2004

Melting Multimedia: A Study of Methods and Processes for Developing Small-Scale Dynamic Multimedia Systems that Can be Easily Updated by the Client

Paul Martin

Melting Multimedia: A study of methods and processes for developing small-scale dynamic multimedia systems that can be easily updated by the client.

A Thesis Submitted to the Faculty of
The College of Imaging Arts and Sciences
in candidacy for the degree of
Master of Fine Arts in Computer Graphics Design

Geoffrey Paul Martin
October 28, 2004

Thesis Proposal for Master of Fine Arts Degree

Rochester Institute of Technology
College of Imaging Arts and Sciences
School of Design
Computer Graphics Design

Title Melting Multimedia: A study of methods and processes for developing small-scale dynamic multimedia systems that can be easily updated by the client.

Submitted by Paul Martin

Date 20 May 2004

Thesis Release

I, Paul Martin, hereby grant permission to the Wallace Memorial Library of RIT, to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

_____ 05/20/04
Date

Thesis Committee Approval

_____ 11/2/04
Chief Adviser Chris Jackson, Assistant Professor, Computer Graphics Design — Date

_____ 11/2/04
Associate Adviser: Jim Ver Hague, Professor, Computer Graphics Design — Date

_____ 11/2/04
Associate Adviser: Nancy Ciolek, Associate Professor, Computer Graphics Design — Date

School of Design Approval

_____ 11/8/04
Chairperson: Patti Lachance, Associate Professor, School of Design — Date

Melting Multimedia

A study of methods and processes for developing small-scale dynamic multimedia systems that can be easily updated by the client.

Paul Martin
Rochester Institute of Technology
Computer Graphics Design
Thesis Report

20 May 2004

Abstract

This report reviews the process and practices involved in creating a multimedia project with an efficient work flow and content that can be edited by the client using a Web browser. Construction methods are reviewed and explained.

Keywords and Phrases

frozen content
centralized code
modular content
naming conventions
content management systems (CMS)
future proofing
data integrity
flat file databases
brochure style Web sites
persistent data
targeting
efficient multimedia workflow
ActionScript
PHP
Flash

Introduction

The melting of media is not a new idea. Gutenberg started the ball rolling with his concept of movable type that drastically lowered the cost of publishing information. The concept of cheap publishing to the masses was one of the major ideas that allowed the World Wide Web to bring us the Information Revolution. The idea was the Internet could:

- Remove the cost barrier of mass publishing by drastically reducing the cost of publishing information.
- Give the ability to be seen and heard by the masses, to the masses.
- Allow anyone with a desire, to be able to go to a computer and upload their thoughts with basic simple Hyper Text Markup for formatting.

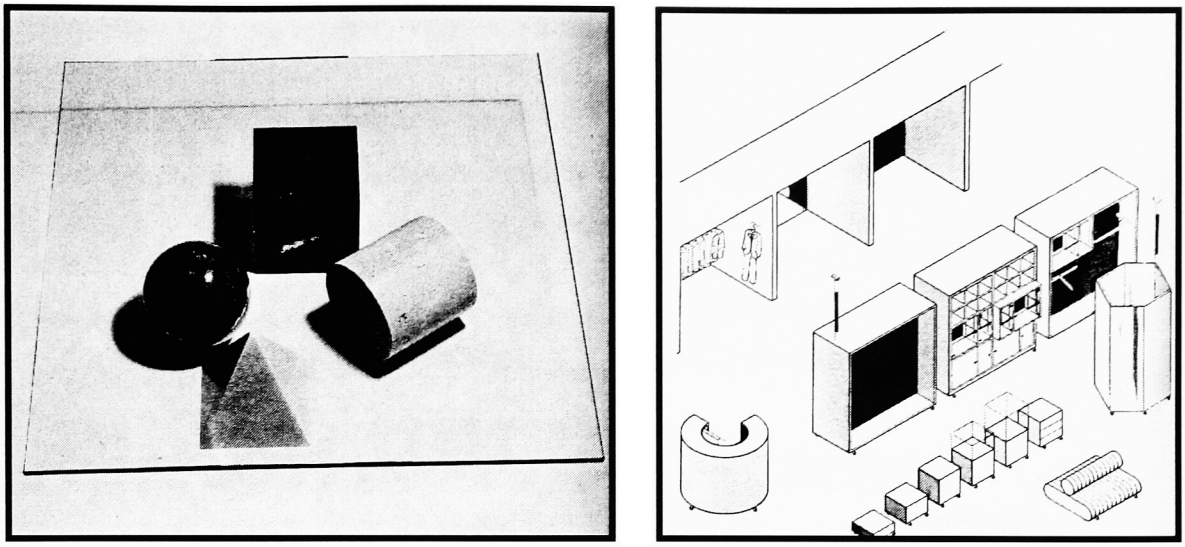
But those dreams of simple Web sites have yet to be fully realized. To be taken seriously, Web sites must adhere to design and construction practices that are complex. It is not reasonably possible for the "bricks and mortar" small business owner or organization to

know enough about the technology to keep their Web site maintained professionally. In order to maintain their online content business owners must hire a Web professional to navigate the technology. As Web professionals, we closely guard the passwords and other editing information. We are afraid the client might try to make an update on his own, and with one mis-guided keystroke crash the whole project. And that's just with a basic HTML based Web site. When it comes to the sophisticated multimedia packages we've been producing for the last two years in Graduate Computer Graphics Design, the barriers to updates are even higher. Often creating content for screen is strikingly similar to designing for Gutenberg's print world. Once the ink hits the paper, there's no changing your mind. Similarly, by the time our slick little multimedia packages are complete they are often unchangeable without great effort. We build fancy navigation with multi-state raster based buttons that must be individually edited when we find a typo. Or we set text in jpegs or gifs to satisfy our need for absolute typographic control.

As technology is advancing, often content is still frozen. And sadly, in multimedia, once a sign-off sheet is signed, and especially if CDs are pressed, the piece is virtually frozen.

My grandfather says, "women are allowed to change their mind." In the 21st century those rights should at least be extended to the rest of us who are involved in content development. This is one of the reasons I am fascinated with the Web for content delivery. The Internet can deliver content around the world, virtually instantaneously after I change my mind and click save, the information is fresh.

Part of my inspiration came from Massimo Vignelli when he and Lella visited RIT in the fall of 2002. He talked of extending the life of a design object by allowing the owner to configure it to fit their current needs. These are the examples he showed:



Vignelli's owner configurable objects.

- A table that "came with no instructions" as to how to place the legs, leaving it up to the user.
- A department store display modules that could be re-arranged to fit the season's merchandise.

As I reflected on Vignelli's comments and examples, I asked, "Why can't we develop multimedia projects with a longer life span? Can we give the client hands on control of the content? Through design and development choices can we turn up the heat and melt multimedia? "

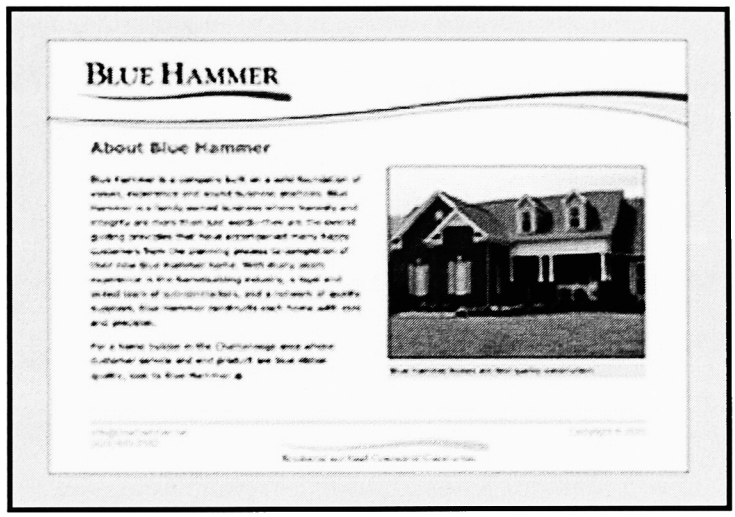
I want to not only push the multimedia development and workflow to be more future proof, but to also take the designer out of the cut and paste content maintenance loop.

Research

My idea was to let content creators and editors actually edit the content in a brochure style Flash site. Large Web sites have sophisticated, often customized, content management systems (CMS) that run from large databases of content. This is the basic idea I was looking for, but the scale of even the small systems is way too grand for a brochure style site. And besides, the cheapest database hosting packages exceed the hosting budget for a small scale site.

Macromedia offers a solution for HTML based sites. Their Contribute software is basically a $150 per installation Web browser with the ability to browse to a page and edit. A designer can build a site and lock down non-editable areas and styles, then give a password to content editors and they can change the editable content while the site is live.

I designed Blue Hammer, a Contribute site, as an experiment to learn how to design material other people can edit. I was immediately realized how the possibility of other people editing a page magnifies the importance of design rules, especially simplicity.



We'll look at more key design practices as they relate to my major project.

Process

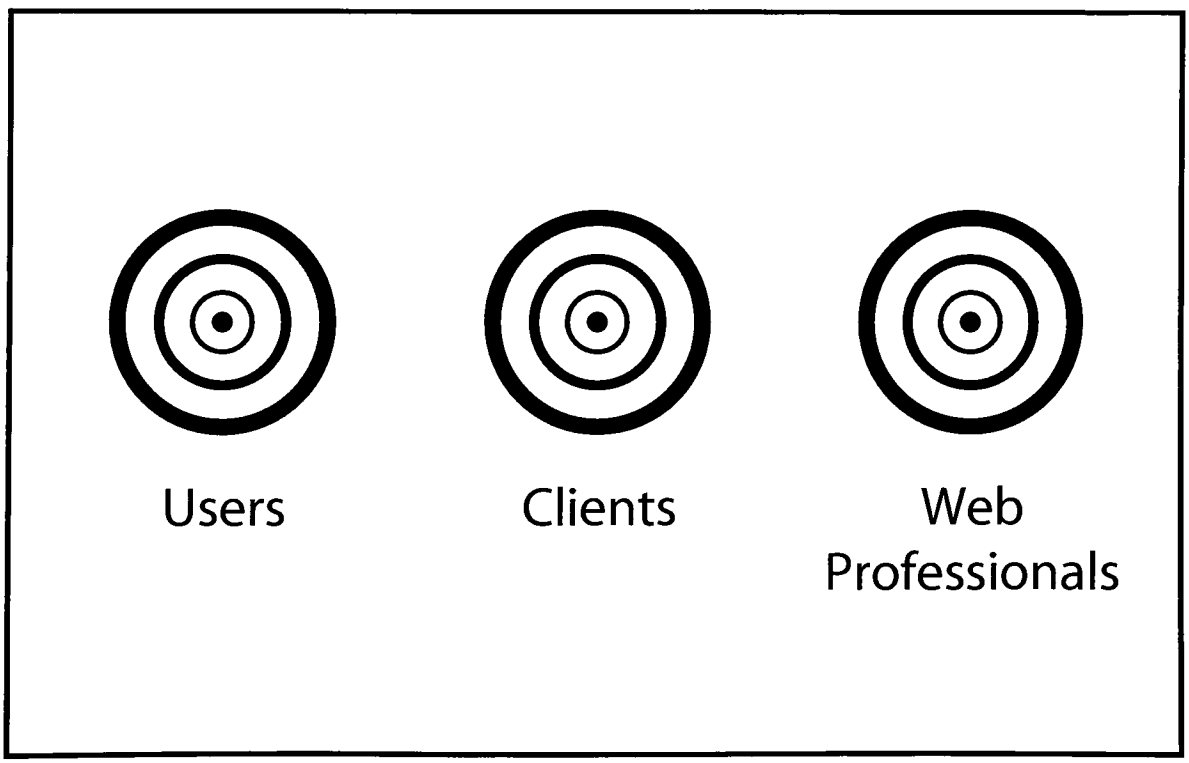
Major Project/Public Site

For my thesis project I chose to redesign the Web site for DLS Photography. The portion of the site I tackled with this project focuses on pre-sales qualifying of potential customers.

Reasons for Redesign:

- The original site had grown and its use has expanded beyond the original design's capabilities.
- The primary objective of the redesign is to improve customer bookings, management, and interaction. The secondary goal is to create a site that is easy to manage and update for the photographer.

Throughout this project I have tried to keep in mind 3 target audiences.



The first target audience is users or potential customers of DLS Photography. The biggest user group is brides looking for the right photographer for portraiture and wedding events. Brides using the Web site are usually young and educated. Families, couples, event organizers, and businesses are also important user groups. The desired action of a first time visitor is to schedule photography. The desired action of existing clients is to review and purchase photographs. (Note: This project focused on first time visitors.)

Reasons customers choose DLS Photography:

- High quality product
- They trust David's personality
- Value
- Style
- Availability

Qualifying the client is crucial for a process like this. The client:

- is computer savvy, but doesn't have Flash or HTML development skills.
- does not have a large Web budget
- needs a small scale/brochure style Web site
- has an up-to-date computer with broadband (Broadband is not a requirement for the project to function, however it makes maintaining an image heavy site much easier.)

The final target audience is Web and multimedia professionals, especially those interested in pushing for a future-proof workflow and easier to manage clients.

The next few sections will look at key structural, design, and technical aspects in how to melt multimedia.

Design

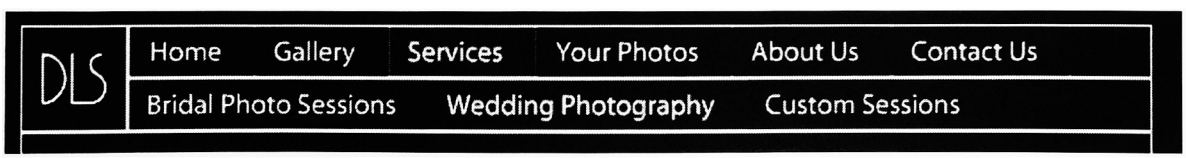
Because the project was a redesign there were interesting design considerations. The client had significant investment in the identity and brand exhibited in the previous site. It was important to maintain that identity, while taking steps to improve the navigation, user experience, speed and content display in the site. Technology has improved since the first release of the site, and the new site takes advantage of Macromedia Flash software to improve the user experience. Computers have also improved, which allowed an increased use of screen real estate. A complaint often received by the client is that images on the Web are too small. So we decided to base the site design around large 640 x 480 px display images. The client said ,"larger images are more powerful and people like to buy large images."

Because the large images were important, certain compromises to accommodate the file sizes and minimize bandwidth needs were required. To keep the project light weight for Web delivery, I chose to develop as much as possible of the site look and feel with code. For example, all of the buttons in the project are created completely with code. There are no buttons in the library or on the stage, they only exist when the file is compiled and the code generates them. Also a module based design structure was developed to deliver or download content to the user's browser only on demand.

Together, the rectangular images, and rectangular text boxes used for the code generated buttons, encouraged the continuation of the modern design choices of the previous site.

Separation

Separation of content from display structure, a guiding principle of CSS implementation, is also key to melting multimedia. In this project, it begins with the navigation. Each navigation button is generated based on content from a text file. Delete or add an entry in the text file and a button is dynamically removed or created. The SWF does not have to be recompiled.



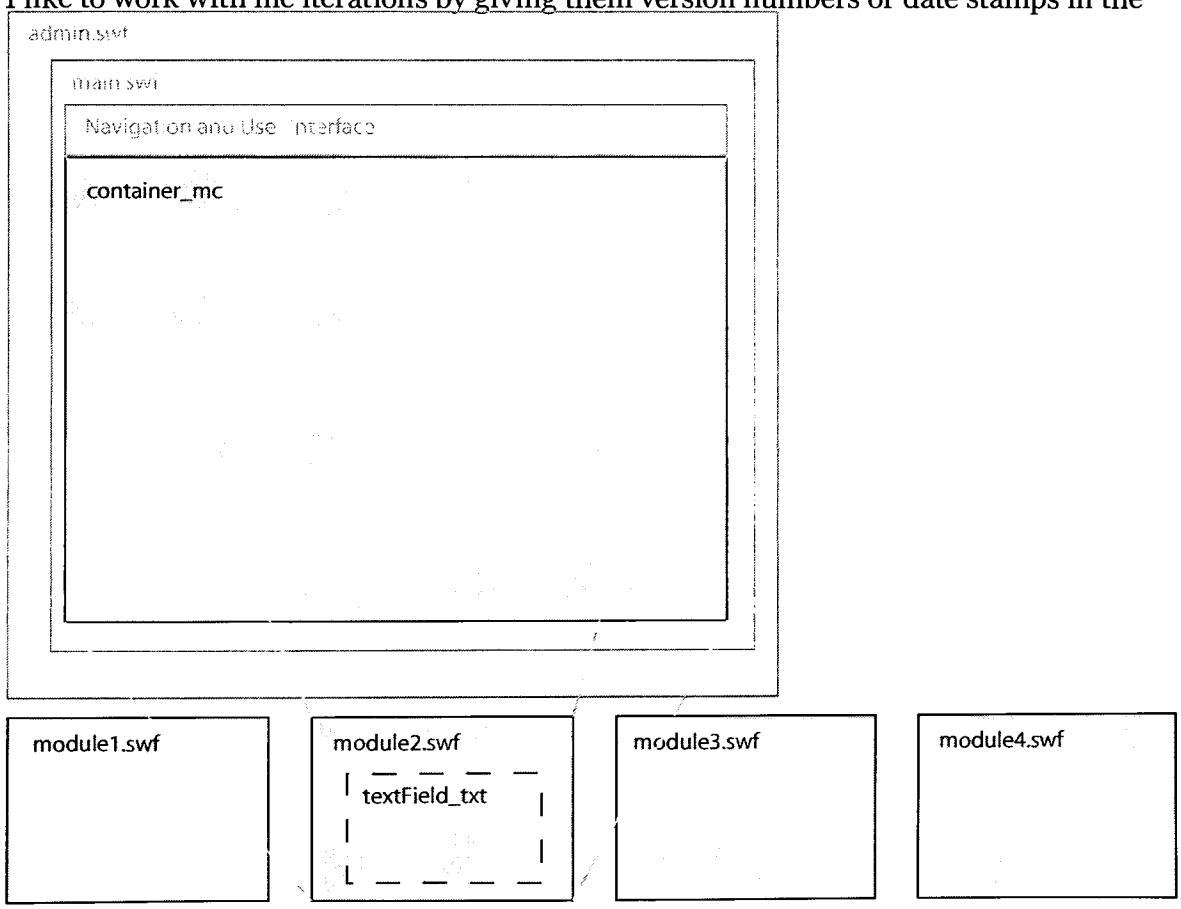
Code

Code is key, the more we can build and especially apply styles with code, the more flexibility we have to make quick project-wide changes in the future. To keep styles consistent across the project, I used an external *.as file to store: color definitions, text formats and frequently used variables. If someone changes her mind and wants to freshen-up the site with a new highlight color, every element created with code can be updated in one place.

Modularization

This site is module based, which makes it easy to update just a single section of the site without having to wade through the entire project. It also helps to keep the site fast, by delivering content to the user's browser only on demand.

I like to work with file iterations by giving them version numbers or date stamps in the



file name, so I can go back in the project history if I break something. This practice of file name iteration numbers cannot be followed in a module based project, because all the files are so interrelated the project must be developed with files that follow a strict naming convention.

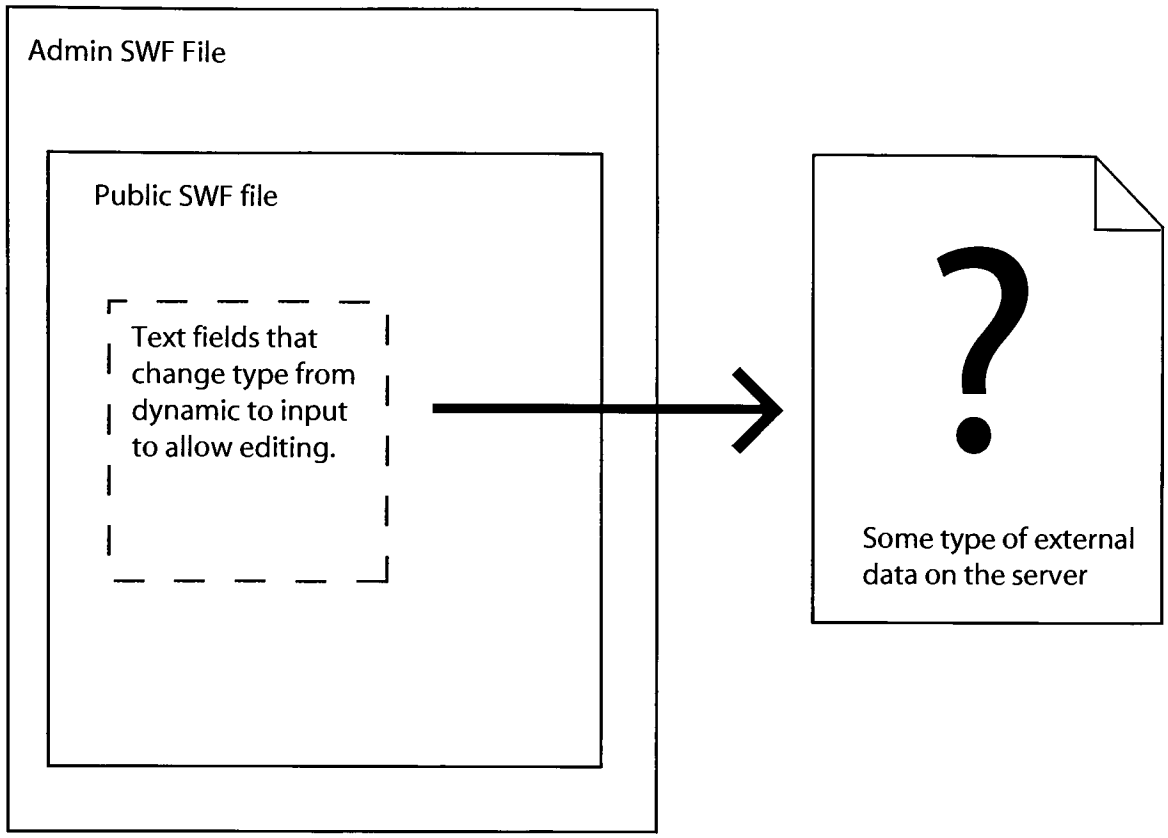
Naming Convention

This type of project requires strict relationships and naming conventions to be efficient. I will elaborate on how developing and following a naming convention is even more important as we move into the administration side of this project.

Administration

How do we relinquish content control, maintain design control and let the client edit content?

My concept to give clients a key to their content is simple. First change the dynamic text fields in the Flash file to input text, second, make edits, then click save! This sounded simple enough, but in implementation "save" was a major problem. For security reasons Flash SWFs are not able to write a file. The next time the SWF is loaded it will still contain the original data. So we have a problem. How do we create persistent data–data that retains its state for the next user. We must somehow preserve these changes for other users to view. We need a method to write the edited text to the server so it can be read back by future user sessions.



Macromedia would like us to solve this problem by purchasing a whole suite of additional products, including Flash Remoting, and a server running a ColdFusion Database. For larger sites with larger budgets this solution or a similar one is appealing. For a small site, it's a concept killer.

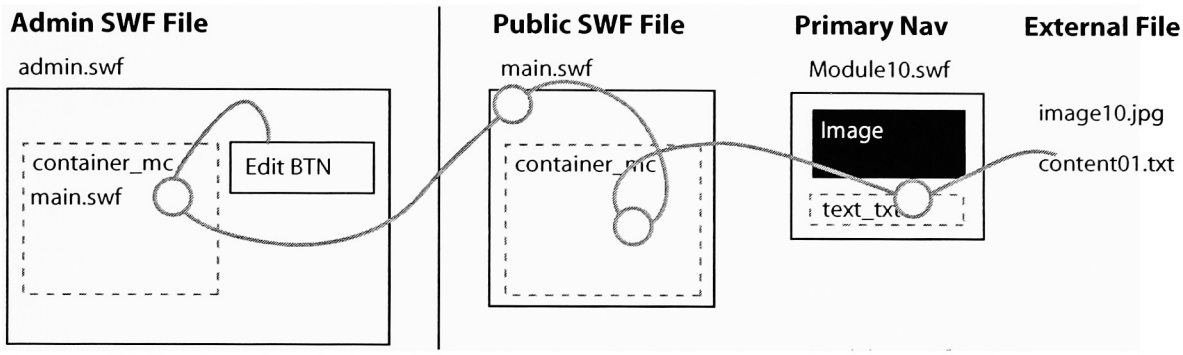
Fortunately, the open source world offers PHP (PHP: Hypertext Preprocessor) which is a general purpose scripting language designed for the development of dynamically generated HTML pages. PHP has many advantages, including:

- PHP can read and write text files on the server
- PHP can communicate with Flash via the standard methods of GET and POST
- PHP is often deployed in low cost hosting packages

Client Interaction

To allow only the client to make changes to the public site, we must provide an administration SWF that loads the public site and gives editing capabilities. The administration URL is an unpublished URL, protected by a password.

Inside the administration site, the client can browse to a section of the site he would like to edit and make an adjustment. In our example, David might need to update his prices for the 2004 wedding season. He just browses to the pricing page, clicks edit to highlight editable fields, makes his changes and clicks save.



How it Works – Naming Conventions and Relative Paths

The administration's functions work by changing the properties of the text fields located in the public file. In order to keep the administration functions and ActionScript off of the public files it is important to maintain naming conventions to ease targeting. Because there are so many levels of embedded content, relative target paths are essential. And it is important to keep a global project perspective when programming. For example, once a SWF is embedded on a lower level, _root in its ActionScript suddenly refers to the highest level document. So relative paths are best. In my project, there are buttons in the content modules that need to target the container movieclip in the main public file. In a normal project it would be easiest to write these paths using _root to target the main public file and then target the container movieclip. However, in this situation the buttons would work fine when the main public file is viewed alone, but because the admin file loads the main public file for editing, the buttons would not work in the administration section. Once loaded into another level in the admin file, the _root path on the module button refers to the admin file and not the main public file. This renders the button use-

less. To avoid this type of path confusion it is best to avoid the use of _root and instead use _parent to target objects in higher levels.

```
gotoEmailBTN_mc.onRelease = function(){
    _parent.container_mc.loadMovie("M61.SWF");
}
```

This section of code shows the relative path to load a new module.

To illustrate how important naming conventions are, lets look at the process the edit button in the admin file must follow to correctly target text fields several layers deep. Like all the other buttons in this project, the edit button is generated completely from code, using text fields and text field backgrounds and borders for the button colors. The edit button works because all the content containers in each module have the same instance name. So it does not matter which module is loaded, since all the names are the same, the same path in the edit button's function will still work. When clicked, it drills down through the project and adjusts the properties of editable text fields in the currently loaded module. Most importantly, it changes the type from dynamic to input. The save button works in a similar way. It follows the same path to the currently loaded text field, to capture the content and send it to a PHP script.

```
container_mc.container_mc.content1_txt.background = true;
container_mc.container_mc.content1_txt.backgroundColor = color7;
container_mc.container_mc.content1_txt.selectable = true;
container_mc.container_mc.content1_txt.type = "input";
```

This section of code shows the paths required to for the edit button to prepare the module content for editing. A code fragment showing the complete functions of the adminstration edit and save buttons are in Apendix 1.

The PHP script for saving the content to a text file is included with detailed comments in Appendix 2. The script is divided into three sections. The first section of the code catches the variables sent by Flash and assigns them to PHP variables. The next section opens the file name sent by Flash, writes the content entered in the text field, and closes the file. Before discarding the variables, the script assembles and sends an email message to the site owner confirming the edits just made. Since the email contains the edits just made the email also serves as a redundant copy of the site content on the server.

Problems Encountered and Solved

This project was my first Flash Web site, so I encountered many problems that required learning the best practices for developing Flash content for the Web. One difficulty was learning how to load SWF and jpeg files into movieclips in order to only deliver content to the user's browser on demand. This was especially important in a site containing a lot of images. Loading the entire site content at once would have required significant load times even with a broadband connection. I learned to design precisely to the pixel, because the external files must match the dimensions of container movieclip or they will be stretched.

I discovered instance names and file names work best when they begin with an alpha character and do not include hyphens. Otherwise, in some situations Flash behaved as if the hyphen was indicating the mathematical subtraction function or handled names beginning with leading numbers or numbers padded with zeros as a number and not an instance or file name. This created a number of situation where a number had to be sliced out of the middle of variables which was slightly more ActionScript gymnastics than if it were just at the beginning of the variable or instance name. The number often was included at the end of the instance names but still required extra slicing because of the ActionScript naming conventions of amending instance names with tags such as _mc and _txt to help with code clarity and allow Flash to provide code hints when in development mode.

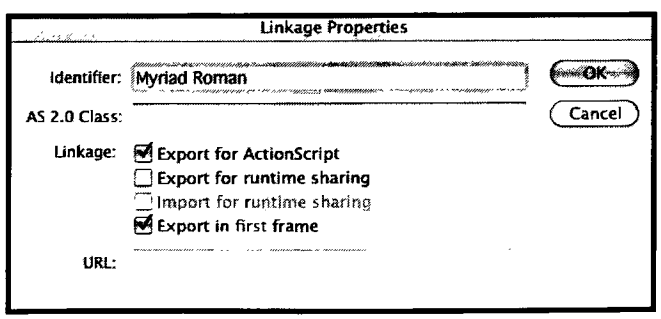
Learning to deal with external text files was also critical. At first I thought it would be possible to keep all the text content of the site internally, and store it in an array or some sort of variable structure. But later I learned that once Flash compiles a SWF file it can-

not be modified. So the requirement for an external storage method became apparent. This was further frustrated by the inability of Flash to save or write files. Searching for a solution to this problem of creating editable persistent data was especially challenging. Most people or resources referred me to methods that would only save content on the users machine or methods for saving content to the server that were very cost prohibitive. As I discussed above, fortunately I discovered PHP could fill these content storage requirements.

PHP presented its own new challenges. First of all it was a new scripting language to me, fortunately, the learning curve was made easier by its similar syntax to ActionScript. The development and testing workflow was challenging because the only machine I had to test on was a remote server. I had to upload the files every time I made a change, then test. When Flash calls a PHP script there is no method for showing errors. So, when I broke something, it just didn't work and I had to guess until I figured out what might be the problem. There were several people on Internet message boards that helped me through problems. The most helpful board I found is part of the Web site kirupa.com.

Since I used virtually all dynamically created text boxes, ensuring that the specific font I chose appeared required a complicated process. With static text fields this is not an issue because Flash automatically embeds the appropriate font, however, with dynamically created text fields Flash does not handle Font embedding automatically. Mimi Netzel, a fellow student, showed me a three step process for ensuring that dynamically embedded fonts work when published.

1. First, the font must be added to the library.
2. In the library the linkage must be set to export for actionScript.



3. Each time a text field is created the font must be told to be embedded with actionScript.

```
embedFonts = true;
```

Learning to target objects in different SWF files took some time. There is documentation suggesting levels of SWF files can be referred to and managed using the _level2 syntax. I found that creating or placing an empty movieclip on the stage then loading SWFs into it was more reliable from a targeting standpoint. It also enabled me to ensure dynamic content loaded on the appropriate level, not in front of graphical user interface elements.

I used text fields to create all the buttons. This was very helpful in keeping the file size low. However, I ran into a problem when trying to vertically position the text inside the text field. I formatted the text using the ActionScript textFormat object which has no controls for vertical spacing or padding of text inside a text field. I tried setting negative and positive leading values with no luck. My final work-around was just to create an additional text field on-top of the button background text field. This only required a few extra lines of code and allowed me to position the text wherever I liked via ActionScript.

Limitations

Here are some of the limitations of this type of project work flow.

- It's easy to publish information without it being proofed.
- Any time you're opening/providing access to files, you are opening a security hole.
- It is easy for the client to edit the content, so it's easy for them to damage the content. And the newest content resides in one place. On the server. (With the exception of the email back-up.)
- The concept is designed for small-scale sites. Managing a large site

with this type of system would be overwhelming.

Benefits

I have become so invested in this type of work flow, I can not resist reiterating some of the benefits.

- Keeping the content fresh is not the designer's problem. This thesis project grew, in part, out of a desire to be more efficient. One of the ways to be more efficient is to eliminate processes. Through redistribution of the work load the designer is only responsible for design, no cut-and-paste updates, while the content editors actually control their content.
- Since everything is modular, it is possible to quickly take a model from one project and apply it to another project. For example, the mail form has standard features, which can quickly be implemented in other projects with minor adjustments.
- Because, as much as possible, the content is separated from the display structure, future redesign is easier.
- The cost of updating content is cheap and is not locked down to one computer. The client does not have to pay a designer for updates, and unlike editing an HTML site with Macromedia's $150 per installation Contribute software, this process is accessible from any Internet connected Web browser.
- All of the benefits above Contribute to giving projects a longer useful life span.

Conclusion

In my thesis project, I have looked at solutions for retaining design control while releasing content control back to the client. And through this report I have reviewed some of the design methods and construction practices I used for developing unfrozen, small-scale, multi-media projects. I look forward to continually pushing the concept forward. I think some of the avenues for accomplishing this include; developing richer methods for editing content and style, simulating fresh content with randomness, harnessing data

feeds from other content providers, and harnessing other technologies such as blogging software to emulate a content management system. This thesis process has encouraged me, and gives me excitement and hope for the future of multimedia. This work to melt multimedia and push for future-proof, efficient work flows has become and will continue to be a life-long passion.

Appendix

Appendix 1: Administration Edit and Save Button Code Fragment

```
editBTN_mc.onRelease = function() {
    // assemble the text field name based on
    // the module number
    var modNumTxt = DLSmodNum + "content1.txt";
    // target the content of the current module
    // highlight the text field by changing the background color
    // allow the text to be selected
    // change the textfield type to input
    container_mc.container_mc.content1_txt.background = true;
    container_mc.container_mc.content1_txt.backgroundColor = color7;
    container_mc.container_mc.content1_txt.selectable = true;
    container_mc.container_mc.content1_txt.type = "input";
    // ask for a submit/save button
    // the edit button calls a button generation script
    // and specifies the functions of the save button
    BTNName = "submitBTN";
    BTNDepth = 2;
    BTNText = "Save";
    BTNx = 680;
    BTNy = 130;
    createBTN(BTNName, BTNDepth, BTNText, BTNx, BTNy);
    // specification of save button tasks
    submitBTN_mc.onRelease = function() {
    sendlv = new LoadVars();
    loadlv = new LoadVars();
    sendlv.sendVars = "&content1="+container_mc.container_mc.content1_txt.text;
    sendlv.moduleText = modNumTxt;
    // the save button returns the text field to it's
    // previous state
```

```
        container_mc.container_mc.content1_txt.selectable = false;
        container_mc.container_mc.content1_txt.type = "dynamic";
        container_mc.container_mc.content1_txt.background = false;
        container_mc.container_mc.content1_txt.selectable = true;
        // Finally, the save button sends the text to the PHP script
        sendlv.sendAndLoad("write2text.php", loadlv, "post");
        submitBTN_mc.removeMovieClip();
    };
};
```

Appendix 2: PHP Code

```
<?
// Listen to what Flash had to say
// When the admin file calls this PHP file, it includes the variables sendVars
// and moduleText. The variable sendVars contains the edited content. The varible
// moduleText contains the name of the text file to write the content to.
// This section of code captures those variables from POST and makes them
// PHP variables for use later in the script.
$sendVars = $_POST['sendVars'];
$moduleText = $_POST['moduleText'];
$file = $moduleText;
//
// This section opens the appropriate text file and writes the content
// and closes the text file.
$fp = fopen($file, "w");
fwrite($fp, $sendVars);
fclose($fp);
//
// While we have all this information, it's easy to prepare an email message to
// send an edit confirmation message to the site owner.
$sendTo = "gpm9103@rit.edu";
//
$headers = "From: Edit Confirmation";
$headers .= "<gpm9103@rit.edu>\r\n";
$headers .= "Reply-To: gpm9103@rit.edu";
//
$subject = "Web Site Edit Confirmation          ";
$subject .= $moduleText;
//
$message = "This message confirms these edits made to your site: ";
$message .= "\r\n\r\n";
```

```
$message .= $sendVars;
//
// This calls the PHP mail function and sends the message with the variables
// that we built above.
mail($sendTo, $subject, $message, $headers);
?>
```

Appendix 3: Frequently Asked Questions

Though out this process I've kept a record of questions people have asked me as I've talked about or shown my project. I have included them here with my answers to provide addition clarification.

Q. Does this put designers out of a job?
A. Many designers have suggested they favor FROZEN multimedia projects because they have many opportunities to bill clients every time they want a change. And they suggest that if my thesis idea catches on it will put them out of work.

This approach has many advantages for the designer as well as the client. It allows the designer to reach a wider market by servicing more clients and allowing them to do more of the content work. It relieves designers from content "emergencies" and allows clients to make updates on their schedules. The clients also benefit due to the lower cost of updates, therefore they can afford to keep their content fresh, which even benefits users.

Q. Are there not Web sites that do this now?
A. I am not aware of any. There are some Web sites, like http://www.balthaser.com, that are basically Flash built with Flash, which a designer can use without the input and constraints of a trained designer. My solution results in a much more professional product and still requires that a designer do the content evaluation, information architecture, branding, design, style development and construction of the site.

Q. Is this basically a template?
A. My project is a small-scale, low cost, content management system that takes advantage of template concepts to implement the system. A template or boilerplate is a frame or container into which content is poured. What I've built is similar, with a distinction that I know basi-

cally what the content is as I'm designing. My system, which only requires basic browser software–no design software for the client, is more sophisticated than just a set of templates handed over to a client.

Q. Is it worth the time and effort it takes to design this way?
A. That's an interesting question. Recently, I did another small flash site. Due to time constraints imposed by the deadline and the demands of various professors, I very much needed to push the site out the door in a hurry. So I did not bother with some of these methods like building as much as possible of the graphical elements with code. And I did not develop an admin tool. However, I did use the same component based practices I've used developing this site. I also tried to keep content separate from the presentation with external images and text. I think it will be somewhat easier for me to update the site in the future thanks to this. Once a workflow is developed, taking a large chunk of code from one site and implementing it on another site speeds the process along.

Q. What did you learn?
A. I learned an amazing amount. Here are a few of the highlights:

- I learned about Flash and ActionScript. Prior to beginning work on this project my experience with Flash and ActionScript was very limited.
- Design rules are exceptionally important. Simplicity is essential. Both organization and placement of off nav navigation is critical.
- I was disappointed to realize that 'dynamicness' results in a less *Flashy* product–transitions, animations, and special effects are more difficult to achieve with dynamic information.

Q. What is a brochure style site?

A. A brochure site, is an electronic version of a brochure you might be handed to learn more about a product or service, no fancy features, just flashy images, and content.

Q. How does this solution compare to Contribute?

A. Contribute has a richer content editing environment. A significant advantage my project has over Contribute is that it is not locked down to specific computers. Any Web browser is a potential edit point.

Appendix 4: User Testing Results

The user testing process has been very interesting and I have received some very valuable feedback. One of the major items I discovered during user testing was that even though the files are generated for the latest Flash Player, earlier players will play them with almost no perceivable problem until on a sub-level of the site the text in dynamically created text fields does not wrap. It just continues on off the edge of the screen. I likely would have never discovered that I needed to require Flash Player 7 for this project without doing user testing. I had tested it on Flash Player 6, but did not notice the error because everything appeared to be functioning correctly.

A sharp tester pointed out an inconsistency in interface labeling: I referred to the same module as the Email Form and as the Mail Form.

Evaluations were mostly very generous. Some users scored the site speed as slow even though they were using a broadband connection. I wonder if the server was not performing well when they looked at the site. Possibly the images could be compressed some more, and I could probably enhance the preloading process to improve the user experience.

Here are some of the comments from testers:

> I love the pictures, and the design does a good job of enhancing the photos while giving the info needed. It was easy to navigate around.
>
> So the red is overwhelming on the admin site, I like red, but it takes away from some of the pictures. It dominates the field of vision. I like the black and white better...or maybe gray or even a dark burgundy, a classic color. His pictures are classic, and that red is very modern.
>
> And that is one BIG lock. What you need is a black and white photo of a really cool lock. That would totally fit with your theme.

—

> Everything looks great. I like the ability to edit text directly on the web. That's pretty neat.

—

> Looks Great,

The copy switched fonts under the Services / Custom Sessions section
The About Us Section seems like it could have had more information

–

Admin:

- Lock icon is HUGE!!!
- Red is very overpowering, though it is clear that you are not on the real site. If you could leave the background of the SWF black and just make the HTML background red, that would be better.
- It might be nice to have a logout button or something like that to ensure that users know how to get out of the editing section.

–

The site itself is great. I love your use of flash–the appropriate balance between fancy enough and overdone. I'm not sure about a couple of the photo selections, but I guess that's more his area than yours

Resources

Choi, Wankyu, et al. Beginning PHP4. Birmingham, UK: Wrox P, 2000. A comprehensive PHP guide.

Useful for examples and discussion of PHP.

Cotler, Emily, and Kelly Goto. Web ReDesign | Workflow that Works. Indianapolis: New Riders, 2002.

Used for managing redesign process. An excellent book.

Design: Vignelli. Comp. Michael Bierut, and Massimo Vignelli. Ed. Joan Ockman. Milan, Italy: Comune di Milano, Ripartizione Cultura/Raccolte d'Arte, 1981.

Contains photographs of Vignelli's work.

Lash, David A. The Web Wizard's Guide to PHP. New York: Addison Wesley, 2003.

Used for examples of PHP syntax.

Lowery, Joseph. Roadmap to Macromedia Contribute. Indianapolis: Macromedia P in association with New Riders, 2003.

A basic how to use Contribute book. Also useful for learning to design structures that allow others to edit the content.

Macromedia - Flash Developer Center : PHP Articles. Macromedia, Inc. <http://www.macromedia.com/devnet/mx/flash/php.html>.

A good starting place for learning about PHP and Flash interaction.

Negrino, Tom. Macromedia Contribute 2 for Windows and Macintosh. Visual Quickstart Guide. Berkeley, CA: Peachpit P, 2004.

A Contribute how-to book. Also has disccusion about design practices when others are editing the content.

Vash. kirupaForum - Sending Variables to PHP from Flash And Back again! Kirupa.com. 29 Jan. 2004 <http://www.kirupaforum.com/forums/showthread.php?s=&threadid=17604>.

Part of this tutorial was especially useful for learning to connect Flash and PHP inorder to save files.

Webster, Steve. Foundation PHP for Flash. Birmingham, UK: friends of ED, 2001.

Has examples of how to make Flash communicate with PHP.

Thesis Proposal for Master of Fine Arts Degree

Rochester Institute of Technology
College of Imaging Arts and Sciences
School of Design
Computer Graphics Design

Title	Melting Multimedia: A study of methods and processes for developing small-scale dynamic multimedia systems that can be easily updated by the client.
Submitted by	Paul Martin
Date	20 May 2004

Thesis Committee Approval

Chief Adviser: Chris Jackson, Assistant Professor, Computer Graphics Design — Date

Associate Adviser: Jim Ver Hague, Professor, Computer Graphics Design — Date

Associate Adviser: Nancy Ciolek, Associate Professor, Computer Graphics Design — Date

School of Design Approval

Chairperson: Patti Lachance, Associate Professor, School of Design — Date

Paul Martin
Melting Multimedia
Computer Graphics Design
Thesis Package