The history of software is marked by a series of so-called "killer apps": VisiCalc (the first spreadsheet), email, the browser, and even eBay or Amazon are all--arguably--killer apps. Each one inspired users to adopt it and software producers to copy it. While no one really knows what the next killer app will be, chances are good that it will be built using Flash, Flash Remoting, and the Flash Communication Server. To make this possible, developers must be aware what the tools can do. Come to this session to see both what's possible and why the next killer app is waiting to be uncovered using Flash.


Designing

The main focus of this paper is to help people see the technical possibilities of Flash. However, all the technology in the world won't make a bad idea good. Before looking at technical solutions it makes sense to analyze your project objectives. With a goal of making the next killer app, it can be easy to try to do too much.

Think Specific

I think the fatal flaw of failed apps is they try to do too much. I love to point out my cell phone's calculator (shown in Figure 1). Who's idea was that? First, I challenge you to make a less usable calculator. Second, who said I wanted a calculator in my phone? Anyway, the phone works fine as a phone but unless they're going to do a better job with the calculator then don't bother. I often see cases where people will make an entire email client inside of Flash, for example. As a technical achievement this is great—but there are some really great email clients out there. Unless your app will do something unique or do something better, you really need to ask whether it's worthwhile. I'm not saying a branded email client—for example—isn't a good idea, just be sure you're not doing something just because you can.

Figure 1: As useless as it is unusable: my cell phone's calculator.
I have two related points to make: don't try to do everything, and, make sure you're offering something new. I was recently asked whether it was possible to come up with a new idea that hadn't been tried before. Of course you can—but even if "all the good ideas have been taken" you can still pick something and do it better. For example, you could simply brand something with your client's colors and logo. But even more than that, as you focus your objective it becomes easier to separate from the pack. For example, you've seen countless chat apps but make one that translates typed text as it's sent and you've got something unique. In fact, I've got just such a demo for this presentation plus you can see a more refined implementation at http://teknision.com. Does this mean, don't bother? No, just think of something different or more refined.

Tools Overview
Assuming you have a good idea, the next step is to select the right technology to reach your goals. Here's an overview of available tools so that you can better select the appropriate solution for your project.

Flash MX. It's the tool used to create Flash animations and applications.

Flash Remoting: a way to send data to and from application servers over an optimized binary protocol called AMF (action message format). Additionally, data is automatically parsed so that, for example, database queries arrive as a Flash object.

Flash Communication Server: a standalone server to which your Flash created .swfs can make persistent connections (allowing real time interaction with others) and optimized to both record and transmit live or recorded audio/video streams. Probably Macromedia's coolest product ever.

Application Servers (including Cold Fusion): a webserver (configured and connected so people can browse to it) that delivers customized pages based on a programming language. Instead of creating static pages by hand, an application server can dynamically create pages on the fly.

Macromedia Central: almost like a replacement to the browser but designed to only display Flash movies and web data. Features unavailable in browsers include: special on/offline features for occasionally connected users; support for collaborative data sharing (between different apps); and direct support for outside web services (that is, it supports SOAP). Depending on how many people jump on this, it may become Macromedia's coolest product ever.

Breeze: both a tool and a service. The tool coverts a Microsoft PowerPoint presentation into a Flash movie and lets you add audio and chapter delineations. Users can watch your Breeze presentation online (on a hosted or dedicated server). Additional navigation tools let users pause, play, and jump to different chapters in the presentation.
**Breeze Live**: geared to make online meetings better. Not only are there different roles (like a presenter who can take control and an attendee who can only watch) but there's integrated chat, webcam streaming, and an awesome screen sharing feature where the presenter can let everyone else see his desktop or any application he's running. As a participant in a Breeze Live meeting, I can vouch that it's a great experience.

**Dreamweaver**: A text editor. Sounds funny but that's all it produces! Granted there are a million of related features which make editing, previewing, and deploying powerful web pages easier.

Figure 2 attempts to show how everything fits together.

At the top left you have Flash authoring where you basically create .swfs that play within the browser (directly below). The bottom left signifies how a user can interface using a mouse, keyboard or webcam and mic. Any standard .swf can take user input and, through programming, perform operations inside the movie.
Flash can also reach out. The simplest manner is by launching new windows or affecting the browser in other ways. In addition, you can connect a .swf to the Flash Communication Server (top right). These connections are persistent so every connected user can remain synchronized. Another way for Flash to reach out is to make requests to (and handle replies from) application servers. The most elegant way to do this is through Flash remoting. I show Cold Fusion, but nearly any application server will work. (Incidentally, Flash Communication Server can also use Flash Remoting thus reducing the total bandwidth—compared to having each client make requests.) The fact you can send and receive data from an application server means you’ve got access to the whole world. Both local databases and remote web services.

Figure 2 doesn't show everything. For example, Breeze isn't there, but it's really just a turnkey service that uses existing technologies such as Flash Communication Server. Perhaps the most biggest omission is Macromedia Central.

Details: Flash

Naturally, I can’t discuss everything here. However, there are a few important features that are worth learning. When employed properly they can make your app more usable and refined. Basically, this is a list of "features worth knowing about".

Arrays vs. Generic Objects are two ways to store groups of data in single variables. That is, where a regular variable has a name and a value, a variable's who's value is an array can have multiple values. Arrays are good when the number of items is unknown or needs to grow or shrink as the number of records vary. Arrays also have lots of sorting abilities. Generic objects are nice when the contained values need context. For example, an array might be good to store a list of your kids. But, a generic object might be good for storing firstname, lastname, and phonenumber because each value is identified with a property name.

DataProvider Class is basically an array full of generic objects. The cool part is you can bind a DataProvider to a component so that as the data changes so does the component's display.

TextField and TextFormat Objects are necessary for dynamically presenting formatted text. A TextField is an instance (of Dynamic or Input Text) onscreen. A format object is like a stylesheet, so it has no effect until you apply it to a specific TextField. Also, Flash supports simple HTML tags including a href. Such hyperlinks can additionally trigger scripts inside your movie.

XML is simply a way to structure data so that the nature of the data is inherent in how its stored. Sort of like how the slashes in a date: 8/21/03 help put some meaning to the numbers. The slashes are the structure not the data. Anyway, it's a hierarchical style of formatting data that either loads into Flash or gets sent out of Flash to an application server.
Callbacks for Asynchronous Operations is a necessary way to structure certain operations. For example, making a request to a remote database won't give you an immediate reply. Instead you set up how you're going to handle the response first, then you make the request. Asynchronous is simply the idea that you can make a request (say load data from a database) and Flash will continue to operate (say, animate a graphic) instead of hanging while it waits.

Dynamic Graphics includes a variety of runtime drawing capabilities. These include drawing lines, curves, and filled shapes. In addition, you can load external images or Flash movies, as well as display symbols contained in the movie—but positioned on the fly.

SharedObject Local is like a browser cookie—it allows a movie to save data on the user's computer so that next time they visit your site it can remember and restore the display or automatically log in. Cookies aren't as useful because they only hold string values. With Shared Objects you can store any data type (such as array or generic object). Then, when resuming, that data comes back into Flash in the same form.

LocalConnection Object is a way for multiple Flash movies to send messages and trigger events in other movies playing on the same computer. (Interestingly, the syntax is nearly identical for when using the remote NetConnection object with Flash Communication Server.) Anyway, if nothing else, LocalConnection is the primary way your application elements built for Macromedia Central can all work in concert.

onResize is an underutilized event that triggers whenever the user resizes their browser. This means, if you want to rearrange the screen based on their layout you can do it when the resize. To use this your publish settings must use Scale: no Scale (or just put this script in your file: Stage.scaleMode="noScale"; ).

Details: Remoting
More a "feature" than a "product" as it's built into ColdFusion MX (though you can buy it for .NET and Java servers). It lets Flash movies send and receive data to application servers (which can, in turn, link to databases). In addition to parsing common data types (so you don't have to), data travels over HTTP using a small and fast binary message format (called AMF—action message format) which is asynchronous (meaning Flash can proceed instead of waiting for data to arrive).

There are only a handful of methods related to remoting, but volumes of details from which you can learn. It's really a simple idea: Flash makes a request to an application server, that server processes the request (perhaps accessing a database or web service) and returns a result, Flash receives the result. The actual sequence is Flash defines the request by building data to send as well as defining how it will react when data comes back. Then, it sends a request. The point is that you want to make sure Flash is ready to handle the result before it makes a request.
The two huge advantages remoting has over alternatives using GET and POST is that:
1) you don't need to convert all values to string (you can send and receive numbers, arrays, generic object etc.) and,
2) data travels over an optimized binary format called AMF (Action Message Format).

The Macromedia documentation flips and flops between two primary ways to make requests and handle results--I'll just show one skeleton form here:

```actionscript
#include "NetServices.as"

// RESPONDER
myResponder = function(){
  this.onResult = function(data){
    trace("result is "+data);
  }
  this.onStatus = function(info){
    trace("onStatus called : " + info.description);
  }
}

// INIT
myGateway = NetServices.createGatewayConnection(); // connect
myService = myGateway.getService("path.to.cfc"); // get service

// TRIGGER
myService.myFunction(new myResponder(), "param");
```

First, you'll need to include the NetServices ActionScript Library. Then, the homemade responder object is defined. The two events it's set up to handle are part of remoting: onResult and onStatus. Notice, onResult will receive a single parameter from your remote method (called data here). Also, onStatus receives a generic object (called info here) that contains the property always called description. (There are other properties, but this is the explanatory one.) The initialization need only be executed once. It first points to you installed version of remoting, sets up a gateway instance, then gets a service instance. The service points to the file containing your methods (say a Cold Fusion Component or "CFC")--in this case, a file called "cfc" inside a folder "to" inside a folder "path" (basically a folder path but using dots not slashes). Finally, the last line shows how you can trigger a method (called myFunction) that resides in your CFC. Notice that the first parameter received by your remote method is the second one you pass. That is, if you pass a responder object as the first parameter Flash knows not to send that parameter.

The myResponder has just two properties for the two events. However, you could add more properties so that each instance you create could distinguish itself from the other. For example, if you make repeated calls to the remote method you'll need a way to sort out which reply is from which call. The responses don't necessarily come back in order.

Flash remoting simply gives you direct access to application servers. This means Flash can be the visual front end to any data, whether it's complex or contained remotely. This also means that, via remoting, Flash has access to any web services.
Details: Flash Communication Server
The Flash Communication Server (FCS) lets you build features into your Flash Player files that let users share data in real-time with others. In addition, live or recorded streams of video and audio can be included in your applications. This is probably the most exciting product in Macromedia history.

(For additional details about FCS, read the handout from my session "Exploring Practical Uses for Flash Communication Server")

Flash Communication Server really is cool. There's a free developers version plus service providers that are very economical (for example, adding FCS to my site hosted at (mt) Media Temple costs only $15 a month). You owe it to yourself and clients to check it out.

Details: Macromedia Central
Although I suspect Central will be out by the time you read this, it has been embedded in the latest Flash Player for a long time. That is, the first time you visit a page attempting to install Central, it uses the Flash player to perform the install. Once installed it's basically an alternative browser that hosts Flash movies. The cool part is your applications have access to slightly more powerful privileges especially geared toward the "occasionally connected" user. For example, you can control what files get cached so the app works while offline. Additionally, you can build in synchronization features that automatically bring a user up-to-date when they reconnect.

Central apps have three main components: 1) shell apps that play in the main window; 2) pods that are little mini apps that users can decide to view at all times; and 3) agents that run in the background so that when certain events occur (like your stock price peaks) notices are brought to your attention. Sending messages between your app components is all handled through Flash's LocalConnection object.

Not only can your app components talk to each other but--provided the user has enabled it--data from one app can be sent to another. For example, it's possible to select data gathered by a stock price monitor and send that to a graphing app built by another developer. Data is sent in a standard RDF form. The idea is called collaborative apps--multiple single-purpose apps can be snapped together by users. Here again it pays to have an app that's just good at one thing instead of trying to solve all world problems in one uber app.
In fact, you could do anything Central can with Flash and a third party projector making tool (say, Northcode's SWF Studio for example). After all, Central is just Flash player 6 with a few extra features. There are several reasons why Central is attractive however. First, it's easy to install. Users don't have to trust that your projector is safe—they have Macromedia lending its credibility to the main install. Plus, a lot of people will get Central quickly... and with little trouble. Once installed additional apps are easy to install and remove. Also, the primary built-in app for Central is an application finder where users can get linked to your app. On top of all this, Macromedia has included a try/buy mechanism and hosts a merchant server to let you charge for your apps. You can require a one time charge after the trial period or even trigger requests for more money when the user enters certain areas of your app. Naturally Macromedia takes a cut of the revenues.

Even if you don't immediately have an idea for a commercial app that's going to let you retire, do check out Central because it's really cool. I can think of plenty of non-public apps. You don't have to charge for the app but rather charge for access to information where appropriate. Finally, it's really just a new environment where your apps can get exposure. Oh, and don't forget modem users fall under the "occasionally connected" definition.

Read more: www.macromedia.com/software/central/ (and be sure to read the white paper—it has a great "day in the life of" story).

With the permission of the author and publisher, significant portions of this handout are excerpted from the forthcoming book "Flash for Rich Internet Applications" by Phillip Kerman and published by New Riders (www.newriders.com, ISBN: 0-7357-1377-4).