and intricate Flash animations as well as seemingly faster loading pages. We are also finding a good deal of functionality and customization that allows for the user to interact with an otherwise ordinary interface," says Javid.

Many of the artists featured here chose Flash for similar reasons, but when asked about their inspiration, they gave answers as varied as the sites themselves, ranging from their company's branding, to a new challenge, to their "favorite things." Thousands upon thousands of artists are using the same technology, for largely the same reasons, to create a very unique representation of themselves and their work. Says Walters, "I didn't want to make something that wasn't me...because I was selling me." The images on the following pages here are accompanied by statements from the artists about what inspired them, and the URL so that you can, of course, get the full experience.

If they inspire you to enter your own site, all the better. When asked to provide some advice for newer designers, Javid had this to say: "Balance! Balance is very important not only in colors and layout, but also in Flash. Sometimes less is more and unless properly planned, 'more' can lead to a Web site's being too busy and cluttered, unfriendly to the user, and to most modern users, slow and cluttered."

And Romain Gruner (December 2003 winner) puts it very simply: "To succeed in this business we need to be better or different. I choose the difference."

To learn more about the ADA, please visit www.americandesignawards.com.

ADA Criteria

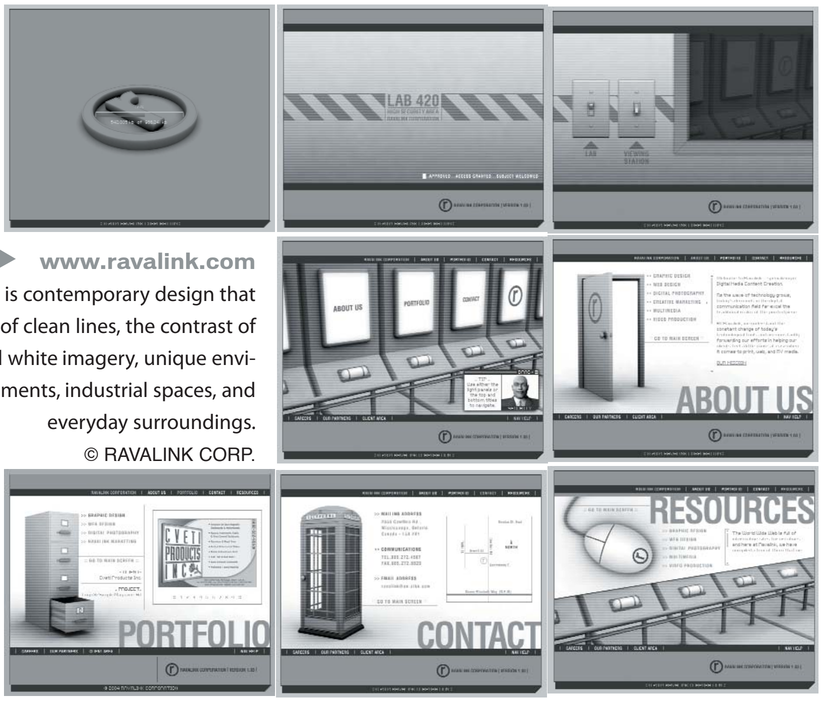
Each design piece is graded based on the degree of creativity (30 points), design potential (40 points), and originality of idea (30 points). Based on the final grade received, the contestant will receive:

1. ***Platinum Award:*** This award is presented to the top 5% of our designers, with a score of 90–100.
2. ***Gold Award:*** Presented to designers who score between 80 and 89 on our judging score sheet.
3. ***Silver Award:*** Presented to designers who score between 70 and 79 on our judging score sheet.

www.filigrooves.com

I am a big retro fan in general. I really appreciate the creative accomplishments of the past. I think that today's designers, musicians, architects, etc. have a lot to learn from the past decades. I'm not talking about imitating, but letting yourself be inspired by the works of others in order to create something new.

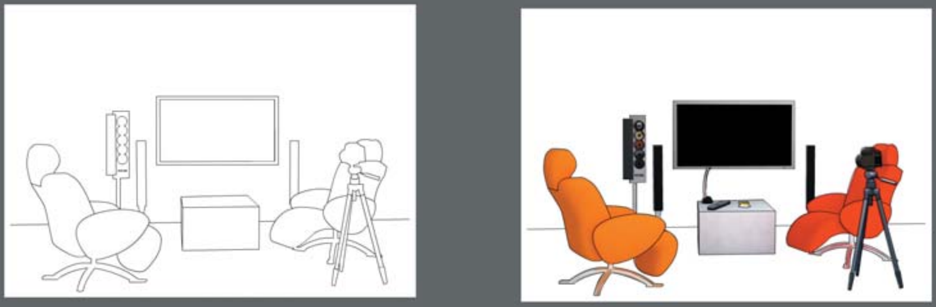
© FILIGROOVES



www.ravalink.com

Our muse is contemporary design that consists of clean lines, the contrast of black and white imagery, unique environments, industrial spaces, and everyday surroundings.

© RAVALINK CORP.



www.romaingruner.com

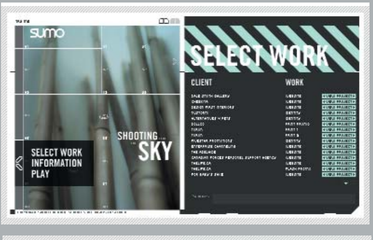
My inspiration was to show magazines (architecture, design, mode...). I wasn't inspired by other Web sites, only print.

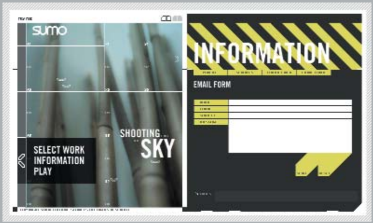
© ROMAIN GRUNER

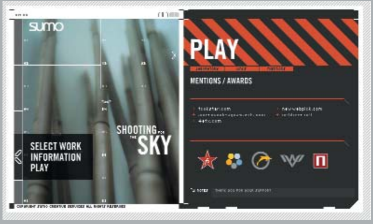
www.omegadawn.com

Our inspiration was the need for change and the desire for an easily maintained, database-driven Web site.

© OMEGADAWN GRAFIX, INC.



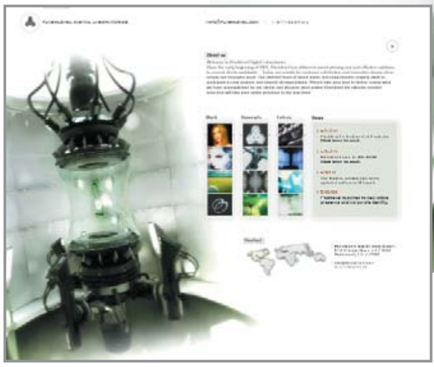




www.sumoxl.com

We based everything on our branding. A lot of the elements you see on the Web site are also found throughout our collateral.

© SUMO CREATIVE SERVICES

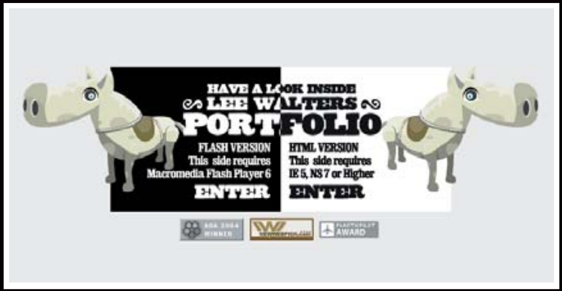


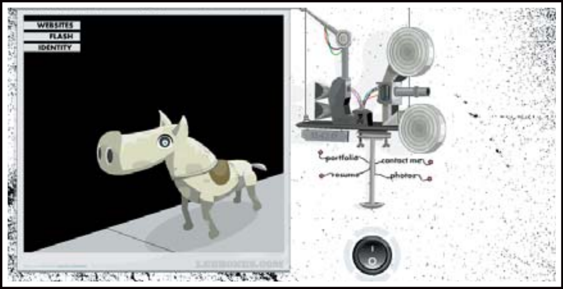
www.flashlevel.com

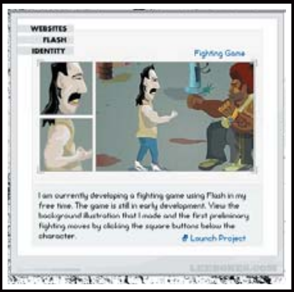
We are inspired by technology, motion pictures with amazing special effects, nature, and of course the challenge of a new project.

© FLASHLEVEL, INC.

Back
Username
Password
Sign up
Enter

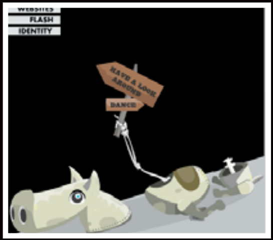






I am currently developing a fighting game using Flash in my free time. The game is still in early development. View the background illustration that I made and the first preliminary fighting moves by clicking the square buttons below the character.

Launch Project



www.leebones.com

I love antiques, cats, parades, film projectors, plastic buttons, abandoned buildings, stupid dance moves, illustrations, and drawing. So when I thought about all of those things put together, the end product was leebones.com. I didn't want to make something that wasn't me...because I was selling me.

© LEE WALTERS

CSV in Flash MX 2004

Make your classes easy and flexible

by danny patterson

CSV is a format that has been around for a long time. It's a simple way to store multiple rows with multiple columns and it can be easily imported into a variety of commonly used desktop applications. This article examines the use of CSV-formatted data in Flash. We will write a class in ActionScript 2.0 to handle all of our CSV functionality and then we will use this class to populate the Macromedia v2 Data Grid Component.

Why Use a CSV File?

CSV is a very simple format. At its core it is essentially a large string that has rows separated by a delimiter and columns within each row delimited by a comma. Because of its use of rows and columns, this data format is perfectly suited for record sets, whereas an XML-formatted record set is more complicated.

Another good reason to use CSV is that it can be viewed by many common desktop applications. Any spreadsheet program, such as Microsoft Excel, can view, modify, or create CSV files. Unlike an XML-formatted record set, users can open a CSV file and easily read the information.

Like XML, CSV is data formatted within a file. It's not tied to any specific software or platform and it doesn't need a server in order to deliver it to the client. This makes CSV and XML files a very portable way to handle data. You could build a Web- based Flash application using a CSV or XML file and then easily use that same code to build a CD-ROM–based application.

CSV will not always be the right solution for your project. XML is usually a better solution for delivering data to the client because of its versatility. Any time your data is not formatted in a series of rows and columns, CSV should be avoided. You should also refrain from using the CSV format if you have multiple levels of complex data types. For example, if you have an array or a list in one of your cells, you should probably use XML to define your data. XML is better at separating out different data types and handling multiple levels of embedded data. CSV should only be used for primitive data types (String, Number, and Boolean) that are housed in the row-and-column format.

CSV Specifications

The first thing you'll notice about a CSV file is that it has rows and columns. The rows are delimited by a line break, a carriage return, or a combination of the two. The columns within each row are separated by a comma. Many times the first row of the file is used to define the column headers; we'll talk more about that when we begin parsing the data in our custom class.

Many CSV files will use something called a qualifier, the most common qualifier being the double quote ("). This character is used to surround each column's data. Below you can see the difference between a file with a qualifier and one without a qualifier.

CSV data without a qualifier:

Field 1,Field 2,Field 3,Field 4
Field 5,Field 6,Field 7,Field 8

CSV data with a qualifier:

"Field 1","Field 2","Field 3","Field 4"
"Field 5","Field 6","Field 7","Field 8"

Building the CSV Class

Before we begin programming our CSV class, we need to define exactly what we need this class to do:

1. The class will need to load in an external CSV file.
2. The class will need to parse the data into a Flash object. The format we want to get the CSV data into would be an array of objects (or associative arrays), with the object's key being the column header.
3. The parsing will need to be able to pull the column headers from the first row of the CSV; however, we will also want to give the user the option to override this feature by specifically defining the columns in an array prior to parsing the file.
4. We also need the parser to be able to handle qualifiers. We will allow the user to define the qualifier prior to parsing the CSV data.

We'll start our class by defining it. Our class won't be using the constructor for anything, so we can leave that method empty.

```
class CSV {
 function CSV() {}
}
```

Next we need to define our class's parameters:

- csvData is a private array that will hold the parsed CSV data.
- onLoad is a public method that the user can overwrite to get a callback when the data is loaded and parsed.
- columns is a public array that the user can define prior to parsing the CSV data to manually define the column headers.
- qualifier is a public string that the user can define prior to parsing the CSV data. Qualifiers assist with more accurate parsing. We will default this value to an empty string.

```
private var csvData:Array;
```

```
public var onLoad:Function;
public var columns:Array;
public var qualifier:String = '';
```

The first task that we need our class to do is load data in from a file. We'll accomplish this by creating a load method, which will simply take in the CSV file's path as its only parameter. The load method will use the LoadVars class to load the file into Flash, but we will actually incorporate an undocumented method within the LoadVars class, called onData. This method works exactly like the XML.onData method. It allows us to get the raw file data without having it parsed by the LoadVars class.

```
public function
load(csvPath:String):Void {
 var csvLoad:LoadVars = new
LoadVars();
 csvLoad._parent = this;
 csvLoad.onData =
function(rawData:String) {
  this._parent.onData(rawData);
 }
 csvLoad.load(csvPath);
}
```

Notice that in our LoadVars.onData method we are calling a method within our class, called onData. We will create that method to handle the parsing of the CSV data after the file has loaded. Once the data is parsed it will pass it back to the user through the onLoad method. We won't actually define the onLoad method. To handle the load method callback, the user will define this method. Users could also override the onData method if they didn't want the data parsed right away.

```
public function
onData(rawData:String):Void {
 csvData = parseCSV(rawData);
 onLoad();
}
```

Next we'll add the parser method. We'll make this method public so users can utilize it without loading in the CSV data from a file. The parser is the most complicated part of this class – it is also the part of the class where the most can go wrong.

Our parser method will take a raw CSV string as its sole parameter and return the parsed data as an array of objects. The first two things our parser method must do is determine the row and column delimiters. We already know that the row delimiter is a carriage return (\r), a new line (\n), or both (\r\n). A quick search of the string will help us determine what the actual row delimiter is. The column delimiter is always a comma in the CSV specification; however, we can make our parsing more accurate by adding the qualifier before and after the delimiter. This new concatenated string will be our column delimiter.

To begin parsing the CSV string we will split the string into an array of its rows. This will allow us to loop through the rows and parse each row individually.

If the column's parameter has not been defined prior to running this parser method, then we can assume the first row in the CSV file holds the column headings. We will populate the column's array from this row.

As we loop through the rows we will split each row into an array of its columns. If the row doesn't have the same number of columns as the columns parameter, then we will skip that row. As we loop through the columns within each row we will place the data into an code I

```
public function parseCSV(rawData:String):Array {
 var ii:Number, columnArray:Array, rowObject:Object;
 var returnArray:Array = new Array();
 var rowDelimiter:String = (rawData.indexOf('\r\n') > -1) ?
'\r\n' : (rawData.indexOf('\r') > -1) ? '\r' : '\n';
 var columnDelimiter:String = qualifier + ',' + qualifier;
 var rowsArray:Array = rawData.split(rowDelimiter);
 if(!columns.length) {
  columns = removeQualifier(rowsArray.shift().toString(),
qualifier).split(columnDelimiter);
 }
 for(var i:Number = 0; i < rowsArray.length; i++) {
  columnArray = removeQualifier(rowsArray[i].toString(),
qualifier).split(columnDelim);
  if(columnArray.length == columns.length) {
   rowObject = new Object();
   for(ii = (columnArray.length - 1); ii >= 0; ii--) {
    rowObject[columns[ii]] = columnArray[ii];
   }
   returnArray.push(rowObject);
  }
 }
 return returnArray;
}
```

code II

```
private function removeQualifier(originalString:String,
qualifier:String):String {
 var modifiedString = originalString;
 if(modifiedString.charAt(0) == qualifier) {
  modifiedString = modifiedString.substring(1);
 }
 if(modifiedString.charAt(modifiedString.length - 1) ==
qualifier) {
  modifiedString = modifiedString.substring(0,
(modifiedString.length - 1));
 }
 return modifiedString;
}
```

object using the column heading as the key. We will then append this object to the end of the array that we will be returning (see Code I).

We also need to create a private method called removeQualifier that supports the parser. This method will pull out the qualifier at the beginning and end of each row (see Code II).

In order to expose the private csvData parameter, we will create a getter method called data. This method will simply return the parsed data to the user. We do this to protect the csvData parameter from being set outside the class.

```
public function get data():Array {
 return csvData;
}
```

Using the CSV Class with the Data Grid Component

In this example we will use Macromedia's Data Grid Component to display our CSV data. The data grid's dataProvider parameter accepts an array of objects as a valid value, so we won't have to manually populate the grid. The first thing we need to do is place an instance of the data grid on the stage, then give it an instance name of dataGrid_mc. Then we will create a new layer and name it Actions. We will put all of our code in frame 1 of the Actions layer. Make sure your CSV.as class file is in the same folder as your FLA file.

In frame 1 of our Actions layer we will begin writing our code, creating an instance of the CSV class.

```
var csvLoad:CSV = new CSV();
```

Next we define our columns and qualifier parameters before we parse the CSV file.

```
csvLoad.columns = new Array('Column
1', 'Column 2', 'Column 3');
csvLoad.qualifier = '"';
```

We then define our onLoad method, where we populate the data grid with the parsed CSV data. Finally, we call the load method and send it the path to our CSV file as an argument.

```
csvLoad.onLoad = function() {
 dataGrid_mc.dataProvider = this.data;
}
csvLoad.load('sampleData.csv');
```

Conclusion

I hope this article gives you an idea of how you can use CSV data within your Flash application. But more important, I hope it has given you a concrete example of how you can build an ActionScript 2.0 class and how you can make your class easy and flexible to the developers that implement it. This CSV class may not be the solution for your application, but the principles of object-oriented programming should be used in nearly every application you build with Flash.

Note: Class code can be downloaded from www.sys-con.com/mx/sourcec.cfm.

Danny Patterson (www.DannyPatterson.com) is the senior Flash engineer for POPstick (www.POPstick.com) in Boston, and has been developing for the Web since 1996. He is a Certified Advanced ColdFusion MX, Flash MX, and Flash MX 2004 Developer and works with the Flash community as a member of Team Macromedia. dpatterson@popstick.com

It’s everybody’s PDF™

Finally, a software company that offers affordabe 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!

activePDF®
Leading the iPaper Revolution

www.activePDF.com

Creating Games in Macromedia Flash MX 2004

Escape from reality

by glen rhodes

MACROMEDIA FLASH MX 2004 GAME DEVELOPMENT
• Covers the entire game creation process from design and story, to sound, physics, optimization, 3D, and deployment
• Teaches the fundamental programming techniques needed for creating a variety of games from simple to complex
• Uses eight fun games to teach Flash MX 2004 game programming
• Includes a CD-ROM with all of the files needed to create the eight games from the book, including graphics and sound files
Game Development Series
GLEN RHODES

Creating games in Macromedia Flash MX 2004 is one of the most rewarding endeavors an aspiring or seasoned programmer can pursue. People could choose to develop games using Flash MX 2004 for the sheer enjoyment of seeing a creative idea come to life, a specific project for a client, an exercise to learn the details of programming in Flash, or as a way to express themselves artistically.

Before we look at the motivation, let's briefly look at the history of computer games in general.

A Brief History of Games

Ever since the early days of humankind, games have been an integral part of our existence. Games allow us to escape from reality, while at the same time allow us to mimic reality in ways that we might never be able to experience in real life. My perfect example is when I ran around the fields as a child playing soldiers in a war, or explored far-off lands in my backyard.

Enter the computer. The computer allows us to delve into worlds as never before. The computer can be a portal into distant lands, and can place us in the storylines of the greatest science fiction and fantasy stories.

Before the electronic game era, we had mechanical pinball machines. Pinball is sometimes considered a precursor to modern video games, and moreover as the catalyst that spawned the first general interest in computer games. Looking back at the history of games, it is amazing just how far they have come in a relatively short period of time.

In 1958, Willy Higginbotham, a physicist who'd worked on the Manhattan Project in 1947, created the world's first electronic game. It was a basic tennis game entitled Tennis for Two. In the 21st century, the use of the third dimension in games is almost a necessity. However, over 50 years ago this 2-dimensional game showed how electronics weren't limited to performing business and scientific functions, but could be used for entertainment as well. Although Tennis for Two felt like a computer game, it was really hardwired electronic circuitry set up through an oscilloscope, and did not technically involve a computer.

The origins of the modern computer game can be traced back to the early days of computer science and the invention of the transistor. At this point, the computer actually became somewhat accessible to enough people to warrant making the first games. These games were very simple in their nature, and the earliest example of a true computer game, Spacewar! was created in 1961 to run on the first PDP-1 computer at MIT. This game consisted simply of a CRT screen showing a few dots meant to represent spaceships, which were controlled by primitive handmade joysticks, but even then the potential for future game advances became evident.

Ralph Baer, considered by many "The Father of Video Games," was at the forefront of video game development in the 1960s. A pioneer of the television itself, it was his goal to merge both the computer game and television to create the home console video game. His earliest achievement was the chase game, which consisted of two objects, such as a fox and a hound, moving along the screen. By the late 1960s, Baer and his team of developers created a shoot-the-dots style of game, and followed that with a table-tennis game for two players. Ultimately, what Baer developed was what he referred to as the "Brown Box," which was the original incarnation of the home-based console. PlayStation 2 and Nintendo GameCube are examples of today's home-based consoles.

By the 1970s, a new but humble computer game industry had emerged. Games such as Pong and Space Invaders filled the conversations in many homes. One of the largest companies that emerged from this revolution was Atari. Behind it all was video game pioneer Nolan Bushnell. Although other creators, such as Baer, believed that the idea for Pong was taken from earlier forms, it was Atari that marketed the game first. Pong to this day is still a landmark, a milestone in the advancement of the gaming industry. The 1970s and early 1980s also saw the exponential rise of other games such as Pac-Man and Frogger.

The emergence of handheld games (today's equivalent being the Nintendo Game Boy) unfortunately did not help the already growing stigma that video games were too violent or anti-social. This concept originated in the 1970s and it is still a controversial issue today. Nevertheless, all these achievements paved the way for advancement in the computer gaming industry in the coming decade.

By the 1980s, arcade games were the central hub of an entire cultural generation. Children filled the arcades, spending most of their pocket money so that they could try the new Donkey Kong, Ms. Pac-Man, or the improved version of Space Invaders through game systems made by such manufacturers as Galaga and Galaxian.

Perhaps the most important development allowing for the rapid advance of video games was the affordability of home computers that first occurred in the early 1980s. In 1982, the Commodore 64 was considered by many the computer to have. Many games had become a permanent fixture on the Commodore 64, for example The Olympics and Mission Impossible. Eventually, the personal computer allowed for the everyday person to learn and use computer programming in their home, thus opening the door to many an aspiring programmer to begin creating their own PC games, and players playing at home rather than solely at the arcades.

In the years that followed, games have expanded into many areas of life, and have spawned many genres from works of fiction and nonfiction. Today, many of the best games feature real-time rendered 3D graphics with thousands of polygons of detail, millions of colors, lighting, reflection, and full surround sound. In this book we'll be looking at several genres of games including the top-down action game, the side-scroller, the behind-the-player 3D, and others. We'll be pushing Flash to its limits.

Motivation

Whatever your reason for making games, Flash provides the perfect medium to do so. Its interface is very straightforward and easy to use, while its programming language is robust and powerful, yet simple once you get the hang of it.

On top of all that, creating a game with Flash allows you to deploy your games to millions of people within minutes, giving game developers today more opportunity to make a name for themselves and gain exposure than ever before.

Enjoyment

There's no denying it; it's fun to make games. Once you've overcome the technical hurdles, nothing is more satisfying than watching your creations and ideas come to life.

The Potential

One of the most critical roles that Flash plays today is to provide people with the ability to entertain an audience. In the online world of mass marketing, this audience translates to advertising revenue. When this happens in a medium such as the Web, we inevitably see an emergence of corporate clients who want to make use of that technology. In the case of Flash MX 2004, games are a perfect way to generate advertising revenue through the Web.

In the Words of Craig Swann

Value of Gaming in Today's Industry

Craig Swann, CEO of CRASH!MEDIA (www.crashmedia.com), had these words to say on the importance of games in Macromedia Flash in the modern online world.

"There should be no doubt about the importance of gaming in the online community. In an industry in which content is king and the medium is nearly as flexible as thought itself there is almost no limit to what content can be. Since the birth of the World Wide Web, we have seen a shift in people's experiences. Before the Web, we were living in a world in which we received our content in a 'lean-back' form of communication. Very little was expected of us. So there we sat on our couches, in a vegetative state, leaning back and accepting whatever was broadcast to us as information; the only interaction being the clicking of the remote. When the Web came along everything changed. We entered the age of 'leaning forward.' We began leaning into our screens, searching for information. We were slowly gaining some control, and more importantly, interest in getting involved with data, information, images, text, videos, and of course games.

"I was lucky enough to have been born alongside the birth of mainstream video games in the early 80's. Pac-Man, Donkey Kong, Frogger, Space Invaders. I was in the arcade popping quarters into the machines on my tippy-toes when they first came out. I've seen these classic games transformed over the decades into games that are now very close to real-life multi-player simulations. It's incredible, and the one thing that has continued to advance is the interactivity of the games – the feeling of the games and how they are played.

"Games take us into a world, the world of the developer. Through the clever use of motion, sound, music, animation, and most importantly interactivity, entire worlds are created. Games take us to another place. Our minds focus on the environment of the game – everything else fades out in our perceptions. It really is amazing how mentally involved and focused we become when we play games. This is one of the main reasons games can play an important role in online development of interactive brands. Games give something. As developers, any time we create something that goes online we are vying for a piece of a person's life. Think about that: their time; their existence. Yet games are something that most of us are guilty of spending hours and hours playing. A good word for it is 'escapism.'

"Without a fancy survey to back me up, and going only on my personal experience of eight years on the Web as both a developer and surfer, I would have to say that as far as entertainment on the Web goes, games take the cake. In my experience, online games are the number one overall activity for people around the world looking for fun on the Web – clean fun that is! This is largely due to the very existence of Macromedia Flash. People have been producing the most amazing variety of games using Flash for years now. Flash, due to its ubiquitous nature, has evolved rapidly to become the number one platform for online game development."

Advantages of Flash as a Platform for Games

"Macromedia Flash is an excellent tool for creating games due to several factors. First, its scripting environment is sophisticated enough to enable real-life physics and motion element to games. Secondly, being a vector-based application allows for the creation of tiny graphic elements. This means that sophisticated interactive games can be created and experienced with a very little effect on bandwidth. Whether on an ultra-fast optical T3 connection or on a 28.8 dial-up modem, Flash games, as long as they are created properly, can provide the same experiences.

One of the unique aspects of games is the fact that they take us into a world, a world where, as developers, we create the rules. We create the characters and the environment. When you have the ability to code an environment with its own sets of natural laws and physics, you truly do create unique interactive experiences. The rich integration of sound and music games allow you to go even further and create strong experiences that tie in all of the senses, as they are tightly focused on gameplay.

This sort of intense user focus can be channeled to help strengthen a brand or concept to to the user. Unlike the average visiting times of other types of Web sites, users will often spend as much as an hour playing games. This unusual length of time on an Internet property offers many ways to communicate to the user, whether it is through the game concept itself, or through interstitials and other ads, and marketing.

The Internet by its very nature is viral. We tend to hear about things and check them out. The ease of communication on the Internet allows us to easily pass along ideas as well as share things that we find in this seemingly never-ending universe. Games are an amazing way to get people to talk about something and participate. Contesting for high-scores has proven to be an excellent way to generate traffic to a site. What better way is there to win something than by playing fun games!

With Flash, we don't have to stop at single-player games either! There are many ways now possible including XML sockets and the Flash Communication Server to facilitate real-time multiplayer games. This is where things start to get really interesting! Today we can create games and environments that can involve any number of people based on the concept of the game. From our experience, we had the opportunity to create a multiplayer game as part of contesting for a large beverage distributor. The game allowed people from around the world to play and battle each other head to head in real-time. The ability to experience an event with someone halfway across the world is something new to us, something that is continuing to grow and flourish. This is the direction we are headed. To communicate, collaborate and most importantly play with each other. We are living in fabulous times – where the power is truly in our hands to create the very things we envision. It blows my mind to imagine what sort of ways we'll be gaming and sharing our lives with people through the amazing ways we can create games and experiences with Macromedia Flash MX 2004."

Summary

We've seen that games have a rich history, and as young as the games industry is, it's a large and growing one. The propagation of the Flash 7 player across the Net allows us to take these games to a new level, bringing them instantly to an audience of hundreds of millions, creating limitless opportunity for creative expression and commercial success.

Reference

- DeMaria, Rusel, and Wilson, Johnny Lee. *High Score! The Illustrated History of Electronic Games.* McGraw-Hill Osborne Media. (2002).

Excerpted from Creating Games in Macromedia Flash MX 2004, Charles River Media.

Glen Rhodes (Toronto, Ontario) has been developing games professionally for 10 years. He is CTO of CRASH!MEDIA. Glen has developed games such as Catch the Train for Nike and Domino Dementia for www.shockwave.com. He has also written and coauthored a number of successful titles on Flash MX.

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*

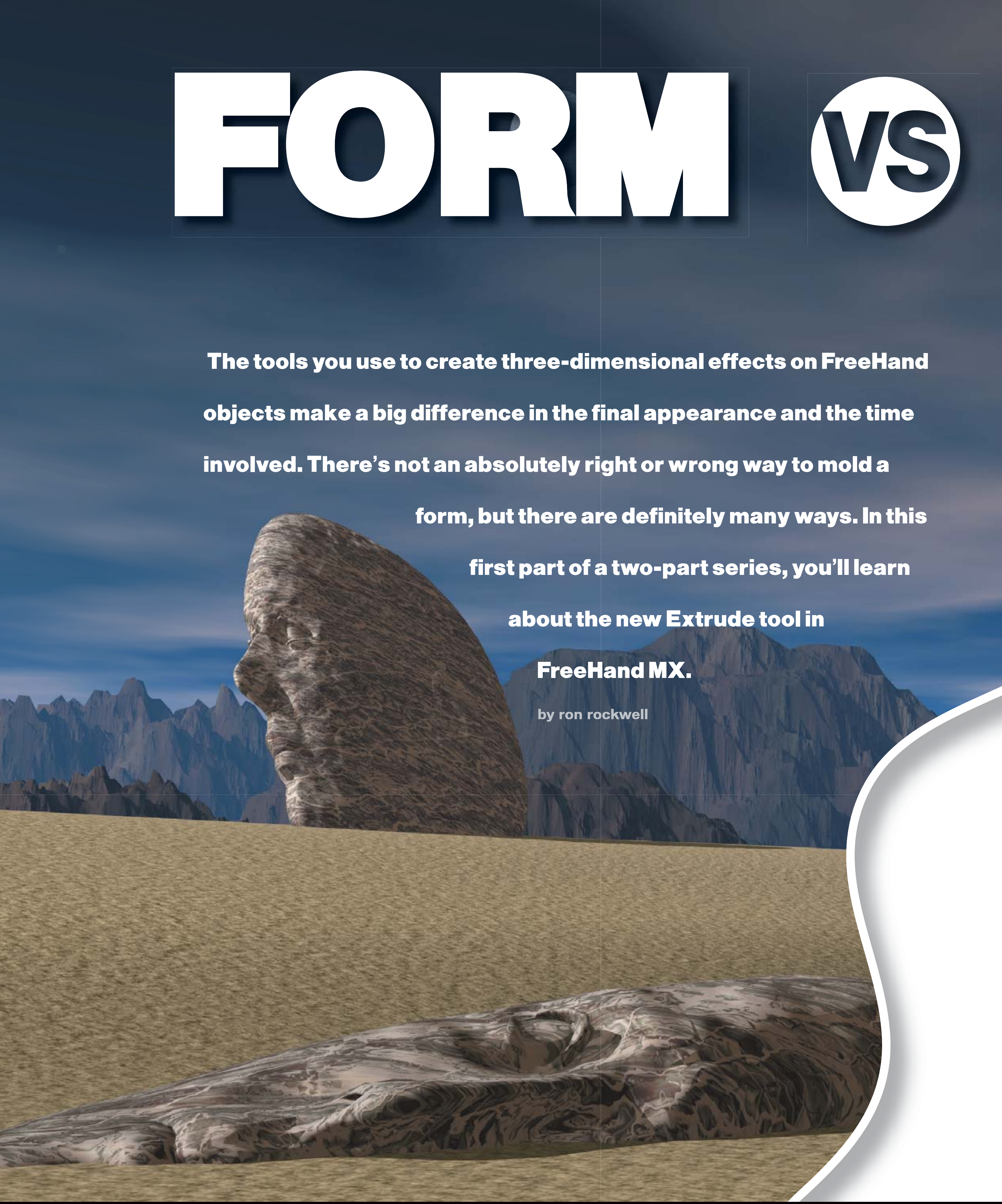
"When it comes to ColdFusion hosting, HostMySite.com rules them all!"

James Kennedy
mbateam.com

*offer applies to any annual shared hosting plan

call today! **877•248•HOST** (4678)

FORM VS



The tools you use to create three-dimensional effects on FreeHand objects make a big difference in the final appearance and the time involved. There's not an absolutely right or wrong way to mold a form, but there are definitely many ways. In this first part of a two-part series, you'll learn about the new Extrude tool in FreeHand MX.

by ron rockwell

FEATURE

PART 1

There are many ways to create the illusion of three-dimensional objects with the multitude of drawing tools and functions within FreeHand MX. Blends, gradient fills, vector and bitmap effects, and Xtras will be covered in Part 2 of this series. In this installment, the new Extrude tool will be explored and explained. It is both simple and complex to operate, depending on your level of experience with the tool, and the effect you are attempting to achieve. If you want "Superman" style text zooms, then the Extrude tool will fix you up in seconds. However, if you want hollow objects, donuts, spirals, and high degrees of rendering, then your work will be a little tougher to complete. But that's not to say it's hard – it will just take a bit more time. The hot dog illustration in Image I has six extrusions and took about three hours to draw. Even with 1.5GB of RAM, a lot of time is spent waiting for the program to complete its calculations and redraw the screen.

The Extrude Tool

Initial use of the Extrude tool is pretty simple: click the tool on a vector object and drag the cursor to create a vanishing point. Unfortunately, you'll rarely be finished that quickly, so a few pointers might be welcome. You can see the Extrude tools in Image II. All of these commands are found by choosing Extrude from the Modify menu.

- ***Release Extrude:*** Clicking this icon or choosing it from the menu will convert the extruded artwork into the polygons that make up the shape. This could be as few as four polygons for a simple cube, or many thousands of polygons in a complicated extrusion. The artwork cannot be modified in any way with the Extrude tool at this point as it is a simple vector graphic.
- ***Remove Extrude:*** This option will take away any extruded effects and return the graphic to its original shape.
- ***Reset Extrude:*** After you've manipulated an extrusion in certain ways, you cannot apply changes without resetting the extrusion, which takes you back to the original extrusion. In essence it's a lot of Undos in a single click. After using Reset Extrude and making your changes (usually color), you must redo any rotation and scale adjustments you had previously done. The bright side is that you have an idea of what you want it to look like, so it takes less time than originally.
- ***Share Vanishing Points:*** With multiple objects in a drawing, there are times you wish to have them all "pointing" to the same spot on the horizon. To use this option, select all the extruded objects necessary, and click the Share Vanishing Points option. All the objects will then show their vanishing points. The next mouse click will apply a single vanishing point that you can drag to any location.
- ***Rotate Extrude:*** If you have a lot of extra time on your hands, you can click this icon or choose it from the menu. It serves the same purpose as double-clicking an extruded object: it displays the rotation circle. Different strokes for different folks, I guess.

I find it easiest to have all the icons/buttons for extrusion actions installed in my main toolbar to save time.

If you've just played with the Extrude tool a couple of times, it may seem daunting to work with. But it's actually quite easy to master. As with other things in life, there are rules to remember, however. For one, in order to create an object in 3D, FreeHand converts the object into hundreds and thousands of tiny polygons – usually triangles – that create facets (or faucets if you're drawing plumbing fixtures) that give the illusion of three dimensions. Therefore, the higher the degree of rendering you ask FreeHand to do, the more polygons that will be created, and coincidentally, the more RAM and time required to manipulate the object. The key here is to work at low levels of rendering, or "steps" as it's called in the Object panel, until you have completed the sizing, positioning, lighting, and other factors. Then, when you're ready to print or commit to a different format, increase the number of steps in order to smooth out the facets.

Another rule is that the Extrude tool works only on vector objects and text; you cannot extrude a bitmap. Well, you can trace the bitmap and extrude the various objects, but I can't imagine why you would (the multitude of points in a tracing would take up so many computer resources that it would be impossible to get any work done).

More Rules

You can't simply change the color of an extruded object (but you can cheat – see Color Switcheroo). If you have rotated and changed the size of an object, you will be prompted to Reset Extrude, which brings you back to the first step in your extrusion. At that time, you must choose Modify > Extrude > Reset Extrude; double-click Contents in the Object panel to Subselect the object as a group; then double-click Contents once more to actually change fill or stroke color. Then you redo your rotation and size changing with the new colors in place. You must have the Extrude tool active to change any 3D attributes. As an added bonus, any change you make to the artwork (live on the page with the Extrude tool) can be accomplished by entering numbers in fields in the Object panel. That means you can maintain consistency between extruded objects by entering the same numbers for X, Y, and Z-axes and other attributes such as lighting arrangements. You probably know that Adobe Illustrator's extrusion function requires you to open a dialog box to change attributes, then close the window to see the result. If it's not to your liking, you must return to the dialog box. FreeHand is much more user-friendly in that respect. Last, if you have multiple objects that you wish to share a common vanishing point, it's a matter of selecting the objects and choosing Modify > Extrude > Share Vanishing Points.

Oh, best of all, you can extrude live text; just click the extruded text with the Text tool. The text editor will open, and you can make any changes you wish, including changes in color on an individual-letter level.

image II

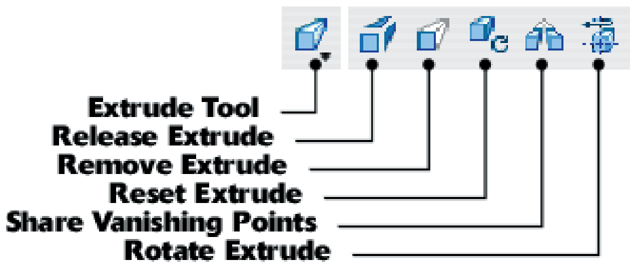


image I

Using the Extrude Tool

Before applying an extrusion, choose the base color you want for the object. Since FreeHand adds shadow and highlight colors, it's best to select a color with a middle value – not too dark or light.

Click the shape or text with the Extrude tool, dragging in the direction of your vanishing point (you are, in fact, dragging the vanishing point). Seconds later you have a 3D extrusion.

The most common extrusion you'll probably make will be a text object. In Image III, "POW!" was extruded in the with default settings, resulting in a solid mass receding to the horizon. But to show that extrusions don't have to be simple, the path that is shown was applied as a profile to give the text a boxing glove appearance. A full description of profiles appears further in these pages. For the record, when the extrusion was complete, it was cloned and the extrusion released. Then all the components of the extrusion were deleted, leaving only the rotated text which was colored and modified.

I must apologize for throwing so many features at you in one illustration, but the magic of the Extrusion tool is the ability to combine various attributes. By reading through to the end, you'll see that everything will have been covered in good detail. With that in mind, the top of Image IV shows the extrusion profile shape drawn to give my extrusion a jelly jar shape (notice that the shape is nearly closed: the extrusion will create a shell

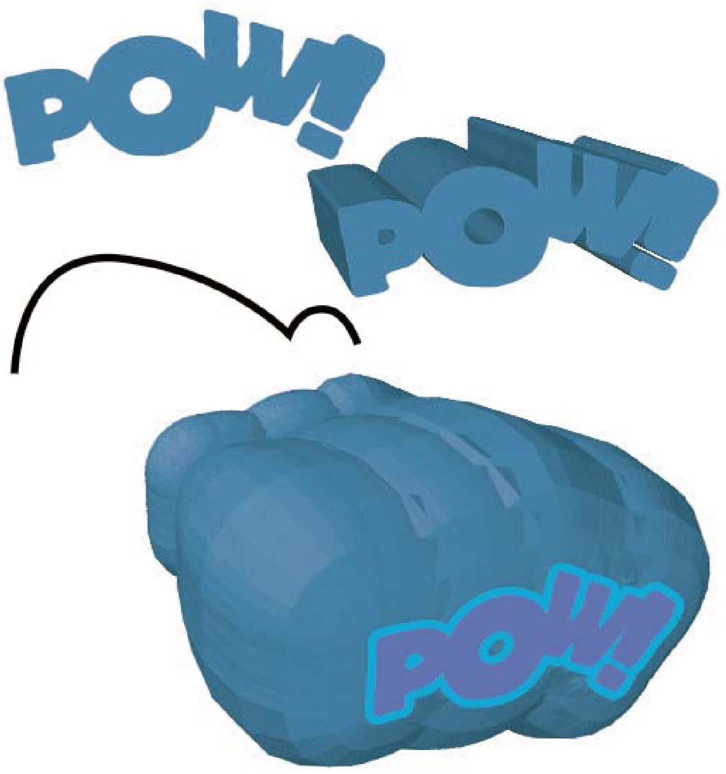


image III

image IV

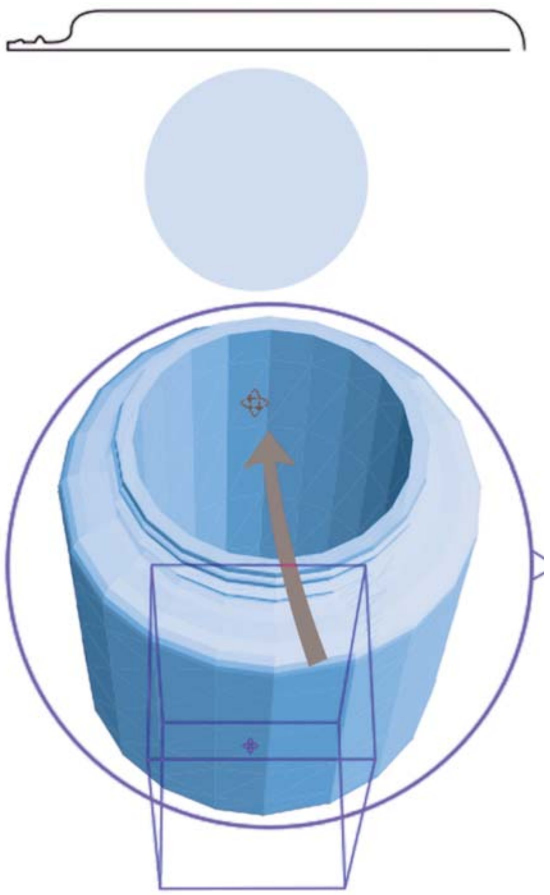
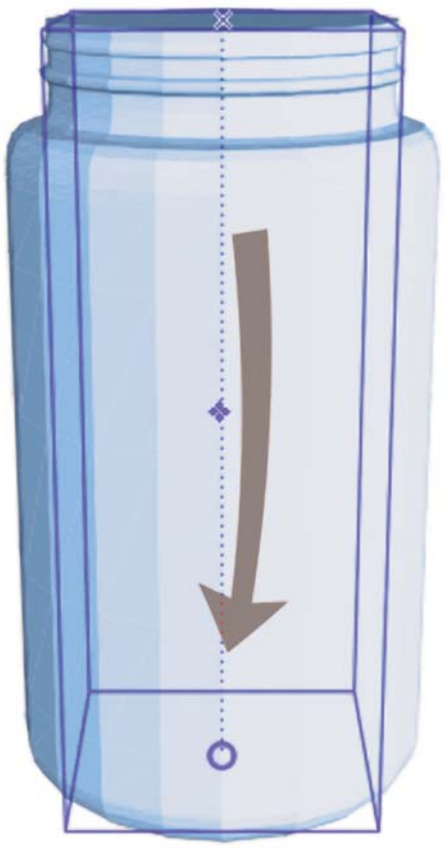


image V



around the object that is extruded). Immediately beneath the path, you can see the circle that was extruded. At the bottom, the extrusion was double-clicked with the Extrude tool to bring up the Rotation Circle. Then the tool was clicked inside the bottom of the Rotation Circle and dragged upward to rotate the extrusion vertically. Please note that the object you choose to extrude will be the core of the extrusion. That is to say that all the extrusion work will be done around the object. Small objects result in skinny extrusions; large objects gain a wider extrusion. You may have to try two or three sized objects to get the appearance you're looking for.

Clicking and dragging the Extrude tool inside the Rotation Circle rotates the object around its X- and Y-axes. Rotation around the Z-axis is achieved by clicking and dragging the cursor outside the Rotation Circle. The triangular point on the circle is a Z-axis reference marker.

When the object is rotated to your satisfaction, double-click the Extrude tool on the desktop to deselect the object. Then click the object once with the Extrude tool. The object center point is indicated by the "X" at one end of the dashed Z-axis line. At the other end is a circle that controls the depth of the extrusion. Click and drag this circle to lengthen or shorten the object. The vanishing point is marked by a cluster of diamonds. A rectangular box will appear when the object is selected with the Extrude tool, showing the actual perspective. Sometimes this box will be partially off the object, but it doesn't affect your editing or the final result.

It's important to remember that, yes, you can skew, rotate, scale, and mirror an extruded object, but once you do so, you can no longer use the extrusion tool for editing the object without using Reset Extrude and starting over.

Using a Profile

Simple extruded circles, squares, and text can be pretty boring. FreeHand MX allows you to create a profile for an extrusion, and the best way to explain the tool really requires figures to do the heavy lifting. Image VI shows a circle, a square, and a compound path that have been extruded and share the same vanishing point. Each example has the same profile, but a different approach: either bevel or static, and a static with a twist applied.

I hate to admit it, but the Extrusion tool does not create a true 3D extrusion on the first click, due to the math involved in the extrusion process. There, I've said it.

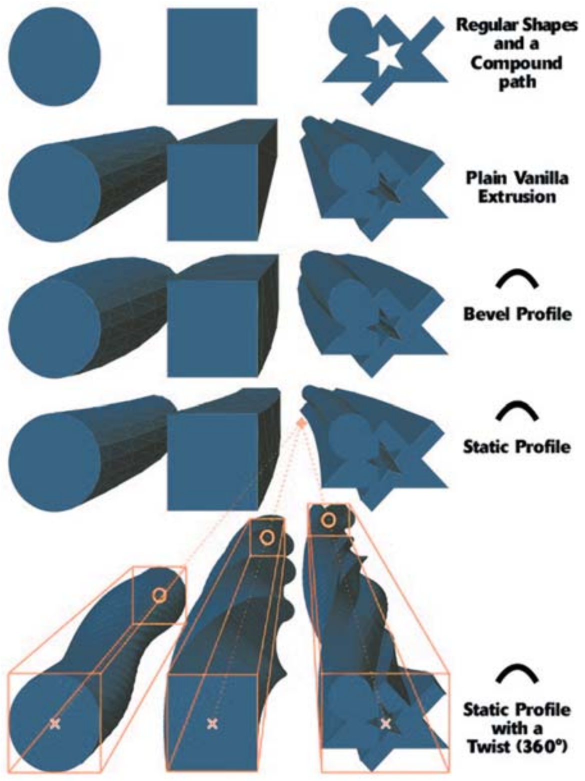
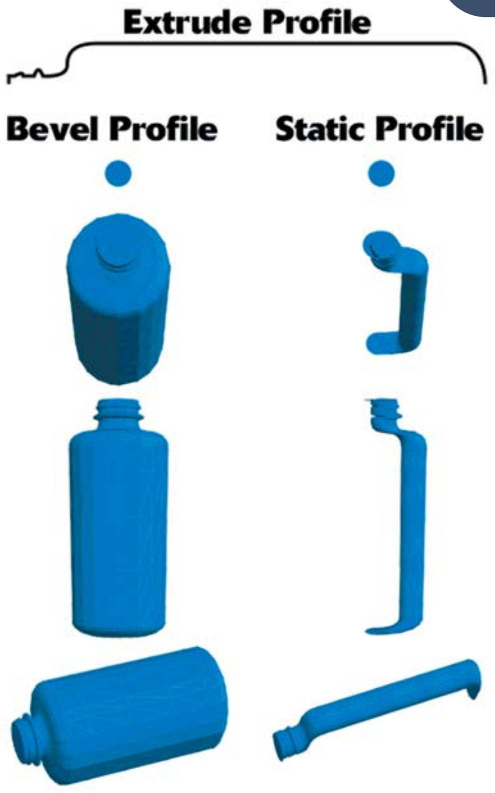

Regular Shapes and a Compound path
Plain Vanilla Extrusion
Bevel Profile
Static Profile
Static Profile with a Twist (360°)

image VI

image VII



Look particularly at the bottom of Image V – the length of the extrusion accents the fact that you're looking at the objects all on the same plane, without perspective. A true 3D extrusion would rotate and skew the original object in conjunction with the extrusion's vanishing points. Instead, the object remains at its original orthographic placement and shape, square to the page. Once you rotate the extrusion (explained below), distortion occurs, but the original object is not distorted correctly. The solution to the situation is extremely simple and results in a correctly distorted object: when you first apply the Extrude tool, click as near the center of the object as you can, and do not drag out a vanishing point. The vanishing point can be manipulated later.

In the extrusion shown in Image VII, the path is modified from the path in Image IV, and is more open, which will create a solid extrusion. You can see how the bevel profile wraps around a straight-line extrusion of the object. A static profile, however, acts as a spine to the extrusion. An object with a static profile will maintain a consistent shape from one end to the other, but follow the spine of the profile. The bevel profile in this image is a good example of how the object size changes the appearance of the extrusion – compare to virtually the same profile in Image V.

More than likely, you've noticed that most of the extrusion examples so far have a rough, faceted look to them. They were done with minimal rendering resources – the default of 5 in Lighting and 5 in Profile. It bears repeating and can't be overstated that extruding objects is extremely RAM intensive, so you must be patient and do any profiles, rotations, twists, and other adjustments before bumping up the number of Steps in either the Profile or Lighting windows. The simple act of moving an extruded object to a different location on the page can cause a wait of several minutes – if not an unwelcome "Could not complete your request – not enough memory" message. The boxing glove-type extrusion in Image II has nearly 28,000 objects with 5 steps. Increasing the steps to 10 would require 56,000 objects, and the program comes to a halt.

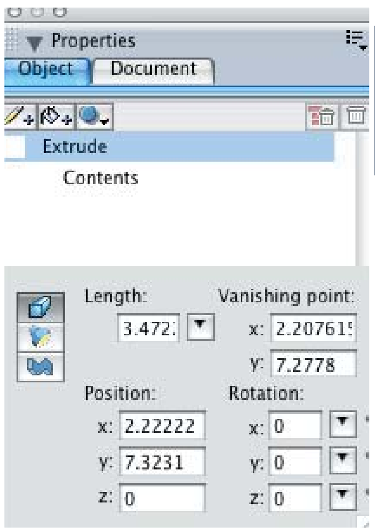
Edits Made in the Object Panel

If you're like most people, you'll want to change the length of the extrusion, its orientation, vanishing point placement and rotation live on the page. Although it can be done in the Object panel (for consistency with other objects, for instance), it's much more satisfying and less frustrating to work directly with the object. However, there are many options to consider in the Object panel. The screenshot in Image VIII shows the available options. Entering numbers in any of these fields will affect the appearance of the object, and will take the same amount of time to render.

Lighting Options

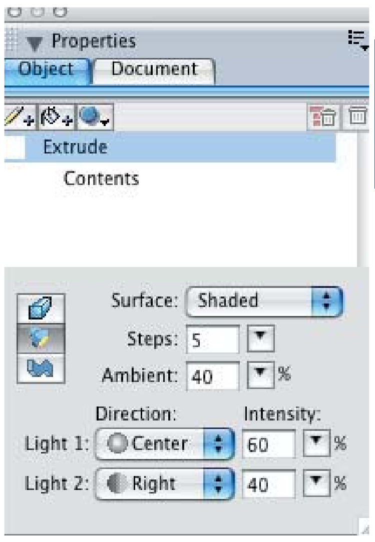
You have a choice of surface rendering approaches, the number of steps involved in the rendering, and how much ambient light surrounds the object. Beyond that, you can control the position and brightness of two light sources. FreeHand is not a 3D rendering program, but supplies several lighting choices as seen in Image IX, reached by clicking the light source icon in the Object panel that appears when an extruded object is selected.

image VIII



Length: 3.472
Vanishing point: x: 2.20761 y: 7.2778
Position: x: 2.22222 y: 7.3231 z: 0
Rotation: x: 0 y: 0 z: 0

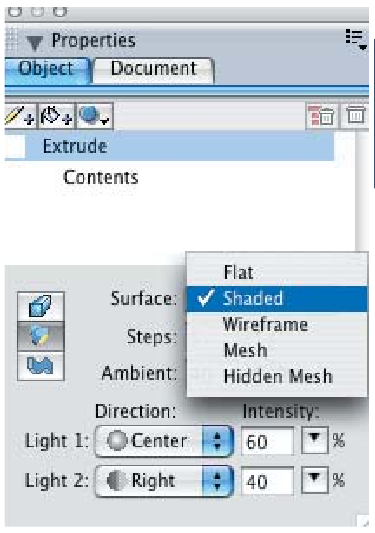
image IX



Surface: Shaded
Steps: 5
Ambient: 40 %

	Direction:	Intensity:
Light 1:	Center	60 %
Light 2:	Right	40 %

image X



	Direction:	Intensity:
Light 1:	Center	60 %
Light 2:	Right	40 %

Surfaces

Surface choices are shown in Image X, and bear a little explanation.

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

image XI

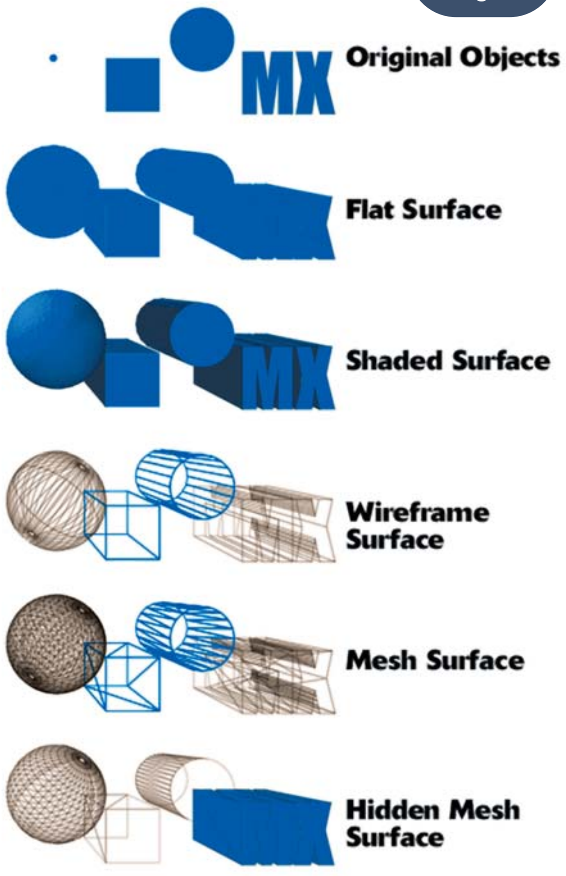


Image XI shows the effects of the different surfaces. The Flat surface option creates an extrusion, but does not give a modeled light. As it's named, the object is flat, but the polygons that make up the extrusion are stroked with a lightened version of the fill color (this happens to be a bug). A stroke on the original object will separate the face of the object from the extrusion. The usual option is the Shaded surface, and here's where the lights come into play. Wireframe creates a skeletal outline following the profile of the extruded object; the paths that make it up are 1-point lines with the color of the object's fill. A Mesh surface shows the same wire outline, with the addition of outlines of all the triangular polygons that make up the object. Both Wireframe and Mesh show all sides of the object, as if it is hollow. Last, there's the Hidden Mesh surface. This surface is different than the others in that you must start with a stroked path, and the end result is that the rendering hides all the construction lines that aren't in view. Good to use when you don't want to see the dark side of the moon. The Wireframe and both Mesh surface treatments are super if you're creating a techie background and need a blueprint effect.

Lights

If you choose to have a shaded surface, then you'll find the various light sources by clicking their drop-down menus shown in Image XII. Their locations are simplistic, but they get the job done.

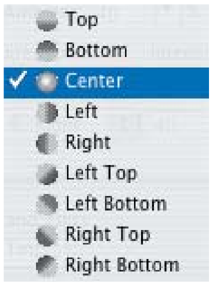
Profiles

The workhorse of the Extrude tool lies in the application and use of profiles. As seen earlier, a profile can surround an object, or act as a spine. Image XIII shows how to create various objects that seem challenging at first. In all the examples, the object to be extruded is on the left, followed by the profile and the resulting extrusion. The extrusions have been rotated to show off their best sides, and don't share a common vanishing point.

To create a sphere, start with a very small ellipse. Imagine the area of a ball that touches the table – that's how small an area you want, or you'll end up with a flat spot. The profile is a semicircle, oriented as shown, or rotated 180-degrees. If you rotate it 90-degrees as shown in the next example, the original object lies at the bottom of the funnel shape (out of sight), with the extrusion rising up and traveling outward from the center.

A torus or donut is tougher. Create a compound path: draw a circle; clone it; reduce the size of the clone; and center the two circles upon each other. Use Modify > Join to make the compound path (or you can use the Punch Xtra). Draw another circle that you want to be the thickness of the donut, and Ungroup it. Select the top point in the circle and choose Modify > Split. Then rotate the circle 45-degrees counterclockwise, and copy it to the Clipboard. Now select the Extrude tool and click the compound path; in the Object panel, click on the

image XII



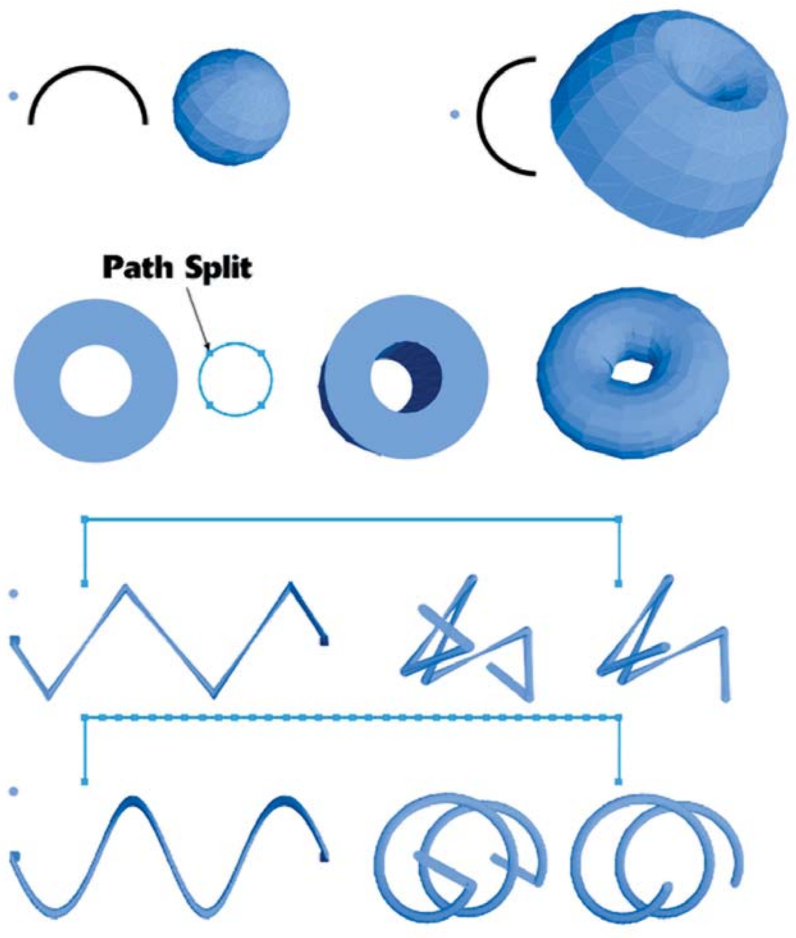


image XIII

Profile button and select Bevel from the drop-down menu. When the Paste In button is active, click it, and the default straight line in the preview box will be replaced with your broken circle. Wait a minute or two and your donut will appear. If you see a space station or Saturn-looking object, rotate the original profile, copy it, and paste it into the extrude panel.

The bottom two examples in Image XIII use the Twist function. The difference is that in the upper one, a U-shaped path containing only four points is used as the profile. The Twist entry is 720 degrees, creating two complete revolutions. A U-shape is necessary to provide a center point from which the spiral will revolve. So, with the simple profile, a path is created of straight lines rotating and stopping at 90-degree intervals. The next example has been rotated end-on. To create a smooth spiral or spring shape as shown on the bottom, use the Add Points Xtra several times on the long stretch of the profile. I split the end paths off first and joined them later. Now the profile rotates extremely smoothly. To get rid of the end paths as in the far-right examples, you will have to get the object oriented exactly as you want, then use the Release Extrude function to convert the spiral into a group of polygons. Use the Lasso tool to select the end objects and then delete them.

Both the Static profile and the Twist option are unique to FreeHand in 2D drawing programs. In Image XIV, two squares and two circles arranged in a square were grouped, then extruded. A 360-degree twist was applied. If you ever have to draw the staff of Aesculapius (the two intertwined snakes on the winged staff symbol used by doctors), this is the technique.

And, in the "because I can" department, I took that same extrusion with a twist angle of 720-degrees, and applied a circular static profile, resulting in the snarl of rubber bands shown in Image XV.

Color Switcheroo

Okay, you've got the extrusion exactly where you want it, the profile is perfect, even the lighting and profile steps are just right. Then the customer says they want more of an robin's-eggshell blue instead of this off-white-blue color that was approved earlier – now what? Well, if you follow the generalizations of the program, you'll select the extrusion and attempt to change colors using all the various methods you know. All to no avail – you'll be greeted with a message telling you to use Reset Extrude. "That's just perfect! Back to square one," you say, pulling your hair out. Well, if you use this simple method, you'll be flying high in no time. Simply use a named color when you select your fill color. Doing so allows you to create a new color and drag it on top of the named color in the Swatches panel. Just a few minutes later, the extrusion will change color, and you'll still have most of the hair on your head. In case you're not up on named colors, create a color in the Mixer or Tints panels and click the "Add to Swatches" button next to the color well, or drag a swatch from the color well and drop it on the Swatches panel. You'll be prompted to name the color. Choosing a new color and dropping a swatch of it directly onto the color square in the Swatches panel will change the color of every object that has the named color, so be specific when you name the color for your extrusion.

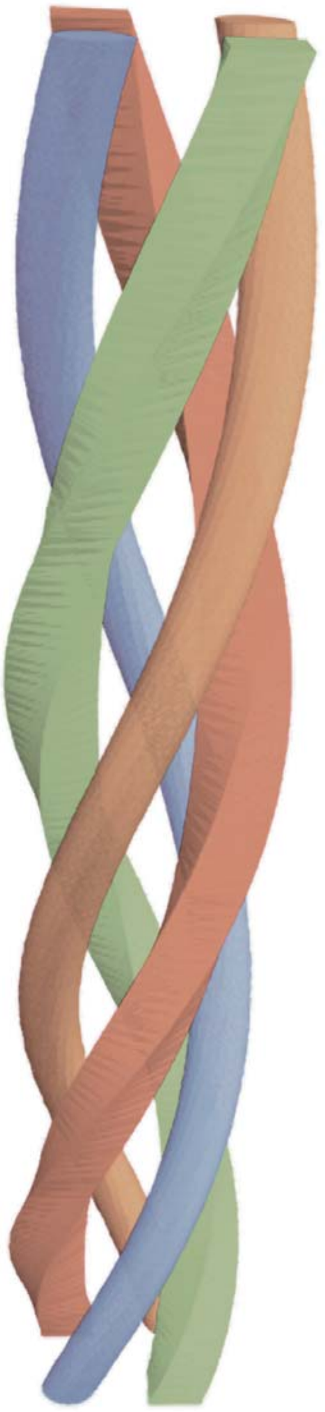
Adobe Illustrator has two features that I'd love to have in FreeHand: plastic rendering and image mapping. Since we don't have either, you must create workarounds. Simple rendering techniques using gradient fills can give you a highly polished surface appearance. For flat-sided image mapping, it's easy enough to apply an Envelope to the art you want mapped and adjust it to fit. Beyond those features, I'll take FreeHand MX's Extrude tool any day, considering the static profile, mesh modeling, and the fact that since the extrusion is a top-level object we make live adjustments for length, rotation, and attitude instead of going back and forth through a dialog box.

Acknowledgments

Many thanks to Peter Moody, David Spells, and other engineers at Macromedia for the technical support they provide.

Illustrator, designer, author, and Team Macromedia member Ron Rockwell lives and works in the Pocono Mountains of Pennsylvania. He is the author of FreeHand 10 f/x & Design, *and is about to introduce a FreeHand MX course. He has Web sites at www.nidus-corp.com and www.brainstormer.org. Contact him at guru@brainstormer.org with questions or article requests.*

image XIV



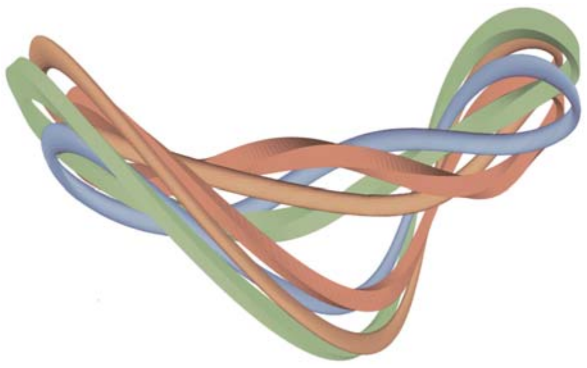


image XV

An OO Approach to 'War'

With the advent of ColdFusion components (CFCs), introduced in ColdFusion MX version 6.0 and greatly improved in version 6.1, ColdFusion MX allows CF programmers to enter the mainstream of object-oriented (OO) programming. With the overwhelming success of the J2EE and .NET platforms, OO has become the dominant paradigm for building commercial software and gaining a thorough working knowledge of it is essential to any programmer's long-term career success. In this article, we'll build a simple version of the child's card game "War" – hopefully learning something about OO design and implementation along the way.

by hal helms

Here's a refresher on the game: two players each receive a card. The cards are then compared to see which has a greater value (the ordering of suits is ignored for this game). The player with the higher card is the winner and the game continues until the deck of cards runs out. As simple as the game is, it does provide some insight into object orientation as implemented in CFCs. First, though, we need to understand what OO is and how it requires a shift in thinking.

As most ColdFusion programmers write code, data and program logic are separate. Data is stored in databases while program logic is fit into CFML files. When we need to read or modify a piece of data, we commonly extract it from its database home, use it, and then return it (see Image I).

OO takes a different approach: both data and logic are combined into a single software "packet" known as a class. Data is stored in instance variables and logic is stored in methods.

In the address example I've used, we would build an Address class. What sort of data would an Address class have? A minimal representation of an Address class would probably include street, city, state, and zip. Because the Address class isn't expected to do much, its methods might be restricted to "getters" and "setters" for the instance variables, yielding methods like getStreet and setStreet. These methods (and similar ones for the other instance variables) simply provide access to the data that is kept private.

In ColdFusion MX, classes have a special name: ColdFusion components (or CFCs, for short). Code I shows one CFC implementation for an Address type.

A class is a static representation of something in the real world. But we don't want a static address; we want an actual address. Our class definition acts as a sort of blueprint or cookie cutter, from which individual objects can be created – and it's objects we're counting on to do the work in an object oriented application.

Creating an object from a class is quite simple:

```
<cfset myAddress = CreateObject('com-
ponent', 'Address') />
```

That's it. We now have an object referred to by the name, myAddress. Our object is pretty barren at present. All it has for instance variables are empty strings. We'll give the instance variables more meaningful data, but first, we need to understand a very important OO idea – perhaps the most important idea.

Encapsulation

Encapsulation is a principle that urges us to keep our data private to the outside world, allowing access to it only through methods. With CFCs, we achieve this by placing the instance variables in a private scope (private to the object itself) known as the variables scope. From outside the object, those variables don't seem to exist. We can try to access them – perhaps using dot notation:

```
<cfoutput>
 #myAddress.street#
</cfoutput>
```

But ColdFusion will tell us that no such variables could be found. That's right – since they're private, the only way to access these is through the getter and setter methods we created. Here's how we can get real data into our object:

```
<cfset myAddress.setStreet('4034
Whetstone Ct') />
<cfset myAddress.setCity('Marietta')
/>
<cfset myAddress.setState('GA') />
<cfset myAddress.setZip('30062') />
```

Our setters work to set values for instance variables; our getters provide the information stored in those variables.

```
People often ask me if it's hot in
#myAddress.getState()#.
```

Once we have an object, we can get it to do some useful work. We do so by sending the object a message. We saw a simple example of this with our getters and setters. The methods defined by an object define the messages that can be sent to that object and regardless of how complex the object may be, we always use this message-sending to accomplish things.

In addition to encapsulation, OO is upheld by two other chief tenets: poly-

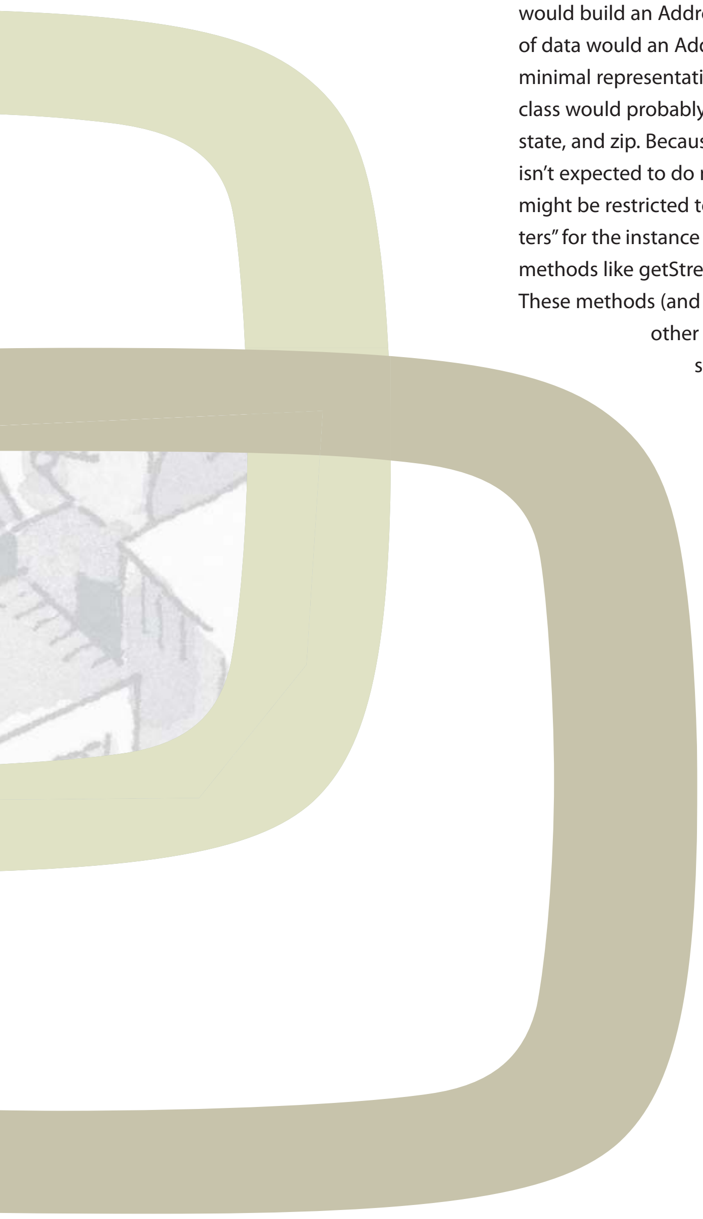
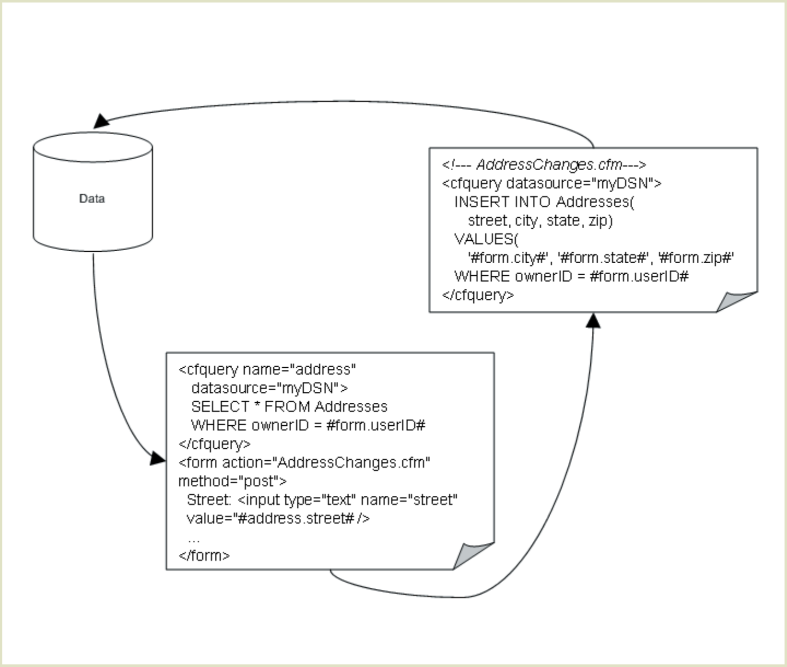


image I



```
<!--- AddressChanges.cfm--->
<cfquery datasource="myDSN">
  INSERT INTO Addresses(
    street, city, state, zip)
  VALUES(
    '#form.city#', '#form.state#', '#form.zip#'
  WHERE ownerID = #form.userID#
</cfquery>
```

```
<cfquery name="address"
  datasource="myDSN">
  SELECT * FROM Addresses
  WHERE ownerID = #form.userID#
</cfquery>
<form action="AddressChanges.cfm"
method="post">
 Street: <input type="text" name="street"
 value="#address.street# />
  ...
</form>
```

morphism and inheritance. With CFCs especially, these two are intimately related. The word polymorphism is an unhappy transplant from its native Greek, where it meant "many forms." As applied to OO, it might be better translated "much confusion." Polymorphism is notoriously hard to define – and that's too bad, because it's a powerful tool in the hands of accomplished programmers. Still, let's see if we can't discover what polymorphism is.

I'll start by asking a simple question: Who are you? Well, that depends. You may be a child to some, a parent to others, an employee to your company, a spouse to your mate, a friend to your comrades – the list is probably a very long one. What's important for our understanding of polymorphism is that you belong to many different groups. Here, we've listed Child, Parent, Employee, Spouse, and Friend.

OO languages allow us to create a software model of the world, or at least those parts of it that are under consideration. Part of the language's job is to make sure that conversations between objects (via message sending) is safe – that is, the object that is being sent a message has a method that correlates with the message sent. The way that class-based OO languages (Java, C#, CFCs) provide for this is through type checking. The logic behind this is as old as Aristotle, who laid out the basic law of the syllogism like this:

1. Socrates is a man.
2. All men are mortal.
3. Socrates is mortal.

If the first two premises are granted, the conclusion is inescapable. Likewise, OO languages implement their own version of syllogistic logic:

1. You are a parent.
2. All parents worry.
3. You worry.

Put in slightly more techie terms, I could say that since you're of type Parent, you can respond to the worry() message. But parents are not all the same. Some are NaturalParents. Others are AdoptiveParents. Still others are SpiritualParents. No matter; they all know all too well how to respond to the worry() message. Again, translating this into TechTalk, we might draw a UML class diagram that looks something like this Image II.

Now we get to the polymorphic part: let's say that we have a Counselor type, who specializes in helping parents deal with worry. The Counselor has a soothe() method.

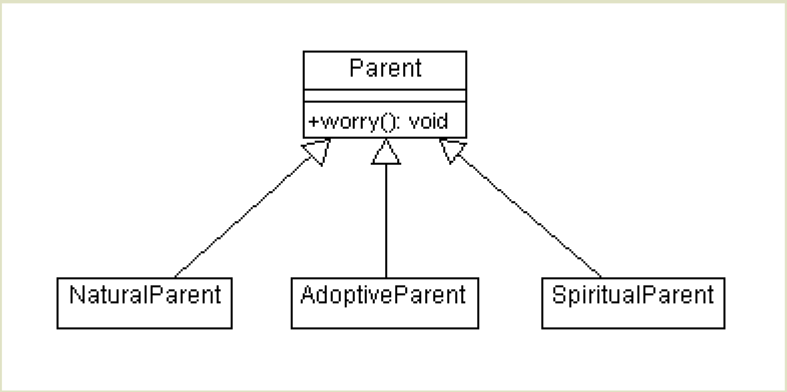
```
<cffunction name="soothe" access="pub-
lic" returntype="void" output="false">
 <cfargument name="parent" type="???"
required="true" />
 <!--- secret parent-soothing method
goes here --->
</cffunction>
```

We specified an argument named parent, but of what type? We certainly don't want to have methods for each type – sootheNaturalParent(), sootheAdoptiveParent(), and sootheSpiritualParent(). That's just a bad solution. Not only is it ugly, it's fragile. If we add a new kind of parent in response to changing requirements, we're going to have to find every method that accepts parent types and add new methods to correspond with the new parent type.

Luckily, polymorphism comes to the rescue: we can specify that the argument type is simply Parent. At run time, we can pass a Parent type, or – this is the important part – we can also pass any Parent subtype. The generalized Parent type provides a type specification for which any subtypes will be considered valid. So we don't need different methods for each type of parent. We can specify that the type is Parent and our code will run correctly. If we add another Parent subtype later, none of the code that accepts type, Parent, will break. This new Parent subtype can safely be passed to the existing methods.

There's another aspect to polymorphism. Image III is another UML diagram

image II



showing a polymorphic relationship.

Now, in a Boss class, we might have a method called delegateWork() that accepts an Employee and calls the Employee's work() method. The code might look like this:

```
<cffunction name="delegateWork"
access="public" returntype="void" out-
put="false">
 <cfargument name="employee"
type="Employee" required="true" />
 <cfset arguments.employee.work() />
</cffunction>
```

As each Employee is sent to the Boss's delegateWork() method, the Boss tells the Employee to start working.

What happens? The PayrollClerk starts paying people; the Engineer starts designing things; and the Programmer starts writing code. Even though a generic Employee is accepted as the type of delegateWork method, the actual work method called on that Employee will be that one the corresponds with their more specific type. This prevents the Boss from needing a series of methods such as delegateWorkToPayrollClerk(), delegateWorkToEngineer – and so on.

Polymorphism (also known as subtype polymorphism) is a powerful tool to be used in building scalable and maintainable code. To make use of polymorphism in our OO code, we need to think in terms of general types, for which individual subtypes can later be substituted. This provides flexibility to our designs.

With CFCs, subtype polymorphism occurs by means of inheritance – whereby one class is a specialized type of another, existing class. A SportsCar is a specialized type of a Car, for example. To reflect this relationship in code, the subclass (SportsCar, in this example) declares that it extends a base class:

```
<cfcomponent displayname="SportsCar"
extends="Car"…>
```

A class that extends another class has access to the base class's properties and methods, as though they were its own. Since Car will already have a start() and stop() method, for example, SportsCar doesn't need to define these itself – SportsCar inherits them from Car. (If a subclass wishes to implement a method differently from its base class, it may simply redefine the method in its own code. At run time, the overridden method, as it's called, will be invoked instead of the base class's method.)

Back to the Game

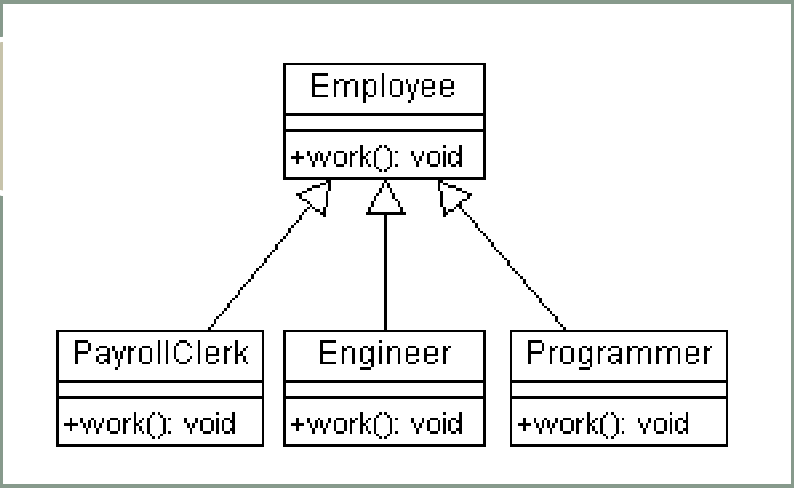
With that quick primer on object orientation done, let's return to our War card game. The game uses a standard set of poker cards. I began by modeling a single card in a CFC called Card. A Card needs to do only a few things. It must be able to return a string representation of its value ("Jack", for example), a numeric representation of its value (11 in the case of Jack) and it must be able to determine if it outranks another card. The public methods for Card are therefore getStringValue(): string , getNumericValue(): numeric, and compareTo(Card): numeric. (The syntax for expressing method signatures as they're called is this: any arguments accepted by the method are shown in parentheses; if the method returns anything, the return type is specified after the colon. Parsing the "compareTo(Card):numeric" method signature would tell us that the compareTo() method accepts a variable of type Card, and returns a value of type numeric.)

At about this point in the process, I remembered that different games use different card types, differing chiefly in the numeric value assigned to cards. To build in for this flexibility, should the need later arise, I created a specific PokerCard class that extends the Card CFC. To reflect the fact that different card types might have different numeric representations for the same card (an Ace might be counted as 1 or 11, for example), I decided that all specific Card subtypes should determine the numeric values for each card.

I decided to call this needed method initializeValueMap(). Since it would be specific to each subclass of Card, I wanted a way to ensure that all subclasses would implement the initializeValueMap() method. In languages such as Java and C#, I could mark the method in Card as abstract. The compiler would force all subclasses to implement the method.

In CFCs we have to find a different mechanism, as no concept of abstract yet exists. I chose to have the base Card CFC throw an error as its implementation of initializeValueMap(). If each subclass overrides the method, the error will never be thrown. If a subclass fails to override, however, the base class's method will be called and the error raised.

image III



The compareTo(Card) method receives an object of type Card (note that the type is deliberately made general). The Card object uses the value map set by the subclass to determine its ranking compared to the argument sent to it. It returns a number: 1 if it outranks the passed-in Card, -1 if it is outranked, and 0 if both Cards have the same value.

A single playing card is part of a card deck, of course, and our next class must model the deck. I used the same type-subtype scheme to create a CardDeck and a PokerDeck class. The methods of the base class, CardDeck, include dealCard(): Card, hasNext(): boolean, shuffle(), getDealtCards(): array, and getUndealtCards(): array.

In addition to these public methods,

CardDeck defines the private method, newDeck(), which generates individual Cards and clears the undealtCards array. The shuffle() method uses the newDeck() method and both are meant to be overridden in subclasses. To help ensure this, the methods throw errors in the base class.

One of the advantages of OO is greater reusability of code, and we note that these four classes can be used to create many different card games. Our War card game will be implemented as a CFC, War, that uses both PokerCard and PokerDeck and their superclasses (superclass is another term for base class).

The class has the following public methods: play(): string, setDeck(CardDeck). In addition, all of the classes we've seen have an init() method, which deserves some explanation. Most OO languages have the concept of a special "method," called a constructor that is automatically called when a new object is created. This is used to initialize variables and prepare the object for use.

Unfortunately, at this point, CFCs have no built-in constructor, so ColdFusion developers (intrepid lot that they are) have coalesced around a pseudo-constructor called init(). While it's not called automatically, developers adopt the habit of always instantiating a CFC using the following syntax:

```
<cfset objectName = CreateObject('component', 'CFCname').init() />
```

The init() method can accept arguments and always returns the newly created object, once any initialization occurs. I recommend that all your CFCs have an init() method – even those CFCs that need no initialization. This allows developers to always create new objects using the syntax shown above.

Note that the init() method of our War.cfc accepts a CardDeck. If you wanted to create your own version of War that worked slightly differently, you might do so by passing a CardDeck of Card types that implement compareTo(Card): boolean a bit differently. You could, for example, create a War game in which the lowest card always wins.

The play method deals cards to two players, identified as Red and Blue. It calls one of the card's compareTo() method, sending it the other card. Finally, it returns a string indicating who won and what the current score is.

To test out the system, I created a very simple Tester.cfm file:

```
<cfset pd = CreateObject('component',
'PokerDeck').init() />
<cfset game = CreateObject('component', 'War').init(pd) />
<cfoutput>
 <cfloop
condition="#game.getDeck().hasNext()#"
>
  #game.play()#
 </cfloop>
</cfoutput>
```

While there are remaining undealt cards, the game will play the cards and will report whether Red or Blue finally won.

Conclusion

If you're new to CFCs and/or OO, you might want to take this code and expand it. You might add code to account for the situation when a tie occurs. In the real game, a "war" is declared and three cards are dealt. The winner of the next deal wins that hand plus all six of the "war" cards. Or, you might decide to build an interface to the system that will allow it to be played by real people. You can download the code at http://halhelms.com/index.cfm?fuseaction=code.detail.

From this simple example, we can see encapsulation, polymorphism, and inheritance at work. These three tenets of OO, when properly used, can help us write code that is robust, reusable, and maintainable.

Hal Helms (www.halhelms.com) is a Team Macromedia member who provides both on-site and remote training in ColdFusion, Java, and Fusebox. Hal is cofounder of the Mach II project. hal@fusebox.org

code I

```
<cfcomponent displayname="Address"
hint="I represent an Address">
 <cfset variables.street = "" />
 <cfset variables.city = "" />
 <cfset variables.state = "" />
 <cfset variables.zip = "" />

 <cffunction name="init" access="public" returntype="Address"
output="false">
   <cfargument name="street"
type="string" required="true" />
   <cfargument name="city"
type="string" required="true" />
   <cfargument name="state"
type="string" required="true" />
   <cfargument name="zip"
type="string" required="true" />

   <cfset setStreet(arguments.street)
/>
   <cfset setCity(arguments.city) />
   <cfset setState(arguments.state) />
   <cfset setZip(arguments.zip) />

   <cfreturn this />
 </cffunction>

 <cffunction name="getStreet"
access="public" returntype="string"
output="false">
   <cfreturn variables.street />
    </cffunction>
 <cffunction name="setStreet"
access="public" returntype="void"
output="false">
    <cfargument name="street"
type="string" required="true" />
    <cfset variables.street = arguments.street />
    </cffunction>

 <cffunction name="getCity"
access="public" returntype="string"
output="false">
    <cfreturn variables.city />
    </cffunction>
 <cffunction name="setCity"
access="public" returntype="void"
output="false">
    <cfargument name="city"
type="string" required="true" />
    <cfset variables.city = arguments.city />
    </cffunction>

 <cffunction name="getState"
access="public" returntype="string"
output="false">
    <cfreturn variables.state />
    </cffunction>
 <cffunction name="setState"
access="public" returntype="void"
output="false">
    <cfargument name="state"
type="string" required="true" />
    <cfset variables.state = arguments.state />
    </cffunction>

 <cffunction name="getZip"
access="public" returntype="string"
output="false">
    <cfreturn variables.zip />
    </cffunction>
 <cffunction name="setZip"
access="public" returntype="void"
output="false">
    <cfargument name="zip"
type="string" required="true" />
    <cfset variables.zip =
arguments.zip />
    </cffunction>

</cfcomponent>
```

Subscribe Today!

– INCLUDES – FREE DIGITAL EDITION! (WITH PAID SUBSCRIPTION) GET YOUR ACCESS CODE INSTANTLY!

*The major infosecurity issues of the day... **identity theft, cyber-terrorism, encryption, perimeter defense, and more** come to the forefront in* **ISSJ** *the storage and security magazine targeted at IT professionals, managers, and decision makers*

Editorial Mission

Greater Collaboration and Efficiency Through Education

✓ ***ISSJ*'s editorial mission is to showcase proven solutions** that will guide, motivate, and inspire senior IT and business management leaders in the planning, development, deployment, and management of successful enterprise-wide security and storage solutions.

✓ ***ISSJ*** brings together all key elements of data storage and protection, and presents compelling insight into the benefits, efficiencies, and effectiveness gained by focusing on these two critical areas of IT simultaneously.

✓ ***ISSJ*** is an objective, critical source of information that helps storage and security managers make informed management decisions about what is working today, and what they need to plan for tomorrow, and is the only publication that focuses exclusively on the needs of IT professionals who are driving the enterprise storage architecture/ infrastructure while pursuing and incorporating the latest security technologies.

✓ ***ISSJ*** achieves our mission by delivering in-depth features, practical "how-to" solutions, authoritative commentary, hard-hitting product reviews, comprehensive real-world case studies, and successful business models, written by and for professional IT storage and security practitioners.

SAVE 50% OFF!

(REGULAR NEWSSTAND PRICE)

Only $39.99 ONE YEAR 12 ISSUES

www.ISSJournal.com

or

1-888-303-5282

SYS-CON MEDIA

The World's Leading i-Technology Publisher

The Best Kept Secret in Town

Application development with Macromedia Director

by jason macdonald & paul-catalin oros

Macromedia Director is best known for building games, 3D simulations, and animations. What a lot of developers probably don't realize is that Director can also be used to build elaborate applications, and even more so now with the release of Director MX 2004, which supports ECMAScript-compliant JavaScript syntax.

Although Director is seldom mentioned as a serious contender in the applications development arena, it may very well be the best-kept secret in town! This article discusses what an application is by today's standards, and why applications are still required when everything else seems to be moving to the Web. We then introduce Director and show how it can be used as a great application development tool. In order to do so, we break an application down into its three main components – user interface, data management, and logic – and we explore how Director can be best used to implement these three main components.

Relevance of Applications

What Is Today's Definition of an Application?

Ten years ago it was easy to define what an application was; it basically was anything that ran on a computer, and the content was referred to as "documents." Today it is harder to define "application," as the boundary between documents (data) and applications (programs) has since blurred. Some documents, such as PDFs with embedded JavaScript or interactive PowerPoint presentations, do pretty much what an application use to do. On the other hand, some Web-based applications are very basic and can hardly be classified as an application as they lack the logic, personalization, and interactivity that is normally associated with an application. For the intent of this article, an application will be defined as a stand-alone medium that interacts with end users and performs some nontrivial logical tasks. Examples of such applications are:

- An electronic catalog that informs and guides a client through the selection of a company's products
- An interactive e-learning environment or a personalized itinerary planner for conference attendees

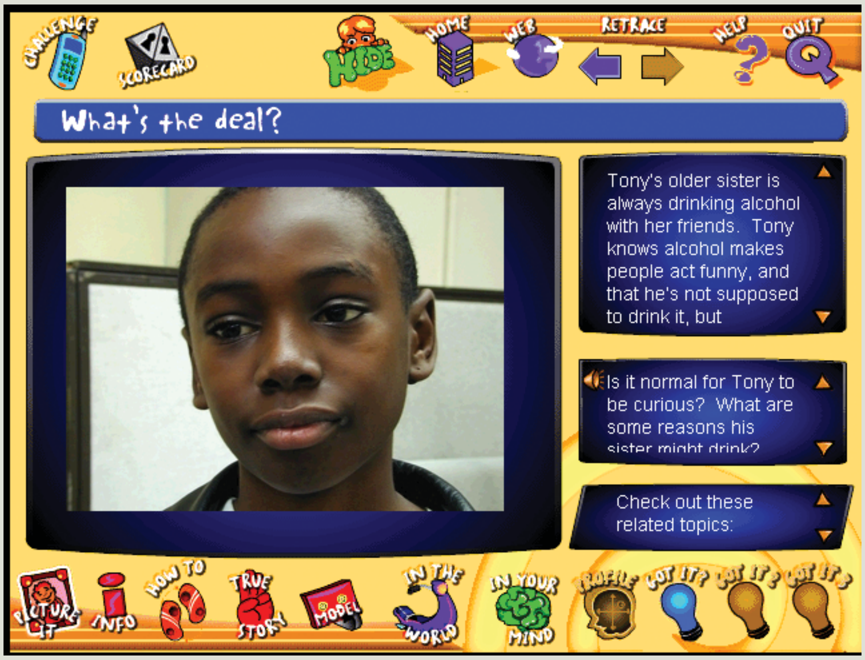
Such applications can be delivered as CD-ROMs, information kiosks shared by users, or downloadable installers to be executed on end users' computers. Image I shows an advanced application built with Macromedia Director (screenshot from Relate for Kids software, by Ripple Effects, Inc.).

Long Live Stand-Alone Applications

Before getting started, we should address the question of why we still need offline applications when everything else seems to be migrating to the Web. There is no doubt that the Web is a great tool for diffusing information and applications to millions of users. It also ensures you can easily keep your application fresh and content up-to-date.

Offline applications still, however, maintain their value and offer many advantages for certain types of applications and in specific situations. For example, delivering applications to today's booming mobile market in many cases needs to be done offline due to hardware and bandwidth restrictions. In addition, although bandwidth capabilities for the Internet continue to improve, certain ultra-rich media applications cannot be handled today even with broadband Internet. Furthermore, certain markets still lag

image I

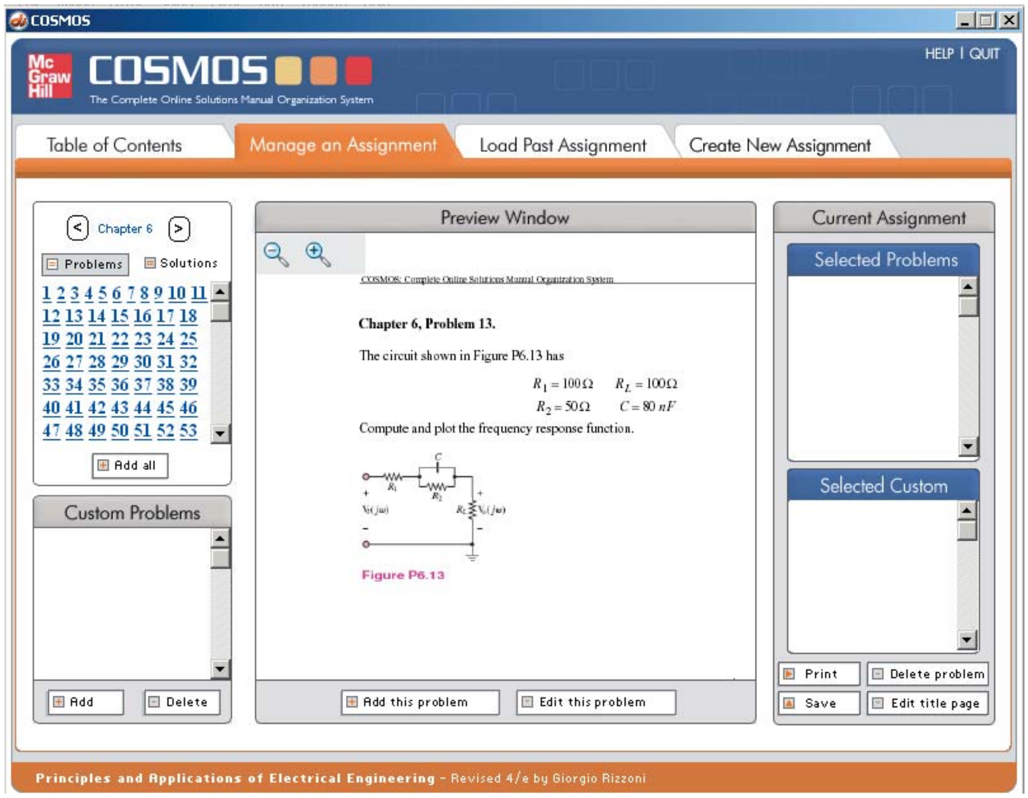


Tony's older sister is always drinking alcohol with her friends. Tony knows alcohol makes people act funny, and that he's not supposed to drink it, but

Is it normal for Tony to be curious? What are some reasons his sister might drink?

Check out these related topics:

image II



COSMOS: Complete Online Solutions Manual Organization System

Chapter 6, Problem 13.

The circuit shown in Figure P6.13 has

$$R_1 = 100\,\Omega \qquad R_L = 100\,\Omega$$
$$R_2 = 50\,\Omega \qquad C = 80\,nF$$

Compute and plot the frequency response function.

Figure P6.13

Principles and Applications of Electrical Engineering – Revised 4/e by Giorgio Rizzoni

behind and do not have the capabilities to easily stream large files over the Internet or to handle bandwidth-greedy applications. In cases where the Internet connection is poor or where networks are unavailable, delivering an application offline is sometimes the only way to go. Locally installing such applications can help ensure your project is optimally viewed and that end users don't tune out waiting for the information to be downloaded.

CDs/DVDs are proven money-making distribution methods that can be designed to reasonably protect content against illicit use. Online distribution of content and applications, on the other hand, is more difficult to control – and is a hot topic being discussed in courtrooms these days! If you are selling your application, it's definitely worth considering a distribution method that will ensure the highest return for you. For increased control over licensing, offline applications tend to be more secure than those distributed on the Web. The profitability of online distribution can be severely hampered by users sharing their IDs and passwords with others to access content. Having end users install the application on their local drive and enter a license key can make it a lot more difficult for your application to be distributed illegally. In addition, information and files tend to be more secure when locally saved than information stored in a Web-based application.

Another advantage of offline applications is from a completely nontechnical viewpoint. Offline applications can be delivered on CD-ROMs, which make for great tangible sales and marketing tools that can be distributed to customers and prospects. Having a physical item in hand has a stronger impact and is more memorable then simply giving out a URL. Furthermore, CD-ROMS can be customized and uniquely packaged to really stand out from the crowd. The nice thing as well is that, if positioned properly and designed to compliment existing online applications, they can actually be used as an effective tool for creating additional qualified traffic to your Web site.

Director, the Best Kept Secret in Town

Macromedia Director obviously is great for creating appealing and engaging user interfaces. With the predefined behaviors (which we will talk about later) and its cross-platform capabilities, development time is very efficient as you do not need to worry about low-level implementation. In addition, unlike some programming languages typically used for building applications, Director is very easy to learn and offers a user-friendly development environment similar to other Macromedia products.

Director also contains functionality that pertains to network integration, which makes creating hybrid (online/offline) projects fairly simple. The media support is excellent and now even includes support for DVD. This means you can use the best media type for the task at hand and not have to worry about converting it to another format.

How Director Differs from Flash

There is no doubt that both Flash and Director are great development tools, each with its own advantages. There are, however, a few key differences between Director and Flash when it comes to application development that are worth pointing out.

One of the main differences is that with Director, your application can access the end-user's operating system. This is ideal for applications where you need to read and write files to the user's hard drive or for when you need to access external applications and specialized hardware.

Another key difference is that Director has an extensible plug-in architecture. Plug-ins called Xtras allow you to add custom features and functionality to your application. You can create fully featured applications that can access, launch, and control other applications from within the Director executable. There are hundreds of existing Xtras on the market to choose from, and if you can't find what you are looking for, you can always build your own.

One final difference worth pointing out is Director's excellent and extensive support of media formats. Director supports practically all formats including DVD-Video, Flash, MP3, 3D, and AVI just to name a few. Because Director is capable of using these file formats directly, there is no need to convert them, which means the quality of the file will remain intact and there is less chance of errors occurring. This is also a great time-saver both when you are developing your applications and when you are updating it.

Image II shows a Director project using various media including PDFs (screenshot taken from the Cosmos project, by Hunt & Gather, Inc.).

Application Development – Best Practices

To properly develop an application, as with most things, great planning and quality management are essential. Application development can greatly benefit from an additional step called "architecture." This step essentially aims at devising a framework that adequately depicts the application and proposes a skeleton upon which every single feature will be built.

In the construction industry, buildings are designed first with an architectural drawing and, unless your building is identical to other ones, this drawing will be unique. The same applies for software architecture. One generic architecture cannot be applied to a collection of applications, and it is only when an application has been properly specified that it can be architected.

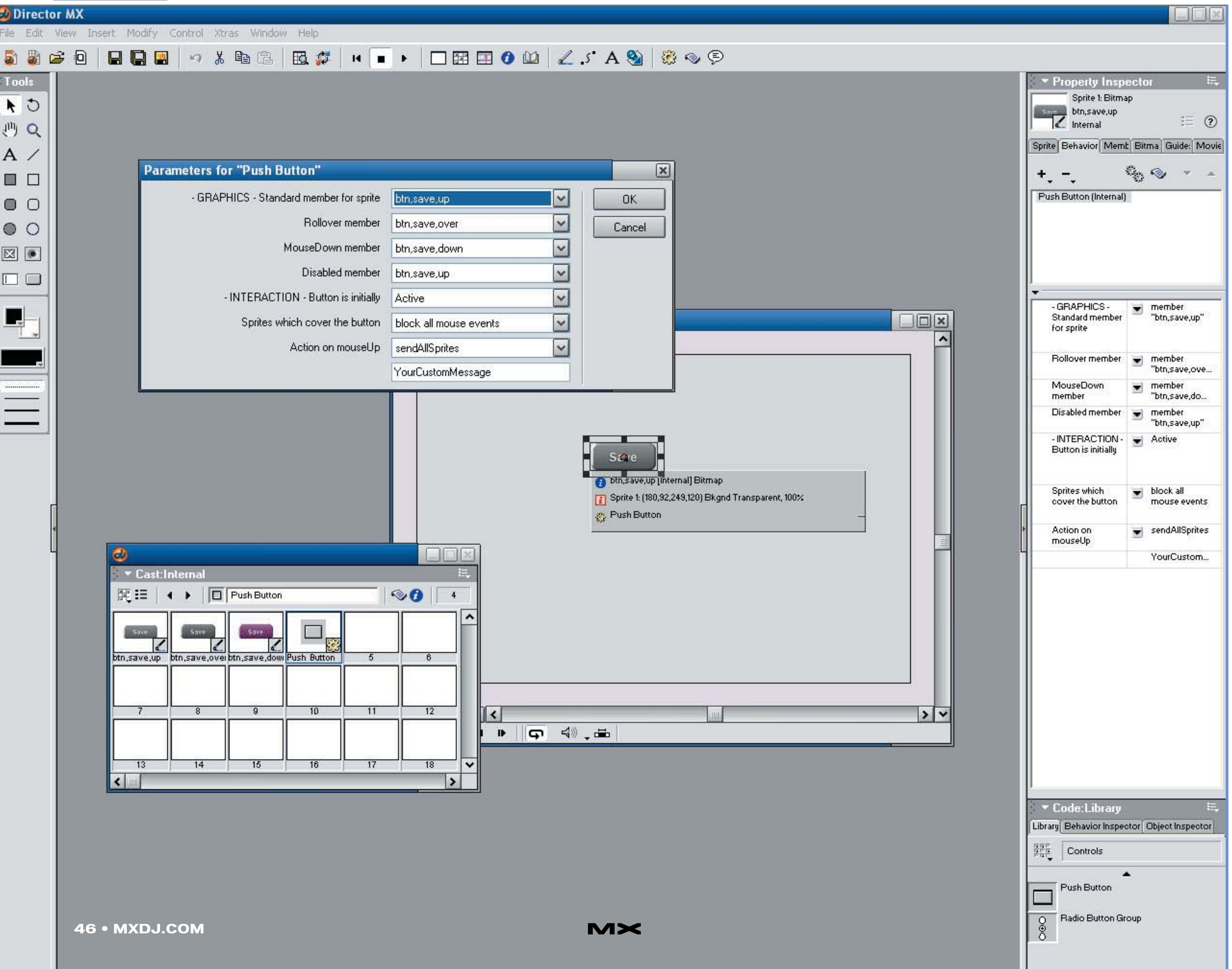
There are several proven methodologies that can help with the architecture of an application such as, for example, Design Patterns. Discussing the fundamentals of software architecture is in itself an article, so we won't go into it here. We will, however, discuss one approach that works well and that can provide a clean framework for designing applications. This approach involves breaking down your architecture in terms of three main layers: User Interface, Logic, and Data Management. Doing this will help provide a framework to properly design and plan your application.

The User Interface

Right out of the box Director provides all the basic tools required to quickly and efficiently implement a custom user interface. Items such as a text controls, push buttons, radio buttons, and check boxes are available and save a lot of development time. These buttons are essential in order for your user to be capable of interacting with your application. In addition, Director comes with widgets or visual controls such as menus and scroll bars.

While these elements are great in most situations, if you have spent a lot of time and effort designing a customized graphical user interface you may want to use something less generic that will integrate seamlessly with your design. Any bitmap or set of bitmaps can be combined with behaviors to create customized user interface widgets. For example, if the gray square button provided with Director is a little bland for your project you can create your own custom button using bitmaps, which have the same look and feel as the rest of your project.

image III


Parameters for "Push Button"

- GRAPHICS - Standard member for sprite	btn,save,up
Rollover member	btn,save,over
MouseDown member	btn,save,down
Disabled member	btn,save,up
- INTERACTION - Button is initially	Active
Sprites which cover the button	block all mouse events
Action on mouseUp	sendAllSprites
	YourCustomMessage

btn,save,up [internal] Bitmap
Sprite 1: (180,92,249,120) Bkgnd Transparent, 100%
Push Button

Director MX
File Edit View Insert Modify Control Xtras Window Help
Tools
OK
Cancel
Property Inspector
Sprite 1: Bitmap
btn,save,up
Internal
Sprite Behavior Memb Bitma Guide Movie
Push Button (Internal)
Cast:Internal
Push Button
btn,save,up btn,save,over btn,save,down Push Button
Code:Library
Library Behavior Inspector Object Inspector
Controls
Push Button
Radio Button Group

When your button is ready, simply put the default state on the stage and attach the push button behavior. In addition, when Director imports layered image files from applications such as Adobe Photoshop or Macromedia Fireworks, each layer of the image file is placed into separate cast members, removing the need to cut out each layer individually, which is extremely time-consuming. Image III shows a custom button using three graphics and Director's behavior

While the above example shows us how to create a customized standard button, we can take this one step further and create a button that has a custom animation when the user rolls over it. To achieve this there are a couple of options; however, the most reliable method is to create a button in Flash with the different states and import the .swf file. When the .swf cast member is dropped onto the stage it will automatically interact with the up, down, and over states of the button.

If your application needs to adopt the look and feel of the operating system that it is running on, you can use a third-party product called the OSControl Xtra. The OSControl Xtra allows you to create a user interface based on the operating system's settings and desktop themes. The interface will adapt to each end user's system and will look as if it was created for that particular OS.

The Text Asset Xtra, which comes with Director, is probably the most commonly used Xtra and is useful in almost every project. It can be used as a means to display your data to the end user or allow the end user to input or edit content.

Before Director MX 2004, advanced widgets such as a treeview, calendar, or datagrid used to be very costly and expensive to include in your Director project. Now, however, with the addition of Flash components in the latest version of Director, Macromedia has allowed us to include these items for free with little or no hassle whatsoever.

A very important issue at the moment is rendering software compliant to Section 508 of the U.S. Rehabilitation Act (accessible software for the disabled). Since Director MX, Macromedia has included a Speech Xtra and behaviors with Director, which developers can use to help render their project accessible. Developers can use these free tools to enable text to be read using any standard Text-to-Speech software on the end user's computer.

Logic

The logic of an application drives the user interface by using services provided by the Data Management layer. Logic should be considered distinct from the user interface and the data management, even though it is closely related to them both. Most of the programming happens at the logic level: we describe here what the application should react to, how it should react, and where to find the data needed in order to accomplish the required tasks.

Traditionally, Lingo script has been the programming language available for Director developers. Lingo, which is completely cross-platform, has been around for many years and is a well-established and documented language but is limited to Director development. Now, however, with the release of Director MX 2004, programmers have a second option: the increasingly popular JavaScript syntax.

The JavaScript syntax is ideal for developers who are accustomed to developing for the Web as they can use their existing knowledge to code in Director (no need to learn a new syntax). Using either of the Director syntaxes, developers can develop the following types of scripts in order to create the back end of their application.

Behaviors

Behaviors in Director are pieces of reusable script that are applied to sprites to make them behave in a certain manner or to do specific tasks. Behaviors come in very handy when you have a functionality that you would like to repeat on different sprites, as parameters can be included. Behaviors also allow for a new kind of teamwork. More advanced programmers can produce behaviors that nonprogrammers can easily use by dragging and dropping them onto their user interface elements. This is particularly useful when a number of user interface elements require the use of the same behavior.

Movie and Parent Scripts

Movie scripts can be used to respond to key presses, mouse clicks, movie

Advertising Index

Advertiser	URL	Phone	Page
ActivePDF	www.activePDF.com		25
CFDynamics	www.cfdynamics.com	866-233-9626	9
Electric Rain	www.erain.com	888-613-1500	Cover 3
FuseTalk	www.fusetalk.com	866-477-7542	17
HostMySite.com	www.hostmysite.com/mxdj	877-248-4678	29
Information Starage+Security	www.ISSJournal.com	888-303-5282	43
Interakt	www.interaktonline.com		5
IT Solutions Guide	www.sys-con.com	201-802-3021	49
Macromedia	www.macromedia.com/go/dwupdated		Cover 2
Macromedia	www.macromedia.com/go/max		Cover 4
PaperThin	www.paperthin.com	800-940-3087	35
Seapine Software	www.seapine.com	888-683-6456	6
SYS-CON e-newsletter	www.sys-con.com	888-303-5282	27
SYS-CON Publication	www.sys-con.com/2001/sub.cfm	888-303-5282	51
SYS-CON Reprints		201-802-3026	57
Web Services Edge East	www.sys-con.com/edge	201-802-3045	23

events such as start and stop, and your own custom events. They are available to the entire movie regardless of which frame the movie is in or which sprite the user is interacting with. They function as pieces of code that you want to use throughout your movie. Parent scripts are Director's way of providing basic object-oriented features to Lingo. Image IV shows an example of Director's scripting alternatives.

Data Management

The management of data in an application is a key consideration. When analyzing and planning your data there are two important items that need to be looked at: the structure of your data and how you plan to store your data. There are several methods available; however, the method you choose will depend greatly on the amount and kind of data you are dealing with.

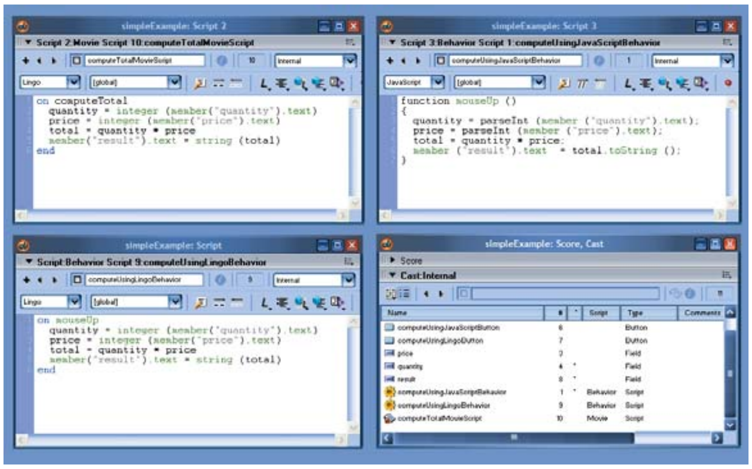
Let's use an e-catalog as an example. If your e-catalog only has a few products, you can store your content in separate cast members (one cast member for every piece of data). Each product would have its own frame. While this is probably the easiest way to manage your data within Director, if you are planning to update your content often this is not the best approach.

When dealing with larger scale projects (i.e., 30–60 products in your e-catalog) you can dynamically link your cast member to external files. This is done by having one cast member serve as the container for your content, while your data is organized using files and folders on the hard disk. You can load your external files by setting the filename of the cast member to the desired file on the hard drive.

When loading text into your application you can also use the FileIO Xtra that comes with Director. The FileIO Xtra allows you to open the desired text and read the content of the file into your text cast member. Using this approach also offers the major benefit that if the end user modifies the content of the cast member, you are able to save their changes back to the file located on their hard drive, an option that is not available when setting the filename of the cast member.

If, however, your e-catalog will contain hundreds of products that share similar attributes (item number, dimensions, color, price, etc.), or if your products require frequent updating, you should consider using a database Xtra to manage your data. This greatly facilitates the updating of your content by avoiding duplication of related information. There are several third-party database solutions available on the market. Choosing the right one for your project depends significantly on what your needs are. Some of the databases, such as the V12 Database Engine and FileFlex, offer a universal widest-reach kind of solution. These databases are cost effective for simpler projects, where they can reduce development time. Should your project require more advanced queries such as combinations of AND, OR, NOT queries, then databases like Valentina are more suitable. These more powerful database solutions, however, do require additional skill and experience to work with. If your project is Windows-only, Datagrip offers a good combination of both power and convenience. If you want to access a live database through a network, GoldenGate, ADOXtra and Arca are both worth considering, or if you don't shy away from Web server–side scripting,

image IV



simpleExample: Script 2
Script 2:Movie Script 10:computeTotalMovieScript
computeTotalMovieScript
10
Internal
Lingo
[global]

```
on computeTotal
  quantity = integer (member("quantity").text)
  price = integer (member("price").text)
  total = quantity * price
  member("result").text = string (total)
end
```

simpleExample: Script 3
Script 3:Behavior Script 1:computeUsingJavaScriptBehavior
computeUsingJavaScriptBehavior
1
Internal
JavaScript
[global]

```
function mouseUp ()
{
  quantity = parseInt (member ("quantity").text);
  price = parseInt (member ("price").text);
  total = quantity * price;
  member ("result").text  = total.toString ();
}
```

simpleExample: Script
Script:Behavior Script 9:computeUsingLingoBehavior
computeUsingLingoBehavior
9
Internal
Lingo
[global]

```
on mouseUp
  quantity = integer (member("quantity").text)
  price = integer (member("price").text)
  total = quantity * price
  member("result").text = string (total)
end
```

Name	#	*	Script	Type	Comments
computeUsingJavaScriptButton	6			Button	
computeUsingLingoButton	7			Button	
price	2			Field	
quantity	4	*		Field	
result	5	*		Field	
computeUsingJavaScriptBehavior	1	*	Behavior	Script	
computeUsingLingoBehavior	9		Behavior	Script	
computeTotalMovieScript	10		Movie	Script	

“With the release of Director MX 2004, programmers have a second option: the increasingly popular JavaScript syntax”

Director’s own HTTP communications functions can access CGI scripts.

XML is a very popular format, especially for data exchange. For instance if your company’s inventory system provides an XML file or an XML Web service, you can use Director’s XML parser to retrieve data from your Web site to parse, read, and use the content within the application. This can be very useful for updating the user’s local content to ensure that the application has the most recent product information.

In most situations using one of the above solutions is not enough. To get the best results you will probably have to mix and match the different methods to come up with the approach best suited to your project’s requirements. Along with the above mentioned solutions, it is worth mentioning that Director also provides two very useful data structures for managing small amounts of data: lists and property lists. These structures are excellent for managing the data behind the user interface, such as populating drop-down menus with item categories and recording user’s choices and selections.

Conclusion

Although the trend today is toward online applications, the need for offline applications is still evident and the future remains bright. There are a lot of opportunities still out there for those willing to venture off the online path or for those considering creating hybrid online/offline projects. If this is the route you are looking to explore, then Director is definitely something to consider for your arsenal.

However, as with any kind of development, the success of a project is not solely based on the tools you select. Having the right tools in hand is an important aspect, but a successful project is often also a result of having the combination of the best methods and the required engineering skills to get the most out of the tools you select. With the right skills and methods, Director can be a very powerful tool; in fact when used optimally it could very well be the best-kept development secret in town!

Useful Links

For those who are new to Director, Macromedia offers a free fully functional trial version of the product on their site, www.macromedia.com/downloads.

Third-Party Xtras

- *OSControl Xtra*: http://xtras.openspark.com
- *V12 Database Engine:* www.V12DBE.com
- *Valentina:* www.paradigmasoft.com
- *Datagrip:* www.datagrip.com
- *FileFlex:* www.fileflex.com
- *ADOXtra:* www.mediamacros.com
- *Arca Database Xtra:* http://xtras.tabuleiro.com
- *GoldenGate:* www.ggdbc.com

Images

- *Relate for Kids Project: Ripple Effects, Inc.:* www.rippleeffects.com
- *Cosmos Project: Hunt & Gather, Inc.:* www.huntandgather.com

Jason MacDonald is a Multimedia Project Manager for Integration New Media. He has been developing projects with Macromedia Director for the past seven years, and is an active member of the Director community.
jason@integrationnewmedia.com

Paul-Catalin Oros has been working for Integration New Media as project manager for the past eight years. Paul specializes in database management and Web applications and works closely with Jason on all multimedia projects requiring advanced data management functionalities.
paul@integrationnewmedia.com

xile

written & illustrated by louis f. cuffari 8

I think the webmaster is suffering from delusions of grandeur.
I am honored to accept this award for web design excellence...
...and I would like to thank the following who have inspired me...
Quit dreaming!
Sit down!

A LIMITED TIME SAVINGS OFFER FROM *SYS-CON MEDIA*

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO $340 AND RECEIVE UP TO 3 FREE CDs!*

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

3-Pack

Pick any 3 of our magazines and save up to $210.00
Pay only $99 for a 1 year subscription plus a FREE CD

- 2 Year – $179.00
- Canada/Mexico – $189.00
- International – $199.00

6-Pack

Pick any 6 of our magazines and save up to $340.00
Pay only $199 for a 1 year subscription plus 2 FREE CDs

- 2 Year – $379.00
- Canada/Mexico – $399.00
- International – $449.00

9-Pack

Pick 9 of our magazines and save up to $270.00
Pay only $399 for a 1 year subscription plus 3 FREE CDs

- 2 Year – $699.00
- Canada/Mexico – $749.00
- International – $849.00

CALL TODAY! 888-303-5282

Pick a 3-Pack, a 6-Pack or a 9-Pack

☐ 3-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 6-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 9-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.

TO ORDER •Choose the Multi-Pack you want to order by checking next to it below. •Check the number of years you want to order. •Indicate your location by checking either U.S., Canada/Mexico or International. •Then choose which magazines you want to include with your Multi-Pack order.

☐ **LinuxWorld Magazine**

U.S. - Two Years (24) Cover: $143	You Pay: $79.99 /	Save: $63 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $32
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $79.99 /	Save: $4
Int'l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **JDJ**

U.S. - Two Years (24) Cover: $144	You Pay: $99.99 /	Save: $45 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $69.99 /	Save: $12
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $89.99 /	Save: $40
Int'l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **Web Services Journal**

U.S.- Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **.NET Developer's Journal**

U.S. - Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **Information Storage + Security Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $93 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $39
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $88 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $34
Int'l - Two Years (24) $216	You Pay: $89.99 /	Save: $126 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $59.99 /	Save: $48

☐ **MX Developer's Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $93 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $32
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $88 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $34
Int'l - Two Years (24) $216	You Pay: $89.99 /	Save: $126 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $59.99 /	Save: $48

☐ **ColdFusion Developer's Journal**

U.S. - Two Years (24) Cover: $216	You Pay: $129 /	Save: $87 + FREE $198 CD
U.S. - One Year (12) Cover: $108	You Pay: $89.99 /	Save: $18
Can/Mex - Two Years (24) $240	You Pay: $159.99 /	Save: $80 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $99.99 /	Save: $20
Int'l - Two Years (24) $264	You Pay: $189 /	Save: $75 + FREE $198 CD
Int'l - One Year (12) $132	You Pay: $129.99 /	Save: $2

☐ **WebSphere Journal**

U.S. - Two Years (24) Cover: $216	You Pay: $129.00 /	Save: $87 + FREE $198 CD
U.S. - One Year (12) Cover: $108	You Pay: $89.99 /	Save: $18
Can/Mex - Two Years (24) $240	You Pay: $159.99 /	Save: $80 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $99.99 /	Save: $20
Int'l - Two Years (24) $264	You Pay: $189.00 /	Save: $75
Int'l - One Year (12) $132	You Pay: $129.99 /	Save: $2

☐ **PowerBuilder Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

*WHILE SUPPILES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Subscribe Online Today www.sys-con.com/2001/sub.cfm

SYS-CON MEDIA

Best Practices in Kiosk Design

kiosk systems provide a unique set of development conditions and challenges. Whereas most multimedia development produces a software application, a kiosk is a collection of physical, hardware, software, and support systems. This article attempts to look at the entire kiosk project, in hopes that you will have control over most of it.

by roy crisman

Design the Process

You already know to do your information architecture (IA), user interface (UI), and usability design for software before the visual layout and software development. You may even have the time, budget, and foresight to do usability testing focused on your target user base with wire frames on paper or with quick software mockups before starting on graphics and code. But for a kiosk, the product is more than an executable file; it is the entire system and process. Besides creating a simple and intuitive UI, you want to simplify the installation, deployment, and support of your kiosk.

Run the entire process through an IA/UI/usability design phase. Create end-to-end use cases; remember to include service and support personnel as users as well as the end user. Consider every aspect you are responsible for or interact with. Don't forget:

- Computer hardware
- Peripherals
- Cabinetry
- The operating system (OS) and third-party software
- Networking
- Installers
- Documentation and training manuals
- Online portions
- Administration and configuration components
- UL-type consumer safety testing and certification
- Contracts with third-party vendors
- Support systems

If no one considers a component, such as the base system OS, as a part of the end product, no one will be assigned responsibility for it and have time budgeted for it. Parts of your process that fall through the cracks will fall to whoever has the free time or gets stuck with it at the last minute, not the best person for the job.

Your initial sketches, charts, and use cases don't have to be incredibly detailed. It is better to realize that these "work products" are just as important to plan, document, and have in source control as the code. Continue to add to them and assign responsibility for the sections as the project grows.

We Control the Hardware

Choose Your Platform

Look at the entire scope of the project to determine which platform(s) you are going to focus on and develop on. It may be helpful to develop for full cross-platform compatibility even if the in-kiosk machine specification is for only one. A cross platform–compliant program can quickly turn into a demo CD-ROM or be viewed by that one client who uses "the other platform." You also want to try to keep your options as open as possible to limit your risk to hardware going out of production. There are several limiting factors on your hardware selection: certain required hardware components, components without drivers or features for a particular OS, sponsorship arrangements with hardware providers, and required software components such as an ActiveX controller for a digital camera.

Use industrial hardware components whenever possible for parts that the end user interacts with. Or, use cheap consumer devices with plenty of spares and a quick replacement process. Most consumer keyboards, mice, trackballs, and joysticks weren't made to withstand chil-

dren playing on them and drinks being spilled on them daily. I have seen consumer components with impressive technical specs used – and replaced after a year's worth of sub-par performance.

With keyboards there are several options:

- ***On-screen with a touch-screen monitor:*** This works best for a small amount of data. It may be more prone to vandalism and require more frequent cleaning, and have a major impact upon the UI/usability and visual design.
- ***Custom industrial keyboard:*** This is a more expensive option and probably requires a third-party vendor, may require a Quality Control/Quality Assurance cycle of its own, and allows you to define exactly which keys are available to the user.
- ***Standard keyboard (industrial or consumer):*** This keyboard already exists, is less expensive and more thoroughly tested than a custom one, and will require some work with the OS to lock out certain keys and key combinations to protect your system.

Choose your industrial hardware supplier and/or physical kiosk fabricator carefully, though. My last kiosk project had a running joke about the vendor's "Don't worry about it, we'll handle it" attitude and inability to complete even simple tasks. They shipped us a "new replacement trackball" that had been removed from service three months prior. The stainless steel kiosk cabinetry arrived on location in a distant city without a hole for power cables to exit. Custom industrial keyboards had their letters rubbed off by use within six months.

Just as you may separate generic code and graphics from specific, it can be helpful to keep user-generated data separate from the OS and application. When the OS hard drive becomes physically damaged, you won't lose any of the previous data. You may even separate the OS from the application.

And We Control the Software

The basic kiosk is just an application program (the Show) running in a box. It probably has an "attract loop" that runs between user sessions, and a time-out mechanism that detects when someone has walked off and returns the application to the attract loop.

The Show (and other kiosk applications) should create at least a couple of levels of logs. A user-interaction log will record the path the user takes through the application in order to data mine for information about how the kiosk is used. Each component may also have its own process or debug log to record the progress through the code, the beginning and finishing of functions and methods, relevant variables and states in the case of trapped errors, interactions between components on the kiosk, as well as attempts to talk to other machines and network processes. These debug logs are indispensable in solving problems if the kiosk does lock up, crash, or error, or even in knowing whether the problem is the code module, the data, the network, or a change to another component having side effects. All log files should be written locally in case off-kiosk communication isn't working and may also be remotely accessed or uploaded to a central location. The data may also be transmitted to a server as it is generated so you can track exactly where a user is, or as part of some multi-user system.

Logging data can be very important for understanding the long-term usage of your kiosks. It also may be important to have archived for historical data: I have parsed through debug logs to discover that the 1.3 release was actually more stable than the 1.2 release – despite the client's convictions to the contrary. Version 1.3 was receiving refocused attention due to the upgrade, showing the weakness of relying on anecdotal evidence of kiosk performance.

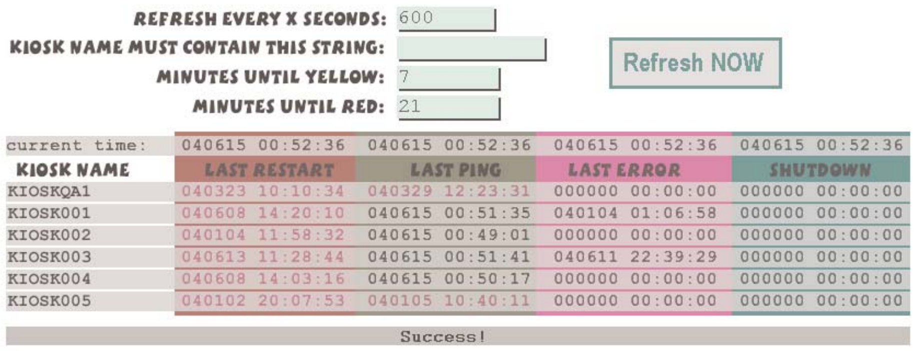
The Show and kiosk components have a variety of ways to transmit data depending upon your situation: it can be sent immediately via a direct connection, sent as tasks are completed (at the end of a user session), or queued up in batches to be sent only late at night.

Use audio sparingly and wisely. Audio should work with the entire kiosk, not be something someone just throws in at the end. If you do have audio, consider what can be done to make it more tolerable to any people who work around your kiosk installation. There can be quite a lot of intentional damage to your kiosk installation if it aggravates the people who have to work around it. Even during development, it may not be long before someone who sits near the QA machine asks someone to turn off the sound. (A cheap pair of headphones left plugged into each machine may help ensure office tranquility and allow people to hear the audio when they need to.) Attract loop audio can be especially annoying. Adding randomness and variability to the attract loop audio can greatly increase the experience for people exposed to the kiosk for long periods of time.

Herd animals use deep bass "lowing" sounds because these can travel over greater distances and are hard for predators to pinpoint the location of. Unless you are creating a kiosk about cows, dinosaurs, or the science of sound waves, you should avoid bass, especially when you have a group of kiosks together. This also means you can save money by not buying a subwoofer for the kiosk sound system.

One way to reduce the risk of the effect of downtime due to computer errors is a software or hardware "watchdog" component that reboots the machine if the show quits giving it a sig-

image I



REFRESH EVERY X SECONDS:	600
KIOSK NAME MUST CONTAIN THIS STRING:	
MINUTES UNTIL YELLOW:	7
MINUTES UNTIL RED:	21

current time:	040615 00:52:36	040615 00:52:36	040615 00:52:36	040615 00:52:36
KIOSK NAME	LAST RESTART	LAST PING	LAST ERROR	SHUTDOWN
KIOSKQA1	040323 10:10:34	040329 12:23:31	000000 00:00:00	000000 00:00:00
KIOSK001	040608 14:20:10	040615 00:51:35	040104 01:06:58	000000 00:00:00
KIOSK002	040104 11:58:32	040615 00:49:01	000000 00:00:00	000000 00:00:00
KIOSK003	040613 11:28:44	040615 00:51:41	040611 22:39:29	000000 00:00:00
KIOSK004	040608 14:03:16	040615 00:50:17	000000 00:00:00	000000 00:00:00
KIOSK005	040102 20:07:53	040105 10:40:11	000000 00:00:00	000000 00:00:00

Success!

nal on a regular interval. A hardware version is more reliable than a software solution because the machine may lock up beyond the ability of the software to restart the system. The watchdog doesn't solve any problems with the kiosk crashing, and any user interacting with the kiosk when it crashes will still have a bad experience; however, if the kiosk is automatically back up and running shortly, it may still give many users a good experience. A kiosk that has crashed or locked up (possibly with sensitive user information left on screen) and sits for hours or days can give a lot of potential users a bad experience.

The kiosk may also require access to configuration settings, utility functions, and the standard OS. It may be helpful to be able to view logs, turn "restricted keys" on the keyboard back on, or to re-enable the cursor on a touch-screen kiosk for development and testing with a mouse. These configuration components and utilities may also be useful as parts of the Console.

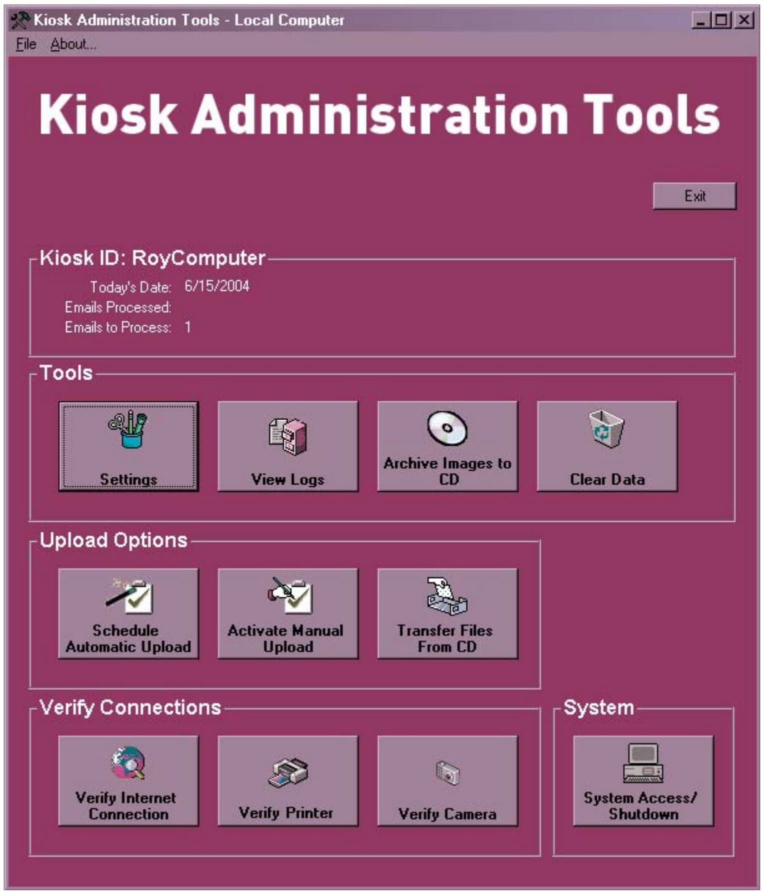
The basic idea of the Console is to be able to control any kiosk from anywhere. Technically proficient on-site support is expensive; design and develop as if you will never be able to have a live human configure or support it again. If programmed well, a configuration application may work on kiosk, on a local console machine, and as a remote console, possibly through a Web interface. By making these features usable anywhere, you can change settings or fix problems from around the world or right in front of a suspect kiosk with a wireless device, with the same interface.

The console should be able to make configuration changes to the kiosks, either to a specific kiosk, or broadcast changes to groups of them. It may be useful to be able to control a kiosk by sending it a message that simulates a button push, allowing you to remotely "drive" the kiosk through a workflow.

So, How Are the Kiosks Doing?

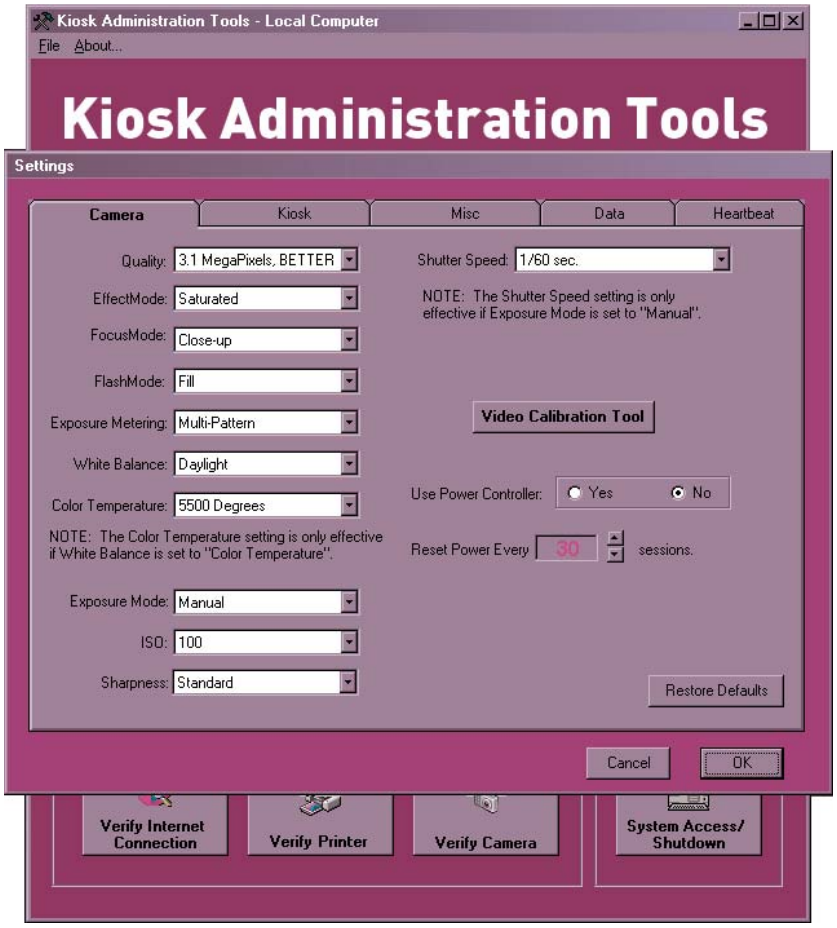
One of the most useful features for your remote console is a "heartbeat monitor." (see Image I). The Show can send a "heartbeat" at a regular interval to a remote server, and the heartbeat monitor allows you to view that information. The

image II



Kiosk ID: RoyComputer
Today's Date: 6/15/2004
Emails Processed:
Emails to Process: 1

image III



Quality: 3.1 MegaPixels, BETTER
EffectMode: Saturated
FocusMode: Close-up
FlashMode: Fill
Exposure Metering: Multi-Pattern
White Balance: Daylight
Color Temperature: 5500 Degrees
NOTE: The Color Temperature setting is only effective if White Balance is set to "Color Temperature".
Exposure Mode: Manual
ISO: 100
Sharpness: Standard
Shutter Speed: 1/60 sec.
NOTE: The Shutter Speed setting is only effective if Exposure Mode is set to "Manual".
Use Power Controller: Yes [✓ No]
Reset Power Every 30 sessions.

heartbeat contains a minimal amount of information: kiosk ID, time and date (if the kiosk is scheduled, otherwise skip it and just use the server's time), and maybe what state the kiosk is in. Some useful states are "StartUp" when the machine reboots, "Error" if an error has been trapped and the kiosk is in an error state, "OK" when the kiosk is running fine, or any other modes or states the kiosk has, such as a "Night" or "Closed" mode, or periods of inactivity. The heartbeat monitor then allows to you see what is going on with your kiosks. The minimal heartbeat monitor may just display a green light next to a kiosk name that is okay, a red light next to one that isn't.

Without this, you don't have an answer to the question, "How are the kiosks doing?" You can also display more information such as state, timestamps for last error, last restart, last OK ping, etc. With this information, you may end up alerting support staff to things such as the network being down or kiosks needing rebooting, or that maybe something else is wrong because, for example, no has used Kiosk 12 all day.

The heartbeat monitor can also come in handy when you're running QA tests. Your team can monitor the 10 machines across the hall in the QA lab all day instead of walking back and forth only a couple of times a day.

The holy grail of modern kiosk development is remote updating. If you never have to physically intervene with a kiosk in order to change the underlying software, you can greatly decrease the cost of bug fixes, content changes, upgrades, and updates. The console should be able to schedule remote updates and view versioning information for all the kiosks.

The console application may also be able to create and display other kiosk information, including data-mining information. Any information that can be used to chart user interactions, display usage statistics, orders, purchases, media usage, kiosk uptime or downtime, revenue, etc. Image II shows the console application main screen; Image III shows the console "settings" screen.

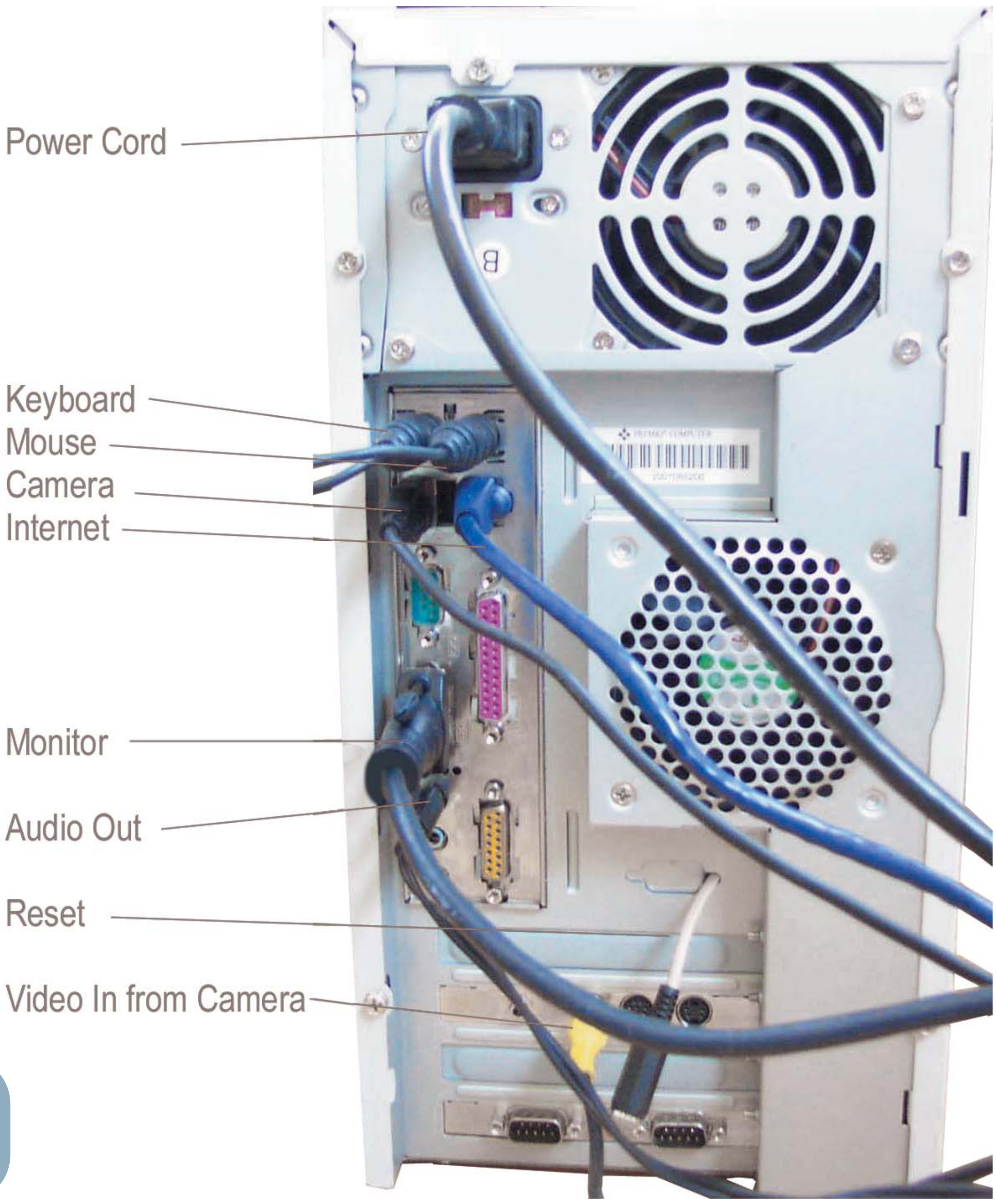


image IV

The Operating System and Supporting Applications

The creation process of the base OS and the installation of any drivers or supporting application programs is just as important to the success of your kiosk as the kiosk show application. Any third-party software required for your kiosk to run should be handled here. This process should be well documented, controlled, and versioned. Small things like the order of installing drivers may be important. Don't upgrade any component without realizing the inherent risk that change can pose. Make changes one at a time with testing, just as you would test major software changes. The more open-ended you can build this process, the better. Work to have a single OS recovery package that will adapt and work for variant hardware.

Quality Assurance, Too

Develop on Your Target Platform and Environment

The more restricted the target hardware/OS/supporting application environment is the more important this is. I know...I've got a snazzy laptop and I don't like the "other" platform either. But it takes only a couple of seconds to test a feature or make a screenshot to illustrate a problem on the target platform. And if I can't control that camera with my preferred system or I can't interface with the database because we don't have a network, the development process will take much longer.

Visual designers should also inspect their artwork on the target hardware displays. Graphics look a lot different in a dark cubicle on a perfectly tweaked 21" flat-screen monitor than they may on an out-of-the-box touch-screen monitor in a brightly lit room. LCD monitors may have problems displaying high-contrast images. Find out that the black, white, and red interface looks like with badly compressed JPG graphics on the kiosk's LCD screen before the client has signed off on the design and all the hardware has been purchased.

Test on Your Target Platform

Independent of your development system, you should have a target platform with controlled and documented hardware, OS, and supporting applications.

Test for the Long Term

Most multimedia application programs are run for only a relatively short amount of time in one session and can be recovered by the user if necessary. But tiny memory leaks can add up over 12 or 72 hours, and may be caused by your application, faulty supporting applications, or even a specific combination of motherboard, audio card, video card, and drivers.

Have at least a basic test constantly running on the target platform. I've been known to utilize "RECORDER.EXE" from Windows 3.1 to record primitive user interactions and then loop them for a week. There are much more sophisticated testing programs available. Target a full week of uptime, but build in a daily reboot if your product allows.

You'll also want to have a real QA process. Because you have a narrow target platform, QA can spend their time on the software functionality, rather than testing for compatibility issues. QA should evaluate and verify the entire process, not just the program, whenever possible, from building the OS, adding supporting applications, installing the application, configuring and deploying, and hardware component replacement.

Support

While you may not be actively involved in supporting the kiosk over the full life of its deployment, there are many things that you can do to aid the support of the system.

Keep a backup of the base OS and of the OS plus the kiosk application at hand at any deployment. This can be used to wipe a troublesome machine clean or to format a replacement machine.

Ralph Waldo Emerson said, "A foolish consistency is the hobgoblin of little minds," but a reasonable consistency is necessary for kiosks. Imagine trying to fix an upside-down kiosk monitor in a remote location and discovering that two of the screws in the bracket holding the monitor require a star screwdriver bit. Work for consistency. Watch for computer hardware consistency – chipsets change in production runs and you never realize the variability in what appears to be the same video card until something doesn't work. Demand consistency from any third-party providers.

Take pictures of the hardware when it is set up correctly, as an illustration. Take pictures of the finished and deployed kiosks for reference later (see Image IV). It may take a long time to figure out remotely that the problem with the kiosk is related to which USB port the digital camera is plugged into.

Keep an adequate supply of replacement hardware on hand at the deployment site. Hardware seems to fail in pairs, but it may be that hardware failures are only noticed and reported by local staff after multiple units have failed. If you have a bug reporting console component that can be used on-kiosk or on a console machine, you will receive more accurate and helpful data than someone trying to decipher a handwritten log a week later. Remember that local personnel are already fully employed; don't count on adding any task to someone's job, even if it will "just take a minute." The security guard doesn't want the added task of shutting down all your machines every night any more than you do. Make sure any contracts and agreements define who is responsible for and who pays for upkeep, cleaning, checking on the physical system (blown monitors) and theft/vandalism.

How hard or easy it is to support your kiosk systems is going to depend upon how well you are able to document, control changes, and simplify systems during development.

Conclusion

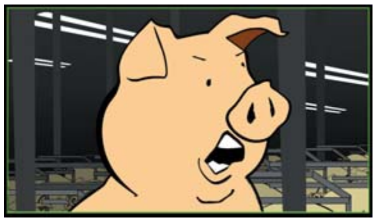
Kiosk systems can be rewarding to develop, allowing for far greater control over the system and the ability to push the limits of multimedia development. A kiosk is so much more than the interface the user interacts with, and thus requires more planning and development than typical multimedia software.

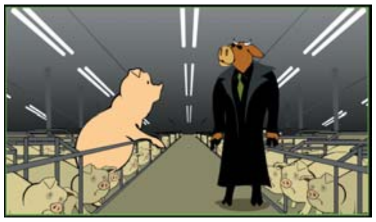
Roy Crisman is a multimedia developer who has developed kiosks systems for the Adler Planetarium, The Smithsonian Institution, Epcot Center, the El Capitan Theatre, National Parks, and traveling special events for Eastman Kodak and Disney. MXDJ@brokenoffcarantenna.com

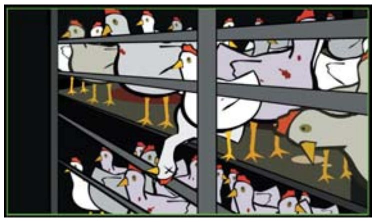
The Meatrix

To date, over 5 million viewers have logged on to The Meatrix, (www.themeatrix.com), a cutting-edge Flash animation piece. Through a cartoon pig named Leo, and a trenchcoat-clad cow named Moopheus, The Meatrix is educating consumers on the consequences of factory farming while touting the benefits of sustainable meat through a highly effective persuasive tool – humor.

The Meatrix was created by Free Range Graphics (www.freerangegraphics.com), a design firm serving non-profit groups, as part of its annual Free Range Graphics Flash Grant, which was awarded in 2003 to the Global Resource Action Center for the Environment (GRACE, www.gracelinks.org).







electric rain swift3D® version 4

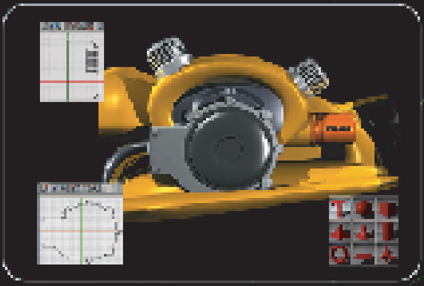
THE ULTIMATE 3D FLASH™ TOOL

ThereÕs 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 3DÕs 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.

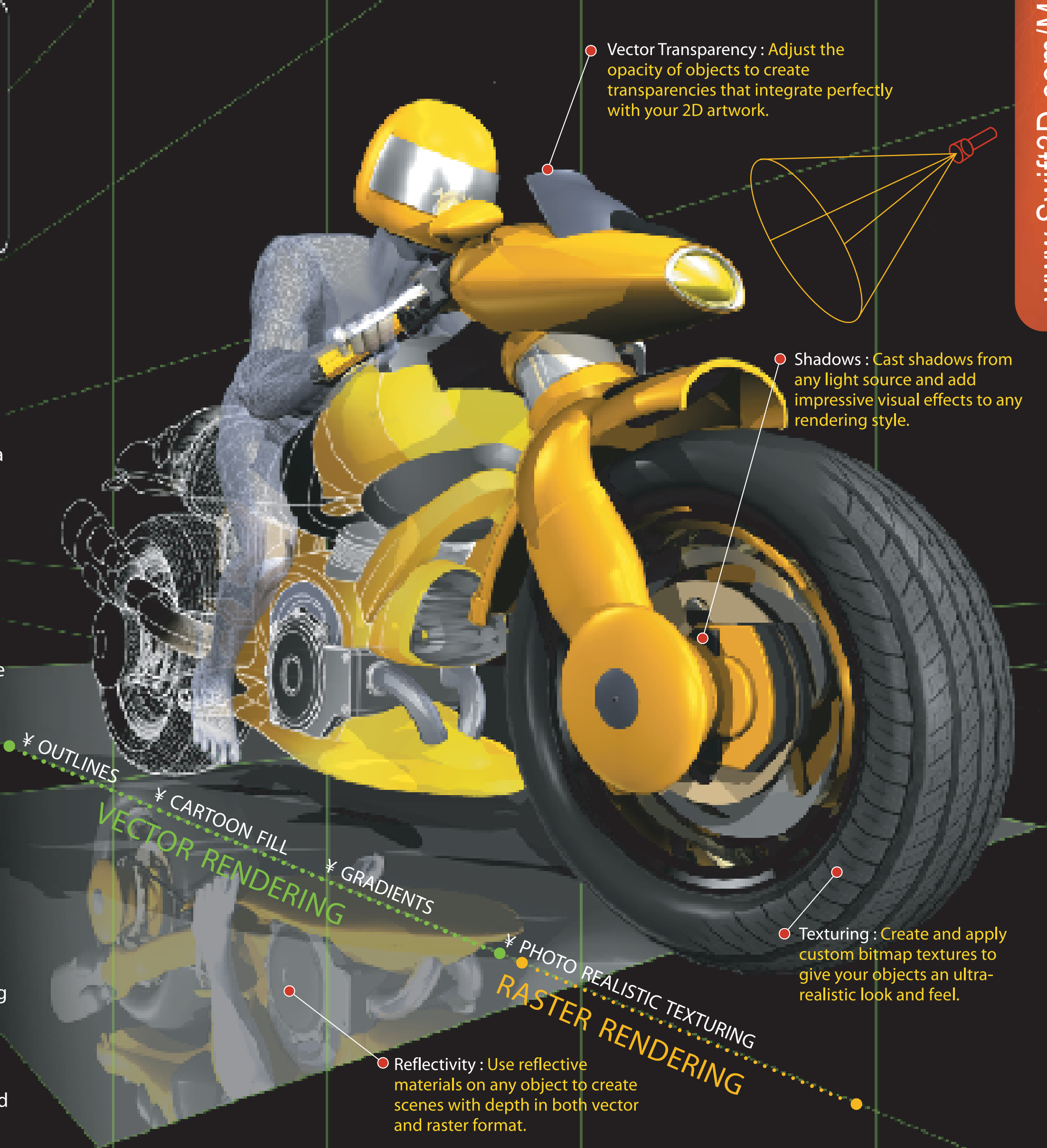
Use a frame-based timeline: Start with pre-built animations for quick results or tap into the full-featured Flash-based keyframe timeline.

GROW into advanced features

Polygonal Modeling: Use a powerful modeling system thatÕs 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.

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

electric rain®
www.erain.com 1.888.613.1500

Add the third dimension to your Flash!
Swift 3D v4 Product Info: http://www.Swift3D.com/MX