A Survival Guide: Shockwave for Director or OK What Do I Do Now?

Rob Dillon, President, Digital Design Group Limited

As I write this, January 1996 everything that I'll be discussing is still in Beta test version. But most of the basics should hold true and the software should progress to greater states of finesse and usability.

Some basics for this Survival Guide: I'm expecting that you are a current user of Macromedia Director and have some understanding of Lingo, you've worked at least a little with HTML and that you've read the Shockwave Developer's Guide.

So first we'll go over some broad concepts and then we'll talk about what makes Shockwave so different from *regular* Director and ways to ease the pain of transition.

All we have to work with so far is the Shockwave plug-in for Netscape's Navigator Web Browser. There's a beta plug-in for Windows 3.1 and for '95 / NT. There's a developer build of the Mac plug-in circulating and the beta release version is promised for the first of February. There are more Shockwave plug-ins expected for other browsers, Navigator is the first. If you don't have the Shockwave plug-in, you can download it from Macromedia's web site. If you need Navigator 2.0, you can get that from Netscape's site. I have a full listing of reference material at the end of this Guide.

Overview of concepts:

What are plug-ins?

Netscape defines plug-in modules as

> ...software programs that extend the capabilities of Netscape. Plug-ins are dynamic code modules, native to each Netscape platform. Plug-ins are complementary to architectures such as OLE and platform-independent programming languages such as Java.
>
> Plug-ins can have one of three modes of operation: embedded, full-screen, or hidden. An embedded plug-in is a part of a larger HTML document, visible as a rectangular frame within a page (embedded plug-ins are specified in HTML with the EMBED tag). A full-screen plug-in is a self-contained viewer, completely filling the content area of a Netscape window. A hidden plug-in runs in the background. [1]

The Shockwave plug-in is the embedded type. Plug-ins didn't exist three months ago, but they're now a growing industry. Visit Netscape's plug-in site and see for yourself.

This begs the obvious question - are plug-ins applets and is this the same as Java? In a word - no. Java is a programming language. You can use it to write applications. If the application is small and meant to be run inside another application, then it could be called an applet. So Java can be used to write applets.

Java needs a plug-in to run on Navigator. The Java plug-in is the hidden type. There will shortly be a plug-in to run Java authored applets in Netscape Navigator, but so far only in the 32 bit version of the Windows browser. Netscape seems to be focusing heavily on bringing Windows products to the market first, 'nough said.

1. Netscape Navigator Handbook. Questions and Answers. On-line publication, 1996

Server Push - Client Pull
Most internet communication is really a series of one way message transmissions. Data is sent from one place to another, there is no immediate feedback. I don't really want to get into a communication vs. transmission debate here, so let's leave our mutual beliefs at this common ground.

OK, so to feed additional material to a user without their input you have two choices, server push or client pull. The server is, of course, the origination point of the data, and the client is the software on the user's receiving computer.

So in the normal internet way of communicating, point A would send a message to point B. At some later time, minutes, days, years, point B might answer back to point A. That's fine for e-mail. This system leaves the internet with plenty of free time, since it's only connected for the brief period that it needs to dump the data to point B. But with the advent of the web and it's graphical capabilities, it became necessary to further enhance the communication system.

So along came server push and client pull. With server push the connection between server and client remains open, not just when data is transmitted, but all the time. Now the server can "push" whatever it has down the line to the server. This data is most often changed web pages, the result might be an animated graphic or a color change. This "push" doesn't require any input from the client. You can just sit back and watch it happen!

Client pull on the other hand is a little more sophisticated in that it codes instructions into a page to have the browser automatically reconnect and take some action. It might be to move to a new page or play a sound file.

Both of these enhancements are wildly variable in their effectiveness and highly reliant on data transmission speeds for their success. They are also only workable on some browsers. Both server push and client pull are terminated when the user opens a new page.

Synchronous and Asynchronous Communication
Most of us are used to synchronous or simultaneous communication. That's the way a conversation works. That's also the way most software applications work on your computer. An instruction is given and the response is immediate. That's not the way the internet works. As I described above, the feedback comes later. This is asynchronous communication. And it's the demon that you have to understand and deal with when working with Shockwave.

Some Basic Concepts
Data Security
My personal view is that there is no real data security on the internet. There are systems in place to give you some security for certain types of transactions on the internet, but it is far from secure.

Macromedia has taken great pains to make sure that Shockwave could not be used for evil, but only for good. Seriously, in the *Shockwave Developer's Guide, Creating Movies for the Web*, there is a page long list of Director features and commands that are disabled in Shockwave. The point behind disabling these features and commands is so that no one can write to or erase from a user's hard drive. I guess you can see how easy it would be to write the "virus form hell" as a Shockwave movie and have all sorts of poor souls innocently download this movie and have it wreak havoc on their system.

While some feel that this places an unusual set of restrictions on authoring it also opens up a whole new set of design possibilities.

Communication Speeds
Because a Shockwave movie is an application that has to be fully downloaded before it can be run, you need to be very concerned about the data transmission rate and the file size of your movie. Once again this is a design opportunity. You really need to consider your audience. Just what are they using to download your pages and how much time do they have. *Never overestimate the amount of interest your audience has in your presentation.* Unless you're working with a closed network, you can't anticipate the data transfer rate of your work. So if you've got a really great Shockwave application that's pushing 500k, you'd better figure out how to epoxy your client's hands to the arms of their chair, otherwise they're going for that *stop* button!

Application vs. Process or Shockwave vs. CGI
The Common Gateway Interface has been a useful tool for enhancing the functionality of the web. CGI's most obvious use is in forms. But CGI is also very useful for dynamically building pages, and other reactive user interpretation. One of the negatives to CGI, aside from the need to write code, is that it requires a client pull to make something happen. You have to send your code back to a program on the server and have the result sent back out. This is not always quick. Server side image maps use this system. There are now client side image maps and these work much more quickly, but they don't work on all browsers.

Shockwave can be self contained. By this I mean that a Shockwave movie can function the same as a client or server side image map and do it much more quickly. Shockwave is not a good substitute for a form submission, yet. You can only do an approximation of a <FORM ACTION= "..." METHOD=GET> right now, and not a good one at that, you can only get text back. You can't truly *post* to a CGI and use the result.

Shockwave - Some Immediate Differences
OK so now you want to make a Shockwave movie and send it out to the world. What special things do you need to know? Well, right off, as we covered above, you need to make it small and you need to take into account the differences between playing back from a local drive and playing back over the internet.

Before we go any further, if you haven't downloaded, read and/or memorized the Shockwave Developer's Guide do that now. It's available on the web at
http://www.macromedia.com/Tools/Shockwave/sdc/Dev/index2.html. Methods for minimizing movies are well covered there. A lot of the following is based on the Guide.

I'm going to cover the problematic areas first and then talk about some interesting workarounds.
Director doesn't understand the Shockwave Lingo commands. This is a problem and a statement of fact. If you put PreloadNetThing "http://www.someplace.com/mymovie.dcr" within a director movie and run it you'll get an error. You can comment out the ShockLingo as you work or you can build a set of handlers that will sideline these offending commands.

You'll need to download your Shockwave movies to the server and run them on the browser from the plug-in in order to see if the ShockLingo works properly. This is a short term problem. I'm expecting Director 5 to have Shockwave integrated.

The movie runs slower through Shockwave. Yes you may see a slowdown in performance in animations run on Shockwave. Remember that you're sharing the processor with Netscape Navigator. How can you plan for this? The movie is running locally on the user's computer and so it should run at the same relative speed everywhere. By this I mean that if you see a 5% reduction in performance on a particular platform, then you can expect that level of performance on every similar platform. With respect to the debate over score animation vs. script animation, my meager experiments show that script animation is more predictable than score animation in Shockwave. Performance is always based on the particulars of the animation and so your results may differ.

The movie was over before it started. I've seen very short animations that have finished running as they finished loading. You need to consider what your user might be doing while waiting for your movie to download. If your movie is set to run only once and then stop, it might get missed. Especially if it's small and/or short.

The screen colors shift when I move from the page with the Shockwave movie. This seems to be the most apparent on web pages with a background color other than black or white. Navigator doesn't seem to handle palettes very well yet, at least on the Windows versions. I've found that if you insert an intermediate page you can "clean" the palette. I've done this with a simple client pull. It only works if the user uses a button on one of my pages and not the Navigator interface. More on this further on.

The movie runs on when told to GotoNetMovie or GotoNetPage. These are both calls to the internet. Aside from taking huge amounts of time to execute, comparatively speaking, these calls go out to a server to get something. This is that demon asynchronous communication. The command will get executed, it'll just take a LOT longer than you expected. meanwhile the playback head is still rolling along. More on this later.

The <EMBED...> tag line shows up on some browsers. HTML is a rapidly evolving language. There are many people creating many new ways to use HTML. It takes time for all of the new bits and pieces to filter down. Browsers are supposed to ignore HTML that they don't understand, and sometimes they do. You should give the user a way to download the software that they'll need to see your Shockwave movie. You can do this inside the <NOEMBED> tag. More on this further on.

Yes, you can't branch back and forth between movies and you can't run a movie in a window. But you can run many movies on the same page. So now instead of simple static graphic elements on a page, you can make animated movies and place them on your page.

You can use the same movie on many pages but you can't use the same movie in more than one place on a single page. This means that if you have an animated bullet and you want to use it 3 times on a page you'll have to make 3 copies with 3 different names and load all 3. You can use these same 3 bullet movies on as many pages as you like.

While you can't branch back and forth, you can branch forward. This has 2 advantages: first you can build a movie in segments and play the segments one at a time. Because the segments are small, you can even load more than you need, branches that never get selected. Second you can make a small movie to get on the screen fast and entertain the user while you load additional movies.

All of these movies have to have the same stage size, but they don't have to use the same stage color or even use the whole stage area. If you make a small "first" movie that has the same stage color as the page background you can make a small animation that will hide the wide open space left on the screen and keep the user busy. You can use PreloadNetThing to continue to load in additional movies. These new movies can begin as soon as they are loaded, or respond to some other action like user input or Lingo. This goes a long way toward giving the illusion that there is more happening than there really is.

What Shockwave Lingo Can Do For You

PreloadNetThing is a pretty powerful command. Much more than you might expect, or at least than I expected. You can load anything. And since you're loading into a cache, you're not bound by the normally expected limit of the user's memory.

As an example, you can use the <EMBED...> tag to load a movie onto your web page. This first movie can be pretty small, it looks like the minimum size is 12k, that will load very quickly and get the plug-in started. It can be the same color as the page background and have some small animation or it could have a sound, a small sound looped and no visual effects. Now you can use PreloadNetThing to bring in a second movie that's bigger. You can set as many PreloadNetThing commands as you like from that first movie. Even if a second movie has loaded and started, the original PreloadNetThing commands are still executing. Remember a handler once begun never ends until it's finished or you kill it.

Now you're probably going to want to branch at some point to a new page. Well why not PreloadNetThing that new page? Remember you can Preload any HTTP item. If you don't want to load in a bunch of complete pages, you can load in graphics. If you have a bunch of GIFs or JPEGs that you use often, you can PreloadNetThing those so that your subsequent pages will load faster. If your user decides to move on to a page that uses these graphics and they aren't fully loaded, they'll load in the usual manner.

Now how do you keep track of all of the Preloaded stuff? Well you can name each of these events, or more properly, give them a value. You can use GetLatestNetID() to apply a parameter. Now you can use that parameter to track these loading events.

A good place to put this is in the second frame score script. I haven't tried it as part of a startMovie handler yet.

So the handler would contain some Lingo like this:

```
on exitFrame
  PreloadNetThing "http://www.someplace.com/first.dcr"
  set firstMov = getLatestNetID()
  PreloadNetThing "http://www.someplace.com/second.dcr"
  set secondMov = getLatestNetID()
  PreloadNetThing "http://www.someplace.com/third.dcr"
  set thirdMov = getLatestNetID()
  PreloadNetThing "http://www.someplace.com/fourth.dcr"
  set fourthMov = getLatestNetID()
end
```

– you can continue this for as many things as you want to load into the cache.

So what happens here is you call for a new movie to be loaded into the cache then you give it a value, a parameter - firstMov, do this in steps so that you are assigning a value to each file that you're preloading.

and then when you need this movie that you've loaded into the cache:

```
on mouseUp
if NetDone(firstMov) then
   GotoNetMovie "first.dcr"
else
NetAbort(firstMov)
  GotoNetPage "someOtherPage.html"
end if
...
end
```

here you check to see if the Preload is complete by asking if NetDone for firstMov is true. If it is then it will get played. If the PreloadNetThing isn't done then NetAbort will stop the download and then send the user off to another page in this instance. If you are sending the user off then a NetAbort() would stop all Preloading in progress. You could also specify a "go to the frame" for the "else" so that you would have a loop until the download is finished.

And again, this doesn't have to be a movie, it can be a page or a graphic.

CAUTION - If you're writing a handler that relies on a network event, be sure to give the movie something to do while you're waiting for the event to happen. As an example: In an normal Director movie you can have an exitFrame or mouseUp handler branch to a new movie. As in:

```
on exitFrame
  go to frame "start" of movie "second"
end
```

You would normally expect this handler to just fire up the movie second and start playing it from the "start" marker. If you try this in Shockwave, you get something *completely different*. When the playback head reaches the exitFrame handler, it will read the GotoNetMovie ... and start to process it, but since it needs to go out and find this movie, an asynchronous event, the playback head can't dive into the new movie because it isn't there and so it will continue on in the current movie until it finds some reason to stop. Possibly with less than expected results. So whenever you initiate a "GotoNet..." event, always give the playback head something to do.

So in summation: PreloadNetThing is a very useful and powerful command and GetLatestNetID, NetDone and NetAbort give you some control over what's going on in the background.

And now some solutions to those problems listed above.
Be very careful of loops. It's very easy to make a simple little animation with a repeat while... loop in a frame. But when you put that on a page and download it, you may have created the perpetual lockout machine. If your repeat loop is sufficiently small and tight, it can take over the processor and lock out the user. This is probably not what you had in mind. Be sure to test your work thoroughly before you unleash it on the planet.

Instead of creating an animated piece that runs once and then is finished, make a short simple animation piece that responds to some input and catches the user's eye. Take a little 32 by 32 movie with a logo twirl in it as an example. Consider having the animation run in response to a rollOver or a timer event. Have it run once and then wait for another input. Make Easter Eggs. Create small movies that use the same background color as the page. Have them start up on a blank frame and wait for a rollOver!

If you have background color problems, you can get back control of the screen palette by putting in an in-between page. This will re-sort the colors. This color change is most apparent in 8 bit Windows pages that use a background color or background tiled GIFs or JPEGs. My solution to this was to make a blank page with this as the HTML:

```
<HTML><META HTTP-EQUIV=Refresh CONTENT="3;URL=http://www.someplace.com/newpage.html">
<HEAD><TITLE>The Cleaner Page 3 back to feedback</TITLE>
</HEAD>
<BODY BACKGROUND="greenback.gif" BGCOLOR="#000000" >
</BODY>
</HTML>
```

The META HTTP-EQUIV= is a client pull.
Refresh CONTENT tells the browser to make a change. “3” is a time period in seconds.
URL= is the thing that you want to change to. In this case I’m swapping in a whole new page.

<BODY BACKGROUND=”greenback.gif” BGCOLOR=”#000000”>
What you get is a black page. My greenback.gif is a 16 by 16 pixel swatch of my background color.

This simple refresh content pulls in a new page after 3 seconds. This seems to be a necessary evil, for now.

To shortcut the playback head from running when an asynchronous call is made give the playback head a place to go. You’ll find that you can’t put “go to the frame” and “GotoNetMovie...” in the same handler. So you need to have an additional frame to put the “go to the Frame”. This means that in a frame where you are issuing the “GotoNetMovie...” command, the playback head won’t stop moving.

Instead of Shockwave immediately branching to that movie as you’re used to, there will be some internal thrashing to find the movie. This asynchronous communication eats up processor cycles. The playback head, now left on its own will continue down the score. It will try and find something else to do. You won’t like what it finds. To stop this simply put a “go to the frame” in the next frame. This will hold the playback head and your network command will get executed.

In order for the <EMBED> and <NOEMBED> tags to work, they have to be recognized by the browser. They are recognized by most. The rule of HTML is that if a tag is not recognized it’s supposed to be ignored. This usually works. You can try this if you have an old version of some browser laying around.

So in an ideal world, any browser that can read and understand the <EMBED> tags would read the SRC and try to do whatever was to be done. In this case start up the Shockwave plug-in and run the movie. If it couldn’t do that, it would then default to the stuff in the <NOEMBED> tags. If the browser can’t read the <EMBED> tags then it will just ignore whatever is there.

There is a workaround listed at the Macromedia web site for hiding the <EMBED> tagged information inside a comment that includes a JavaScript. The process of hiding the <EMBED> information inside a comment and then running it through JavaScript was/is a short term solution to the problem of Netscape Navigator browsers that could read and interpret the <EMBED> .dcr command but crashed while trying to do that.

This was/is the case with the Nav 2.0b(x) for Macintosh. hiding the <EMBED> inside a comment and using JavaScript is not a universal solution. Netscape Navigator is the only browser that can run JavaScript right now. So you see the list of browsers that can interpret the <SCRIPT LANGUAGE...> tag is even shorter than the ones that can read <EMBED>.

To further explain these processes I’ve built some example movies. Actually I’ve built an example web site that shows an HTML site and a Shockwave version of the same site. You can go through these pages and make your own judgments on Shockwave. In most cases the Shockwave element is not much different in size from the static GIF element that it replaces. So the Shockwave’s page is up and running in approximately the same time span as the static page. If you’re interested in the code that went into all of this, e-mail me and I’ll send off the actual movies to you. I don’t have an FTP site and so I can’t make them available on-line.

There are additional movies available for downloading at Macromedia’s Shockwave Movie Lab. Be sure to look at those and pick them apart.

As I write this there are still some problems with writing and running Shockwave movies. I hesitate to call them bugs, and I don't know where to point the finger. So until these products, Navigator 2.0, and Shockwave 1.0, are in general release, you will probably find *unexpected features.*There are resources, so be sure to take advantage of them.

Place to Go, Things to See
The following are web resources for working with Shockwave:
Macromedia : http://www.macromedia.com
NetScape: http://home.netscape.com, or simply use the **Handbook** button at the top of the browser.
Shockwave Developer's Center: http://www.macromedia.com/tools/Shockwave/sdc/Dev/index2.html, This will get you right to the latest information on Shockwave development.
Shockwave Developer's Guide: http://www.macromedia.com/Tools/Shockwave/sdc/Dev/contents.html
Maricopa Director Site: http://www.mcli.dist.maricopa.edu/director/This is a searchable archive of Director related information.
My web site: http://www.interact.lm.com, this is full of examples of the techniques that I've explained above. Please feel free to comment.

There is additional ongoing commentary about Shockwave and Director in general on the Direct-L mailing list. You can join this list by sending an e-mail message to LISTSERV@UAFSYSB.UARK.EDU. In the body of the message, type SUBSCRIBE DIRECT-L (your first name) (your last name)

Shockwave is interesting and exciting. It can also be extremely frustrating at times. I hope this short Survival Guide helps you to get started on your way to building some great interactive tools for the web.

Remember this is just the beginning. The Shockwave Lingo command set is very small right now. But that will change, and for the better. Consider that a number of developers have begun adapting existing games to Shockwave. Right now you can use Shockwave to link to other movies, other HTML documents and to other sites. Soon you'll be able to post data from a Shockwave movie. This will bring a whole new level of interactivity - multi-player games, data retrieval from remote sites and much more.

In the future, you're going to get computers as prizes in breakfast cereals. You'll throw them out because your house will be littered with them.

Robert Lucky, interview in *Newsweek,*June 4, 1979