Thom Hines
Mini Thesis
Yury Gitman
May 16, 2011

Tinker: Thoughts and Reflections

In this paper, I would like to introduce and discuss the process behind my first year mini-thesis project, *Tinker*. Tinker, in my intention, was built to be the easiest way to edit a website. Tinker is a Content Management System (CMS), similar to WordPress or Drupal, that allows site owners and administrators the ability to quickly and easily change the content on their sites. What sets Tinker apart from these other systems, though, is that it is intended for smaller, less complex sites, and this distinction allows for Tinker to have a lot of affordances that dynamic, database-driven CMSs can't. Even compared to the already high standards of modern CMSs, Tinker has a much easier installation process, an innovative system for handling web site files and versions, and most importantly, an incredibly transparent interface for viewing and altering web pages.

Summary

As somebody who has been designing and developing web sites for a number of years, I have used many tools and technologies to make my clients' websites better and more useful, while, as much as possible, making the development cycle more streamlined and efficient for me. When I first started working on the web professionally, I made sites like most novice site designers do: by creating simple, vanilla, static HTML pages. I had no idea about other possibilities at the time, and so for somebody who was interested in web development but was trying to teach himself the skills at home, it worked well enough to produce and launch several websites, and so I kept making them. Unfortunately, a few months after every launch, it seemed to be very common that I would get emails and calls asking me to update my client's website. Quite often it was simple to make the changes and didn't take too much time, but occasionally the updates would take hours and distract me from the work I actually wanted to do. After a couple years of this, I was spending almost as much time updating sites as I working on the sites of my current clients.

So, in order to stop this tedious cycle, I taught myself server-side scripting (eg. PHP) and how to developer for a database (eg. MySQL), so that I could create a custom-built CMS for my clients. After that, when I launched a site, my clients could manage their site content without me, which freed me up considerably from frequent site maintanance, and it gave them a deeper sense of ownership and independence. This, in turn, made them more satisfied with my work. Pretty soon I was incorporating this sort of functionality into every site I made.
It was shortly after this that other pre-built blogging platforms, like WordPress and ExpressionEngine, were starting to become more common. While not initially designed to be a CMS, these platforms had ability to change web site content from within an administrative page

- as opposed to accessing the files directly on the server - and this proved to be really attractive to a lot of developers, including myself. And using a pre-built system was much easier and faster than building a custom CMS for each of my clients. Since that time, scores of CMSs have been developed, and each have their focus and advantages.

While these CMSs were a huge step forward, their creators obviously had to make a lot of choices as to what features were most important and who their target audience was going to be. Most are perfect for sites that need to be able to scale to large sizes and need a lot of control over the *creation* of pages and new content, not just the modification of it. For many people, including for most of my own clients, I would continue to suggest using one of these robust CMSs to manage their sites, since they do offer a lot of control and power, and in truth Tinker is not built to do everything. But developing for a CMS does take some extra time, knowledge and server infrastructure to get it up and running, which can add a considerable amount of work to simpler sites. Furthermore, since most CMS software is meant to be used in many different kinds of sites, the interfaces are somewhat generalized and have many more features and options than what the average site administrator will need to deal with on a daily basis, which I can say form experience can be confusing and frustrating for less experience clients.

Because there are very few applications out there for smaller sites, I decided to create what has become Tinker. Instead of worrying about features that make other CMSs work for handling larger sites (and which other CMSs have done much better than I could in one semester), the main focus of Tinker is to make it the simplest and least technical way for people to change the content on their websites.

The first thing that sets Tinker apart from other CMSs is its installation process. While the installation might only affect a site owner at the very beginning stages of managing a website, it can still be a big enough challenge in some CMSs that its not worth pursuing. And so every part of Tinker's installation is designed to be as simple as possible.

To integrate it into an existing site, all you have to do is upload the Tinker folder to your web server, then navigate to the folder in your web browser. Tinker will ask for you to pick a new user name and password, and then which of your webpages you would like it to install to. This last part is all point and click, as Tinker will automatically try to detect which files are eligible HTML files, and provide them in a list to pick and choose from. Other CMSs use databases in order to store site data, which takes a certain amount of server know-how, but Tinker uses a site's HTML files instead, requiring no extra work. After you've chosen your files, Tinker will be available on any of those pages the next time you view them in a browser.
The editor interface is unlike almost all of the CMS interfaces out there. Instead of a separate admin page, you make changes to sites in Tinker by clicking directly on the content on the site, and typing your changes there. It's as if your site has just become an editable file, like a Microsoft Word document, and everything looks and feels just like it did before Tinker was installed. There is a panel at the top with common layout and style options, such as bold, italic, alignment, etc. and buttons to create links and add images.

Saving changes to the document is as easy as hitting the ‘save’ button in the top right. Whatever is shown on your screen will be saved to the HTML file on the web server, and your changes will then be live to anybody else who goes to view your site. Every time you press the ‘save’ button, a version of your page is saved to the server, which you can call up at any time if you decide to revert back to an earlier stage. When you choose a version, it will automatically replace the page with what that version looked like so you can quickly and intuitively see all of your previous versions. Again, once you are satisfied with how your page looks, just hit the ‘save’ button again and your changes are live.

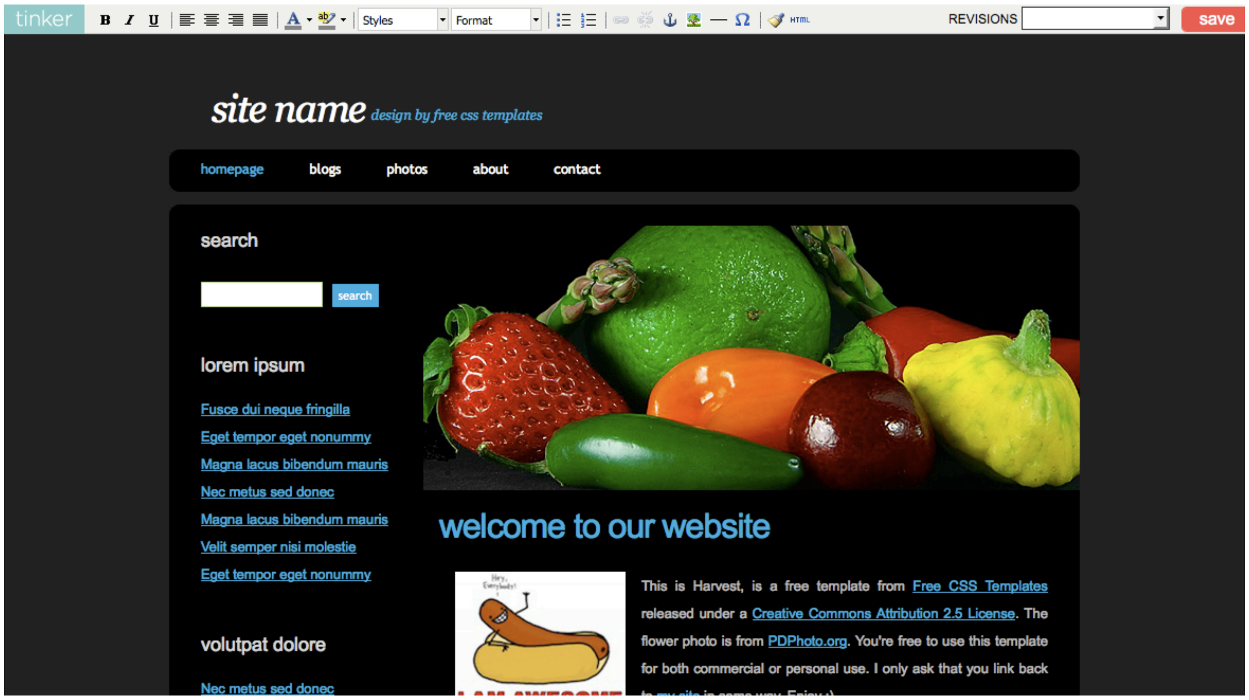
Also, in one of the most recent iterations of Tinker, a drafts system has been introduced to save you from losing data when your browser crashes or you accidentally close your window before your changes have been saved.

Without having to focus so much on more general features, like blog posts or page creation, Tinker is admittedly aimed at a more select demographic. My primary group of focus are web designers, especially those who are either new to web development or who have clients that have small-scale, static web sites. With Tinker, beginning web developers can essentially add complete CMS functionality to a site without having to know any code beyond basic HTML, and yet, even advanced users who may be adept at web development can save significant amounts of time since installing Tinker will usually take less than a minute from start to finish. Tinker requires no modifications to the site files or adjustments to the server, so getting the benefits of Tinker is virtually cost free to those who don’t need a wide array of features.

A secondary target audience might include site owners with already existing small, static sites. At this point, Tinker is much better suited to modifying content than it is in creating new pages or content areas. One of the main features that people have asked me about is the ability to build a site through Tinker, essentially make something from scratch, or at least give site owners the ability to have a basic template to start a new site from. Unfortunately, I haven’t found a way to introduce these abilities without causing more confusion for my core user, or without forcing them to learn more about the principles of web servers and file structures than they would otherwise need to. In the current version of Tinker, there are nearly no trade-offs for non-technical users, in that the average person can operate it without any prior knowledge of how the web works. Before I open up this ability, I want to find a way to make it as seamless and intuitive as the rest of the experience.

Behind the scenes, Tinker uses a combination of server-side (PHP) and client-side (JavaScript with the jQuery library) scripting languages to take input from the user in their browser, and control the HTML files on the server. When a user is logged in, Tinker injects a series of files into the webpage that load up the interface on top of the static web site in the browser. When the user clicks inside the main browser area, they are actually clicking into an iframe, or a page within a page. This iframe is generated by Tinker utilizing the open source text editor, TinyMCE. As the user makes changes, these changes are recorded in the interface on the user’s machine, and every minute or so this record is sent to the server to be stored in a draft file. A similar process happens when the user presses the ‘save’ button, at which time a version is saved into

Tinker and the live version of the HTML file is overwritten with the most recent content. Almost all of this work happens behind the scenes using AJAX (Asychronous JavaScript and XML), so it is fast, doesn't refresh the page, and doesn't interrupt the user's work flow.



This is Harvest, is a free template from Free CSS Templates released under a Creative Commons Attribution 2.5 License. The flower photo is from PDPhoto.org. You're free to use this template for both commercial or personal use. I only ask that you link back

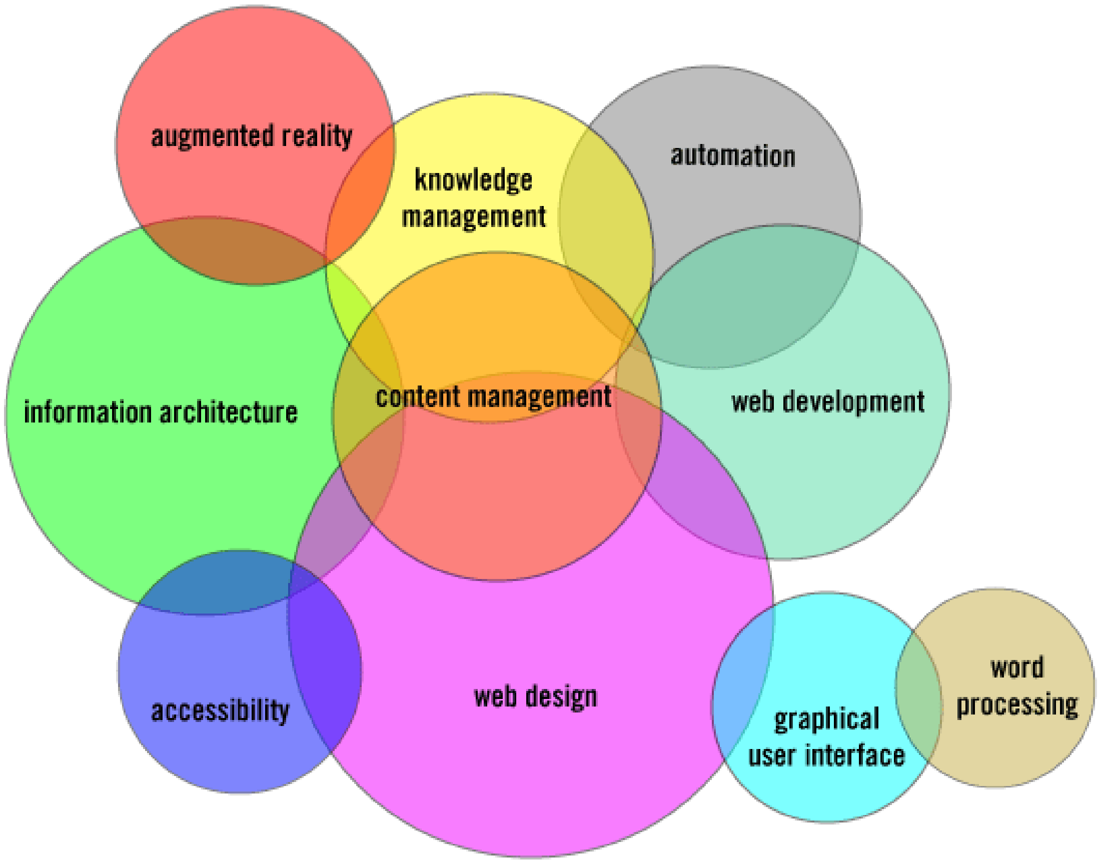
A screenshot of Tinker in use. The gray bar at the top is the only visual indicator that you are logged in, while the text in the main body of the page is selectable and editable.

Domains

The domains of experience and skill that Tinker crosses into are much more diverse than I had anticipated when I first started this project. I initially thought that it was strictly a web development project, as I was making Tinker to address problems I came across as a web developer. I saw that current CMSs had some shortcomings, and figured that by applying a bit of code, I would just be reducing the effort of editing, not adding to the experience.

Since then, I have learned that making something simpler is much more difficult than making some more complex, and I've had to reconsider a lot of choices and get a deeper understanding of how and websites are built the way they are in order to reduce them to their most base forms.

In addition to web design and development, I've also had to work in GUI design, content management (updating content), knowledge management (organizing content), automation, information architecture, accessibility, and have gained insight from augmented reality and word processing. There are definitely overlaps between many of these domains, especially the ones dealing with design or technology, but it wasn't until I started researching some of them that I was able to make more deliberate decisions regarding the entire system.



A visual map of project domains

For instance, the decision to use HTML files to store the site data was not just a usability issue, but a knowledge management issue as well. Where most CMSs rely on a database to store their site's content, which allows for rapid changes and lots of flexibility when mashing up the data, the setup process for a database can be tedious or difficult for some site owners. HTML, on the other hand, is already present anywhere a website exists, and from its earliest stages, was designed to store and format information effectively. Content and Knowledge Management aren't just about storing and retrieving data, but also about presenting it, and so in this way, the HTML is is the perfect container since it can do both. Another big part of usability is that an product or item work well and consistently. Static HTML loads much faster for the user and puts far less strain on a server, making it far less likely to crash.

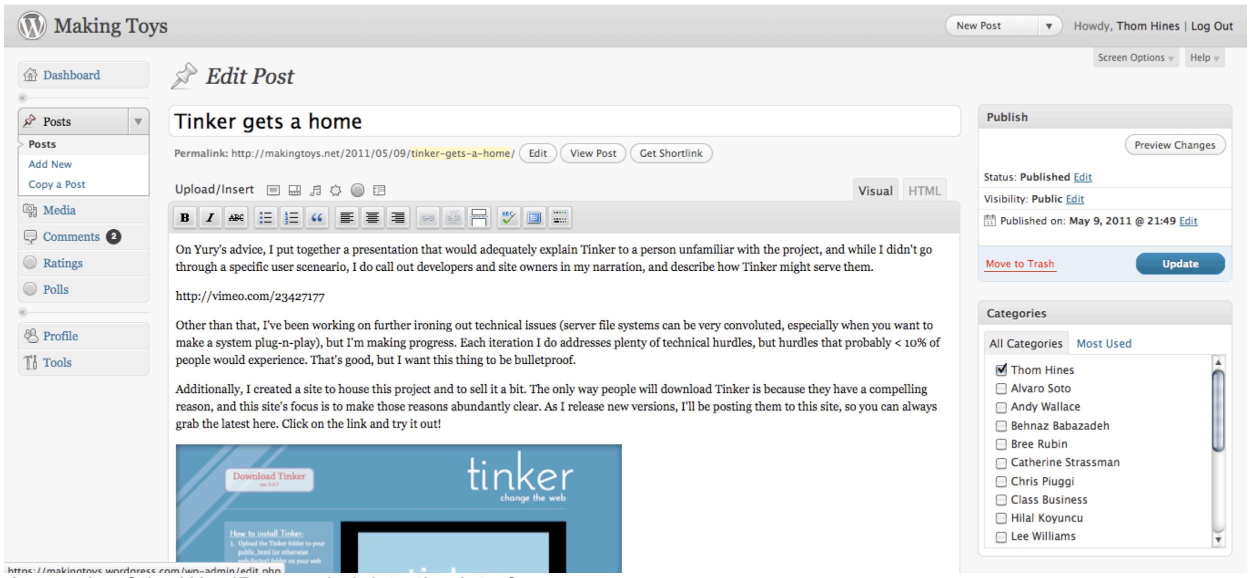
Furthermore, every decision to streamline the process of the interface eventually became a matter of automation, usuability, and information architecture. With each iteration, I had to find the points of friction for my test users. In the setup process, there was a lot of confusion about the complex file structure on the server that had to be configured before Tinker would work. At first I was hoping that this could be overcome with some simple instructions, as it didn't actually require much effort or time, but there were too many variables and I could find no way to give advice that would be useful to everyone that would assure the correct input. So, through a series of stages, Tinker moved to a completely point-and-click install, and by using typical file patterns and a small amount of searching, Tinker intelligently suggests which files or folders are most likely to be the ones you need.

On the front end, moving from pop-up windows to an inline content editor was a big switch, and probably the most profound conceptual change to my project. Part of what makes an interface more intuitive to users is that it removes the levels of abstraction between their actions and their goals. We've seen this over and over with the iPhone, as more apps succeed at convincing you that your finger is touching the content, not the app. It was with these thoughts that I realized that the entirety of tinker should exist as if it were an extension of the site, and as little as possible would be dealt with outside of that.

After all of the research I had been doing for my Augmented Reality (AR) app, NextWorld, I could see the promise of AR in nearly all modern interfaces, and especially so in Tinker. While there are many different kinds and uses of AR, to me, the purpose of AR is nothing more than the attempt to reduce the distance between reality and interface. Once I had changed the interface to its current orientation, it hit me that the implication in the interface was that user was no longer editing their site's content, but they were editing *the site itself*. Even though this doesn't change anything on a technical level, it was a huge paradigmatic shift, and so it guided every decision after that to maintain that feeling. From then on, the site itself became the tool, and Tinker was just the key to open the door that was already there. This realization was so big to me that every time I thought of making edits to my test site through Tinker, I couldn't help but forget about the HTML files and the javascript and just visualize that I was reaching through the screen and making changes to what I was seeing on my screen, as if they were block letters that I could pick up and rearrange. In fact, this is how I ended up with the name Tinker; the experience was so tactile and effortless and whimsical, it always felt like I was just playing around in the guts of my site.

<u>Precedents</u>

As I mentioned before, there are scores of CMSs in the market, and almost every feature of Tinker can be found in at least one of them. From a semi-automated install process to page versions, to flat-file storage, and even the in-page editing, there is little that other people or projects haven't implemented before. But every configuration I've found utilizes these features either poorly, or without regard for inexperienced users, which above all else, is the true point of Tinker.



Tinker gets a home

Permalink: http://makingtoys.net/2011/05/09/tinker-gets-a-home/ Edit View Post Get Shortlink

On Yury's advice, I put together a presentation that would adequately explain Tinker to a person unfamiliar with the project, and while I didn't go through a specific user sceneario, I do call out developers and site owners in my narration, and describe how Tinker might serve them.

http://vimeo.com/23427177

Other than that, I've been working on further ironing out technical issues (server file systems can be very convoluted, especially when you want to make a system plug-n-play), but I'm making progress. Each iteration I do addresses plenty of technical hurdles, but hurdles that probably < 10% of people would experience. That's good, but I want this thing to be bulletproof.

Additionally, I created a site to house this project and to sell it a bit. The only way people will download Tinker is because they have a compelling reason, and this site's focus is to make those reasons abundantly clear. As I release new versions, I'll be posting them to this site, so you can always grab the latest here. Click on the link and try it out!

Status: Published Edit
Visibility: Public Edit
Published on: May 9, 2011 @ 21:49 Edit

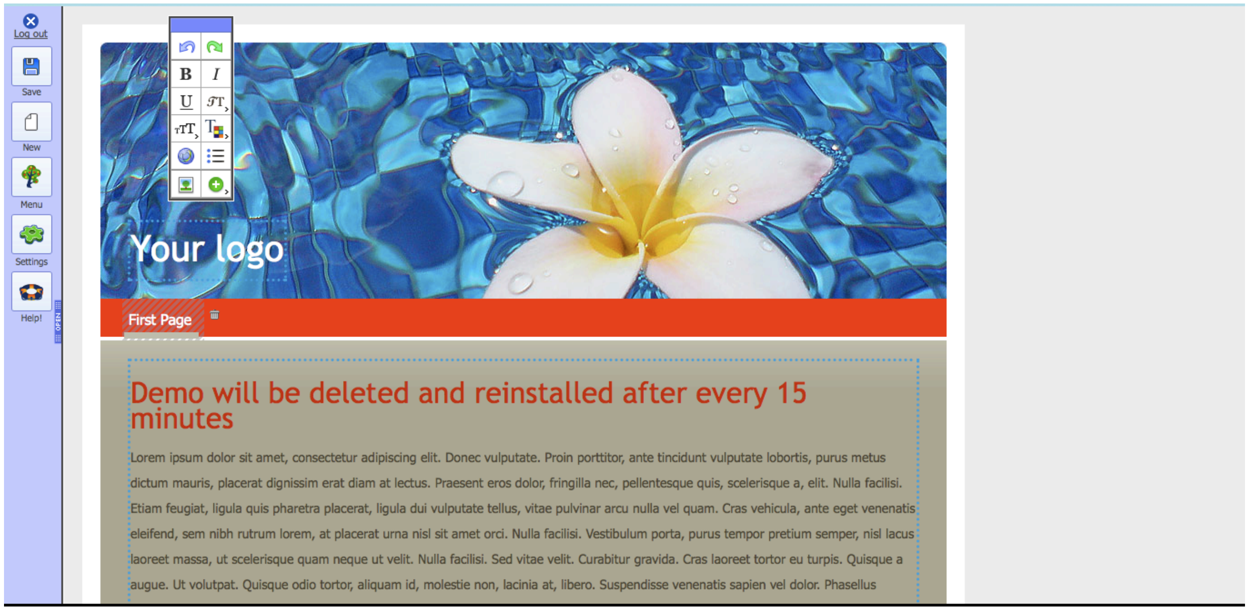
- [✓] Thom Hines
- [ ] Alvaro Soto
- [ ] Andy Wallace
- [ ] Behnaz Babazadeh
- [ ] Bree Rubin
- [ ] Catherine Strassman
- [ ] Chris Piuggi
- [ ] Class Business
- [ ] Hilal Koyuncu
- [ ] Lee Williams

A sample of the WordPress administrative interface.

Fortunately, to reach my goals, I have a lot of places to look for ideas and inspiration. While it is not the first CMS released for general usage, WordPress was the first that introduced me personally to the idea of utilizing code to build and update sites dynamically. WordPress stores data about every page, user, setting and whatever else is specific to a website in a database. This gives it the advantage of nearly limitless scale, and an incredible amount of flexibility in how data can be configured and presented in a website. It was also one of the first examples I saw of how the work of turning content into usable HTML could be automated through the use of server-side code. While some knowledge of a web server is necessary, pretty much anybody can run a fully-functioning site through WordPress without knowing any code at all. Many other CMSs, such as ExpressionEngine, Drupal, Joomla, Movable Type, and Frog CMS, have been developed that use a similar combination of HTML-based templates and database tables to serve as an all-in-one solution to creating new and fully-featured sites from scratch.

But many times a site developer might not want to start from scratch to make a site, and converting an already designed and developed site into one of the CMSs listed above can take a considerable investment of time and effort, especially for beginners. For these situations, other CMSs have been developed with site integration and modification in mind. One popular CMS that does this well is Concrete5. Concrete5 uses a database like the other CMSs, but uses standard HTML files as the basis of its templates. Also, in a similar vein as Tinker, changes to site content can be made right in the web site itself, as opposed to a dedicated, but separate, administrative page like WordPress. So, when a site owner wants to update content, they need only click on a page element, and if it is marked as editable in the template files, a popup will appear allowing them to edit the content. Both of these features remove some of the burden from the user by making the content more accessible. Thus, it is a more intuitive experience.

Another CMS that works well with pre-built sites is CushyCMS. To integrate CushyCMS with an existing site, all you have to do is add a small bit of text to the elements of your site you want to be able to change, and then set up an account with CushyCMS on their server. During this process, you give access to your site's files by giving them them your FTP user name and password. From there, every change you make to your site's content in the CushyCMS admin pages will be updated in the HTML files on your server, and immediately viewable to your users online. CushyCMS's use of a flat-file storage system (which uses text files instead of a database) is incredibly efficient and easy to understand, and was one of the main inspirations for me to work with straight HTML in Tinker. CushyCMS has a few drawbacks, however. It is a commercial service, which isn't bad per se, but is a hurdle for anybody who doesn't have much of a budget for managing sites. Also, all administrative work is done on CushyCMS's servers, which is one more level of abstraction between the content and the site, and its interface can become unwieldy if your pages have a lot of editable regions or elements. Still, it is a great service and has innovated content management in many ways.



Demo will be deleted and reinstalled after every 15 minutes

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec vulputate. Proin porttitor, ante tincidunt vulputate lobortis, purus metus dictum mauris, placerat dignissim erat diam at lectus. Praesent eros dolor, fringilla nec, pellentesque quis, scelerisque a, elit. Nulla facilisi. Etiam feugiat, ligula quis pharetra placerat, ligula dui vulputate tellus, vitae pulvinar arcu nulla vel quam. Cras vehicula, ante eget venenatis eleifend, sem nibh rutrum lorem, at placerat urna nisl sit amet orci. Nulla facilisi. Vestibulum porta, purus tempor pretium semper, nisl lacus laoreet massa, ut scelerisque quam neque ut velit. Nulla facilisi. Sed vitae velit. Curabitur gravida. Cras laoreet tortor eu turpis. Quisque a augue. Ut volutpat. Quisque odio tortor, aliquam id, molestie non, lacinia at, libero. Suspendisse venenatis sapien vel dolor. Phasellus

Zimplit, a CMS with an in-page content editor similar to Tinker.

The interface for Tinker, while thought of and created without knowledge of it, shares a lot of commonality with another CMS, Zimplit. Similar to Tinker, Zimplit uses an iframe overlay in order to make changes to a site's content in a way that mimics editing the page directly. You still make the changes in an administrative area, but you get a more complete idea of how your content will appear to end users as you make changes. This type of interface is a style of content management called "What You See is What You Get" (WYSIWYG). WYSWIYG editors gives users a much more intuitive way to interact with their site, and a lot more accuracy when trying to manage the appearance of content without having to resort to several rounds of trial and error. While Zimplit is much more limited in its use of templates and file management, it is better than Tinker at setting up a new site and does not rely on a pre-built site to work. Zimplit has a few other advanced features that I plan to improve upon and integrate into Tinker over time, as well.

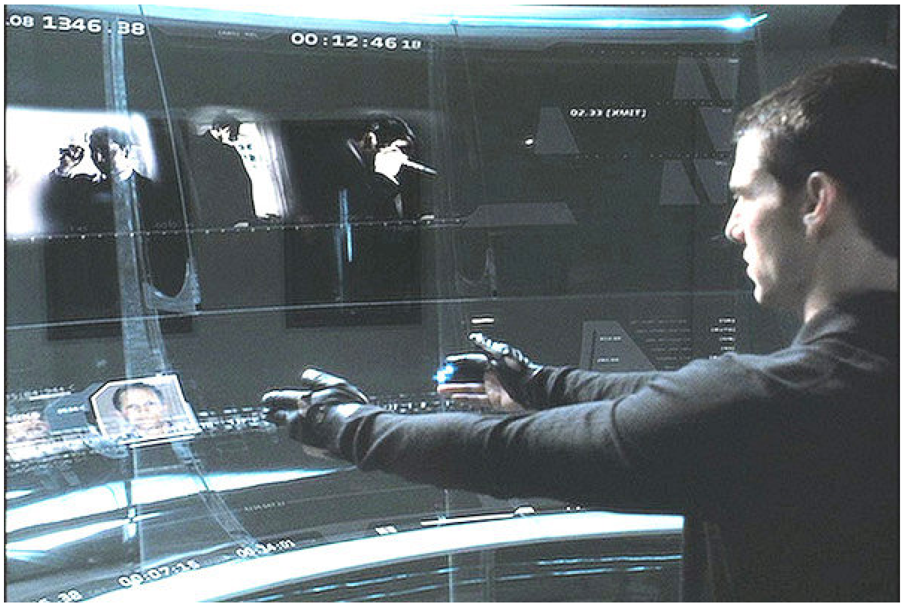
As opposed to my work on the underlying technologies, when I started considering interfaces, my research led more further from the internet and more into the realm of science fiction and design theory. At first, when considering making a CMS that is simpler to integrate and use, most of my ideas revolved around infrastructure. However, the more I tested and the more I thought about it, the more I realized that nearly *everything* in the CMS could be, and should be, simplified. In early versions of my CMS, before it was even named Tinker, I would use popups and in-window alerts to allow the user to edit content, much like Concrete5 handles editing content. This was easier to understand than a separate admin page like most CMSs, but at some point I realized that every time I clicked on an element I wanted to change, the exact same content would appear virtually right on top of the original.

This redundancy irked me somewhat. At this point in my process, I was doing a lot of research on augmented reality for another class, and so I saw the potential for making a transparent interface, or an interface that hides the functionality from the user. Apple's iPhone is a classic example of this. When using many of the apps on an iPhone, you don't worry about menus or buttons, but instead use intuitive gestures to manipulate objects on the screen, which in turn changes data behind the scenes. Microsoft's Surface table is another excellent example in which you pinch, swipe, and touch the surface to interact with media, making the images and video on the screen seem to be real objects instead of files or visualizations.

Microsoft Surface, a tactile, multi-touch interface to mimics real-world objects.

Other precedents for my concept are the 2002 movie, *Minority Report*, which was preceded the Microsoft Surface and was an early mainstream example of such a direct and tactile interface. Also, Vernor Vinge's Science Fiction novel, *Rainbow's End*, which dealt heavily with the future of augmented reality, and Neal Stephenson's *The Diamond Age*, which employed nanotechnology to create media - such as printed posters and books - that were responsive and adaptable, were both very inspirational for the effect and experience I hoped to cultivate in

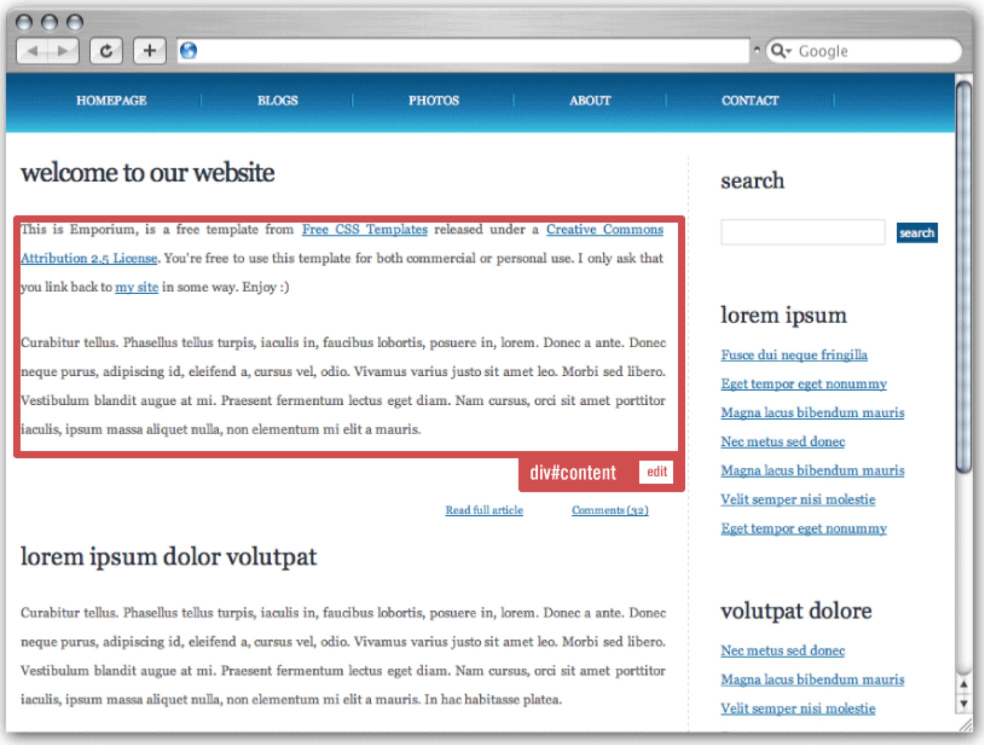
Tinker.


Tactile computer interface, featured in Minority Report starring Tom Cruise

Iterative Design Process and User Testing

Like everybody else in major studio, my project changed over time, and through a series of user tests and conceptual changes, Tinker became a much better product. Looking back to where I started, the role didn't change much from what I originally set out to do. I wanted to make a simpler way to make smaller static sites easy to edit and update, and though I dabbled with other ideas along the way, this is pretty much exactly where I've ended up.

Since so much of my project is technical in nature, it took me a couple weeks to get a prototype ready to test. The first iterations I came up with were studies in exploring a server's file structure, dynamically adding interfaces to static sites, and saving changes to files in a way that would allow them to retain as much as possible of their original formatting and settings. I was far more focused on implementation than look and feel, so I didn't spend enough time on making something that people could work with or that they could give useful feedback to. I wasn't even sure what I wanted to do was going to be possible or if I was capable of executing it, so I wanted to figure that out sooner rather than later. Still, from my brief demonstrations, feedback seemed to be pretty positive and most of the classmates I showed it to could see some potential there.



welcome to our website

This is Emporium, is a free template from Free CSS Templates released under a Creative Commons Attribution 2.5 License. You're free to use this template for both commercial or personal use. I only ask that you link back to my site in some way. Enjoy :)

Curabitur tellus. Phasellus tellus turpis, iaculis in, faucibus lobortis, posuere in, lorem. Donec a ante. Donec neque purus, adipiscing id, eleifend a, cursus vel, odio. Vivamus varius justo sit amet leo. Morbi sed libero. Vestibulum blandit augue at mi. Praesent fermentum lectus eget diam. Nam cursus, orci sit amet porttitor iaculis, ipsum massa aliquet nulla, non elementum mi elit a mauris.

lorem ipsum dolor volutpat

Curabitur tellus. Phasellus tellus turpis, iaculis in, faucibus lobortis, posuere in, lorem. Donec a ante. Donec neque purus, adipiscing id, eleifend a, cursus vel, odio. Vivamus varius justo sit amet leo. Morbi sed libero. Vestibulum blandit augue at mi. Praesent fermentum lectus eget diam. Nam cursus, orci sit amet porttitor iaculis, ipsum massa aliquet nulla, non elementum mi elit a mauris. In hac habitasse platea.

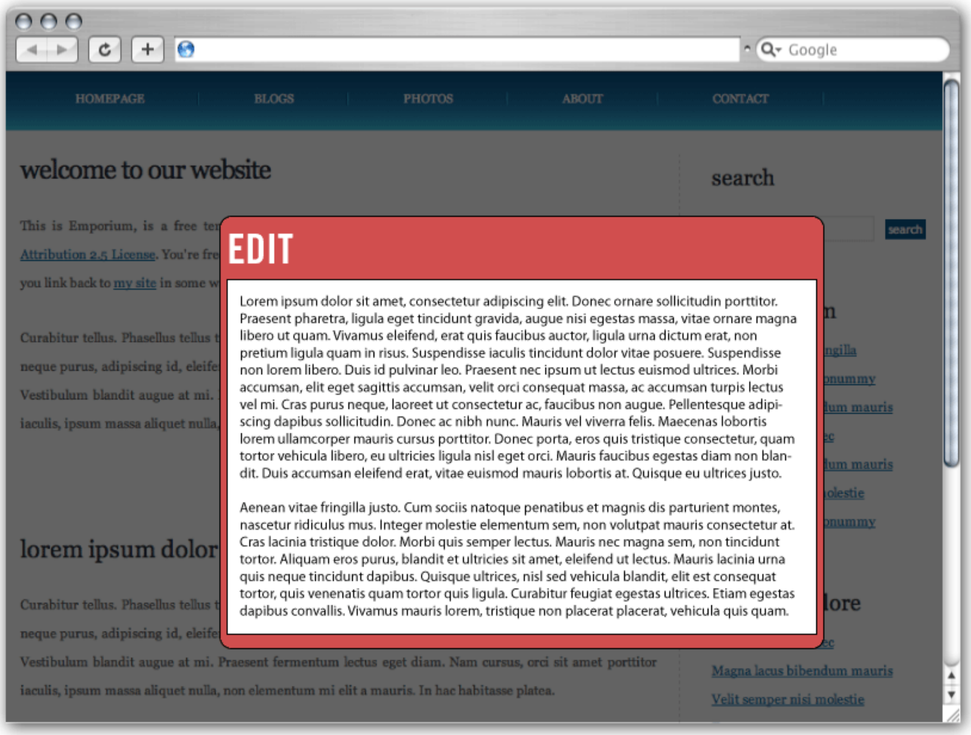
search

lorem ipsum

Fusce dui neque fringilla
Eget tempor eget nonummy
Magna lacus bibendum mauris
Nec metus sed donec
Magna lacus bibendum mauris
Velit semper nisi molestie
Eget tempor eget nonummy

volutpat dolore

Nec metus sed donec
Magna lacus bibendum mauris
Velit semper nisi molestie



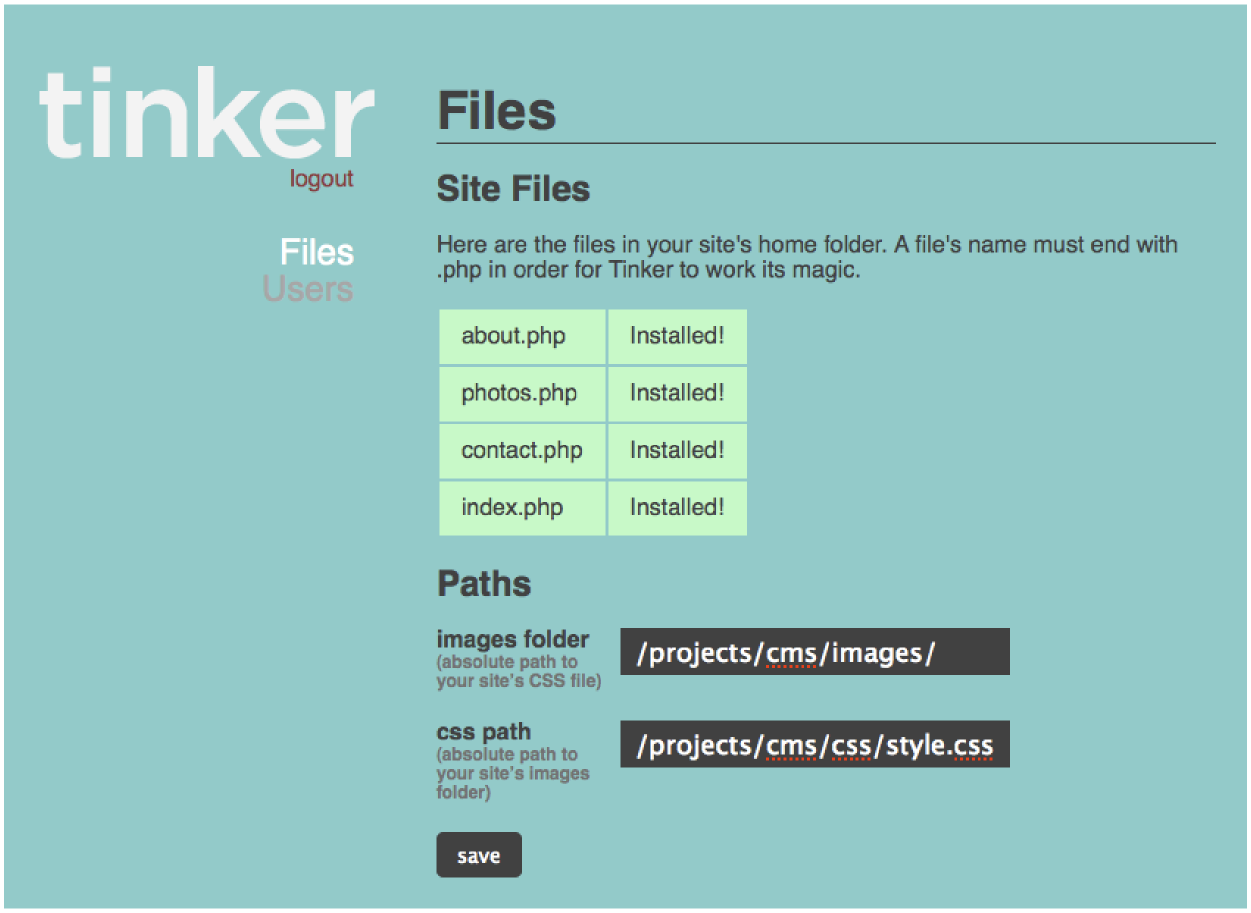
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec ornare sollicitudin porttitor. Praesent pharetra, ligula eget tincidunt gravida, augue nisi egestas massa, vitae ornare magna libero ut quam. Vivamus eleifend, erat quis faucibus auctor, ligula urna dictum erat, non pretium ligula quam in risus. Suspendisse iaculis tincidunt dolor vitae posuere. Suspendisse non lorem libero. Duis id pulvinar leo. Praesent nec ipsum ut lectus euismod ultrices. Morbi accumsan, elit eget sagittis accumsan, velit orci consequat massa, ac accumsan turpis lectus vel mi. Cras purus neque, laoreet ut consectetur ac, faucibus non augue. Pellentesque adipiscing dapibus sollicitudin. Donec ac nibh nunc. Mauris vel viverra felis. Maecenas lobortis lorem ullamcorper mauris cursus porttitor. Donec porta, eros quis tristique consectetur, quam tortor vehicula libero, eu ultricies ligula nisl eget orci. Mauris faucibus egestas diam non blandit. Duis accumsan eleifend erat, vitae euismod mauris lobortis at. Quisque eu ultrices justo.

Aenean vitae fringilla justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer molestie elementum sem, non volutpat mauris consectetur at. Cras lacinia tristique dolor. Morbi quis semper lectus. Mauris nec magna sem, non tincidunt tortor. Aliquam eros purus, blandit et ultricies sit amet, eleifend ut lectus. Mauris lacinia urna quis neque tincidunt dapibus. Quisque ultrices, nisl sed vehicula blandit, elit est consequat tortor, quis venenatis quam tortor quis ligula. Curabitur feugiat egestas ultrices. Etiam egestas dapibus convallis. Vivamus mauris lorem, tristique non placerat placerat, vehicula quis quam.

Images of an early verion of Tinker, which utilized a series of rollovers and popups similar to the interface that Concrete5 employs

About the three week mark, though, I finally had my initial editor interface and a version that people could start to use on their own servers. Scott Peterman was the first to try Tinker out on his own site, and his experience with it was hardly flawless. The installation process was vague and led him to link to his stylesheet files in a way Tinker couldn't understand, which ruined how Tinker presented his pages when editing. Also, the editor was stripping out and reformatting his code, which was breaking the functionality of some of his navigation. Scott found a work-around, but at that early version, it was more work than it was worth to edit with Tinker. To deal with these issues, I altered the installation process to detect files on the server automatically, and give the user the option to just point and click on which files should be used by the CMS. Also, I altered the code handling portions of the text editor, so that it would be more flexible to HTML designs that weren't so standard.

The next person who was willing to try out my CMS on an actual site was Caitlin Morris. Caitlin uses a static HTML portfolio site, and so it seemed like a perfect candidate for updating with Tinker. Again, there were lots of troubles with Tinker. Tinker couldn't access the HTML files on the server, and the server-side code was prevented from saving changes or settings. After a lot of investigation, it turned out that the server was deliberately set up with these features turned off (very rare for most web servers), but it still provided me with some great insight into how Tinker could be improved. We were never able to get Tinker to work on Caitlin's server, but it allowed me to find potential weak points in the installation process and develop a system for making Tinker more responsive and give useful user feedback.


tinker

logout

Files

Users

Files

Site Files

Here are the files in your site's home folder. A file's name must end with .php in order for Tinker to work its magic.

about.php	Installed!
photos.php	Installed!
contact.php	Installed!
index.php	Installed!

Paths

images folder (absolute path to your site's CSS file): /projects/cms/images/

css path (absolute path to your site's images folder): /projects/cms/css/style.css

save

An early version of Tinker's administrative page. The Paths section proved especially confusing to users.

One of the most useful user tests I was able to witness was with an undergraduate product design student named Mo. Mo was willing to sit down an go through a series of user scenarios on a pre-installed version of Tinker. Mo attempted to edit text formatting, work with images, add links and use a bunch of other basic web features in the CMS. He would ask me how to do something, but then before I could even answer, would almost always be able to click the right button or figure it out for himself. In the end, there was nothing he wasn't able to figure out on his own. Even so, he was still frequently caught up with other features that were present in the interface that weren't so obvious. These things were quite often legacy elements from the open source text editor I am using in Tinker, and definitely useful for advanced site owners, but it clearly added confusion to the interface. From this feedback, I see good reason to create a simpler version of Tinker with these options removed, or at least a checkbox in the admin pages that will give people access to them once they've mastered the basics.

Over and over again, the feedback I got was mostly technical. Many things were broken or handled in a way that was confusing, and has mostly been ironed out in subsequent versions. This feedback was critical as I couldn't have easily found it out myself. Each version I release feels mostly complete and intuitive to me, and yet I missed a lot, so it was great to be shown where things could be automated or simiplified further. As for the transparent editing interface, it almost never came up as an issue. People never seemed to have questions about how to make specific changes, and after the initial novelty wore off of being able to type in a web page, it seemed to recede from their minds entirely. I think this is a sign of success in its usability, and it appears to me to be the one feature of this entire system to not have many failings.

Evaluation

Even though a lot of the issues that have come up for this project (and that I am quite often still dealing with) are technical, I feel that its on the technical side that Tinker really has succeeded. On a personal level, I had no idea if I was going to be able to make any of the interface or file editing features work by the end of the semester. Some of it was luck and finding the right tools, because for the first couple weeks, the most I could count on was a solid foundation to a CMS with various prototypes that showed a few of the key features. Still, I've managed to do more than I ever could have expected in just a few weeks.

Furthermore, my CMS seems to be better than any I've found in at making smaller static sites more manageable. It is simpler, faster, smaller, easier to integrate, less resource intensive and more intuitive than the others, and I see so much potential for ways to improve on it.

In fact, as proud as I am of the technical aspects, they are also one of Tinker's two main weaknesses. At this point, it will work on most servers and with most static sites, but there are plenty of exceptions that don't, and so I have plenty that needs to be worked out. Every day I find new problems, and each new problem points to several other potential issues that will need to be dealt with down the road. My list of bugs and necessary features is longer now than it ever has been, and it seems to be growing at every turn. I will continue to deal with these as they come up, but soon I will need a system for managing this, and that looks to be my next big hurdle.

The other thing that has been a problem for me is in defining the role of Tinker to others. This has been a problem since the beginning, and though I have a couple concise and not-so-concise definitions, people still are confused about exactly who Tinker is for and when it should or shouldn't be used. This is partially a problem of branding, but also of so many pre-conceptions on what a CMS can do and the fact that I chose such a niche demographic to support. I hope to figure out better ways of presenting Tinker so that it will be more clea. I also plan to broaden my user base in future versions by giving Tinker features that will help it fill the roles that so many people need.

In a lot of ways, though, Tinker is already successful to me. With a few exceptions, it's a product that I feel will be able to help a lot of people, including myself. I'm in the process of converting a site I recently made for my mom from WordPress into Tinker, and already she's happier with the user experience in editing her pages. Even in its current state, it something that would have saved me lots of time in some of my past work and I fully anticipate it using it in many of my clients' smaller sites. It will save me time on my projects, and it is the most intuitive interface I think I have ever made for a web project.

Beyond that, though, I would like to see Tinker grow as much as possible in the web developer community. I think it could be incredibly useful to other people, and as it matures, I'm sure that Tinker could benefit from a larger array of users and developers. I see Tinker as a long term project, and so it would be exciting to me if Tinker was used widely and received community support for a long time to come. No doubt it will change, and my measure of success would probably change with it, but as long as it was useful and made people's websites easier to use, I would be very, very happy.

Future Directions

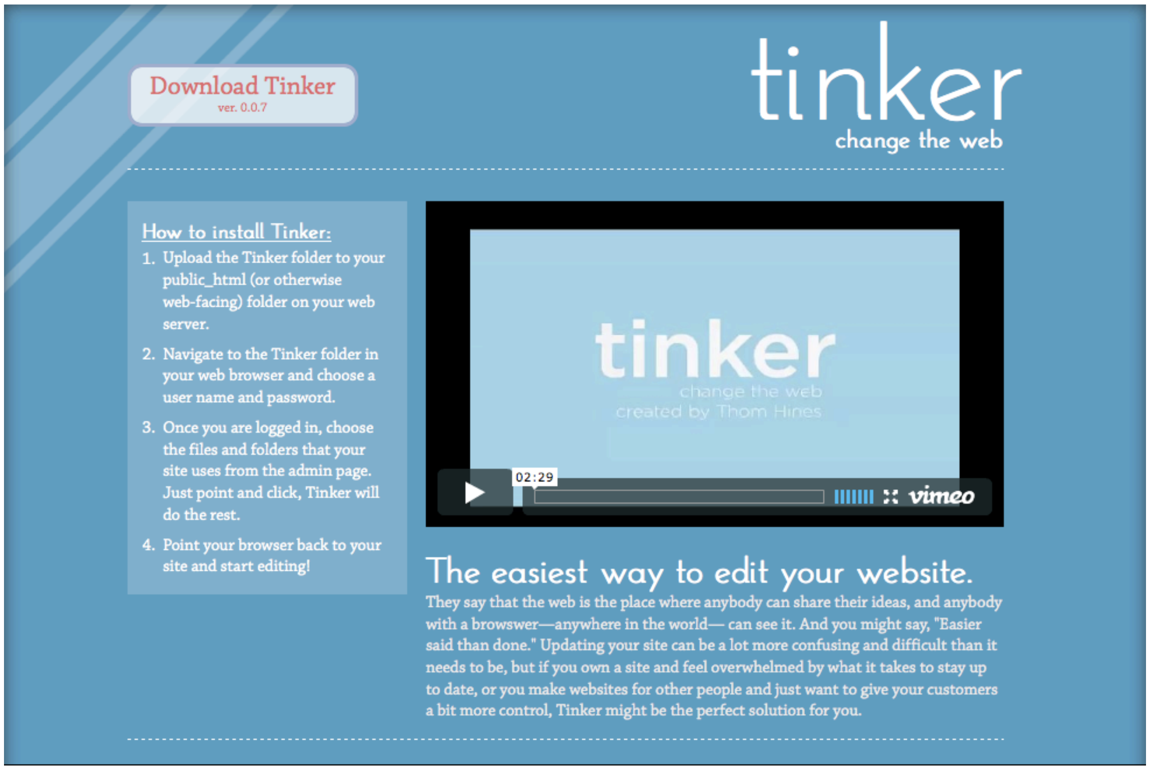
I've had dozens of ideas on how to improve Tinker, and most of them just require the time necessary to implement them. First of all, I haven't addressed all of the concerns I've seen raised through my user testing, and so I would really like to make changes to the code to allow for a more consistent experience for every user and server configuration. I also need to do more testing in other browsers, since it only works fully in Firefox, and I want to simplify the buttons and popup windows to remove any extraneous and unneccessary settings.

From there, I would like to further automate the installation process and require less input from the user, since much of the information about a page is stored in the HTML files themselves. Simplification and usuability are key to Tinker; I would like to lower these hurdles as much as possible. Furthermore, if I can do it in a way that doesn't add complexity to the user experience, I would like to create different user types, so that designers can focus on the styles and layout while editors don't have extra features that get in the way of simple page editing.

Also, I would really like to include ways to make Tinker better at creating pages, not just editing them. Using existing pages as templates, or perhaps including sample pages and galleries might be a great way to get up and running with Tinker. A user-generated collection of pre-built themes might be the perfect way for people to start a web site from scratch.

After that, I think that the transparent interface that makes Tinker so strong could be implemented in other venues. For instance, editing a site from a mobile phone would be a very valuable, and possibly unique, feature of a CMS. Also, once I've gotten better at parsing the information from a user's edits, the transparent interface could be used with other CMSs, such as WordPress or Concrete5. Tinker's interface is modular enough that it might easily become a plug-in for other content management systems, and provide the same ease of use for any kind of site, not just the smaller ones that Tinker works with.

Another avenue which has come up several times is using the foundation of Tinker to target different, non-web-based audiences such as writers or teachers. Since Tinker can work with any type of HTML page or content, building the features and proficiencies toward specific groups could be relatively simple and do more to help those groups out in publishing their work online. If I could take the interface parts of Tinker without having to worry about general concerns like varying file structures and the various ways people set up their own sites, it would be much easier to focus on powerful features that work in more constrained and targeted circumstances.



Download Tinker
ver. 0.0.7

tinker
change the web

How to install Tinker:

1. Upload the Tinker folder to your public_html (or otherwise web-facing) folder on your web server.
2. Navigate to the Tinker folder in your web browser and choose a user name and password.
3. Once you are logged in, choose the files and folders that your site uses from the admin page. Just point and click, Tinker will do the rest.
4. Point your browser back to your site and start editing!

The easiest way to edit your website.

They say that the web is the place where anybody can share their ideas, and anybody with a browswer—anywhere in the world— can see it. And you might say, "Easier said than done." Updating your site can be a lot more confusing and difficult than it needs to be, but if you own a site and feel overwhelmed by what it takes to stay up to date, or you make websites for other people and just want to give your customers a bit more control, Tinker might be the perfect solution for you.

tinkercms.com, the site where people can find information about and download the latest versions of Tinker.

In order to start moving forward with my current iteration of Tinker, I plan to start trying to publicize what it can do and get people to start working with it in as soon as possible. I plan to improve my distribution site and reach out to groups and online communities such as LifeHacker, SlashDot, and Smashing Magazine, and perhaps by submitting my work to festivals and events that many in the web design and development community attend, such as Future of Web Design, South by Southwest, and even some individual developers and firms in the New York development scene. By reaching out and getting some real field-tested and expert feedback from professionals, I hope to make Tinker a better product that could potentially help many people to easily edit their sites.

Resources and References

Adream Blair-Early and Mike Zender. "User Interface Design Principles for Interaction Design." *Design issues* 24.3 (2008): 85-107.

Lev Manovich, "Cultural Analytics: Visualising Cultural Patterns in the Era of 'More Media'."

Domus (2009).

Mark Wigley. "The Architecture of Content Management." *Volume*. 17 (2008): 10-23.

Tony Byrne. “Content Management and Information Architecture,” an interview with Lou Resenfeld, written May 23, 2003, accessed May 7, 2011. http://www.realstorygroup.com/Feature/90-Lou-Rosenfeld

Steven Tanimoto. “Transparent Interfaces: Model and Methods.” Presented at the Workshop on Invisible and Transparent Interfaces, ITI 2004, Gallipoli, Italy, May 25, 2004.

“WordPress.” http://wordpress.org. Publishing Platform (2003).

“Concrete5.” http://concrete5.org. Content Management System (2003).

“Zimplit.” http://zimplit.com. Content Management System.

“Minority Report.” Directed by Stephen Spielberg (2002). Motion Picture.

“Surface.” Microsoft Corp. (2008). Multi-touch computing surface.

“iPhone.” Apple Corp. (2007). Multi-touch smart phone.

Neal Stephenson. *The Diamond Age*. Bantum Spectra, New York (1995).

Vernor Vinge. *Rainbow’s End*. Tor Books, New York (2006).