Macromedia Flash on the Client and the Server

Robert Barone
Buffalo State College, USA

INTRODUCTION

The Internet and more specifically the World Wide Web is a global communication environment. Either via a personal or commercial Web site, we seek to express ourselves as individuals or as organizations through this technological forum. We desire to provide an experience that will bring visitors back to our site. Providing rich multimedia content is the contemporary paradigm in achieving these ends. The software company Macromedia has addressed this need with their commercial product known as Flash. Macromedia Flash is used to create interactive multimedia environments in the client. There are also server-based technologies that communicate via Flash directly or indirectly.

BACKGROUND

Macromedia Flash has been evolving at a considerable pace. Up to Flash 4.0, the user interface was menu driven. From version 5.0 on, the user interface is palette oriented. Flash MX is version 6.0. MX designates a suite of programs. The Flash version at the time of this writing is MX 2004 (version 7.2). Components, introduced in Flash MX, allow the rapid building of user interfaces (analogous to HTML form elements). Flash MX 2004 introduces a new component architecture, called version 2 (or simply v2) with features not found in the previous (version 1 or v1) Flash MX component architecture (Statler, 2003). MX 2004 sees a branching of Flash into two separate applications, Flash MX 2004, incorporating the standard elements of the application, and Flash MX 2004 Professional, an extended version. A history of Flash is available at *www.macromedia.com/macromedia/events/john_gay/index.html*.

FLASH AND THE CLIENT

Flash is much more than a graphic arts program. This application allows the designer and the developer to create a true and unique graphical user interface (GUI). This GUI can be precisely programmed to support end user interactivity. Flash has found tremendous treatment in the World Wide Web, although it is not limited to this arena.

Flash employs vector graphics as the primary medium in content delivery. Vector graphics are described by mathematical formulas. These formulas are solved by the video processor of the client computer and a raster image is rendered. Raster graphics are bitmap images. They are composed of pixels that require a large number of bits to convey the information. Bitmap file sizes are inherently larger even when lossy compression techniques like JPEG (Joint Photographic Experts Group) (Knuckles & Yuen, 2005), a standard graphic file format, are utilized. Vector graphics are scalable and retain information when enlarged, although they suffer from aliasing (Bardzell & Bardzell, 2004). Raster graphics are not scalable and become unintelligible when enlarged. Flash makes use of streaming. The information contained in the streaming wave file is displayed in the browser as it is received. The browser does not wait for the entire contents of the file to be downloaded first. These two features, vector graphics and streaming, combine to make Flash a successful medium for providing multimedia content even with low bandwidth connections. Furthermore, the programming aspect of Flash further enhances and extends the Web sites usability by bringing interaction to the forefront.

Concepts in Flash are hierarchical in organization. Drawing on the stage with any of the tools creates a drawing object. These are the basic building blocks of a Flash movie. Objects can be selected, copied, grouped, transformed, skewed, stacked, deleted, and so forth. Essentially any designer's demands can be met with the drawing object. Flash also supports importing of images.

The next level of hierarchy is the symbol. If the desire is to motion animate (motion tween) the drawing object, it must first be converted to a symbol. Interestingly shape morphing (shape tweening) is performed on a drawing object. There is a subtle importance here which cannot be underestimated. Once a symbol is created, it is automatically placed inside a container known as the library. If the designer wishes to re-use the symbol, it is dragged and dropped from the library onto the stage. Multiple occurrences of the symbol on the stage are referred to as instances of the symbol. An instance on the stage may be thought of as pointing to the symbol in the library. Since the symbol is only stored once and reused many times in the form of an instance, file size remains at a minimum.

Another advantage is file-to-file reusability of symbols. A library from one Flash file can be opened in a completely separate file and the symbols reused.

There are three symbol types, *movie clip*, *button*, and *graphic*. Movie clips are self-contained Flash movies that can be treated as instances. Once on the stage they cycle through their animation sequences independently of the main movie. Movie clips are also programmable. Buttons maintain an up state, an over state, a down state, and a hit state. The first three states can be animated and buttons are also programmable. Graphic symbols are simply used to create motion tweens and are not programmable in the sense that an action cannot be directly attributed to one.

The stage also maintains a hierarchy in the form of layers. Think of the stage as a deck of playing cards (a typical analogy). Each card in the deck is a layer—the card at the top being the highest in the hierarchy and those below it are lower. Typically, one animation per layer is the rule. It is not enough to recognize that a symbol must be on the stage to work with. Identifying the corresponding layer it resides in is just as important.

The timeline is the backbone of a Flash movie. A celluloid film strip is composed of still images that are rapidly moved across a movie projector to give the illusion of continuous movement. The same is true of a Flash movie. Each layer has a timeline that is subdivided into frames. A symbol is placed into a layer. Movements of symbols on the stage are broken across a series of frames in the layers timeline. As the software play head moves form one frame to the next, the symbol is given the illusion of movement. When the play head reaches the last frame, it loops back to the beginning and plays over unless a programmable command is given to interrupt the process. Consequently, it is seen that certain symbols can be programmed via the timeline, directly via their instance, as well as the actions and objects that can be attributed in a given situation.

A Flash movie is created in a file with extension *.fla*. This file can create many file types. For Web delivery, the common types are *.html* (hypertext markup language) and streaming wave file *.swf* (streaming wave file or small Web file). The client's browser requests the html from the server. Once the browser has loaded the html file, the browser calls the streaming wave file from the server.

Flash incorporates a programming language know as ActionScript, which was introduced in Flash version 5.0 (Mohler, 2001). ActionScript, like many other scripting languages, is considered loosely typed and prototype based (Hall & Wan, 2003). This means that variables do not have to be declared before they are used and that the data type stored in a variable may change depending upon how program logic flows. Flash MX 2004 introduces ActionScript 2.0, whose distinguishing features are the ability to define a true class, case sensitivity of all identifiers, and strong (or strict) data typing (Statler, 2003). The developer has the ability to choose the version of ActionScript and Flash Player the movie will run in. ActionScript is ECMA (European Computer Manufacturers Association) compliant (Williams, 2002).

With ActionScript, the developer is handed an object toolbox to use. The Flash architects have designed an immense collection of functions and objects which can be utilized to manipulate instances of symbols. Once the concept of using an object in a program is understood, what remains is becoming familiar with the myriad of object methods and properties. It should also be noted that ActionScript and JavaScript can communicate on the client side. Typical examples are pop-up windows.

The end result, the Flash movie, is essentially a unique stand-alone miniature GUI application that executes on a desktop or in a browser window delivered via disk or network.

FLASH AND THE SERVER

Communication to and from a Flash movie, especially from the client back to the server, is possible. One method is to use middleware. Middleware is a server side script that resides and executes on a server computer. Examples of server side scripting environments are Active Server Pages, ASP, by Microsoft (now ASP.NET); ColdFusion by Macromedia; Java Server Pages, JSP, by SUN; and Personal Home Page, PHP, an open source technology. Perl (Practical Extraction and Report Language) (Castro, 2001) was one of the first languages used to build a Common Gateway Interface, CGI, for server side interactivity.

The idea here parallels the concept of creating an ordinary FORM tag in a traditional HTML (hypertext markup language) page and using GET or POST method to send the data collected in the FORM to a server side script that processes it. This is where Flash will make use of program-controlled text boxes and components to produce HTML FORM behavior.

The server side script will process the data and respond in many different fashions. One way is to create an HTML page and send it back to the browser. Another is to store the data in a database manager first and then send a page back to the client. There are many variations of theme depending upon the Web application.

An important consideration (Chambers, 2002) when using a Flash movie to interface with middleware is the following: a Flash movie is loaded into the browser once and can handle many data transfers to and from the middleware. Middleware can build content for a Flash movie without building a new Flash movie (Muller, 2003). An HTML file loaded into the browser interacts with the