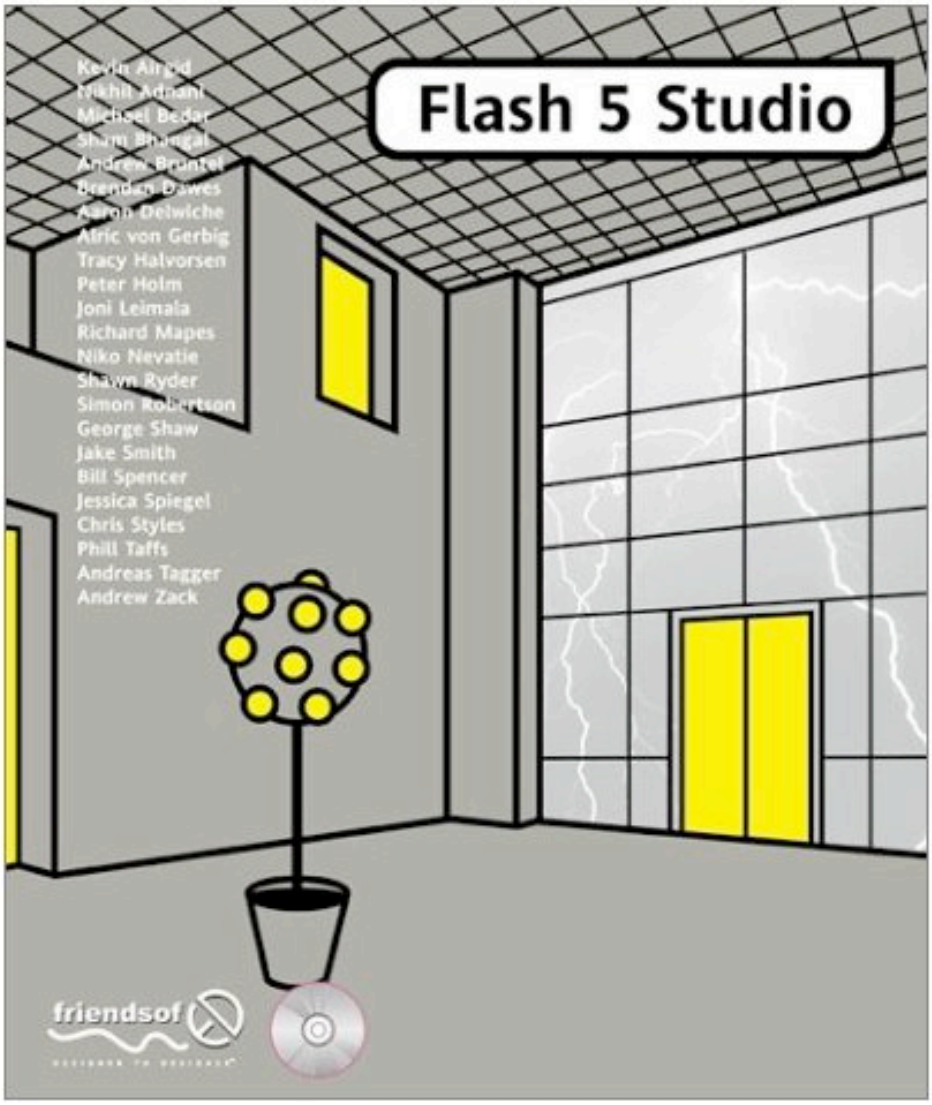

Flash 5 Studio
Kevin Airgid
Mahit Adnani
Michael Bedar
Sham Bhangal
Andrew Brontel
Brendan Dawes
Aaron Delwiche
Atric von Gerbig
Tracy Halvorsen
Peter Holm
Joni Leimala
Richard Mapes
Niko Nevatie
Shawn Ryder
Simon Robertson
George Shaw
Jake Smith
Bill Spencer
Jessica Spiegel
Chris Styles
Phill Taffs
Andreas Tagger
Andrew Zack
friendsof

Flash in its Natural Environment

Flash movies don't live in a vacuum. Without a browser in which to view the movie, a page to view the movie on, and a website to give the movie context, a flash movie is just another computer file. An understanding of some of the principles of site architecture is essential to be able to design a site that utilizes flash. This includes knowing how the movies will sit on a page, how they will interact with other elements on the page, and how that page will be displayed on various web browsers on various platforms.

Even the most beautifully executed animation or perfectly communicative piece of design is incomplete without a carefully considered context. In the simplest of situations, proper architecture can be as easy as building and embedding a movie at an appropriate size. Complexity can range all the way up to multiple movies in several layers embedded in variable sized frames, communicating with each other via javascript or perl…and beyond. By understanding the issues associated with online viewing of flash by various browsers, and even just by understanding that there IS AN ISSUE to understand, you'll be able to create websites that utilize flash in a far more elegant and complete way.

The first step in the design of any web site should be establishing a tentative architecture for the site. This architecture may change as the site design progresses, but without at least a tentative plan, key decisions will be impossible to make.

First is the decision to use flash at all. There is a lot that can be done on a site WITHOUT flash, so it's a bad idea to simply assume that flash will be used. If scalability (the ability of your graphics to grow to fill larger monitors and shrink to fit smaller ones) is important, then flash should probably be used. If high levels of interactivity and animation will be used in the design, then flash is probably the way to go. If sound will be very important, then, again, flash will probably be your best bet. If the site will be largely textual content, or will be database driven, and doesn't contain much animation (most e-commerce sites, for example), then flash will likely be a burden to the design. If the expected user base for the site is not very 'computer-savvy', then flash might not be a great idea as they likely won't have the plug-in installed. There are countless other factors involved in choosing to use flash (or any other proprietary technology) on a website, but a general rule should be – if you can't think of a good reason why you NEED to use the technology, then you probably shouldn't use it. All that being said, it should be noted that flash is generally regarded as the best thing to happen to web designers. When utilized skillfully, flash can USUALLY enhance just about ANY site.

Once the decision has been made to use flash in some capacity on the site, one should decide whether there will be html content in addition to flash content. If there is no html content necessary, one should then decide how flash will be utilized – full screen or partial screen (a design decision that will possibly change once design is begun), one movie or several linked movies, one movie or several stacked movies, and finally how non-flash visitors will be dealt with. If there will need to be html content, the next decision is how the html content will co-exist with flash – flash headers, flash as inline images, or flash pages linked with html pages are all possibilities. From these options, a plan can be devised and design can begin on the site.

Intro to HTML + Flash

The only way to fully grasp the concepts of embedding and working with flash movies in context is to have at least a cursory understanding of html and how to code pages 'by hand'. Applications such as Aftershock, while useful to the novice, can be limiting and constricting to the working professional who must deal with a wide variety of situations regularly. Complex html and esoteric tags are not important for the purposes of working with flash in most instances. Knowledge of the basics is, however, very important.

When you work in flash, the files you create (the movies) are swf files. These files are displayed online via a web browser (netscape, IE, etc) by embedding the files onto pages. Embedding simply means calling an swf from an html page. When a user arrives at the page, the page 'sends' them the swf. The code on the page (the embed tag among other things) dictates how that swf will be viewed by the web browser.

A sample embed tag:

```
<OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"

    WIDTH="100%"
    HEIGHT="100%"
    CODEBASE="http://active.macromedia.com/flash5/cabs/swflash.cab#version=5,0,0,0"
    ID="decontrol">
    <PARAM NAME="MOVIE" VALUE="mainpage.swf">
    <PARAM NAME="PLAY" VALUE="true">
    <PARAM NAME="QUALITY" VALUE="best">
    <PARAM NAME="LOOP" VALUE="true">
    <PARAM NAME="SCALE" VALUE="showall">
    <PARAM NAME="MENU" VALUE="false">

<EMBED SRC="mainpage.swf"
    NAME="decontrol"
    WIDTH="100%"
    HEIGHT="100%"
    PLAY="true"
    QUALITY="best"
    LOOP="true"
    SCALE="showall"
    MENU="false"
    SWLIVECONNECT="false"
PLUGINSPACE="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=
ShockwaveFlash"
    TYPE="application/x-shockwave-flash">

</EMBED>
</OBJECT>
```

While it may appear complex, the embed tag is, in fact, very simple. First, every piece of information is contained twice – once for netscape and once for Internet Explorer (IE uses the 'object' portion of the tag, while NS uses the 'embed' portion). Often, the only portions of the tag that will change from document to document are the source (src in embed, movie in object) and the scale factors (width and height in both sections). Play is almost always true, quality is almost always best, loop is almost always true, scale is almost always showall, menu is usually false, swliveconnect can vary depending on the technology your site uses, but in the beginning can usually be false. Pluginspace refers to the location users without the plug-in will go to download the plug-in and consequently rarely changes. Type doesn't change. It is a common practice to simply copy an embed tag from an existing html page and modify the necessary parameters to work in the new context.

Another aspect of html that is vitally important when working with flash is frames. Frames are a feature of html that allow you to display two or more html pages simultaneously in a viewer's browser window. The height and width of each html page's 'frame' are dictated by another html document called a frameset. The frameset indicates the size and shape (as well as other features such as whether individual frames will be 'scrollable') and specifies which html pages will fill those frames.

Sample HTML frameset –

```
<HTML>
<HEAD>

<TITLE>((D+CON/trol))</TITLE>

</HEAD>

        <frameset  rows="80%, 20%" MARGINWIDTH="0" MARGINHEIGHT="0"
FRAMESPACING="0" BORDER="0" FRAMEBORDER=NO>
          <FRAME Name="flash" SRC="flash.html" SCROLLING="NO"
MARGINWIDTH="0" MARGINHEIGHT="0" FRAMESPACING="0" BORDER="0"
FRAMEBORDER=NO>
          <FRAME Name="bottom" SRC="bottom.html" SCROLLING="NO"
MARGINWIDTH="0" MARGINHEIGHT="0" FRAMESPACING="0" BORDER="0"
FRAMEBORDER=NO>
        </frameset>

</BODY>
</HTML>
```

In this example, we've just specified that a viewers browser window contain two frames, arranged in horizontal rows, one at 80% of the overall height of the window, the other at 20%. The top frame will display the html page 'flash.html' (which presumably contains an embed tag for a flash movie) and the bottom frame displays a page called 'bottom.html' (which might be a blank placeholder or might contain other content or another flash movie).

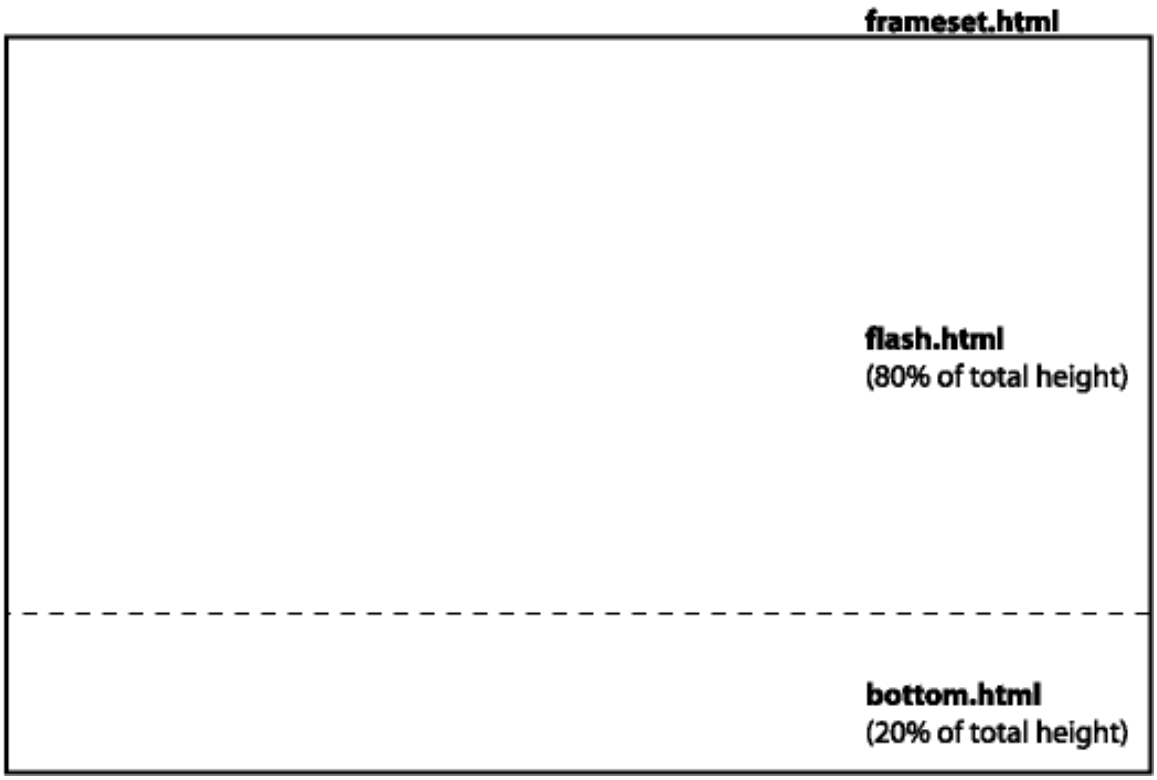
The code for flash.html would look like this –

```
<HTML>
<HEAD>

<TITLE>decon</TITLE>

</HEAD>
<BODY TEXT="#FFFFFF" LINK="#EE7C0B" VLINK="#7D0202" ALINK="#7D0202"
BGCOLOR="#000000" topmargin=0>
<center>

<OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"

    WIDTH="100%"
    HEIGHT="100%"
    CODEBASE="http://active.macromedia.com/flash5/cabs/swflash.cab#version=5,0,0,0"
    ID="decontrol">
    <PARAM NAME="MOVIE" VALUE="mainpage.swf">
    <PARAM NAME="PLAY" VALUE="true">
    <PARAM NAME="QUALITY" VALUE="best">
    <PARAM NAME="LOOP" VALUE="true">
    <PARAM NAME="SCALE" VALUE="showall">
    <PARAM NAME="MENU" VALUE="false">

<EMBED SRC="mainpage.swf"
    NAME="decontrol"
    WIDTH="100%"
    HEIGHT="100%"
```

```
    PLAY="true"
    QUALITY="best"
    LOOP="true"
    SCALE="showall"
    MENU="false"
    SWLIVECONNECT="false"
PLUGINSPACE="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=
ShockwaveFlash"
    TYPE="application/x-shockwave-flash">

</EMBED>
</OBJECT>
</center>

</BODY>
</HTML>
```

The code for bottom.html (assuming it's a blank frame) would be –

```
<HTML>
<HEAD>

<TITLE>decon</TITLE>

</HEAD>
<BODY TEXT="#FFFFFF" LINK="#EE7C0B" VLINK="#7D0202" ALINK="#7D0202"
BGCOLOR="#000000" topmargin=0>

</BODY>
</HTML>
```



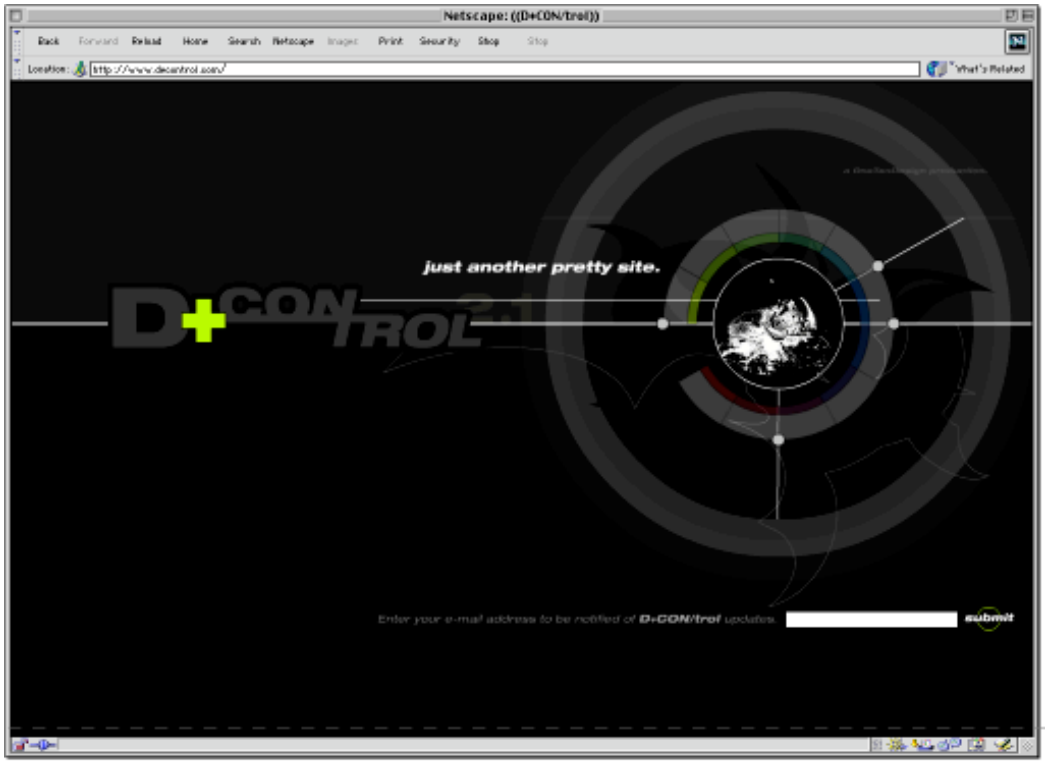
Full Screen Embedding

The most common, and often the most effective, way to use Flash movies is as completely self-contained web pages. The movie fills the user's screen and doesn't allow for other content such as html text or inline images. In order to achieve a predictable, well designed full screen flash experience, several factors must be taken into account. The first is the browser's inherent misunderstanding of a true full screen environment. If you simply embed your movie at 100% width and 100% height (which would seem to be the right choice given that 100% of the screen size is the goal you're after), the browser will usually put scroll bars on the window. Certain browsers will also leave some space at the top and left side of the page when embedding simply at 100%.

An easy frameset that consists of a 1 pixel frame across the bottom (across the top works equally well) will allow you to embed a flash movie at 100% of screen size (minus the nearly invisible 1 pixel frame across the bottom). This "invisible" frame can also be a useful place to hide code such as hit counters, javascript, etc. An example of a frameset built to contain a full screen flash movie is below :

```
<frameset  rows="*,1" MARGINWIDTH="0" MARGINHEIGHT="0"
FRAMESPACING="0" BORDER="0" FRAMEBORDER=NO>
 <FRAME Name="flash" SRC="main/flash.html" Scrolling="no"
MARGINWIDTH="0" MARGINHEIGHT="0" FRAMESPACING="0" BORDER="0"
FRAMEBORDER=NO>
 <FRAME Name="bottom" SRC="main/maincount.shtml" Scrolling="no"
MARGINWIDTH="0" MARGINHEIGHT="0" FRAMESPACING="0" BORDER="0"
FRAMEBORDER=NO>
</frameset>
```

Notice the ‘rows=”*,1”’ in the first frameset tag. This allows the top frame, the one that actually holds the flash movie, to expand to whatever height the user’s browser window is set to.



Window Proportions

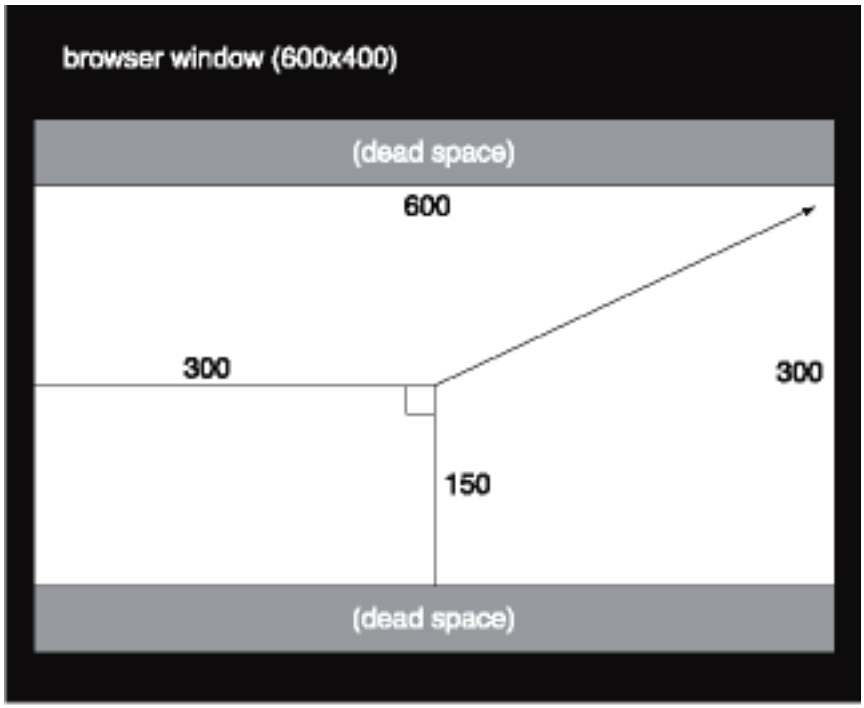
The next point in using full screen flash movies is an understanding of browser window proportions. Screen resolution is a very minor issue with Flash due to a Flash movie’s scalability (which is one of the greatest advantages of Flash). It doesn’t matter whether a user’s monitor is set to 640x480, 800x600, 1024x768 or any other possible resolution – the full screen movie will grow or shrink to fit the user’s monitor and will appear simply as full screen, regardless of resolution.

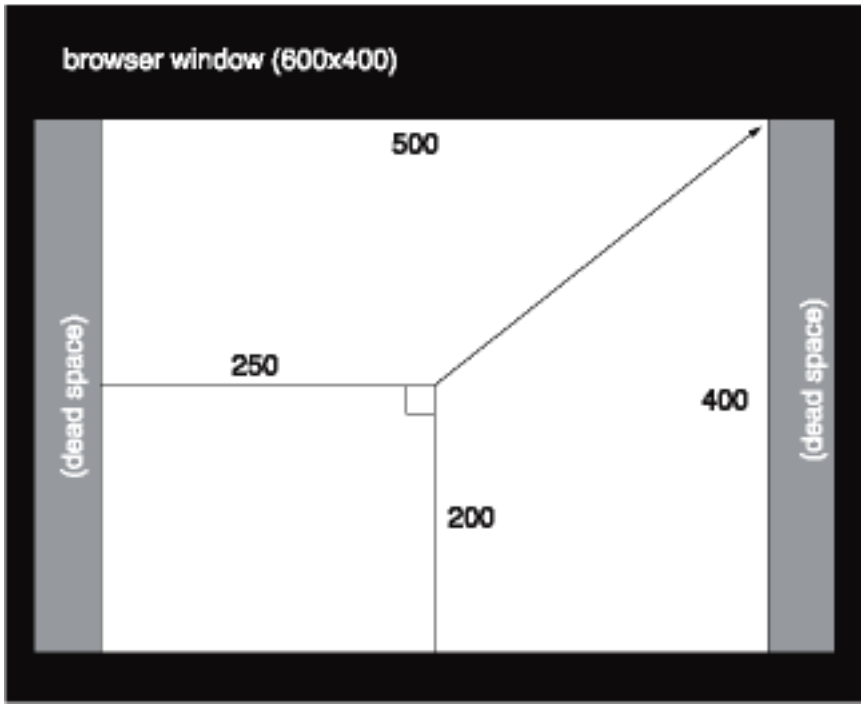
What IS relevant about screen resolution, though, is that all resolutions are exactly the same ASPECT RATIO. 800x600 is a 4:3 aspect ratio (meaning the height is three quarters the distance of the width), as is every other resolution. If Flash movies were really viewed at FULL SCREEN, the question of proportion would be far simpler. You would build your movies with a 4:3 aspect ration and let them scale to fit any monitor they might be viewed on. The problem, though, is that the actual content area of the browser isn’t quite as large as the viewable area of the monitor, and even worse, varies considerably from browser to browser and user to user. Depending on whether the user is on a PC or a Mac, using Netscape or IE, how they’ve got their toolbars configured, what button style they use on their browser, and whether they’ve got their window maximized, the viewable area for an 800x600 monitor can vary enormously. It is very possible, or even likely, that a user might not have their browser window maximized. Often, users will view the web with their browser window at full screen HEIGHT, but at a width that is much narrower. This not only changes the size of the viewable area, but changes the shape substantially (from a horizontal rectangle to a square or vertical rectangle). As a designer, you have very little idea at what proportion a user’s window will be set.

The question of what proportion to build your movies at is a question more of design than of technical considerations. Factors such as page composition, legibility, and stylistic considerations must all be taken into account when determining the correct stage proportion for a given movie. With a solid

understanding of the mechanics of scaling movies and how flash reacts to different situations, one can make educated decisions that also satisfy these design criteria.

When Flash displays a movie, it will scale the movie so that EVERYTHING inside the confines of the stage is visible. If a movie is embedded at 100% width and 100% height, flash will do one of three things depending on which option you select in the 'scale' parameter in the embed and object tags. If you select 'exactfit' for the 'scale' parameter, flash will stretch the movie so that the width and height fit exactly to the window dimensions (which will stretch and distort your movie – generally an undesirable effect). If you select 'noborder', flash will fill the entire screen with the flash movie, cropping where necessary so that no stretching is necessary. This is also generally less than ideal as it is difficult to plan exactly how a movie will be cropped, making it impossible to position important content and guarantee it will be seen. If you specify 'showall' (or if you leave the 'scale' parameter out of the object and embed tags), flash will maintain the movie's proportions and fit the movie at the largest size possible without cropping. If your movie is, for example, 300 pixels wide by 150 pixels high, and you wish to display it full screen in a window that happens to be 600 pixels wide by 400 pixels high, flash will scale your movie to 600x300 – which leaves 50 unaccounted for pixels at the top and bottom of the page. What this means when planning a flash movie is that you'll need to plan what users might see in this dead space beyond the edges of your movie.

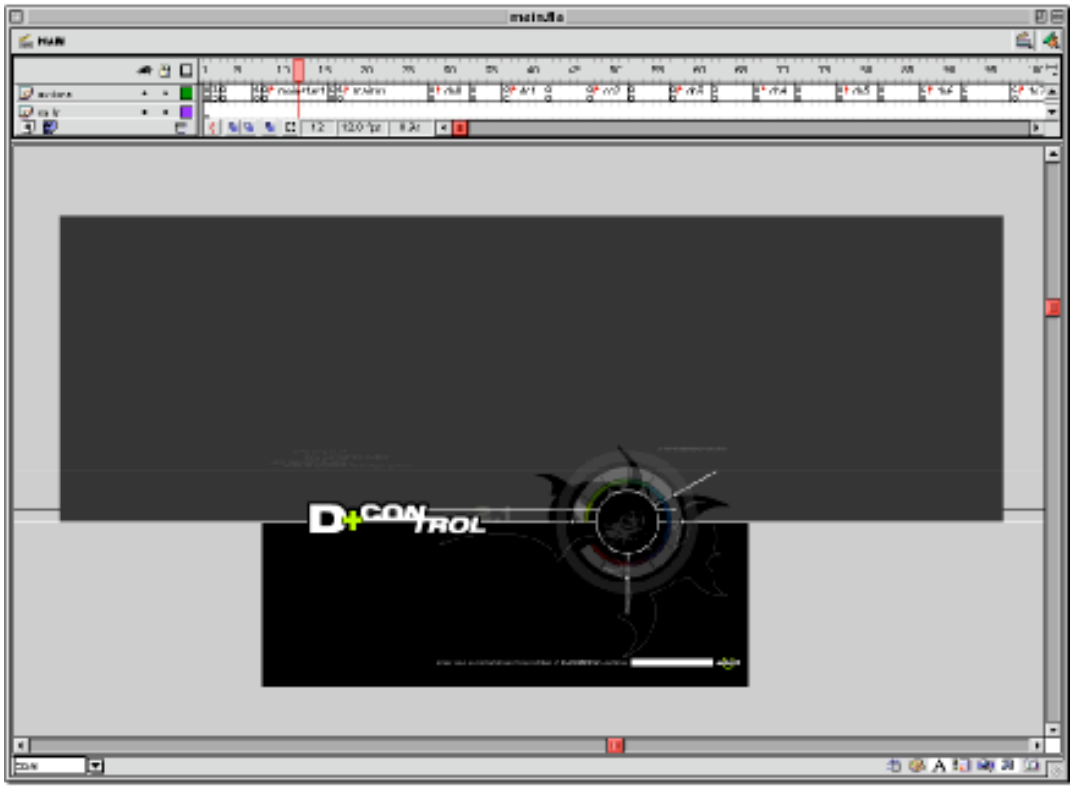


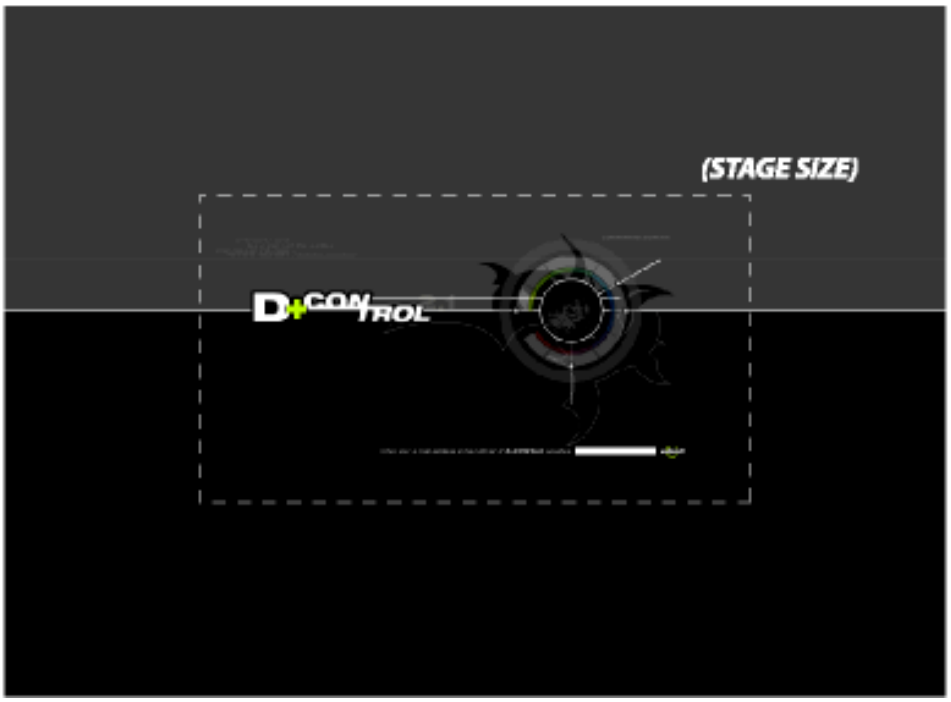


There are a few ways to deal with this 'excess'. Flash allows you to do a very useful thing when embedding a movie at 100% (unfortunately, this technique does NOT work with any other embed size than 100%). Artwork that's put into a Flash movie, but falls beyond the edges of the stage WILL show up if there is extra room on the page. This is by far one of the most useful aspects of embedding flash at 100% as it allows the designer to control with almost complete certainty what viewers with almost ANY browser window size will see.

Assume, for example, that you wish to create a movie with a stage size of 600x300. If that movie is to be displayed in a window with a 600x300 viewable area, then the user will see just what's on the stage and nothing else. If that movie is displayed in a window with a 600x400 viewable area, the viewer will see an extra 50 pixels above and below your stage. Flash allows you to place art ABOVE and BELOW the actual stage so that those 50 pixels are able to be designed instead of acting as 'dead-space'.

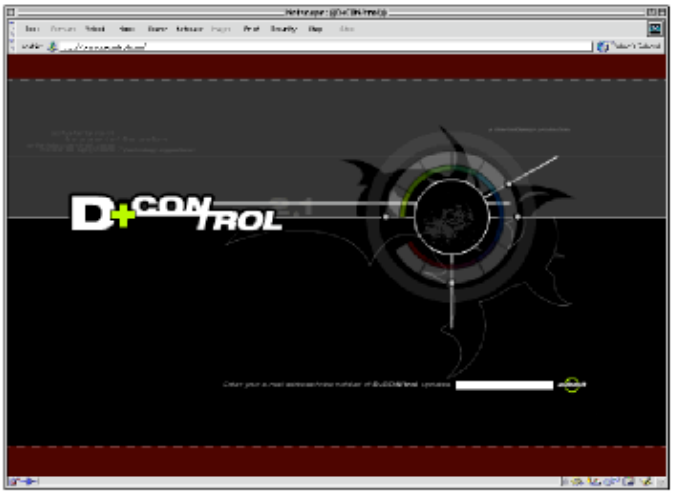
OVERFLOW ART





Another way to take advantage of this ‘bleeding’ effect is to create a mask around the area of the movie you’d like people to see. You would put a black (or white, or any color) frame around the edges of the movie so that all a user would see if there was excess space would be a black frame. You’re still putting art beyond the edges of the stage, but the visual effect is different.

MASKING

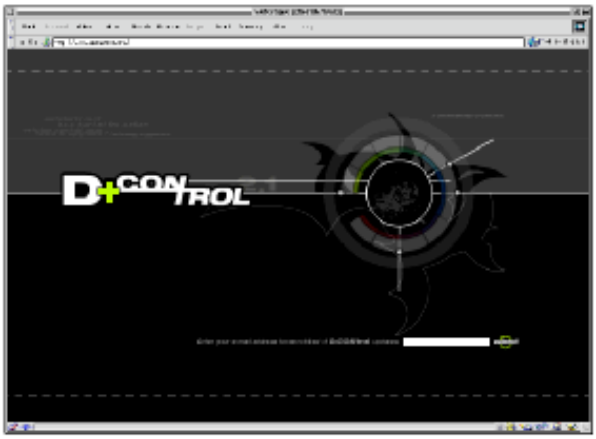


It's possible to fill up the un-accounted for space with 'old fashioned' background images. A background image, though, can be very difficult to align correctly with a Flash movie, especially across browsers and platforms, and offers no real benefit, at least in a full screen situation, over placing the additional artwork in the movie itself. Therefore, as a general rule, background images are not used with full screen flash movies.

There is some real design involved in planning the stage size of a full screen movie based on how you'd like the movie to appear to a variety of users. By experimenting with different stage proportions with different designs, you can control with some precision what users will see in their browsers, regardless of their individual window size. The best way to really understand how the process works is to make lots of movies at lots of different proportions and see how they appear on lots of different window sizes. The one crucial rule, though, is that ANYTHING inside the borders of the stage WILL appear for ALL users (provided you haven't set the 'scale' parameter in the embed tag to 'noborders'), regardless of their individual window proportion. What falls beyond those edges may or may not be seen.

Set up a frameset to hold full-screen flash movies (using the sample code from above). Create a movie in flash that allows you to easily recognize where the edges of the stage are. Try changing the stage height and width and view the movie in a browser (in your frameset). Notice what changing the proportions of the stage does to the amount of art seen beyond the edges of the stage. Also, try modifying the size and shape of your browser window and notice how that affects the way your movie appears on the page.

STANDARD WINDOW



WIDE WINDOW

TALL WINDOW

Percentage/Fixed Size

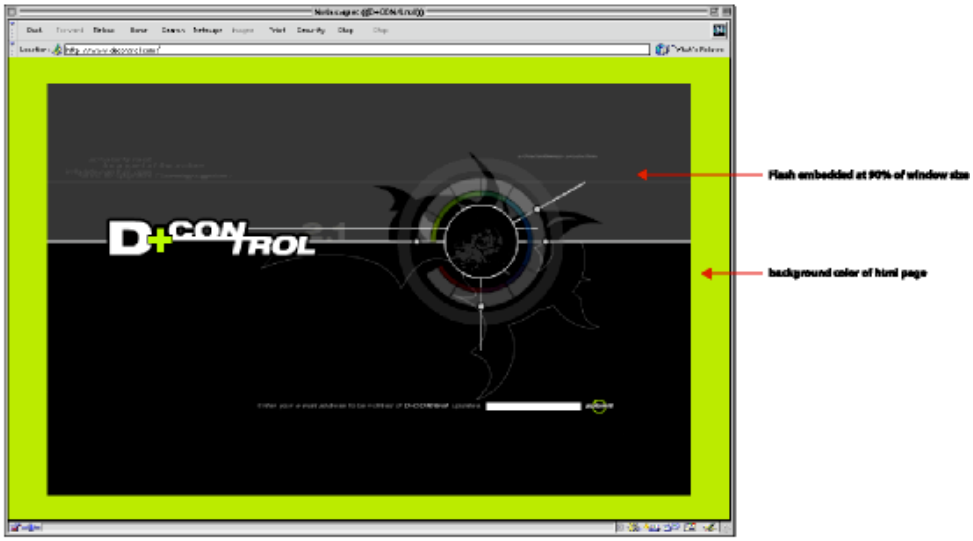
There are times when, for one reason or another, you need to create a page or an entire site that is all in flash, but you can not make the movie full screen. Perhaps your movie contains very complex animation that won't run well if it's displayed at a large size. Or perhaps, for aesthetic reasons, you wish to insist that all users see your movie at a fixed pixel size or at a percentage of their overall window size. In these cases, you'll use either a percentage embed or a fixed-size embed.

For a percentage embed, you specify what PERCENTAGE of the total window size a movie should occupy. Where a movie at 100% will appear to 'bleed' beyond the edges of the window, a movie at 90% will always have a border equal to 10% of the total window size. It's worth noting that the way proportion works (a movie will scale to fit the smaller dimension to prevent cropping) is the same with percentage embedding.

A fixed size embed is the simplest way to embed a movie. If you specify that the movie sits on the page at 600 wide by 400 high, that's exactly the size it will appear on the page, regardless of window size. The movie doesn't scale and therefore works like an inline image (gif, jpeg) in this respect. If the size you specify is larger than the user's window, the window will scroll – just like a simple image.

A problem with percentage and fixed size embedding is that Flash no longer allows you to view art placed beyond the edges of the stage. You must, therefore, work a little harder to account for the area of the page NOT occupied by the movie. Background images or simple background colors are your only options here. Depending on how you choose to place the movie (using html), you may have a difficult or even impossible time aligning a background image to a movie.

PERCENTAGE EMBEDDING



Sample of html for a movie embedded at a percentage (with green background color) –

```
<HTML>
<HEAD>

<TITLE>((D+CON/trol))</TITLE>

</HEAD>
<BODY BGCOLOR="#CCFF00" MARGINHEIGHT="0" MARGINWIDTH="0"
TOPMARGIN="0" LEFTMARGIN="0">

<center>

<OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"

    WIDTH="90%"
    HEIGHT="90%"
    CODEBASE="http://active.macromedia.com/flash5/cabs/swflash.cab#version=5,0,0,0"
    ID="decontrol">
    <PARAM NAME="MOVIE" VALUE="mainpage.swf">
    <PARAM NAME="PLAY" VALUE="true">
    <PARAM NAME="QUALITY" VALUE="best">
    <PARAM NAME="LOOP" VALUE="true">
    <PARAM NAME="SCALE" VALUE="showall">
    <PARAM NAME="MENU" VALUE="false">

<EMBED SRC="mainpage.swf"
    NAME="decontrol"
    WIDTH="90%"
    HEIGHT="90%"
    PLAY="true"
    QUALITY="best"
    LOOP="true"
    SCALE="showall"
    MENU="false"
    SWLIVECONNECT="false"
PLUGINSPACE="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=
ShockwaveFlash"
    TYPE="application/x-shockwave-flash">

</EMBED>
</OBJECT>

</center>
</BODY>
</html>
```

Embedding in Frames

It is possible to utilize the advantages of embedding at 100% (namely the ability to view art beyond the edges of the flash stage) while embedding a movie at less than full screen using frames. A frameset can be designed that will contain a movie at a percentage of total window size or at a specific pixel size while still embedding the movie at 100%, thereby retaining the ability to view art beyond the edges of the stage. If you know, for example, that you wish the flash movie to occupy 80% of the total height of the browser window, you might build the following frameset:

```
<frameset  rows="40%,60%" MARGINWIDTH="0" MARGINHEIGHT="0"
FRAMESPACING="0" BORDER="0" FRAMEBORDER=NO>
  <FRAME Name="flash" SRC="flash.html" Scrolling="no"
MARGINWIDTH="0" MARGINHEIGHT="0" FRAMESPACING="0" BORDER="0"
FRAMEBORDER=NO>
  <FRAME Name="bottom" SRC="content.html" Scrolling="auto"
MARGINWIDTH="0" MARGINHEIGHT="0" FRAMESPACING="0" BORDER="0"
FRAMEBORDER=NO>
</frameset>
```

In flash.html, you would then embed the flash movie at 100% (which now relates to 100% of the size of the frame rather than 100% of the total window size). Using frames, you can almost always maintain 100% embedding of your movies while forcing those movies to scale to percentages or fixed pixel sizes. For this reason, frames are often a better choice than percentage embedding for the flexibility in design that they allow.

Popup Windows

Popup windows are a common feature on many websites. By using javascript, the designer can spawn a new window to display content. This new window is sized according to the designer's wishes (width and height are stated in the javascript). Other aspects of the new window, such as whether it contains a menu bar, location bar, status bar, or even the window's location on the user's screen, are also stated in the javascript code. Given the level of control a designer has over the viewing environment when using popups, it can be quite an attractive option in many instances.

Sample javascript popup code –

```
<HTML>
<HEAD>

<TITLE>decon</TITLE>

<script language="javaScript">

<!-- Hide from older browsers

function OpenPopup()

        {
        window.open(popupwindow.html','window','toolbar=0,location=0,directories=0,status=0,menubar=0,scrollbars=0,resizable=0,width=400,height=300');
        }

//-->

</script>

</HEAD>
<BODY TEXT="#FFFFFF" LINK="#EE7C0B" VLINK="#7D0202" ALINK="#7D0202" BGCOLOR="#000000" topmargin=0>
<center>

<!-- page content goes here //-->

</center>
</BODY>
</HTML>
```

That would be the code for the original page: the one that the popup will open FROM. Depending on whether you're opening the popup window from a flash movie or from html, the code to make the link would be either –

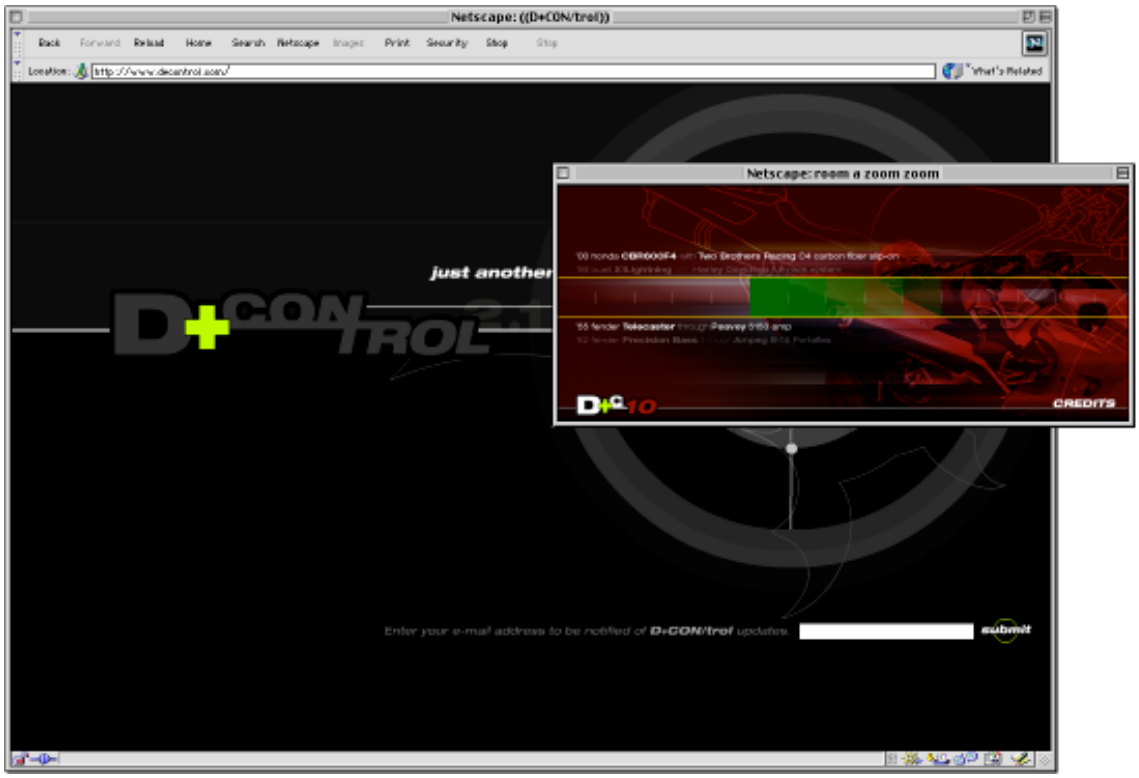
getURL ("javascript:OpenPopup()");

or

<A HREF="javascript:OpenPopup()">

respectively.

Using flash in popup windows is a far simpler proposition, at least from an embedding standpoint, than using flash in a regular browser window. In a popup window, you know the exact size of the window (since you presumably opened the window yourself using javascript) so you can very easily plan the size of your flash movie accordingly. You generally will still put a 1 pixel frame in the popup window to allow for proper scaling of the flash movie, but the issues of differing proportions are gone.



One Movie vs. Several Movies

The next decision in an all flash site is whether to use just one large movie for the entire site or to use several movies; one for each section, for example. There are advantages and disadvantage for each option. All considerations should be explored before design begins as it is sometimes difficult to convert a site from 'one movie' to 'several movies' once design has begun.

One Movie

Using one Flash movie for an entire site can be cumbersome in some ways. The file tends to get pretty large and, unless you're very well stocked with RAM, can be slow to work on. It is also extremely difficult to work together with others if there is only one master file to work from. If, for example, you're working on the main page of a site and another designer is working on a sectional page, you'll need to work in separate files and then combine your work into the single 'master' file by copying frames or symbols. It's a far less than ideal method for working together and allows for quite a few problems.

There are a few very good reasons, however, for working past these problems and using a single movie for an entire site. One of the primary reasons is downloadability. If you've got the entire site contained in a single movie, you can control how the whole site loads into the user's computer. This can be especially valuable if you expect your potential users to have slower connections. You can, for example, display an interesting loading screen while the main page loads, then display the main page while the first sectional page loads; then, while the user is on the first sectional page, you can be loading the next section, and so on. Flash is able to stream content fairly well and, with the additional features of Flash 5, the designer has a lot of control over exactly how a movie loads and what displays while content loads. Proper control over loading can add infinitely to a user's experience on a site. Again, the importance of this issue is greatly diminished if you expect potential viewers to be using fast (DSL, cable, LAN) connections.

Another notable advantage to using a single movie for an entire site is the ability to create transitions between sections and subsections. Since there is no real distinction between sections, you're able to animate out of one section and into another quite easily. Taken a step further, putting all of the content of a site into one movie allows you to potentially dispense with traditional site structure (main page, sections, sub-sections, etc) and create a more fluid, non-linear structure. Pieces of content that open in

areas of the page, floating windows, persistent content tools, etc. are all possible if a site is contained entirely in one movie.

Several Movies

Breaking a site into individual, smaller movies based on sections provides a few distinct advantages. Multiple designers working on a single site becomes far simpler (ie one designer builds the main page in its own file, while another builds a sectional page in its own file; these files don't ever need to be combined). Files are generally smaller, easing the burden on overworked processors. It can be far quicker and easier to work on a smaller file containing only a piece of a site rather than a huge file that contains the entire site. With the new Shared Libraries feature in Flash 5, workflow when dealing with multiple movies and several designers is even further enhanced. The advantages mentioned earlier pertaining to loading and transitions are not present when using multiple movies, however.

If you decide that using multiple movies is appropriate, another decision must then be made. Do you put each movie on its own html page, or embed the first movie and call the new ones into the same page using the 'Load Movie' command? If all of the movies on the site are to be embedded in the same way (ie fullscreen), then it's generally better to call new movies into the same html page. This eliminates some work in creating many identical html pages as well as eliminating the brief 'hiccup' when jumping to a new page (when linking movie, the background color of the page always stays on screen). If you need to embed different movies differently, or if different movies are built at different sizes, then you're stuck using several html pages.

Stacked Movies

A hybrid of the two methods is using multiple movies that are "stacked" or loaded on top of one another. Usually, a "controller" movie is embedded onto the page. This controller then loads the individual movies for the site onto layers above itself. These movies can be downloaded and then "parked" or stopped until they're needed. In this way, you could load movies either as they are needed or all at once, or some combination of the two. If you're on the main page, for example, you might begin loading all of the sectional pages (which are each in their own movies) onto other levels so that they are ready when the user clicks to go to a section.

With stacked movies, it's also possible to create layers that are present throughout a presentation. Movies loaded into layers are transparent anywhere there is no art (where you can see the stage background). Because of this, it's possible to view two or more movies at the same time through layering. Movies are stacked sequentially based on their layer number (higher numbers are at the top of the pile, lower numbers at the bottom).

Since all sections or areas of a site are present at the same time, even though they are in different movies at different levels, the designer is able to provide planned transitions between sections or movies. Provided the site is properly planned and designed, the user experience can be every bit as seamless as when using a single movie.

Stacking movies provides all of the advantages of a single movie without the drawbacks of the need for one large file. It also allows for the practical advantages of using multiple movies without the design issues associated with that method.

Flash Detection and Flat Sites

It is generally a good idea to offer some sort of alternative to flash content. Although a majority of viewers do browse the web with the flash plug-in installed, there are still those that don't. For those viewers, you can implement a detection scheme using javascript that will determine whether or not they have the plug-in (or a specific version of the plug-in if you desire) and send them to an alternate page without flash content if they do not have the plug-in.

Sample of javascript detection code –

```
<HTML>
<HEAD>

<TITLE>D+CON/trol</TITLE>


<SCRIPT LANGUAGE="JavaScript"><!-- used for Netscape

var agt=navigator.userAgent.toLowerCase();
// *** BROWSER VERSION ***

var is_major = parseInt(navigator.appVersion);
var is_minor = parseFloat(navigator.appVersion);
```

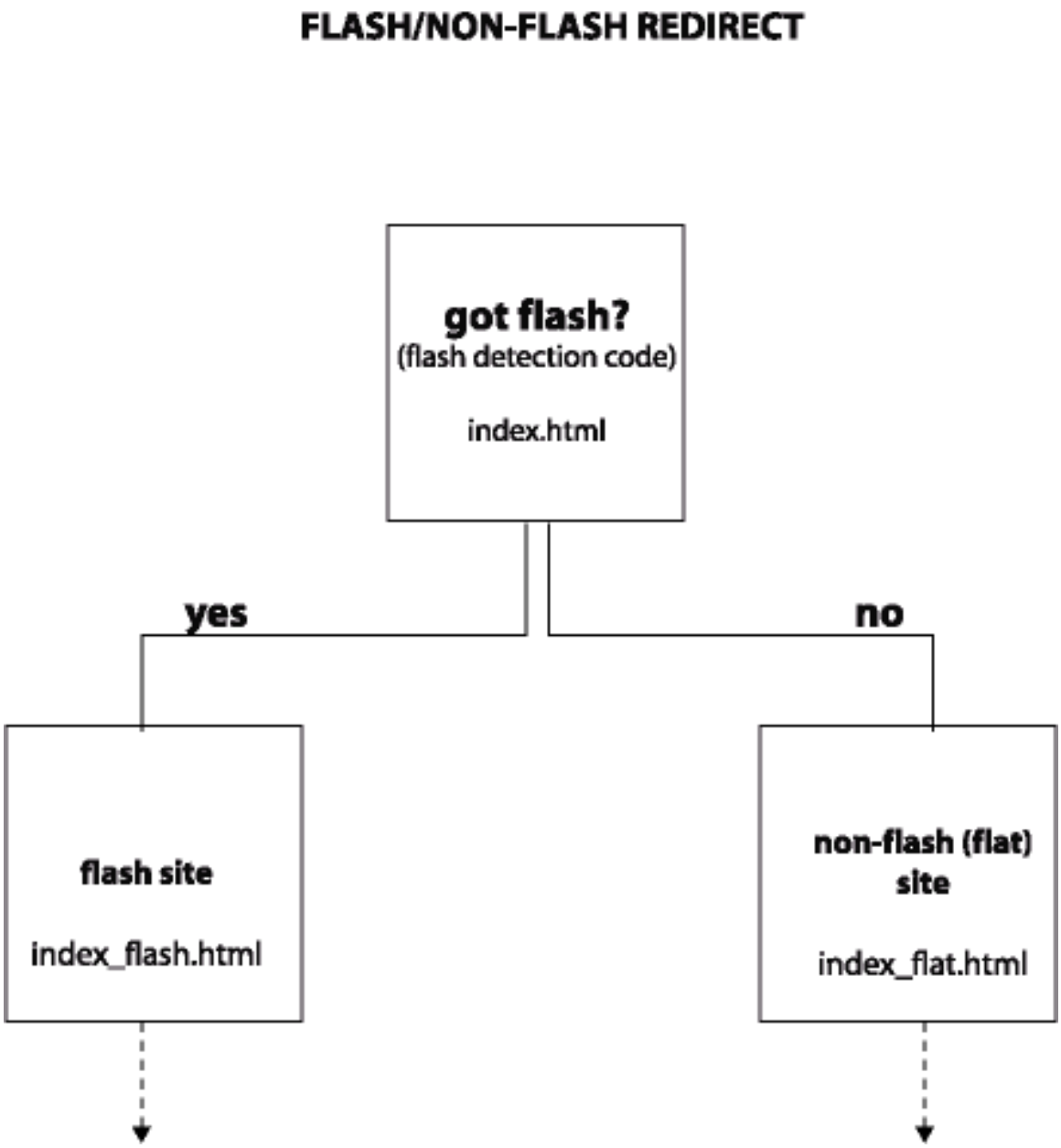
```
var is_ie = (agt.indexOf("msie") != -1);
var is_ie3 = (is_ie && (is_major < 4));
var is_ie4 = (is_ie && (is_major == 4) && (agt.indexOf("msie 5.0")==-1) );
var is_ie4up = (is_ie && (is_major >= 4));
var is_ie5 = (is_ie && (is_major == 4) && (agt.indexOf("msie 5.0")!=-1) );
var is_ie5up = (is_ie && !is_ie3 && !is_ie4);

var is_aol = (agt.indexOf("aol") != -1);
var is_aol3 = (is_aol && is_ie3);
var is_aol4 = (is_aol && is_ie4);
var is_opera = (agt.indexOf("opera") != -1);
var is_webtv = (agt.indexOf("webtv") != -1);

   //separate routine to sort out Mac IE users


var platform = navigator.platform;

if (platform == "MacPPC") {

if (is_ie4){

window.location = ("flat_index.html") }
}
  // assign base variables

          flash = "no"

          flash5 = "no"

          flashver = "1.0"

          action = "ns_exec"

var platform = navigator.platform;

if (platform == "MacPPC") {

if (is_ie5up){

           flashver = navigator.plugins["Shockwave Flash"].description.substring(16,19);

          if (flashver == "4.0") {flash = "yes";}


          if (flashver == "5.0") {flash5 = "yes";}
 } }


   //detect browser


       bName = navigator.appName;

       bVer = parseInt(navigator.appVersion);
```

```
   // detect plugin for NS flash 4 or 5 version

         if (bName=="Netscape") {

            // if (navigator.plugins["Shockwave Flash 2.0"]) {flashver = "2.0"}

           //  else if(navigator.plugins["Shockwave Flash"]) {flashver = "3.0"}

          flashver = navigator.plugins["Shockwave Flash"].description.substring(16,19);


         if (flashver == "4.0") {flash = "yes";}


         if (flashver == "5.0") {flash5 = "yes";}


   // If IE is detected, run the VBScript to detect flash 4

         } else if (bName =="Microsoft Internet Explorer") {
            action = "ie_exec"
         }

   //-->

</SCRIPT>

<SCRIPT LANGUAGE="VBScript"><!-- used for Internet Explorer

         on error resume next

         If action = "ie_exec" then
            FlashInstalled = (IsObject(CreateObject("ShockwaveFlash.ShockwaveFlash.4")))
         End If

         If action = "ie_exec" then
            FlashInstalled5 = (IsObject(CreateObject("ShockwaveFlash.ShockwaveFlash.5")))
         End If

         If FlashInstalled = "True" then
            flash = "yes"
         End If

        If FlashInstalled5 = "True" then
            flash5 = "yes"
         End If

      //-->

</SCRIPT>
<!-- End Scripts for Flash 4.0 detection -->

</HEAD>
```

```
<BODY BGCOLOR="#000000">

<SCRIPT LANGUAGE="JavaScript">

        if (flash == "yes")
                {
                location.replace('index_flash.html');
                }



        else if (flash5 == "yes")
                {
                location.replace('index_flash.html');
                }

        else if (flash == "no")

                {
                location.replace(index_flat.html');
                }

</script>

</BODY>
</HTML>
```

While this code does look rather complicated (indeed, it IS rather complicated), there is very little of it that needs to be thoroughly understood to make it work. Most of the time, though, all that needs to be changed are the page names after 'location.replace' in the last section of code. These correspond to the html pages that contain content for visitors with flash and visitors without flash. As with embed tags, framesets, and a lot of other html, javascript functions are often written once, then copied over and over again to new documents as needed.

It is sometimes desirable to offer viewers a choice when they arrive at a site as to whether they'd like to view the flash site or the flat site. Whether you implement an auto-detection system or give viewers a choice is a matter of personal taste, but can be influenced by such factors as expected viewership (how 'savvy' will the average visitor to the site be?) and the function of the site (do you need to get visitors into some type of content or buying situation as quickly and seamlessly as possible?). In some cases, especially when content must be displayed using flash, a flat page might consist of only a message alerting the viewer that they don't have the necessary plug-ins and that they should probably go and download them and then return. In this manner, all viewers are accounted for and nobody receives a plug-in error message or an ambiguously blank page.



Flash and html Together

Another interesting way to utilize Flash is as an element of a page or site rather than as the complete self-contained site. Flash movies can be used on pages in exactly the same way that static images can be used, but provide far more interactivity and opportunity for design than simple images. In many cases, flash files are also smaller (in file size) than their gif or jpeg counterparts. Flash can then provide even more benefit over static images with little or no drawbacks.

Flash as Inline Image

In many situations, a layout calls for a simple, self-contained image. In most, if not all, of these situations, though, additional animation and interactivity would be desirable. In these cases, there is usually no reason not to use a Flash movie!

When using a Flash movie in a page where you would otherwise use a static image, it's generally a good idea to provide automatic detection for the Flash plug-in and switch out the movie for an image without the user's knowledge if they don't have the plug-in. This method allows for a seamless experience for the user – they don't know that anything has been detected or switched, they simply receive the content that is appropriate for them. A disadvantage to this, however, is that the user then may not know that a richer experience would be possible if they had the plug-in. For this reason, images are often labeled with a message to the effect of "you're seeing a static image where you could be seeing a flash movie. Please download the plug-in" with a link to the appropriate download site.

Flash movies used in this manner on a page are generally not scalable. It is difficult and unreliable to embed a movie at a percentage in a table (as opposed to a percentage of the whole page). Therefore, movies are usually embedded at a fixed size.

Sample of code for putting flash into a page with html content –

```
<HTML>
<HEAD>

<TITLE>-((D+con/trol))-</TITLE>


</HEAD>
<BODY BGCOLOR="#000000" TEXT="#FFFFFF" LINK="#99FF00" VLINK="#669933"
ALINK="#99FF33">

<center>
<table width=85%>
<tr><td>
<font face=helvetica size=-1>
<BR><BR><BR>
<font color="#000000">.....</font>        <B>--D+CON/trol--</B> is an
experimental exercise in the loss of viewer control
over the developing web medium. Web users, weary to lose valuable time and
waste precious energy, have become information scavengers clicking
endlessly in search of vital information yet invariably failing to perceive
their on-line environment in the process. <B>--D+CON/trol--</B>
explores this by
manipulating viewer response to content that is normally provided through
linked visual guideposts to information.  Disoriented and removed from the
rat race of normal web browsing, the viewer of <B>--D+CON/trol--</B> is
unknowingly assimilated into a landscape of seemingly unmarked data and
forced to examine each page encountered.  Ultimately, this results in the
gradually acceptance of loss of control giving the viewer the opportunity
to take the time to digest each area and the space necessary to begin
viewing the visual atmosphere as an end in itsel.

<BR><BR><BR>

<OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"

    WIDTH="300"
    HEIGHT="150"
    CODEBASE="http://active.macromedia.com/flash5/cabs/swflash.cab#version=5,0,0,0"
    ID="decontrol">
    <PARAM NAME="MOVIE" VALUE="logo.swf">
    <PARAM NAME="PLAY" VALUE="true">
    <PARAM NAME="QUALITY" VALUE="best">
    <PARAM NAME="LOOP" VALUE="true">
```

<PARAM NAME="SCALE" VALUE="showall">
<PARAM NAME="MENU" VALUE="false">

<EMBED SRC="logo.swf"
NAME="decontrol"
WIDTH="300"
HEIGHT="150"
PLAY="true"
QUALITY="best"
LOOP="true"
SCALE="showall"
MENU="false"
SWLIVECONNECT="false"
PLUGINSPACE="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash"
TYPE="application/x-shockwave-flash">

</EMBED>
</OBJECT>

..... As an evolving and emerging medium, the web currently presents us with two obstacles preventing visual solutions from being reached successfully and consistently. (A) Masterless, the goals of the form are split between their independent parts; namely, entertainment, information, advertisement and presentation of art. Unlike film which attempts to entertain using a high art platform or television which attempts to advertise using an entertainment platform, the universal purpose of the web and its contents are yet undecided. As a result, the web is noisy; viewers are obliterated by a turmoil of visual, textual and audible information splitting their attention. This leads to an uncontrollable urge on the part of the viewer to seek out pertinent content quickly while disregarding peripheral distractions. (B) As an unrefined medium, the web is struggling for congruity in its content. Presently, the only consistency in on-line content is the lack off any consistency whatsoever; following a given link, the viewer is just as likely to find meaninglessness as they are to find revelation. A and B combine to create a situation in which the lions share of visual content is intentionally ignored by the viewer-taking-matters-in to-their-own-hands in search of browsing "success".

..... <B>--D+CON/trol--</B> rectifies these defects by disregarding traditional methods of "linking" information using visual and textual markers, thus, transforming viewer perception of interactivity and control. Unity of intention, namely the presentation of an art form, allows <B>--D+CON/trol--</B> to resolve the dilemma of incongruous raison d'Ítre. Purposeful content resolves the secondary dilemma of seemingly unsuccessful searching. By taking time to explore, the viewer of <B>--D+CON/trol--</B> can consistently expect to find interesting, entertaining and meaningful content. <B>--D+CON/trol--</B> is an active environment in which users can click and examine, click and digest, and ultimately click and analyze.

..... The nature of contemporary web based communications is such that users of

the medium have taken interactivity to an undesirable level. Rather than accessing information in an exploratory manner using this highly flexible platform, browsers are controlling their on-line experience through the self censorship of clicking forwards and backwards in pursuit of a given goal. By alleviating the most important reasons driving viewers to this behavior, <B>--D+CON/trol--</B> creates an environment in which a reevaluation of
the web as a medium and ones own behavior when faced with this medium is possible. In conclusion, <B>--D+CON/trol--</B> is the loss of viewer command over
the web medium for the sole purpose of heightening consciousness of the flaws and potentials of this developing art form.
**

**
**Go ahead, tell us. We'd love to hear from you.
**
feedback@decontrol.com
**

**
<B>--D+CON/trol--</B> was created by One Ten Design.
**

**

****

</td></tr>
</table>

</BODY>
</HTML>

<examples of inline images> **(inline.ai)**

Flash Headers

Page headers and menus can work very well as Flash movies. In the simplest example, what would normally be an image-mapped gif on the page would become a Flash movie containing similar functionality (ie linking to other pages on the site), but with more dynamic presentation of that functionality. Sound, more robust animation, elaborate button rollovers, and smaller file sizes are all good reasons to use flash for page headers and menus rather than simple gifs and jpegs. If executed correctly, there are few or no drawbacks to using flash in this manner.

A page that is to utilize flash menus or headers can be designed as it would be without flash. Page composition, hierarchy, and structure are all the primary concerns in laying out a page whether or not elements are to be done in flash or static images. When using flash, however, the designer can go much further when adding polish to the design by incorporating appropriate sounds, extravagant rollovers, etc. into the headers and menus. These headers and menus are embedded in exactly the same way (fixed sizes, aligned using tables, etc.) as they would if they were not flash.

Frames

Often, a better way to use a Flash movie as a page header utilizes frames. The page is divided into two frames – a top frame that will contain the Flash movie, and the bottom frame that will contain the content. If the site is structured appropriately, the top frame might not ever need to be reloaded. The user arrives at the site, greeted by a Flash header and some content. When they select a menu item from the top frame, only the content in the bottom frame needs to be replaced. The flash movie can reflect the current content (ie by 'greying out' or highlighting certain menu items). In this manner, users with slow connections can view a very dynamic site while only needing to load a large Flash file one time.

It's sometimes difficult, though, to create a site that works with this simplistic structure. Many sites, especially those that feature primarily html-based content, need to have a main page with a different look and feel (and consequently page architecture) than the sectional pages. In this situation, a problem arises. Do you build separate sectional pages and use individual movies as headers on each page? Then you're forcing the user to load a brand new movie every time they switch pages. Or do you use just one header movie for all of the sectional pages?

One interesting solution might utilize a scalable Flash main page with a single Flash movie header on all of the sub-pages. This format would allow for an exciting full-screen experience for the main page, with easily updateable html based content pages. Additional animation content could be added to the persistent header using several layers of flash movies (a layer for the menu, a layer for animation, and a layer to control the other layers). To make all this work, a single, unchanging header would need to be in place throughout the entire site. The problem lies in linking from the flash main page to the mixed html and flash sub-pages. These sub-pages would all contain the same flash movie so that flash movie would contain menu information for all of the site sections. When going from the main page to a sub-page, it would be necessary to 'speak' to the flash header to 'tell' it what section's menu information it should display.

There are many possible solutions to accomplish the goal of communicating between flash movies on different pages. Each option has many subtleties that will affect other aspects of the site's functionality. All of the methods to be explored have the same goal - to talk from a flash movie to a page, then from one page to another, then from the second page to a Flash movie.

One technique involves making sure you've got a persistent frame – a frame that will be there on the main page and will also be there in all of the sections. Using javascript, you'll tell this frame where you're planning on going when you leave the main page. Once the sectional header loads, it will 'ask' the persistent frame which section it should be displaying (by gathering information contained in the javascript in that frame) and display the appropriate sectional information. It's up to you how to tell the content frame which content is to be displayed. The Flash header can go to the about section, then tell the content frame to load the About page. Or you can have separate frameset pages for each possible sectional page – an About frameset, etc. The About page loads independently of the header. It's also possible to load the About page, then have IT talk to the header and tell it what section to display. Another method might involve sending information to a cgi script that will then direct both the content frame AND the Flash header and make sure everybody's on the same page. Any of these methods might be a viable option and factors such as expected site traffic (and therefore server load), page content (simple pages or complex ones), and programming ability would affect the method to be chosen.

main.html

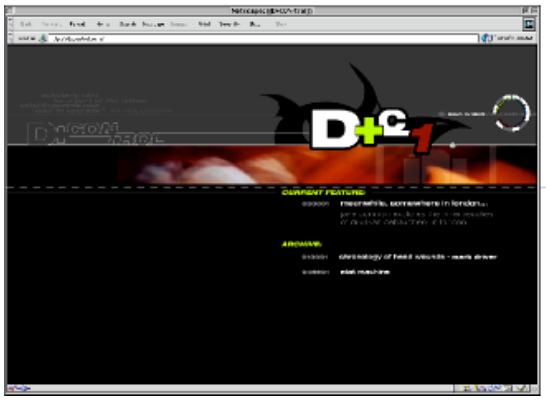
blank.html

flash.html

DC7.html

flash.html

DC9.html



flash.html

DC1.html

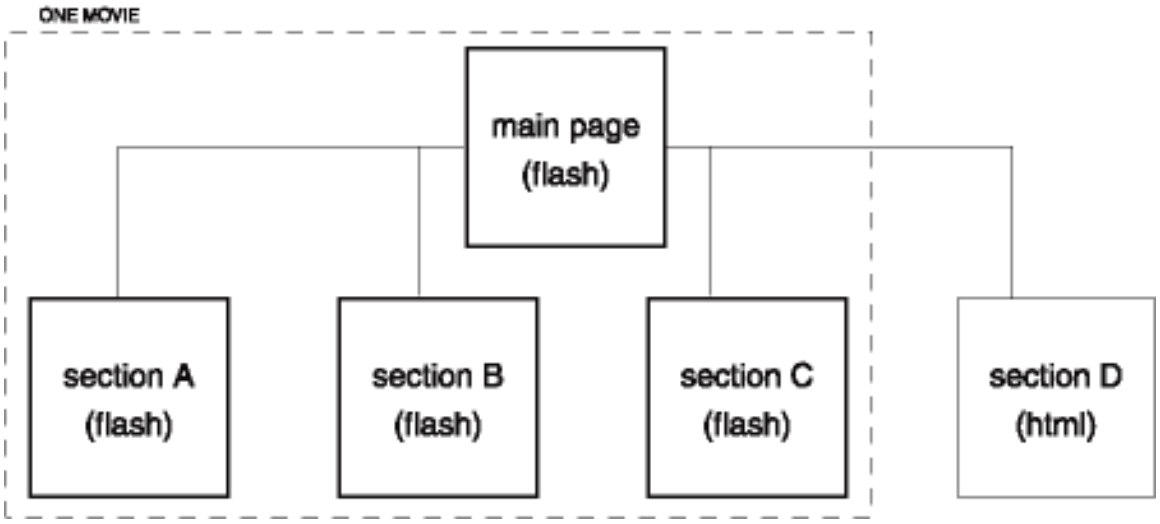
Flash Pages with html Pages

Another way to mix Flash with html would involve certain pages of a site that are entirely in flash, while other pages are entirely in html (or html with inline flash movies).

A situation where you might have a site that mixed all flash and all html pages might occur if you want to do a site entirely in flash (for reasons such as scalability of flash movies or animation capabilities), but need to have some pages entirely in html (such as a cgi-based message board). If you intended to do the flash portion of the site as one large movie or several movies stacked, you would need to link to a new page that contained the message board. The problem with this solution lies when the user attempts to return to the flash part of the site from the message board. You'll be forcing the user to reload the entire flash movie any time they come back to any section from the message board. This would especially be an issue for users with slower connections.

There are a few possible solutions. One, you could open the message board in a popup window – this would work quite well and would separate the message board from the flash movie. Two, you could link from the flash movie to a regular html page with a message board on it – this would work well, but would cause the user to reload the entire site upon returning from the message board. Three, we could break the site up into smaller movies so that every page was its own movie – this would mean that you would load a new movie any time you switched pages, alleviating the strain of loading the entire site at once, but disrupting the experience by forcing lengthy loads any time you wanted to switch pages. None of the possible solutions is ideal, but various factors such as desired user experience (would popup windows be jarring), expected viewership (fast or slow connections), and site design (are there transitions between sections that would be lost when breaking the site up into many movies) will determine which compromise is best.

Flash Linked to HTML pages



Conclusion

Establishing the architecture of a web site is one of the most important, and often overlooked, decisions a designer needs to make in designing a website. Common practice is to determine that the site will be a 'flash site' and simply begin designing – this leaves out a very important step of the design and conceptualization process. Decisions such as whether to use flash alone or with html, what size to embed flash movies at, and how many individual movies should be used are crucial to a site architecture that will provide a compelling user experience. With some knowledge of the possibilities available when using flash, one can make an educated decision that will enhance and hopefully expand upon the overall visual impact of the website.