**Geography 572**

**Lab #3: Slippy Map Iconicity Challenge**

**Lab Objectives:**

* Design a consistent and unambiguous library of highly iconic point symbols
* Serve your own set of tiles using TileMill, CartoCSS, and OpenStreetMap
* Continue to build your knowledge of Illustrator, Photoshop, and HTML5/CSS

**Evaluation:**

This lab is worth **40 points** toward the Lab Assignments evaluation item, which is worth 25% of your overall course grade. A grading rubric is provided at the end of the lab to inform your work.

**Schedule of Deliverables:**

* **October 30th:** Lab #3 Assigned //client contract begins
* **November 6th:** *.csv* File Due (2pts) //input & feedback from client
* **November 13th:** Icon Library Due (6pts) //input & feedback from client
* **November 20th:** Lab #3 Due (32pts) //contract deadline

**Challenge Description**

In an effort to promote the local economy, elected officials from your hometown recently voted to create/restructure the City’s Bureau of Tourism and to hire a new Director of Tourism. The new Director (an admitted map geek!) stated that her first action would be to contract the design and development of a new website featuring the City’s natural, cultural, historical, and/or economic points of interest, or ***POIs*** (i.e., places that tourists may wish to visit during their stay). The Director has declared that the centerpiece of this new website should be a ‘slippy’ map mashup that acts as a spatial catalogue for the City’s rich treasures. Despite recognizing the importance of this central map—and the need to conform to the emerging web map standards driven by popular web services—she does not want the map to have the generic look and feel of most web maps, which typically make use of pushpin symbols atop standard Google Maps tiles. The Director therefore has made clear that a trained cartographer must be hired to design both an aesthetically pleasing set of map tiles and a library of iconic symbols that works in concert with this basemap.

Before advertising the proposal, the Director completed a quick web search to see if there were any cartographers working in the City. Among the first results in the search was your Lab #1 web portfolio (hurrah, the Internet works!). Impressed with your designs, the Director contacted you to see if you would be available to work on the design of the icon library. Upon learning that you are proficient in web design using HTML5/CSS, she also extended you the opportunity to design the template that will be used for the new City website. Happy to make a positive impact in your hometown (and line your pockets with some much needed cash), you gladly accept the job.

***Notes from the Director of Tourism***

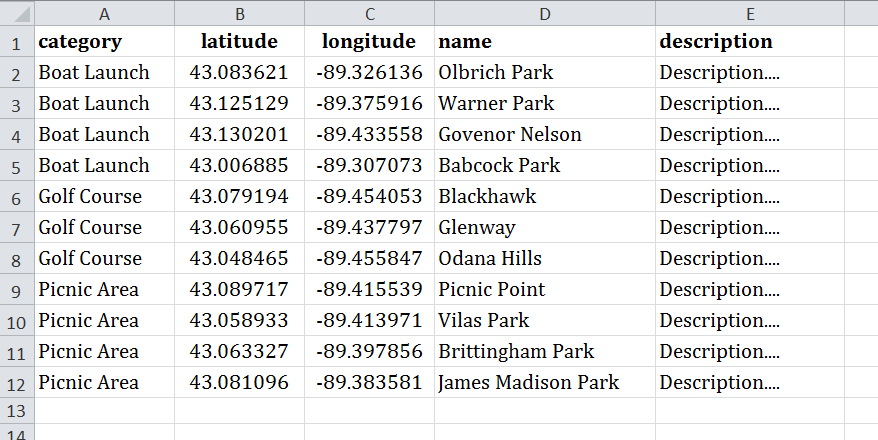
Your slippy map mashup must include ***at least*** *15 unique icons*, with ***at least*** *30 instances of these icons* on the map (i.e., repetitions of icons on the map to show different instances of the same feature type). Selection of an icon instance on the map should activate an information window with a description of the point of interest. The icons must be grouped according to *at least 3 higher-level categories*, with the design of the icon library reflecting this set of higher-level categories; you may include more than three categories, as well as multiple tiers of categories, if logical. To prevent the map from being inundated with detail, you should include ***no more than*** *25 unique icons* and ***no more than*** *75 instances of these icons*. The mashup should not allow zooming out beyond the city, as serving these additional tiles will go over the project budget. It is important to keep in mind that the contract includes ***both*** the design of an icon library for placement on the slippy map (completed using Illustrator and Photoshop) ***and*** the webpage template itself (completed using HTML5/CSS). Finally, you must make use of TileMill for Lab #3, an open source product for serving custom tiles provided by the company MapBox that is introduced in Section 3.

**1. Assembling Your Information**

As with Lab #2, the first step in Lab #3 is assembly of the information that you will be mapping. The information set for Lab #3 emphasizes features at the point dimensionality. While you can include linear or areal features in your map (e.g., a scenic drive, a park), you will need to represent them as points for Lab #3; while this may seem like a limitation, the conceptualization and subsequent representation of features as points is common on web maps due to the provision of integrated map designs at multiple scales, the smallest of which require collapsing to points.

It is up to you to decide which features of interest will be included in your map; if you are from a large city, consider giving your map a specific theme rather than including all possible POIs that may be found on a tourism map. Regardless of which features you ultimately decide to map, you will need to collect five pieces of information about each feature:

1. **Category:** The category representing the type of POI (i.e., the iconic point symbol used to represent the POI); this is not the higher-level category further organizing groups of POIs mentioned above.
2. **Latitude:** In decimal degrees, using negative numbers for latitudes in the Southern Hemisphere. It is recommended that you make use of the iTouchMap tool (<http://itouchmap.com/latlong.html>) to determine the latitude and longitude of your POIs, if you do not otherwise have a georeferenced file. Use a precision of six decimal places to accurately place each POI.
3. **Longitude:** In decimal degrees, using negative numbers for longitudes in the Western Hemisphere.TileMill searches for any column with the terms “lat”/“latitude” and “long”/“longitude”, using the first instance found in the spreadsheet for the X or Y coordinate, respectively.
4. **Name:** The specific name of the instance of the feature. This is different from the category, as several mapped features may receive the same iconic point symbol.
5. **Description:** A brief (100 words or less) description of the feature, perhaps with a link to additional information. You should compose the descriptions yourself such that the set of descriptions are at a common length and contain a common set of summary information. Both the name and description columns will be used to populate an information window upon clicking the icon.



**Table 1**. **An example *.csv* file, formatted for loading into TileMill**.

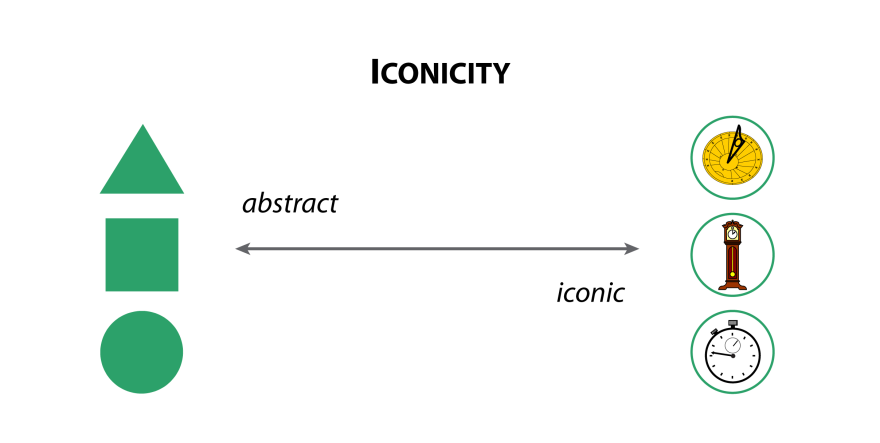
**Table 1** provides the example information for Madison, WI, used in the following tutorial.

Your information must be stored in a *comma separated file*, or ***.csv***, for loading into TileMill. As its name implies, a *.csv* file is a flat text file that delimits rows using the comma (“,”) character. Therefore, you may not include commas in any cell in the file unless you make use of the escape character \_\_. To create your *.csv* file, you either can assemble your spreadsheet in Microsoft Excel (as shown in **Table 1**) and export using *Save As->Save As Type: CSV*, or you can assemble this information in your text editor, delimiting each piece of information using commas and each mapped feature as a different row. You are required to have your *.csv* file assembled and ready for mapping by **November 6th**.

**2. Designing Your Icon Library**

**a. Get Inspired!**

Once you have assembled your information, you are ready to design your icon library! For Lab #3, you are asked to create a set of highly iconic point symbols (sign-vehicles) that should unambiguously represent the depicted phenomena (referent) and evoke the proper concept about the phenomena (interpretant). As introduced in class, ***iconicity*** describes the degree to which a sign vehicle mimics its referent; in other words, it is the semiotic ‘nearness’ between sign vehicle and referent, as mediated by the interpretant. Iconicity is considered a continuum, ranging from the abstract or arbitrary to the iconic or mimetic (**Figure 1**). You are required to create symbols that are more iconic than abstract, although you will find that you are unlikely to design symbols at the level of iconicity you initially hoped due to constraints in the size of the point symbols on your map, and the associated constraint of symbol’s pixel resolution.



**Figure 3: Iconicity.** Qualitative point symbols range from the abstract to the highly iconic.

Before starting your icon designs, we recommend getting inspired by reviewing several prominent projects regarding icon/symbol design:

* [The Noun Project](http://thenounproject.com/): An initiative not specific to mapping that includes icons on a wide array of concepts (and typically multiple icon options for the same concept). You can review the icons in the “Playground” for free, but download of icons has a recurring fee.
* [SymbolStore.org](http://symbolstore.org/): An icon repository maintained by the Penn State GeoVISTA that supports upload and download of point symbols in several geospatial formats for use on maps. There are no usage restrictions on the majority of contributed symbols.
* [Maki](https://www.mapbox.com/maki/): A set of nearly 300 point symbols provided by MapBox explicitly designed for maps. The icons can be downloaded in multiple formats or added directly through TileMill (for details, see: <https://www.mapbox.com/tilemill/docs/guides/using-maki-icons/>). Because of the focus of Lab #3 on design custom point symbols that work coherently as a single library, the following tutorial does not explicitly make use of Maki symbols.

While we recommend leveraging these resources for inspiration, keep in mind that you are required to create a library of symbols that: (a) works internally together, with a clear coherency of design style and an indication of a higher-level categorization, and (b) works well with your custom basemap tileset. Thus, the symbols drawn directly from these resources will require at least minor modification in Illustrator and Photoshop before they effectively can be integrated into your library.

**b. Pen-and-Paper Sketches**

As with Labs #1 and #2, the first step in designing a library of iconic point symbols is pen-and-paper sketching. For Lab #3, we recommend constraining your pen-and-paper sketches using graph paper, as the gridded paper will help you get a sense for the constraints of your selected pixel resolution, while allowing you to explore multiple alternatives. As you work through your pen-and-paper sketches, consider the following guidelines for designing iconic point symbols:

***Evoke a Proper Interpretant***

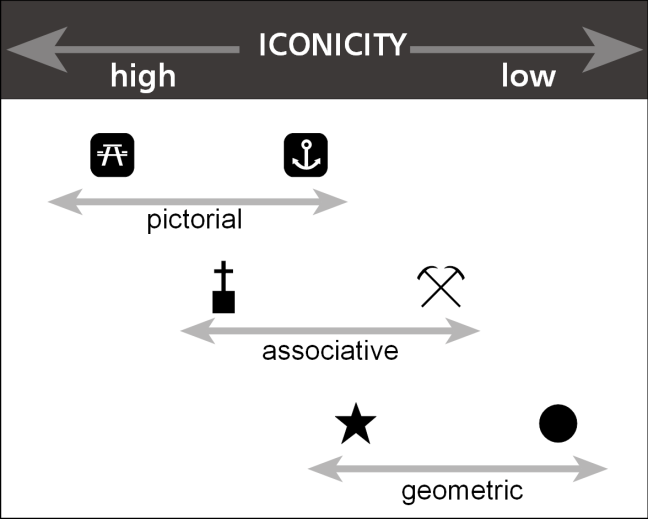
Iconic point symbols can be either pictorial or associative in nature (**Figure 2**). ***Pictorial symbols***look like their referent and typically do not need a legend to be understood (e.g., the use of a bench icon to represent a park bench). ***Associative symbols*** do not look like their referent, but instead look like a phenomenon or concept associated with the referent (e.g., a pick axe to represent a mine). Depending on the feature you are trying to represent, and the constraints in the symbol size, an associative symbol may be a better choice than a pictorial symbol. Consider multiple pictorial and associative designs that may evoke the proper interpretant. It is important to get feedback from your peers to ensure your symbol designs are in fact logical.

***Design for the Map Display Constraints***

You will save yourself a great amount of time and energy if from the onset you design your iconic symbols for the intended symbol size and pixel resolution. Iconic point symbols often are balanced for ***radial symmetry***, meaning that the artboard can be divided equally into four separate sections. Common symbol resolutions therefore include 16px and 24px (for web maps viewed in the browser), 36px (for mobile devices), and 256px (for logo-sized icons). You are recommended to select either 16px or 24px, depending on the visual complexity of your map (i.e., you can have a slightly larger size if you include fewer total icons) and to design for that symbol size. As described in Section 3, TileMill supports the use of different symbol styles and sizes at different scales, so consider how you might reduce the iconicity of your symbols when the display becomes overly cluttered.

***Design Silhouettes***

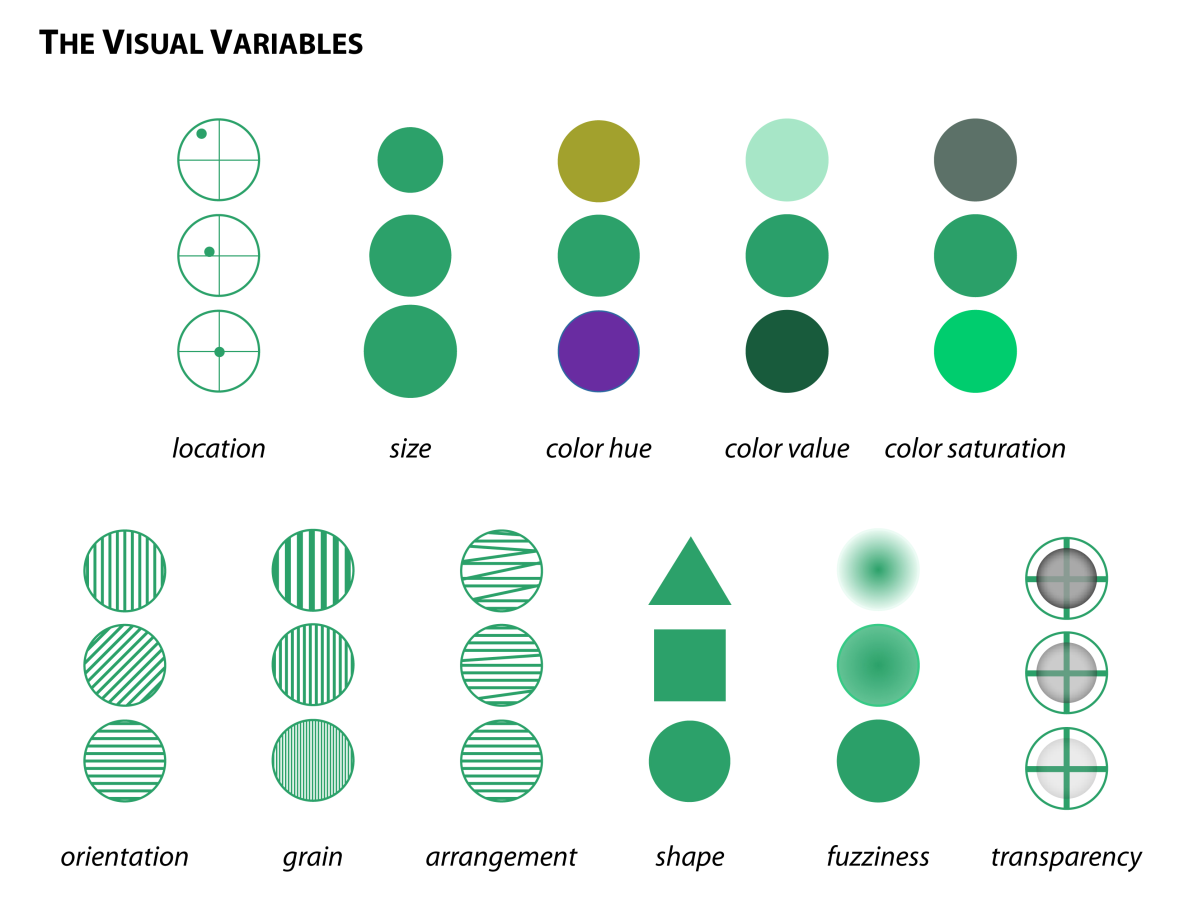
Artboard resolutions of 16px or 24px do not offer a large amount of space for designing highly iconic point symbols. Because of this, you will need to reduce your design down to a simple ***silhouette***, or an icon that is in outline form (e.g., see the pictorial symbols in **Figure 2**). Do not create point symbols that require multiple colors or outline strokes. The use of silhouettes often leads to iconic point symbols that have two colors: (1) the background symbol color (this may be a simple gradient within a single hue) and (2) the silhouette. Consider your custom basemap design to determine which of these should have the lighter color value: if using a light basemap, it is likely that the symbol background should be darker with the silhouette nearing white; if using a darker basemap, it is likely that the symbol background should be lighter with the silhouette nearing black. You also may wish to add a one pixel frame around the outside of your symbol to maintain contrast across a varying basemap. *Keep in mind that any nuance of your design that is less than a pixel wide will not resolve well when exporting from a vector environment to a raster environment*.



**Figure 2: Pictorial versus Associative Symbols.** An associative symbol may be a better choice, given the constraints in symbol size and pixel resolution.

***Signify Higher-level Categories***

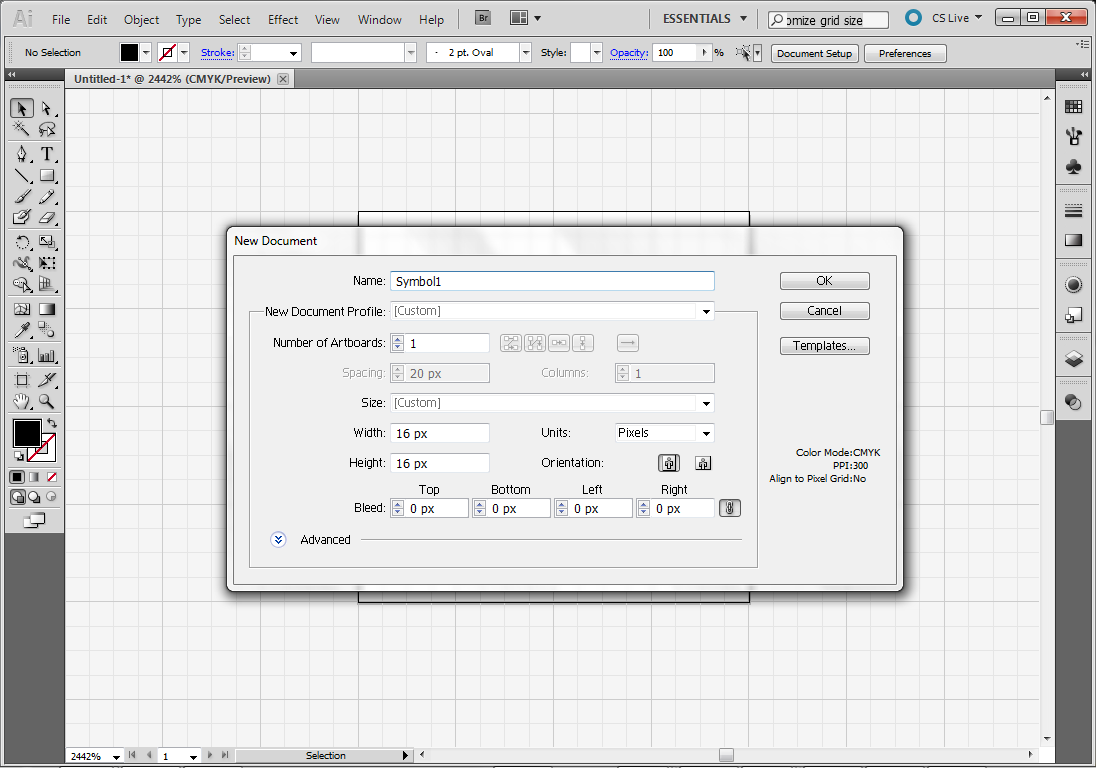
A requirement of Lab #3 is that your point symbols both signify a unique geographic feature and the higher-level category to which the feature belongs. Such higher-level signification will improve the legibility and organization of your map. Because your iconic point features should contain the symbol silhouettes only, you effectively have at your disposal the full range of visual variables to communicate the higher-level category (**Figure 3**). In other words, the higher-level categorization is specific to the design of the symbols and not the loaded *.csv* file, although you may include this higher-level distinction as a column in the *.csv* if you wish to convey the distinction in the information window. Keep in mind the syntactics of the visual variables, as you now know that several visual variables are better purposed for the communication of categorical versus numerical information. Also consider the potential of redundant symbolization.

****

**Figure 3: Using the Visual Variables to Signify the Higher-level Categories.**

**c. Drawing your Icon Silhouettes in Adobe Illustrator**

Once you have explored possible designs, you can start to execute your ideas in Illustrator. The first step in executing your designs—as with the pen-and-paper sketch—is to configure your artboard to the selected pixel resolution. To change the artboard size and resolution, create a new document (*File->New…*) and adjust the *Width* and *Height* parameters such that they reflect your selected symbol size/resolution; in the case of **Figure 4**, a size/resolution of 16px is selected. Also, be certain to indicate that the *Units* are in *Pixels*, and not *Points* or *Inches*. Give your symbol a logical name and click *OK* to create the new document.

****

**Set the *Units* to *Pixels***

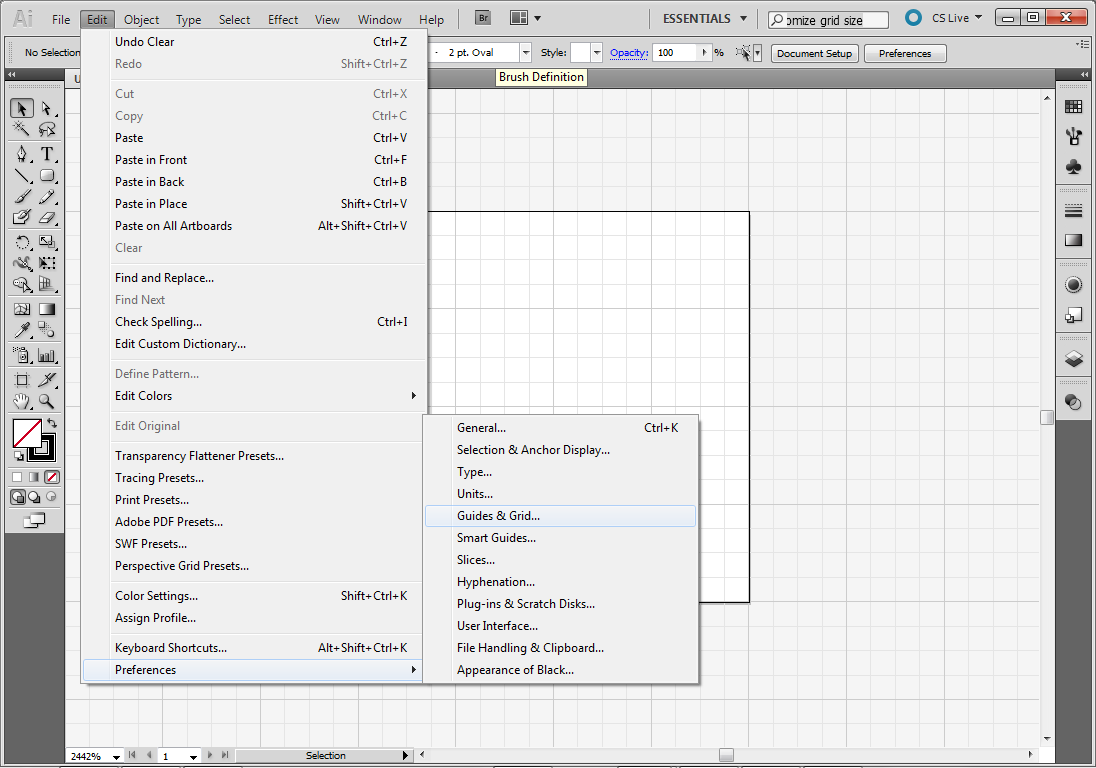
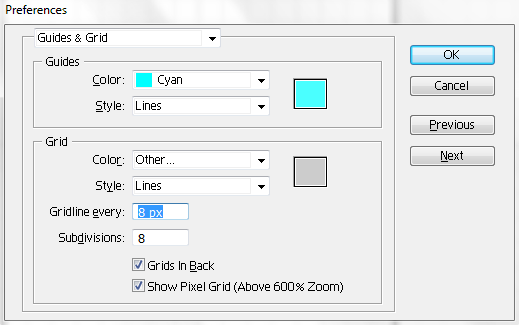
**Set the *Height* and *Width* to your preferred size/resolution**

***File->New…***

**Figure 4: Configuring the Artboard for your Icon Design.**

After configuring the size/resolution of your artboard, it then is necessary to match the grid to your artboard design. Aligning the grid to your artboard will allow you to snap nodes to the corners of actual pixels in the output files, ensuring that all components of your silhouette design are a pixel or larger. To configure the grid, first toggle the visibility of the grid (*View->Show Grid*) and snap your vector artwork to the displayed grid (*View->Snap to Grid*). You will notice that the grid poorly aligns with your artboard size/resolution (it may be that a single grid cell is the entire size of your artboard, or larger). To change the resolution of the grid, select *Edit->Preferences->Guides and Grids...*;this will activate the *Guides and Grids* dialog window (**Figure 5**). In the window, change the *Gridline every* parameter to half of your pixel resolution for radial symmetry (‘8px’ in **Figure 5**) and the *Subdivisions* parameter such that there is a grid square every pixel (‘8’ in **Figure 5**, meaning that the two values should be the same).

You now are free to begin translating your pen-and-paper ideas to the digital design environment. You may take two strategies in executing the linework for your silhouette (and likely will want to use these solutions in combination): (1) drawing and refining the outline of the silhouette, with each node snapping to a corner in the grid to ultimately enclose your shape, or (2) drawing and refining lines of 1px thickness, ultimately converting the lines to polygons using *Object->Path->Outline Stroke.* It is recommended that you maintain a pair of layers in each Illustrator file: (1) a top layer containing your icon graphics and (2) a bottom layer containing the symbol background. Remember again: *Any nuance of your design that is less than a pixel wide will not resolve well when exporting from a vector environment to a raster environment*.

****

**Adjust the *Gridline every* and *Subdivisions* parameters**

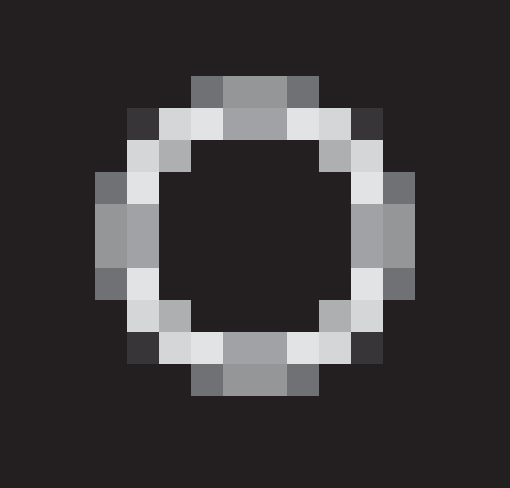
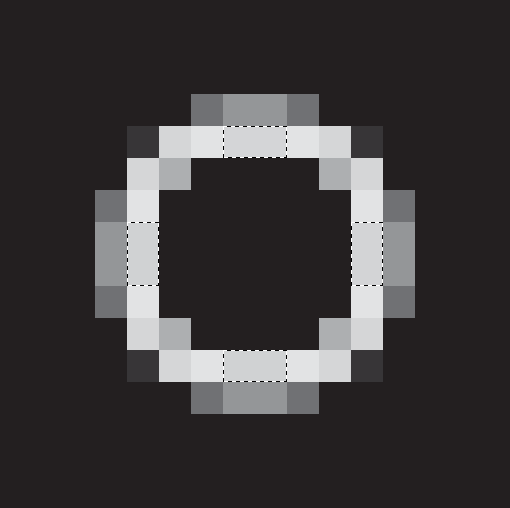
***Edit->Preferences-> Guides & Grid…***

**Figure 5: Configuring the Grid for your Icon Design.**

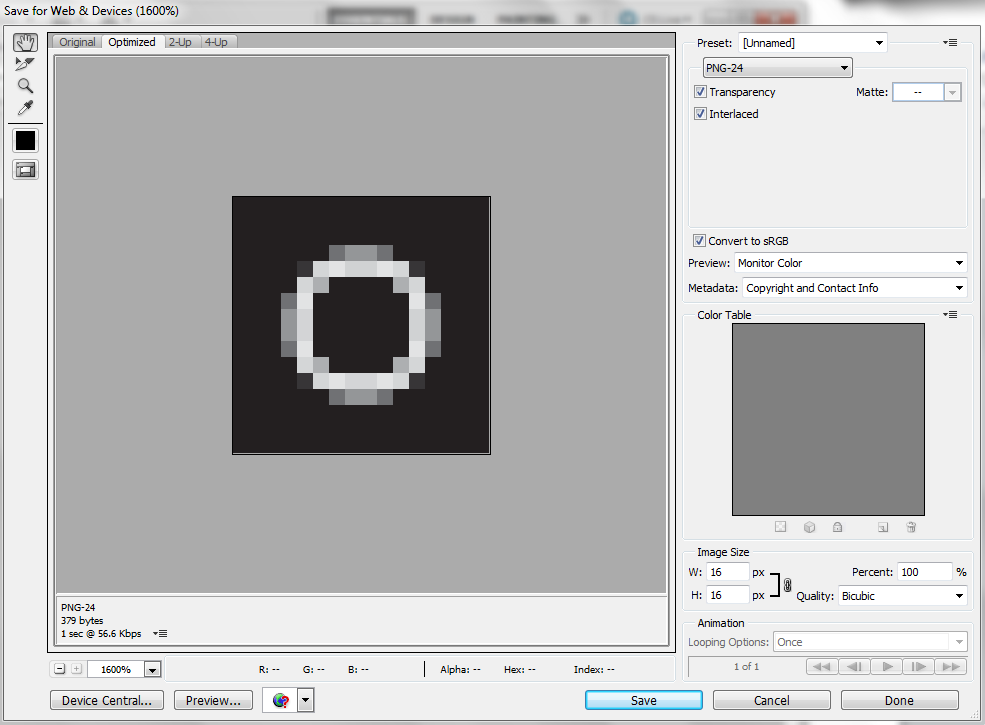
**d. Finishing in Adobe Photoshop**

After you have completed your icon design, it is recommended that you rasterize it for display on your slippy map mashup. The optimal workflow for converting your vector graphics into a raster image is to first export from Illustrator in the .*tif* format (*File->Export…*); when prompted, use the *Screen* resolution to preserve your original pixel resolution. The *.tif* format will preserve the original resolution of the icon and add minimal lossy compression (i.e., there be little blurring/smoothing, if you designed to have the smallest feature be no thinner than 1pt); however, this format is not common on the web due to the relatively larger file size compared to the *.png* and *.jpg* formats. Open the *.tif* file in Adobe Photoshop to preview and resolve any issues in legibility/clarity resulting from the export to a raster format. In Photoshop, you can manipulate individual pixels using the *Rectangular Marquee Tool* to select the pixel(s) and the *Brush* tool to recolor the pixel (see Lab #1 for a review of the Photoshop Tools Panel) (**Figure 6**).

After polishing your icon in Photoshop, create a .*png* version for integration into your slippy web map; be sure to follow logical symbol naming conventions. To create a *.png* of your icon, select *File->Save for Web Devices…* Be sure to maintain the original symbol size/resolution in the .*png* image (**Figure 7**). Assuming you have followed the above steps (i.e., designed for the correct size/resolution from the start), you should not experience any loss in quality when exporting to a *.png*.

** **

**Figure 6: Pushing Pixels in Adobe Photoshop.** The crispness of your icon often can be improved by selecting a problematic pixel or set of pixels and adjusting their coloring slightly.

****

**Figure 7: Exporting to .*png***

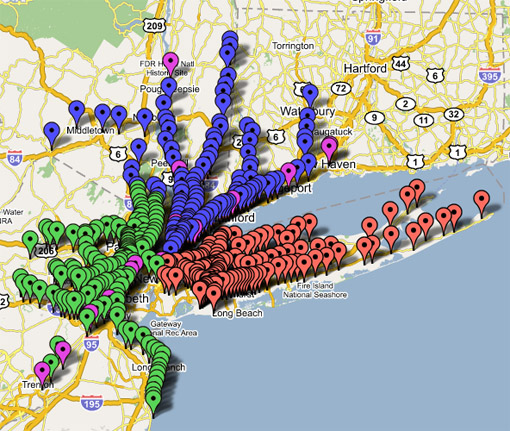
**Maintain the Original Image Size**

**3. Serving Custom Tiles with TileMill and CartoCSS**

**a. Introduction to Slippy Map Mashups**

A ***mashup*** combines disparate data sources and web services from multiple online providers into a new, unique service. Mashups are a primary innovation indicative of ***Web 2.0*** technology, or the transition from the Internet as mere file transfer device (i.e., things moving through it) to the Internet as a full-fledged platform for supporting complex user interaction (i.e., things happening on it). The most powerful mashups are synergistic, where their usefulness surpasses that of the original data sources and web services combined (i.e., the components may be of limited applicability or significance when examined as individual pieces, but a Eureka moment occurs about the big picture when they are combined).

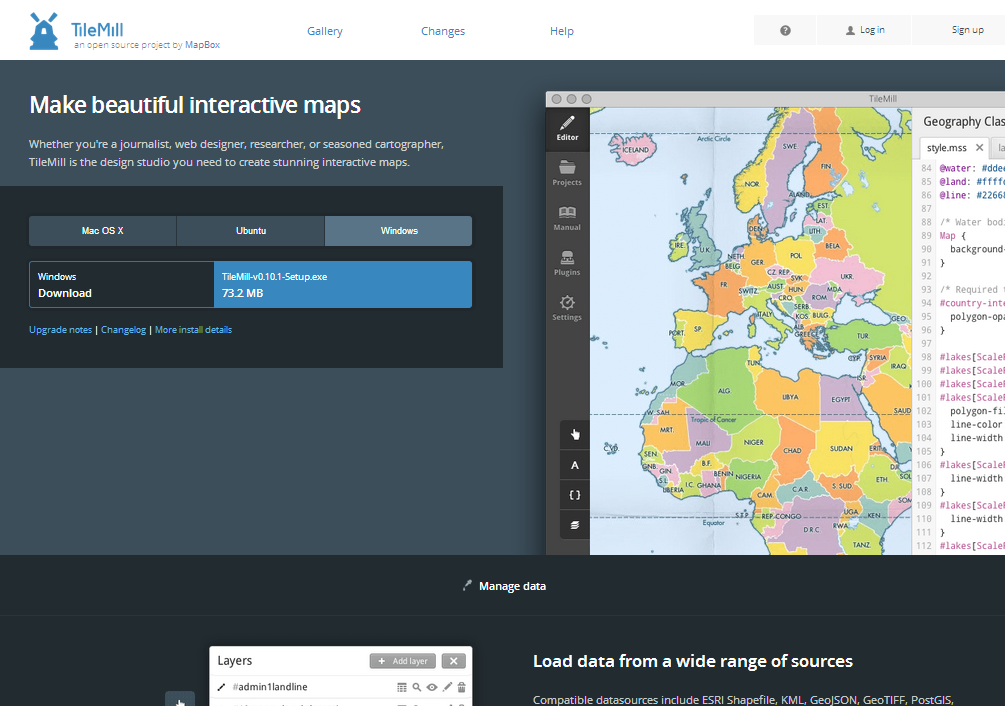
The term ***slippy map*** describes a mashup that draws upon web mapping services to provide integrated, multiscale map designs, typically using a tile-based, raster solution in which the maps at each scale are divided into a set of 256x256 pixel tiles. A slippy map also exhibits basic interactive functionality, such as, panning, zooming (together where the ‘slippy’ term came from), overlay of different basemaps, and retrieval of details through information windows (e.g., your point icons). The advantage of slippy map mashups is that their creation requires very little custom code by combining existing services. However, slippy map mashups were not always considered a positive development in Cartography. Most early slippy maps were ignorant of the time-tested principles of Cartography, leading to the generation of a fleet of poorly designed web maps cluttered by a field of arbitrary “pushpin” point symbols atop an inappropriately detailed and styled road map (**Figure 8**).



**Figure 8: The Oft-Maligned Slippy Pushpin Map.** Can you guess what this map is representing? Your Lab #3 deliverable should not look like this.

**b. Getting Started with TileMill**

Despite these initial cartographic frustrations, the ability to customize the cartographic display has improved substantially in recent years through expansion and refinement of tools for styling and rendering your own basemap tiles. Your Lab #3 slippy map mashup will make use of ***TileMill****,* an open source project developed by a company called ***MapBox*** that supports custom styling and rendering of basemap tiles. MapBox is quickly becoming a leader in the geospatial industry, particularly in the area of web mapping. TileMill allows you load and style your own datasets, such as your set of POIs, or to draw from existing data services, such as OpenStreetMap (introduced below); the following tutorial will introduce both approaches. It is important to note that TileMill is open source in that its styling and rendering features are available for free. However, there is a cost to serving tiles beyond a baseline number of scales and geographic extent (50M). The Lab #3 challenge is explicitly written to keep you within this limit, but you are free to serve additional tiles beyond the 50MB limit if you choose. Finally, we will be serving raster tiles for Lab #3, given the commonality of this solution. MapBox has been highly innovative in the development of vector-based tiling, a promising solution for responsive web cartography in the future.

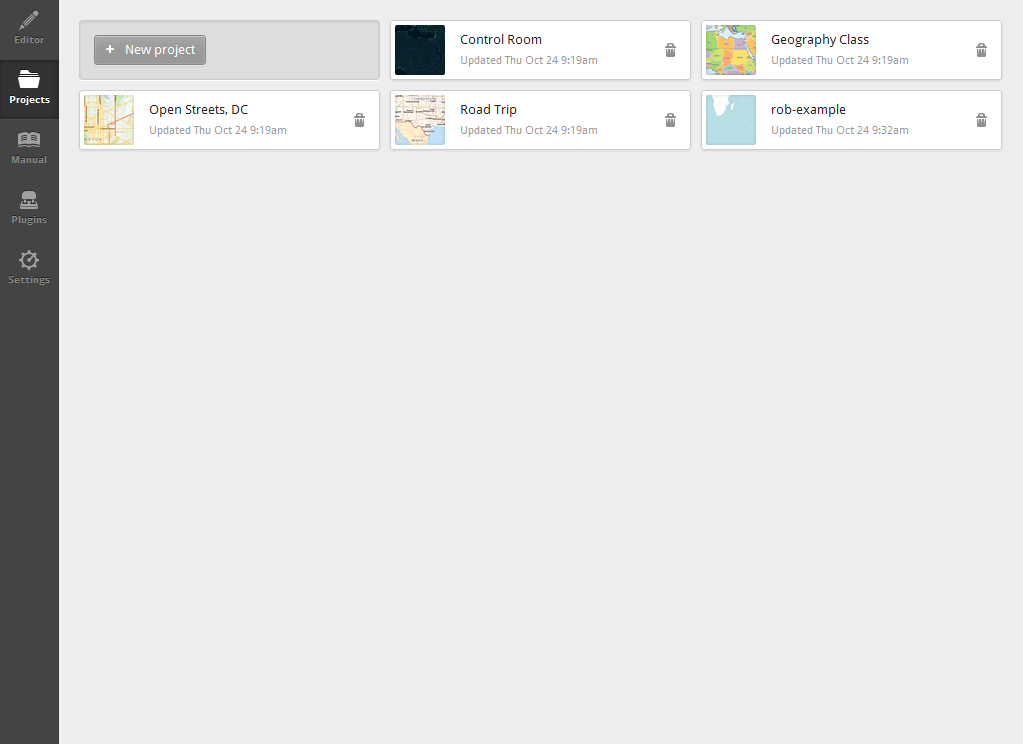


**A**

**B**

**Figure 9: The MapBox Website for TileMill.**

Get started with TileMill by registering with MapBox: <https://www.mapbox.com/tilemill/> (**Figure 9a**). If you prefer to work on your own machine, you then can install TileMill by selecting the appropriate operating system from the download options (**Figure 9b**). TileMill also is available on all machines in Science Hall 380 and M376. If you are working on a Science Hall machine, open TileMill at *Start->* . TileMill will open on the *Projects* tab (**Figure 10**).

****

***Plugins* tab**

***Settings* tab**

**Your projects**

***New Project***

***Editor* tab**

***Manual* tab**

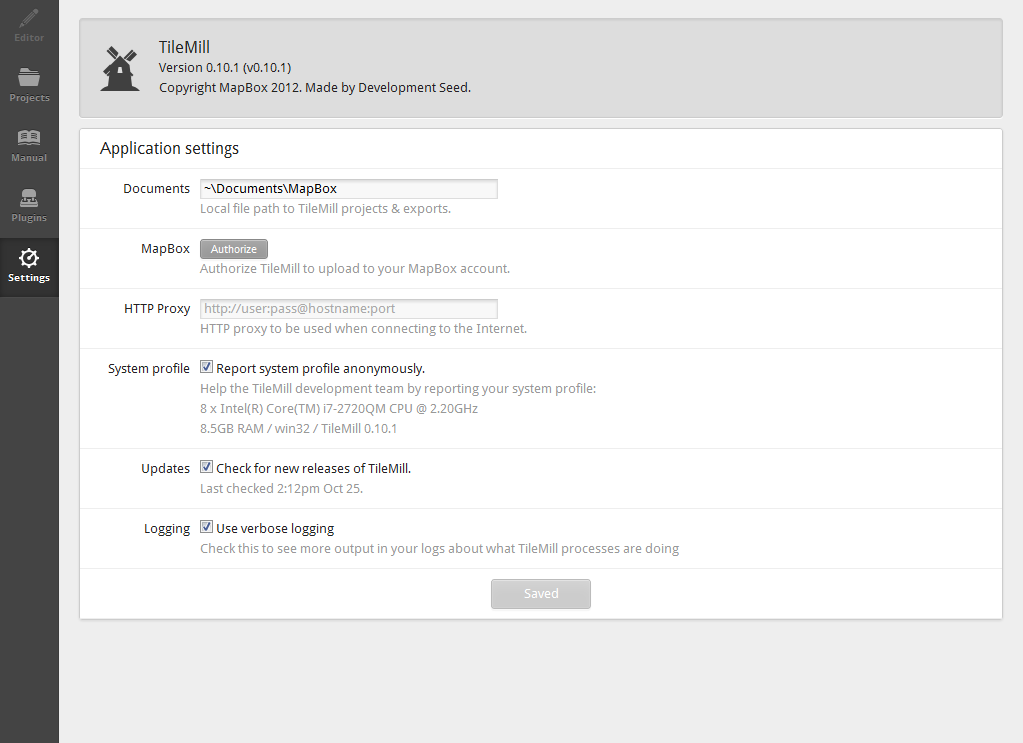
***Projects* tab**

**Projects examples**

**Figure 10: The TileMill Application.**

The TileMill application includes five tabs along the left bar (**Figure 10**):

* The ***Editor*** tab provides a split-panel display, showing your styled basemap on the left and your CartoCSS style rules on the right. You will execute your design in the *Editor* tab; additional details about the *Editor* tab are provided in the next subsection.
* The ***Projects***tab provides several example projects to help you get started and will list the custom projects on which you are currently working. In **Figure 10**, a new project has been created called ‘rob-example’. Clicking on any of the projects will open that project in the *Editor* tab. New projects and their associated assets are stored in a folder on the hard drive called *MapBox*. *Be sure to make a copy of these files if you are working on a university computer and to delete your project in the* Project *tab before leaving your session*. You then can paste these files back into the *MapBox* folder at the start of your next session to have the project available in TileMill.
* The ***Manual*** tab provides an essential introduction to TileMill and CartoCSS. The manual itself is extremely well-written and comprehensive. *The following tutorial is written with the expectation that you have studied the content within the* Manual *tab*.
* The ***Plugins*** tab allows for inclusion of additional functionality through plugins. You will not need to make use of plugins beyond those currently installed for Lab #3, but you may want to explore the other available plugins on your own machine.
* The ***Settings***tab allows you to configure your session. Importantly, you will need to login to your MapBox account before you can export your tiles to the MapBox server; do this by clicking the *Authorize* tab (**Figure 11**). *Be sure to logout of your MapBox account at the end of your work session*.



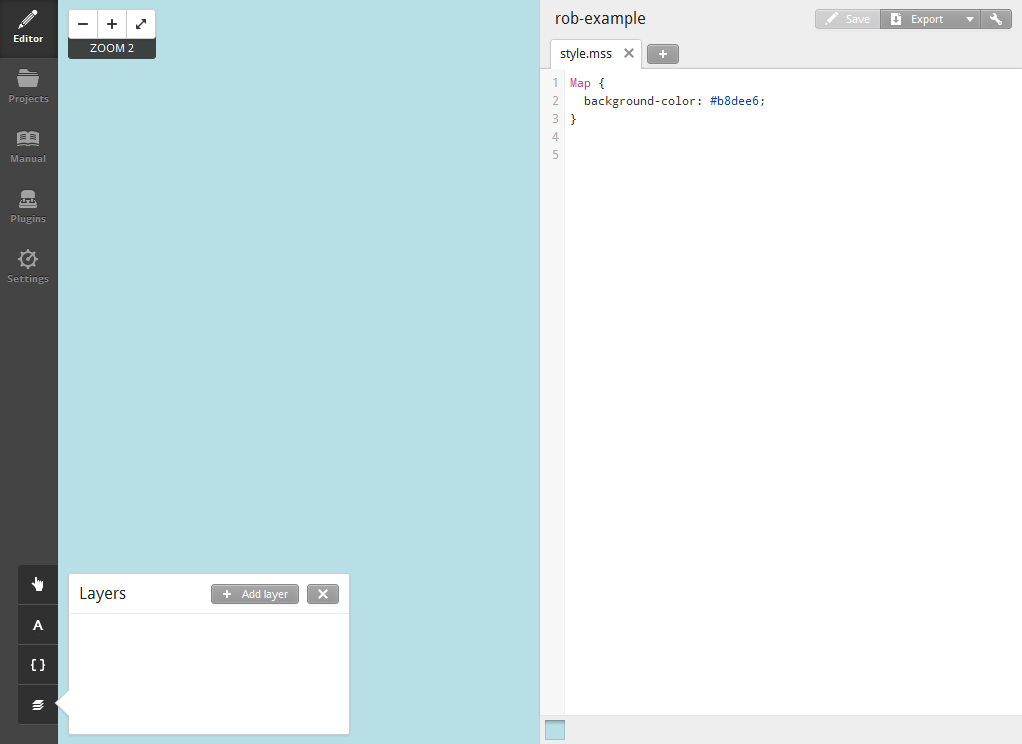
***Authorize* your account before serving your tiles**

**Figure 11: The *Settings* tab.**

**c. Loading Data into the Editor**

Now that you have explored TileMill, let’s get started by creating a *New Project* in the *Project* tab (**Figure 10**); when prompted, give the project a logical *Filename* and uncheck the option to load *Default data*, as your Lab #3 map will not need world layers. Once created, click on the new project to open it in the *Editor* tab (**Figure 12**). You will see the split-pane interface, with the blank map on the left styled with a default light blue (#b8dee6) color and the associated styles for the blank map in the right styles pane.

The first dateset you are going to load for your slippy map is your processed *.csv* file containing your POIs (see Section 1). Activate the *Layers* panel, select *Add layer*, and browse for your *.csv* file under the *Datasource* input box; click *Done* when you have the correct directory path selected. You will need to give the *.csv* file an ***ID***, which is the name of the layer as referenced in the stylesheet. You have the option of adding a ***class***name as well, which is useful for organizing multiple layers into groups, but is not essential for Lab #3. Allow TileMill to *Autodetect* the ***SRS***, or spatial reference system. When you are done, select *Save & Style* to load the POIs and apply a default styling for preview (**Figure 13**). For additional details about loading a .*csv*, see: <https://www.mapbox.com/tilemill/docs/crashcourse/point-data/>.



***Main toolbar***

***Map preview***

***Stylesheet editor***

***Zoom controls***

***Carto* reference**

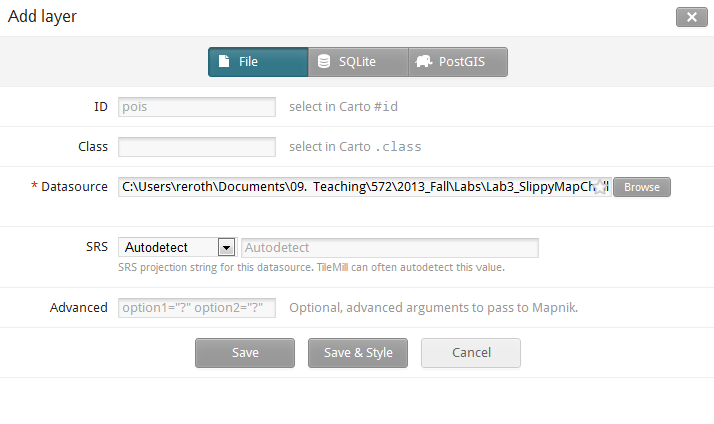
***Add layer***

***Templates***

***Fonts***

***Layers***

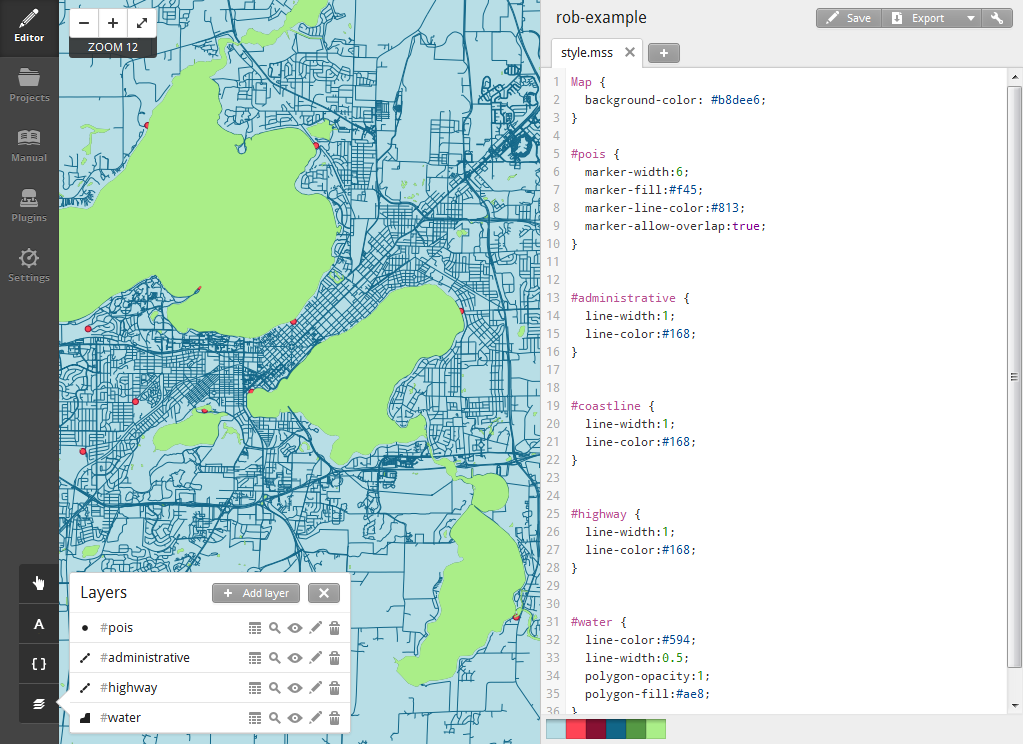
**Figure 12: The *Editor* tab.**



**select *Save & Style***

**Figure 13: Loading your *.csv* data.**

Next, you need background context for your POIs. TileMill allows you to load a variety of spatial formats, including shapefiles, GeoJSON, KML, and GeoTIFF, as well as draw data from SQLite and PostGIS databases. For Lab #3, you are going to download linework from OpenStreetMap to create a basemap for your POIs. OpenStreetMap(***OSM***) is a shining example of volunteered geographic information (***VGI***), or the use of public crowdsourcing to collectively create and maintain geographic information. Details about OSM are available at: <http://www.openstreetmap.org/>. The collective contribution to OSM is now over 30GB, a file size far greater than you wish to download at one time. Instead, make use of the region extracts made available by CloudMade, a spin-off company from early OSM contributors that also provide a proprietary tiling service. CloudMade provides extracts at the state level in the US (and many other admin\_1 levels around the world) at: <http://downloads.cloudmade.com/>. **Figure 14** shows the project updated to include default styles for three OSM layers for Wisconsin: administrative, highway, and water. We recommend that you manipulate this linework in ArcGIS or MapShaper before uploading, particularly if you are going to change the generalization of your linework across scales (see discussion on conditional styling in the next subsection). You are welcome to add other context layers as well from OSM or other sources. Note the stacking order in the *Layers* panel reflects the stacking order of the map preview, but the order of the style rules in the stylesheet editor does not impact the drawing order (at least when there is only one style rule per layer).



**Figure 14: Adding OSM Data to Your Project.** Note: This map shows default styling only.

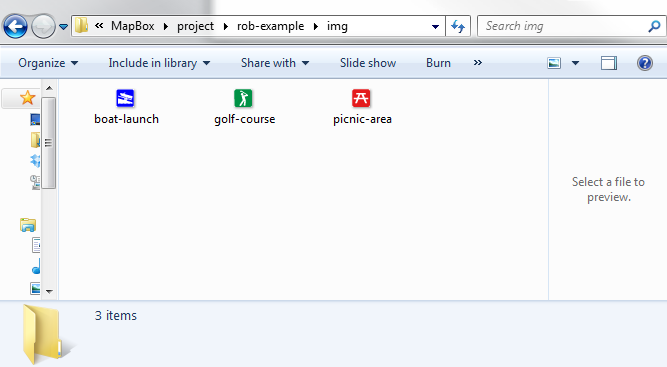
**d. Styling Linework with CartoCSS**

You now have successfully loaded your linework and point features into TileMill. Be sure to saveyour project using the *Save* button in the *Main toolbar*. It is now time to style your map!

TileMill supports the styling of basemap tiles through their ***CartoCSS*** styling specification. In Lab #1, you learned about CSS and the components of style rules (e.g., selectors, properties, values, ids, classes, etc.); return to the second part of Lab #1 if you are not yet comfortable with these concepts. The CartoCSS specification leverages the CSS syntax to allow you to style map features, rather than page elements. Hopefully you now see the value of TileMill: it makes map design much more like web design. The skills learned to build your web portfolio in Lab #1 are now directly applicable to making elegant and aesthetically pleasing tilesets!

Like CSS, CartoCSS apply style rules to map layers by referencing unique IDs (e.g., #pois). You can see in **Figure 14** that default style rules were added to the Map itself, for which there are a small subset of styling rules, as well as each uploaded layer using the unique id you gave it during upload. CartoCSS then is organized into ten different groups of styles based on map features; each group is described as a *symbolizer*, a term derived from the [Mapnik](http://mapnik.org/) rendering engine atop which TileMill is built. Symbolizers include:

* ***Line***: styles for line features and the strokes of polygons;
* ***Polygon***: styles for the fill of polygons;
* ***Point***: styles for point features;
* ***Text***: styles for labeling point, lines, and polygons;
* ***Shield***: styles for symbol annotation for points and lines;
* ***Line Pattern***: styles for applying textures (e.g., dashing) to lines;
* ***Polygon Pattern***: styles for applying textures to polygons;
* ***Raster***: styles for manipulating GeoTIFFs
* ***Markers***: styles for manipulating iconic point features
* ***Buildings***: a subset of polygonal styles explicitly used for build infrastructure.



**Figure 15: Copying your Icon Library into Your Project Folder**

For Lab #3, you will style your POIs using *point* symbolizers, allowing you to symbolize each point location using the appropriate icon. First, create a new folder called *img* in your project directory within the *MapBox* folder and then copy your icon library into the *img* folder; you must do this outside of TileMill (**Figure 15**). Once the icons are copied into your project folder, you then can use the point-file property and set the url() to the relative path of one of your icons (**Code Bank 1**). *Save* your project and wait for the map preview to update.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1 #pois {

2 point-file: url(img/boat-launch.png);

3 }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code Bank 1: Using a Point Symbolizer Style to Load Images from Your Icon Library.**

Congratulations, you now have successfully drawn from your icon library! Unfortunately, all of your POIs currently are symbolized using the same icon. TileMill supports ***conditional******styling*** in two ways: (1) by attributes in the dataset and (2) by zoom level in the slippy map. Beginning with the former, open the *Layers* panel in the *Editor* tab and select the attribute table icon (the first of five icons) for the #pois layer. The information included in the *.csv* file will appear as a table, allowing for inspection of the attribute names (the headers) and the unique attribute values of each POI. For Lab #3, you will be using the nominal variable category created in Section 1 above; CartoCSS also supports conditional styling of numerical attributes, allowing for creation of thematic maps. Conditional styling makes use of bracket notation to specify the attribute and attribute value by which a particular style should be conditionally applied. **Code Bank 2** provides an example of conditional styling by the category attribute.

Two important notes about the syntax for conditional styling with CartoCSS: First, the attribute value must be surrounded by single quotes when a string (i.e., text, or a category attribute); this is not necessary for numbers (i.e., a numerical attribute). Second, the style rule must be contained within curly braces, including the terminating semi-colon.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1 #pois {

2 [category = 'Boat Launch']{point-file: url(img/boat-launch.png);}

3 [category = 'Golf Course']{point-file: url(img/golf-course.png);}

4 [category = 'Picnic Area']{point-file: url(img/picnic-area.png);}

5 }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code Bank 2: Conditional Styling using the category Attribute**

The second form of conditional styling is by zoom level. Returning to principles from G370, you know that the amount of abstraction should change with the map scale; in other words, a map with a large cartographic scale should have more detail, or be less generalized, than a map with a small cartographic scale. The pervasiveness of slippy map mashups requires cartographers to grapple with ***multiscale map design***, providing seamless transitions across scales that have the same aesthetics and style, but that add detail back into the display as you zoom into the map. You can create scale-dependent style rules using the zoom keyword, a numerical attribute that ranges from 0 to 22. **Code Bank 3** provides an example in which two different POI symbol designs are used at different scales. Before zoom level 10-11, a simple circle is used for the POIs using the marker symbolizer; beyond zoom level 12, the iconic point symbols are used. Consider how to make use of zoom levels to resize your icons such that they remain legible at all scales.

With your icons properly loaded, you can now turn your attention to styling the basemap linework. Take some time to experiment with the CartoCSS specification. CartoCSS supports application of multiple styles to the same feature using the double colon (::) notation, allowing you to build up sophisticated and highly aesthetic designs. It is important to note that if you do not make use of the double colon notation, the second style rule will override the first, just as with CSS.

A ‘cheat-sheet’ reference guide is available within the *Editor* tab. The complete CartoCSS documentation is available at: <https://www.mapbox.com/carto/api/2.1.0/>.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1 #pois {

2 [category = 'Boat Launch'][zoom >= 12]

3 {point-file: url(img/boat-launch.png);}

4 [category = 'Boat Launch'][zoom <= 11]

5 {

6 marker-width:6;

7 marker-fill:#00f;

8 marker-allow-overlap:true;

9 }

10 [category = 'Golf Course'][zoom >=12]

11 {point-file: url(img/golf-course.png);}

12 [category = 'Golf Course'][zoom <=11]

13 {

14 marker-width:6;

15 marker-fill:#009245;

16 marker-allow-overlap:true;

17 }

18 [category = 'Picnic Area'][zoom >=12]

19 {point-file: url(img/picnic-area.png);}

20 [category = 'Picnic Area'][zoom <=11]

21 {

22 marker-width:6;

23 marker-fill:#e41a1c;

24 marker-allow-overlap:true;

25 }

26 }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

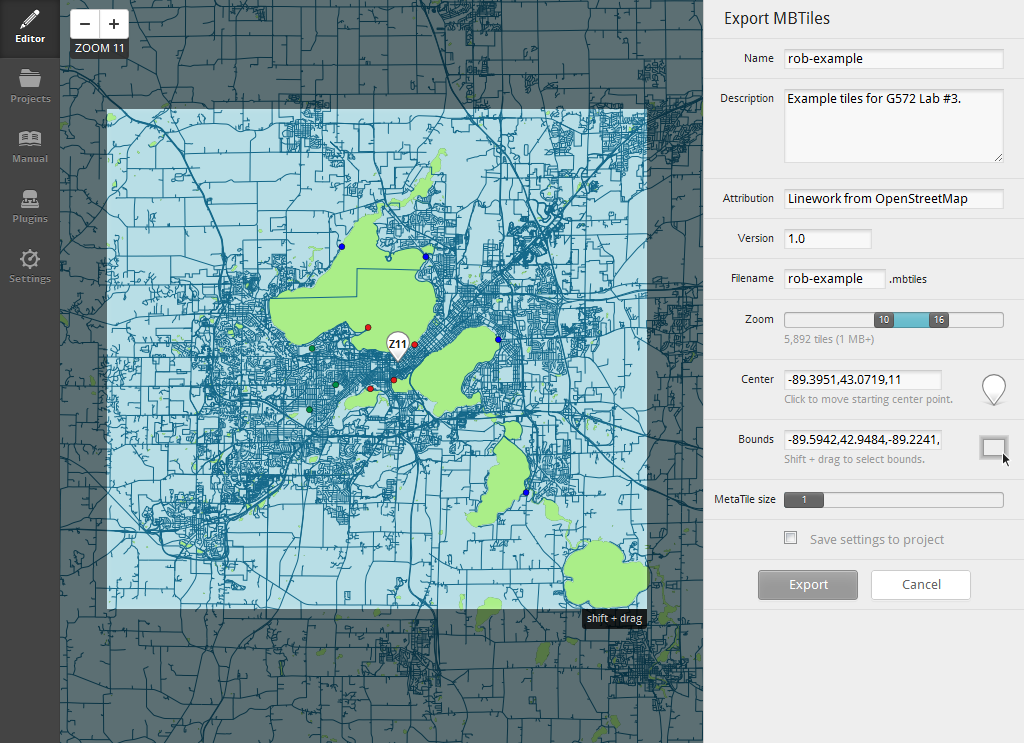
**Code Bank 3: Conditional Styling using the zoom Attribute**

**e. On Your Own: Adding Tooltips and a Legend**

Now that you have designed your tiles, it is time to practice your understanding of TileMill, the TileMill documentation, and HTML/CSS generally by adding interactive functionality for the POIs using tooltips and by adding a custom legend. Make use of the ‘name’ and ‘description’ columns in the *.csv* file when creating the tooltip. For discussion about basic tooltip and legend templates, see: <https://www.mapbox.com/tilemill/docs/crashcourse/tooltips/#legends>. For discussion about advanced legends, see: <https://www.mapbox.com/tilemill/docs/guides/advanced-legends/>. Solutions to both interactive tooltips and a legend will be discussed the week before Lab #3 is due.

**f. Exporting and Serving Your Tiles**

After finishing the design of your tiles, and adding tooltips and a legend, it is time to export and serve them for public viewing on the web. In the *Main toolbar*, click the *Export* button and choose the *MBTiles* option. This will open a dialog window for creating your tileset (**Figure 16**). Provide a logical *Name*, *Description*, *Version*, and *Filename* for your tiles; also list OpenStreetMap under *Attribution*. Importantly, you need to set the *Zoom*, *Center* (click map)*,* and *Bounds* (shift+drag map)of your tileset. Keep in mind that you need to restrict the number of tiles you are producing to 50MB, otherwise you will need to pay for tile service from MapBox or implement the open source [TileStache](http://tilestache.org/doc/) solution; you can inspect a preview of the number of tiles and the tileset filesize under the *Zoom* interface. Once you are satisfied with your export parameters, click the *Export* button. This will activate the rendering and export process; depending on the number of tiles you are exporting, the export process can vary from minutes to hours to even days. Be patient, it is well worth the wait!

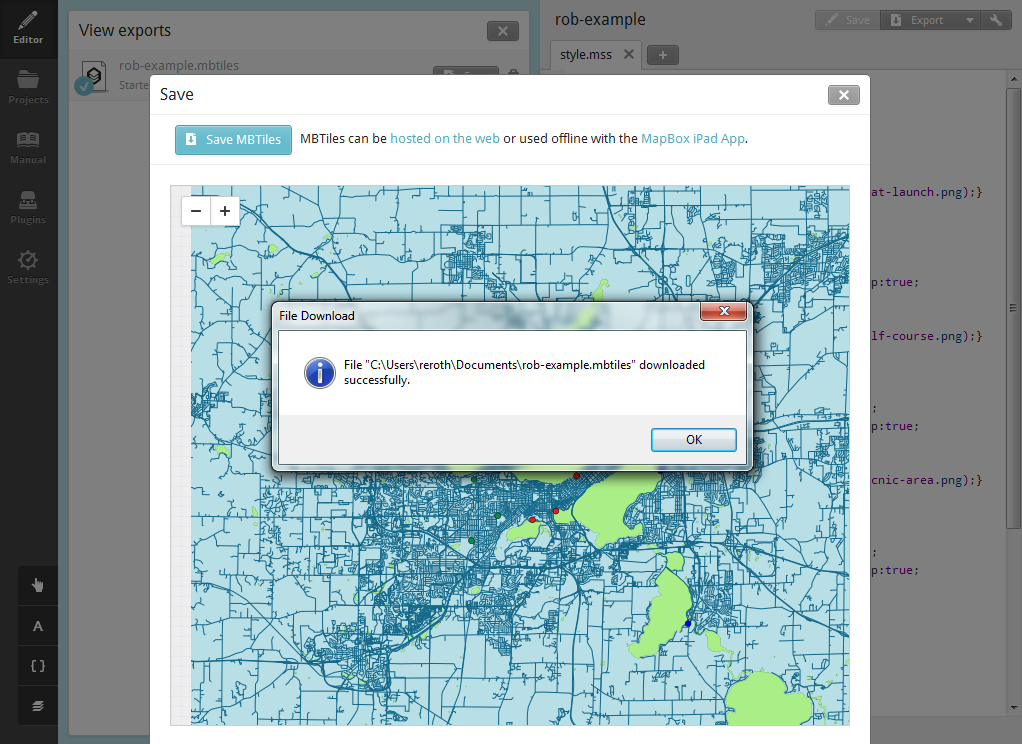


**Figure 16: Exporting Your Tiles.**

Once the tile rendering is complete, press the *Save* button, which will then navigate you to a preview page of your tiles. On the preview page, press the *Save MBTiles* button and be sure to take note of the file location at which your MBTiles are stored (**Figure 17**); *navigate to that location and save a copy of your MBTiles to your USB drive.* One saved and stored, navigate back to the MapBox website and login to your MapBox account.

Once logged in, click on the wrench icon in the top-right corner of your MapBox profile page and select the *Upload Layer* option. When prompted, select your stored MBTiles file and click *Upload File*; give your tileset a logical name once uploaded. You then will be redirected to a preview of your map tiles. Click on the pencil symbol to edit the settings of your tileset (**Figure 18**). Once the dialog window is activated, click on the eye icon beneath *Privacy* so that the tiles become *Public: Visible to everyone*. Save your changes and then click the *Publish* button.

Congratulations! You now are serving your own custom slippy map mashup. Finish the lab by embedding the map into your portfolio page using the provided HTML (i.e., the *Easy Embed*) from the *Publish* dialog window.

****

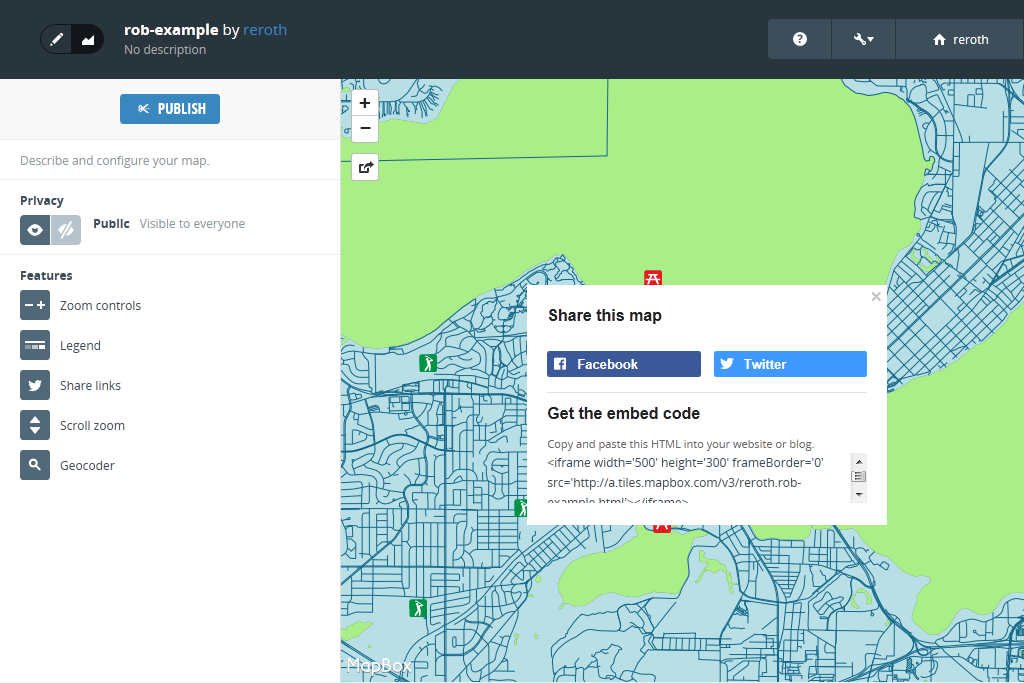
**Figure 17: Saving Your MBTiles.**

**3. return to MapBox website**

**2. note file location**

**1. *Save MBTiles***

***Edit Settings***

****

***Publish***

***Privacy***

**Figure 18: Publishing Your Slippy Map to the Web.**

**Evaluation Rubric: Slippy Map Iconicity Challenge (40pts)**

**Information Check-In (2)**

**2.0pts:** You have assembled your *.csv* file.

**0pts:** You have not assembled your dataset

**Due November 6th**

**Icon Library Check-In (6)**

**5.5-6.0pts:** The icon library is ready for use in TileMill. All four of the above considerations for icon design were taken into account. All icons hold up when rasterized into the *.png* format.

**4.5-5.0pts**: The icon library needs a few small tweaks before it is ready for use in TileMill. One of the four above considerations for icon design was not fully taken into account during design. Several icons break down when rasterized into the *.png* format.

**3.5-4.0pts**: We need to rethink aspects of the icon library before it is ready for use in TileMill. Several of the four design guidelines were violated systematically. Multiple icons break down when rasterized into the *.png* format.

**3.0 pts or less:** The submitted icon library did not meet the expectations of the assigned challenge, or was not submitted at all.

**Due November 13th**

**Slippy Map Mashup (32): Due Online November 20th**

**30-32pts:** The slippy map mashup is attractive, informative, and engaging; the map is a vast improvement over most web map mashups. The icon library unambiguously signifies the POIs and their higher-level categorization. The icons themselves are at the highest level of the visual hierarchy and hold up at all map scales. Your tileset is loading properly and reinforces the aesthetic style of your icon library. The tooltip and legend solutions are working properly and are well-designed. The aesthetic style of the tileset is consistent across zoom scales; what a stunning example of multiscale map design!

**26-29pts:** The slippy map mashup overall is successful as a web map. The icon library mostly works in representing the POIs and their higher-level categorization, but there are one or two icons that remain problematic. The icons work against the basemap and hold up at most, but not all, map scales. Your tileset is loading properly, as are the tooltip and legend solutions, although these solutions could be improved with further refinement. The aesthetic style of the tileset is mostly consistent across zoom scales, but there are unrefined transitions in linework styling between one or two zoom level changes.

**22-25pts:** The slippy map mashup only marginally improves upon typical web maps. Significant problems remain in the icon library, both in the representation of the unique POIs and their higher-level categorization. Many of the icons do not work against the basemap and/or across map scales. Your tileset is loading properly, but the tooltip and/or legend solutions are non-functional or illogical. Little effort was taken to develop an aesthetic style that is consistent across scales. Much more work needs to go into your linework generalization as you go from a large to a small cartographic scale (e.g., elimination of unneeded detail, simplification of lines and polygons, restyling the sizes or colors of features to maintain a proper visual hierarchy).

**Below 21pts:** The submitted slippy map mashup did not meet the expectations of the assigned challenge in multiple and critical ways. Please speak with Rob and Rashauna about strategies to improve the design.