Chapter 20

Enhancing your Biology course and lab WWW pages with the latest in HTML, CGI, and Java script features

Glenn MacDougall
Academic Development Centre
Acadia University
Wolfville, Nova Scotia, Canada B0P 1L0
glenn.mcdougall@acadia.ca

Glenn Bauer
Biology Department, St. Michael's College
Colchester, VT 05439
gbauer@smcvt.edu

Glenn MacDougall is currently the manager of the Acadia Institute for Teaching and Technology (aka The Sandbox) where he works with Acadia faculty to implement the effective use of technology in teaching. To date, the Sandbox has offered over 120 faculty development workshops. Prior to the Sandbox, he spent 8 years in Acadia's Biology Department as a Lab Instructor for first year Biology, and second and third year courses in Cell, Genetics, Ecology.

Glenn Bauer is an Assistant Professor of Biology at Saint Michael's College in Colchester, Vermont, where he teaches general biology, genetics, and microbiology. He received his B.Sc. in Biology from Purdue University (1984) and his Ph.D. in Biology from Washington University in St. Louis (1989). Before arriving at Saint Michael's College, he was a Post-doctoral fellow at Cold Spring Harbor Laboratory and taught part-time at SUNY - Stony Brook.

Contents

Introduction

This workshop, for individuals with basic HTML and web page construction experience, will demonstrate how to enhance your course web pages. We will demonstrate techniques in a platform independent manner that will allow you to add tutorials, quizzes, embedded sound, video, image maps and java script to your course homepages. We will discuss the potential and the limitations of enhanced web pages and the basic differences between the Internet and your own campus intranet. We will also show you how to optimize your pages for delivery in either situation. Templates and evaluation copies of current shareware applications will be given to participants and will used during the session to prepare web pages. **This session, while not advanced, is not really appropriate for individuals who have little or no background in web page construction.**

A web site has been developed for use in conjunction with this workshop. The site is found at:

http://plato.acadiau/ca/courses/biol/macdougall/calgary/

From this site, users can gain quick access to an online tutorial on web page creation, and can quickly link to sites for downloading useful software. This site will remain available after the Calgary ABLE conference so participants can utilize it once they return to their home institutions.

The first part of this workshop will involve a quick review of Hypertext Markup language (HTML). We will begin by creating a course or lab web page using a text editor. Once the page is created, we will look at inserting digital images and the preparation of those images using Paint Shop Pro, Map This, and an animation program. The basic layout and design of lab web pages will be discussed along with the effective use of frames.

The second half of the work shop will look at using the web in an interactive format. Templates of online questionnaires and tests will be provided. We will open them in an text editor to explain how they work. These forms when properly configured to run on your University server will allow your students submissions or test results to be mailed directly to you or added to a database.

Web Design for the Sciences

Many of us have been browsing the World Wide Web without giving much thought as to how far the technology had advanced and what can be accomplished through the various internet protocols. As we search for information or simply browse to see what's out there, we find a constantly increasing number of sites and ever larger amounts of information. While some schools and instructors are making the WWW an integral part of their teaching, it is clear that much of the information which is available on the Web is unfiltered, unreliable, or simply incorrect. Instructors who once relied upon information made available by way of the Web and the Internet are beginning to design their own online resources, presenting information and formats that are more structured, specific, and suited to their needs.

It is becoming increasingly easier to access the Web, and more powerful computers are available with which to do so. This offers students the opportunity to make use of content-enriched and multimedia-enhanced course materials. Instructors are being encouraged to offer their course materials via the Web in order that students may have easy access to basic course information, lecture notes, and other pertinent course-related materials. In laboratories it is now easy to provide students with online lab manuals and to encourage and support a certain amount of interactivity with the information being made available.

Online tutorial packages that were once only available from high-level software development companies or publishers can now be easily created through the use of frames, CGI programming, Java, and JavaScript. With an effective use of frames and online forms, an instructor with only a basic knowledge of Hypertext Markup Language (HTML) can create web pages which offer much more than stand-alone, static, paper-based lab manuals and handouts.

If one agrees with the basic premise that students learn best through engagement and interaction rather than through the traditional strategies of reading and memorizing, one must then agree that that pedagogical reasons for making use of the Web and computer technologies are sound. Electronic media such as these allow a student to guide his or her own path and pace of study, effectively and significantly enriching the learning process. Students who feel that they are responsible for their own progress often achieve better results than those who feel that they have little more than a passive role.

Designing Web-based materials that take full advantage of the asynchronous and interactive nature of the technologies available is not difficult. It involves little more than some careful planning, a clear assessment of the primary objectives of a course, and a variety of easy-to-learn HTML and scripting skills. (It can also involve a significant portion of your time).

In Biology labs, students are often asked to document carefully what they see and, when possible, to prepare accurate sketches. The need for this is obvious, but it can present one of the biggest challenges in a lab. University students, in their first couple of years, tend to be very linear and vertical thinkers. Peering into a microscope and examining a leaf cross-section can, for a moment, become their entire world. In order to sketch the slide, however, students must extract themselves from that world to look at their paper. This constant process of shifting attentions continues until either the drawing is completed or the student becomes frustrated with the process. Several weeks later the world within the microscope has faded from memory, and students are left only with their hand drawn sketches from which to study.

One possible solution to this ongoing problem is through the creation of web-based lab materials using the same slides and course content which the students use during the lab exercises. With the information and slides available via the web, students are thus able to re-visit the lab (albeit virtually) and reacquaint themselves with the materials previously encountered. The reliance on self-drawn sketches will drop, and the result will be that a student's own artistic abilities have less of an impact on their ability to learn effectively.

Web Browsers: A Basic Overview

The two most common Web browsers are *Netscape Navigator* (NN) *and Microsoft Internet Explorer* (MSIE). *Netscape Navigator* has several versions, one of which *Navigator Gold*, allows you to edit and create your web pages. The most recent version, *Netscape Communicator* (NC or Ncomm) includes a whiteboard and other online collaborative options. The next version of *Microsoft Internet Explorer* will also include these (and other) types of interactions.

Designing content for one web browser does not necessarily mean that the information will be seen or supported by the other. Because of the slight incompatibilities between the two browsers, it is better to determine your target audience and the browser they are using before spending too much time developing your web site. This is not generally a problem for basic HTML, but it is a problem for the more advanced HTML formatting options and JavaScript. You should be aware that different versions of the same browser may in fact treat the HTML in slightly different ways, or may not support all of the tags you have used.

Frames

Frames can be very useful in assisting with navigation within a web site. Links in one frame can update information in another frame, allowing the user the ability to view many links on a page quickly without repeatedly going back in the browser. Web pages without frames are more like books, in that they are more or less linear. A link on page one takes a user to page two (Page one is no longer in view). When you create a page with frames, it is possible to send the viewer to page two while maintaining a view of page one.

A much more detailed explanation of frames can be found in: Frames Explanation (below)

Graphic Images

Images used on web pages are commonly from a variety of sources: a graphic created in an image editing program, clip-art from a CD-ROM, a scanned photograph, a screen capture from your computer, or possibly a still image captured from a video. No matter where the image comes from, it has to be taken into an image editing software package and saved as either a GIF (Graphics Interchange Format) or JPG (Joint photographic Experts Group). Most image editors now allow the user to import common image formats [TIFF (Tag Image File Format) or BMP (Bitmap)] and save the image as a GIF or JPG. TIFF's and BMP's are both uncompressed file formats and tend to take up a relatively large amount of memory. GIF's and JPG's are both compressed file formats and therefore require less memory than the files they were created from. Because of the smaller image size, both formats are appropriate for use on a web page. Some thought should go into whether you save the image as a GIF or JPG. The GIF format is a "non-lossy" file, meaning that during compression, and expansion in the web page, all pixel information is maintained. This means that if the sharpness of your image is critical to your viewer (i.e. text or clipart with sharp boundaries), then GIF is the format to use. During compression of JPG's pixel information is actually taken out of the image, and not restored when displayed in the browser. This "lossy" type of compression is fine for photographs because during the loss of image information a slight blurring occurs, which is less noticeable in a photograph.

JavaScript

JavaScript is a language that allows you to increase interactivity on your web pages. Some of the interactivity features JavaScript offers include the ability to create frames, pop-up windows, alert boxes, mouse-over events, and to check for correctly formatted form input. The JavaScript code is embedded in the HTML of the web page, or is called from the server as a .js file. Saving your JavaScript as a .js file, and calling it from the server allows you to include answers to questions you may ask the student, but not allow the student to view the answer by revealing the source code on the web page.

Java vs. JavaScript

Java and JavaScript are languages currently used to provide client-side functionality on the web. Despite the similarity in names, they are only alike in trivial manners. The two common features the languages share is they are object oriented, and they have a very C-like syntax. An object oriented language is a language in which you program by creating objects, and defining how the objects communicate with each other to perform a task. The C-like syntax of the two languages allows programmers to learn the languages with minimal effort. JavaScript, or more recently, ECMAScript, is a scripting language for client-side functionality. It allows a web developer to manipulate the objects in a browser's environment, including the document and the browser itself. For example, it may be used to move layers, change widget labels, and point the browser to new documents. As most scripting languages, it is not strictly object-oriented, and one may fall back to procedural programming methods.

Java is a more general purpose programming language. It was not specifically developed for building WWW applications. Therefore, it can and often is used to develop server applications and general purpose applications, as well as applets which are applications that may be embedded into a web page. In contrast to ECMAScript, Java is a very strict language. Every type must be defined as a class, and all objects must be instances of a defined class.

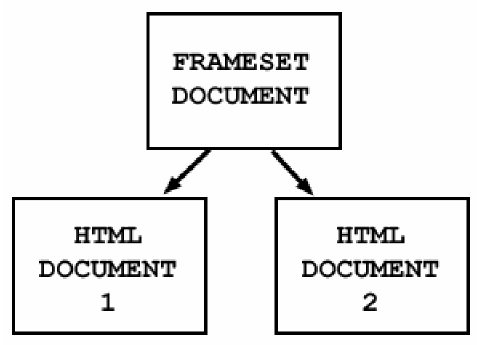
The other major difference between the two languages is that ECMAScript was designed for, and has access to, almost all elements of a browser and web page. However, due to security concerns, Java has very little access to browser functionality. It is self-contained within the applet to which it is tied.

Frames Explanation

Unlike 'regular' HTML documents in which one displayed page requires only one HTML document, framed documents require a 'controlling' or 'frameset' document and however many documents to which that frameset file refers. For example, if you want a page which displays two separate frames, you will require 3 HTML files to build that page: one frameset, and the two 'regular' HTML documents to be displayed in the separate frames.

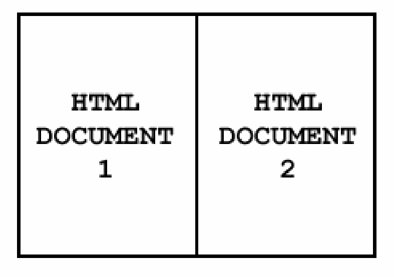
The web browser which calls up the frameset document first interprets that document, and then calls up the documents to which that frameset refers.

If the frameset document in this example was used to create a page that is separated into two column frames, it would look something like this:



In this example the frameset document simply told the browser to split the main viewing window into two columns, and then from there told the browser to call up the two separate HTML files and display them in the separate frames. *Note:* nothing from the frameset document is actually visible in the viewing window of the browser, so no HTML tags other than those used to define the frames can be used.

The code from the frameset document in the above example would look something like this:



```
<HTML>
<HEAD>
<TITLE>Example 1</TITLE>
</HEAD>

        <FRAMESET COLS="50%, 50%">
        <FRAME SRC="html_file_1.htm" NAME="left">
        <FRAME SRC="html_file_2.htm" NAME="right">
        </FRAMESET>

</HTML>
```

You will notice that there are no <BODY> tags in the frameset document. This is because the frameset document, as mentioned before, does not actually display anything in the browser window, so no <BODY> tags can be used.

In the above code sample, you will see that there are two different sorts of tags used in a frameset document. The first is the <FRAMESET> tag. What the <FRAMESET> does is create the different sized columns (or rows) of frames which will be created in the browser window. The other is the <FRAME> tag. This second tag tells the browser what the SRC or 'source' HTML document is for that frame. It is also used to name each frame (as shown in the code above). The <FRAME> tag can also be used to define the various properties of the frame, but these properties don't have to be defined in order for the frames to work.

About the NAME property. In order for the HTML documents displayed in the different frames to refer to each other, each has to have a unique name. If, for example, you wanted a link in the left frame to change the document in the frame on the right, you would have to define a TARGET for that link in the <A> or 'anchor' tag.

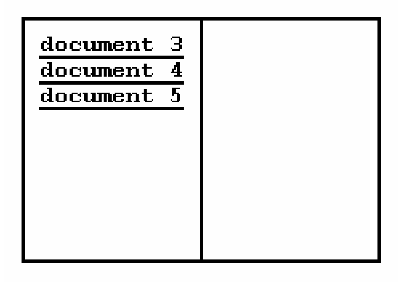
Here's an example of code for the left frame which contains links which would change the HTML document displayed in the right frame:

```
 <HTML>
<HEAD>
<TITLE>Left Frame</TITLE>
</HEAD>

        <BODY>
        <A HREF="document3.htm" TARGET="right">document 3</A><BR>
        <A HREF="document4.htm" TARGET="right">document 4</A><BR>
        <A HREF="document5.htm" TARGET="right">document 5</A><BR>
        </BODY>
```

```
</HTML>
```

This would cause the frameset to look something like this:



Should the user then click on any of the three hyperlinks, the document called by that link would be displayed in the righthand frame.

Any document in any frame can be used to call up different documents in any frame (including itself) through the use of the TARGET property referring to the unique NAME of that frame.

The FRAMESET Tag:

The FRAMESET tag is used to define the type (column or row) of frames, the size of those frames (in percentages or pixels), and the border size for those frames. The basic syntax of the FRAMESET tag is as follows:

```
 <FRAMESET
[COLS/ROWS]="[size values]"
FRAMEBORDER="[1/0]"
FRAMESPACING="[size in pixels]"
BORDER="[size in pixels]">
```

Defining the style and size of the frames is quite simple. If you want columns, simply use the COLS property, and if you want rows, use the ROWS property. The size of the frames can be defined as either percentages of the viewing window (which means that the real size of the frame will change when the browser window is resized or displayed on monitors of different sizes), or by absolute pixel values, which forces the frames to maintain a certain size.

A FRAMESET tag to define two columns with a 25% / 75% relative size would look like this:

```
<FRAMESET
COLS="25%, 75%">
```

The same result can be achieved by using an asterisk (*) rather than defining the second frame as 75%. Essentially the asterisk just tells the browser to fill up the rest of the undefined space, which, if the first frame is defined as 25%, will default out to be 75%.

A FRAMESET tag which defines two rows with absolute pixel sizes of 100 and 540 would look like this:

```
<FRAMESET
ROWS="100, 540">
```

Defining absolute values this way isn't usually a good idea, because it doesn't take different sized monitors or window sizes into account. Therefore, a better way of defining the same thing would be:

```
 <FRAMESET
ROWS="100, *">
```

As mentioned before, the asterisk will simply fill the remaining space (however much that might be) with the document to which that frame refers.

Framespacing / Border:

Unfortunately for the HTML author, the definition of the size of the frameborder is a bit complicated in that there are different codes required by different browsers (such as Netscape™ 3.0 and Microsoft Internet Explorer 3.2).

The Netscape browser completely ignores the FRAMESPACING property definition which MSIE requires, while MSIE completely ignores the BORDER property. Therefore, in order to have a border which displays similarly in both browsers, all three tags are required. For example, if you wanted a FRAMESET to define 2 rows of frames with a three-dimensional style border of 10 pixels between the two, the full tag would appear as follows:

```
<FRAMESET
ROWS="50%, 50%"
FRAMEBORDER="1"
FRAMESPACING="10"
BORDER="10"
>
```

The FRAMEBORDER property is a simple on/off switch; a value of "1" turns the three-dimensional borders on, and a "0" turns them off.

FRAMESPACING defines the width of the border between frames for the MSIE browser, and BORDER defines the width of the border between frames for the NS browser.

It might seem a bit complicated, but to keep it simple just make sure that you always use all 3 property tags in the FRAMESET tag. Make sure that you always use the same value in the FRAMESPACING and BORDER properties.

The FRAME Tag

The basic syntax for the FRAME tag is as follows: (Note: this includes all of those optional property definitions mentioned before):

```
<FRAME
SRC="[source HTML document]"
MARGINWIDTH="[size in pixels]"
MARGINHEIGHT="[size in pixels]"
NORESIZE
SCROLLING=[no/yes/auto]
NAME="[framename]">
```

The SRC property tells the browser which 'regular' HTML document to display initially in that frame.

MARGINWIDTH and MARGINHEIGHT define the margins of the document which will be displayed in that frame.

NORESIZE is an on/off property. If you have NORESIZE as part of the FRAME tag, then the user will not be able to resize the frame. If you don't have NORESIZE in the FRAME tag, then the user will be able to resize the frame as s/he wishes. Most frames are resizable.

SCROLLING defines whether or not and when scrollbars will appear at the edges of the frame. If SCROLLING="no" then scrollbars will never appear in the frame regardless of whether or not the HTML document displayed fits in the frame. If SCROLLING=yes, then scrollbars will be displayed at the edges of the frame even if the document fits in the frame. If SCROLLING=auto, then scrollbars will appear only when necessary such as when the HTML document displayed does not fit into the frame entirely. For the most part, having SCROLLING=auto is usually the best idea.

NAME is the place where you give each frame a unique name by which other HTML documents can TARGET that frame.

A FRAME tag which calls up an HTML document called "document4.htm", giving document4 side margins of 40 pixels and a top margin of 60 pixels, which isn't resizable, has automatic scrollbars, and creates a frame called "frame4" will look like this:

```
<FRAME
MARGINWIDTH="40"
MARGINHEIGHT="60"
NORESIZE
SCROLLING="auto"
NAME="frame4">
```

Stylistically, having a bit of a margin around your HTML documents is always a good idea. It makes the document a lot easier to read, and is usually aesthetically preferable. Unless you have specific design reasons for doing so, it is usually not a good idea to have MARGINWIDTH and MARGINHEIGHT values of zero. (If you leave these properties out entirely, the browser will display the documents with the regular default margin sizing for HTML documents.)

A Note About the Order of Defining FRAMES

Web browsers interpret frameset documents in a specific order and create frames according to that order. Like readers of English and most other languages, a browser reads from left to right and from top to bottom. It also creates frames in that order, so if you have a FRAMESET which defines two frames in columns, the left frame will be created first, and if you have two frames in rows, the top frame will be created first.

What this means is when you are authoring a FRAMESET document, you must define the top most or left most frames first, and then follow with the rest of the frames in order.

For example, a FRAMESET which defines two column frames which display "document4.htm" in the left and "document5.htm" in the right will look like this:

```
 <FRAMESET COLS="50%, 50%">
<FRAME SRC="document4.htm" NAME="left">
<FRAME SRC="document5.htm" NAME="right">
</FRAMESET>
```

If you simply swap "ROWS" for "COLS" in the above example, "document4.htm" will be displayed in the top frame, while "document5.htm" will be displayed in the bottom frame.

Setting Up a Web Page On Your Home Server

When designing a Web page, most people work from and save files to a directory on their harddrive. Some editors like Netscape Gold demand that you work in this manner. Once you have created a web page (using Netscape Gold, some HTML editor, or even Notepad), you likely will want to move it from your harddive onto a networked webserver. Your network administrator may have a directory set up for you, in which case you may simply copy and paste the files in place, or you may need to ask your administrator to place your files onto the web server. Most Departments in universities have some networked directory in which their departmental homepage exists. It may be possible to create a sub-directory for a specific course homepage or personal homepage. (Ask your network administrator about this).

It is essential that your files be put onto the web server and not onto a simple file server or application server. (*Note:* many servers do all three tasks, and thus have specific directory structures from which the various activities take place. Your files must be put in the proper directory.)

If you wish to use various embedded files in your html documents (e.g., embedded excel spreadsheets, or pdf's or animated powerpoint presentations), it is important that your network administrator has set up mime tables on the server. Everything may work just fine from your hard drive version, but once it is placed on the server it may be presented as lines of unreadable symbols. If you see this, then likely the mime tables are not set up. Your administrator should know what to do (Re-boot the web server to have the changes take affect).

Plug-ins and Helper Applications

Quite often web page designers wish to increase the usefulness of their web pages by inserting or embedding such things as PowerPoint slides, excel spreadsheets, word documents, Vector graphic files, movies, animations, sound files, Macromedia Director files, online testing software, or any number of third party software and files.

Both Netscape and Internet Explorer can deal with some of these on their own, but generally require either an inline plug-in or helper application to view and or interact with the new file.

Plug-ins allow a file to be opened within the browser window itself, while helper applications allow the another program such as MSoffice Excel to open and load the embedded Excel spreadsheet

If you embed such files into your files, you are making some basic assumptions on what your audience can do with those files. If your intended audience have browsers without a plug-in to view your embedded file it will appear either as a broken link, or a pop-up box will open asking the user if they wish to download the file for later use. It is therefore important to inform your audience via your web page what is required in order to view your files.

It is also important that the server to which you will ultimately upload the files (for web access) has its mime table set up correctly. (See Setting up a web page on your home server.)

Image Mapping with Gif Construction Set

GIF Construction Set for Windows is a powerful collection of tools to work with multiple-block GIF files. It will allow you to assemble GIF files containing image blocks, plain text blocks, comment blocks, and control blocks. It includes facilities to manage palettes and merge multiple GIF files together. It will make extensions of the GIF specification work for you. Among its other functions, GIF Construction Set for Windows can:

- Create Netscape animations.
- Create transparent GIF files.
- Create interlaced GIF files.
- Add, edit, and delete comment blocks.
- Add non-destructive text to images as plain text blocks.
- Create multiple-image GIF files.
- Serve as a fully-compliant Windows GIF viewer application.
- Be a GIF viewer "helper" application for a World Wide Web browser or Mosaic client.
- You can use GIF Construction Set for Windows to create and modify GIF files with as many blocks as you require.

Running GIF Construction Set for Windows

When you initially boot up GIF Construction Set for Windows , you'll see a list of all the GIF files in the \GIFCON directory of your hard drive. You can open a GIF file either by double clicking on one of the names in the list, or by selecting Open from the File menu. You can navigate to a different directory or drive either by double clicking on the drive and directory entries in the list of the GIF Construction Set for Windows application window or through the File Open dialog.

If you click once on the name of a GIF file which has a Graphic Workshop thumbnail associated with it, you'll see a small version of the GIF file's first image in the application window of GIF Construction Set for Windows. Thumbnails will be discussed in greater detail later in this document.

When you open a GIF file, the main application window of GIF Construction Set for Windows will list each of the blocks in the GIF file, beginning with the header. Clicking once on an image block will display a thumbnail for it. Double clicking on any block that GIF Construction Set for Windows recognizes will call up a dialog to edit it. You can also edit a block by clicking on it once to select it and then clicking on the Edit button in the GIF Construction Set for Windows button bar.

GIF Construction Set for Windows does not allow application blocks or blocks of unknown types to be edited. For more information on Gif Construction Set, please refer to the help files associated with the program. © 1996 Alchemy Mindworks, Inc.

Image Maps with MapThis

What is Map This?

Map This is a program that can be used to create, edit, and convert Clickable Image Maps for the World Wide Web. Map This does this by allowing you to draw the areas to be defined on the graphic you wish to use, attach HTML "url" references to each area, and save the image map data into either NCSA, CERN, or HTML 3.0 format for use on the World Wide Web.

What can Map This do for me? More importantly, why should I use Map This? Setting up image maps the "traditional" way requires you load the graphic into a picture viewer or editor that supports showing the cursor position and writing down the positions of each area you wish to define. Then you would have to type these into a specially formatted text file.

Map This makes all of this really easy (and yes, fun) by allowing you to load the graphic, and draw directly on it with live tools - rectangles, circles, and abstract polygons can all be placed and manipulated. Map This allows you to convert the map from one format to the other by just opening it and re-saving it. Map This also offers several additional tools and features to help with creating image maps:

- Zoom in up to 8x magnification.
- Fly-by AreaTips™
- Cut, copy and paste.
- Supports moving and reshaping (even point addition/deletion) of Polygons.
- Handles images of almost any size.
- Supports both GIF and JPG images, including interlaced GIFs.
- Does not require a TrueColor video board - can operate from 16 colors to 16 million colors.
- Handles multiple open files at once.
- Keeps a tracking header comment block in the image map file. This allows Map This to keep track of the image that goes with the map.
- Supports descriptions per each area, and one for the entire map.
- Supports the HTML "target" tag for HTML 3.0 image maps.
- Opens existing map files that you might have.
- Has docking/floating toolbars.
- Supports NCSA, CERN, and CSIM/HTML data file formats.
- Allows you to save or load as either .MAP, .IMP, .HTM, .HTML, or .SHTML files.
- Supports dragging links from Netscape into the map.

The following is a list of known problems, issues, or just plain "wait for the next version" items:

- Map This is a Windows 32-bit program. I do not feel this is a true limit because of the proliferation of Windows NT, Win95, and Win32s.
- Map This only uses a mouse as input. It's a semi-CAD program, after all.
- Map This supports up to 1024 areas per map. It will generate an error if you try to add more areas or load a map file that has more than this. This is not a true problem, since most servers will die if you have this many anyway.
- Each polygon area is allowed up to 64 points. Some Web Servers put a lower limit - I know of one that only allows up to 16 points.
- The scrolling of the image when you are zoomed-in is slow on some NT systems. This is usually a problem with the video driver in use.
- NCSA image maps allows "points" (closest-point-to-click-wins scheme) to be used for their maps. But since nobody has asked for, and since CERN and CSIM don't support it, there doesn't seem to be a reason to put it in.

- Map This currently only supports Netscape as a drag'n'drop link source. It treats all the others like a text source, so you will get the words, not the link.
- Map This doesn't put the map file into the server - you have to do it yourself.

  For more information on MapThis please see the help files associated with the program Help file.

  © 1995-1996 Todd C. Wilson/Fresh Ground Software

The website located at:

http://plato.acadiau.ca/courses/biol/macdougall/calgary/

will remain on-line and will be updated with new examples and references when possible. If anyone has examples they would like to include, they can e-mail me at glenn.macdougall@acadiau.ca.