HSU Web Map Documentation

Release Version: 1.0.0

Dec 15th, 2017

**Nathaniel Douglass**

**John Cortenbach**

**Kassandra Rodriguez**

**Point of Contact:**

**James Graham**

**Contents**

[**1 PURPOSE**](#_fr08k9m1gvlx) **3**

[1.1 Conceptual Framework](#_jeug92gisjm7) 3

[1.2 How to read this thing](#_g1zvi1rkp6g4) 3

[**2 GETTING STARTED**](#_17bzes3kjc0l) **4**

[2.1 Overview](#_221l07exhhkm) 4

[2.2 Accessing the Test Server](#_u2f9yjqyohe0) 4

[2.3 Contents](#_r4mmcg6islvu) 4

[2.4 Recommended Programs and Software](#_hyl04n3fiq6q) 5

[2.5 Recommended Practices](#_ujyxkdn4ovoq) 5

[**3 SPATIAL DATA**](#_ncauj1bh5enz) **7**

[3.1 Overview](#_9oavyd8irg7o) 7

[3.2 Datasets](#_v27mvrzb9mzk) 7

[3.3 Collection Methods (For sustainability point data)](#_vicag27hwbmr) 8

[**4 CANVASMAP.JS**](#_keimcu4z4ul2) **10**

[4.1 Overview](#_8cq5guusqq2) 10

[4.2 Contents](#_pqhjkm58ky5z) 10

[4.3 CampusMap.html](#_kvoq93xso68d) 10

[**5 INFO BOXES**](#_l9dlritxxzej) **12**

[5.1 Overview](#_a8xo8rtli06) 12

[6.2 Datasets](#_bsx7bz1mih7a) 12

[5.3 How to edit](#_l9cm4m6rup22) 13

[5.4 Formating](#_ej7od7nxqnc9) 14

[5.5 Adding a Photo!](#_sh72ovaonlj9) 14

[5.6 Inserting Links](#_l84cvk36i67z) 15

[**6 IMAGES/ ICONS**](#_kx09ftvoteus) **16**

[6.1 Overview](#_ooqrth6i7ute) 16

[6.2 Building Image Sizing](#_a0pu6rzeqz6g) 16

[6.3 Icon Sizing](#_utwq8h2ilj2j) 16

[**7 SEARCH**](#_uo7u4s7lc5pi) **17**

[7.1 Overview](#_agqqvxy2e749) 17

[7.2 Datasets](#_r3yxh2nc1jj4) 17

[**8 MENU BAR**](#_hwsizqo6u9b2) **19**

[8.1 Overview](#_j59kb16whsut) 19

[8.2 Type](#_hf03xsweoh9) 19

[8.3 Subtype](#_sp5184rjxakn) 20

[8.4 Rules](#_blaix8m07k5s) 20

[**9 PYTHON?**](#_yv3pkrydd4p) **22**

[9.1 Overview](#_hd2xap1v2v1) 22

[9.2 Requirement](#_9x8kqr583fa5) 23

[9.4 Contents](#_5df0j7cvpdbq) 24

[9.3 How to run](#_fz55cfui8xwy) 24

[9.4 Properly utilizing an ID system](#_jwab8k52093j) 25

[9.4 How the scripts work](#_iu9ljcjwjkyd) 26

[**10 BASEMAP (RASTER TILES)**](#_tamgx7ot9i56) **28**

[10.1 Overview](#_g11iq34n227t) 28

[10.2 Tiling Method from Mapublisher to BlueSpray](#_v4ih5e3z9732) 28

[**11 SUGGESTIONS FOR FUTURE RELEASES**](#_xrxjwxahl3tp) **33**

[11.1 Overview](#_war3y2e5g1w9) 33

[11.2 Suggestions](#_tua97r8u9gn2) 33

[**14 CREDITS AND ACKNOWLEDGEMENTS**](#_kp5ahlqy65b9) **34**

[Development:](#_axqj0hd7old6) 34

[Special Thanks to:](#_x6ftkgsnryye) 34

# 1 PURPOSE

## 1.1 Conceptual Framework

A map is a graphic representation of an area, drawn to scale and depicting the physical and social interactions. It uses colors, symbols, and labels to represent features found on the ground. The ideal representation would visualize every feature of the area being mapped. However, “A map is not the territory it represents, but, if correct, it has a similar structure to the territory, which accounts for its usefulness” (Alfred Korzybski, 1993). With this in mind, the Humboldt State Web Map does not, and will not, include everything that the campus has to offer. There are many things that *could* be showcased using this web based platform, and we have mauled over many of the endless possibilities, a lot. However, this map was designed to give incoming students, parents and non students a *brief* overview of the amenities, facilities and resources that HSU has to offer. We hope that this map provides a solid foundation for utilizing web based interactive platforms to display content and provide a hub for students to access. Please see Section 14 for info as to what could be improved and added to this project.

## 1.2 How to read this thing

Below you will find some steps to guide you in editing the web map and understanding what each file does as well as how they are connected. There are several examples to help including clips of code that you can directly copy and paste into a file that you might be changing. There is an overview for each section that gives you a brief introduction as to what to expect. The sections are separated based on what we believe are the most important points.

\*\*Note:

All datasets that are frequently updated or modified are renamed with an appropriate suffix that corresponds to the date in which it was changed (see example below). However, since these are so frequently updated, in this document we’ve decided to eliminate the date, in order to remove any confusion when searching for the appropriate dataset. You can always assume that the most recently dated name is the most current dataset (but who knows, it’s better not to assume).

|  |  |
| --- | --- |
|  | **Dataset Name** |
| **In this document** | BuildingOverlay.js |
| **In the HSUMap folder** | BuildingOverlay\_Nov12.js |

# 2 GETTING STARTED

## 2.1 Overview

This section will take you through some basic steps in how to access the HSU Web Map content, programs and practices. Since the map is a combination of many moving parts, it’s important to follow a few guidelines and tips, just in case you get lost. This document is far from complete, so feel free to add in new sections and troubleshooting tips when possible.

## 2.2 Accessing the Test Server

The HSU Web map is currently stored on the gis-hub-test server. To be granted access, please contact professor James Graham. When connecting to the test server, add this to wherever you enter new server locations:

*PC: \\gis-hub-test\wwwroot\HSUMap\*

*MAC: sftp://gis-hub-test/wwwroot/HSUMap/*

When testing changes, and just looking at the web map, add this to any browser:

<https://gis-hub-test/HSUMap/CampusMap.html>

## 2.3 Contents

|  |  |
| --- | --- |
| **Name** | **What it contains** |
| Folders | |
| Building\_Images | .jpg files of all images used for buildings |
| CanvasMap | Contents for the CanvasMap.js library used to create the map. Section |
| CanvasMapImages | Images called from CanvasMap.js (Only the arrow used in infoboxes) |
| css | Css files used to style web page and map |
| fonts | Additional fonts used |
| images | Holds the icons that appear on the map (the folder should really be renamed to icons) |
| Includes | CanvasMAp javascript files (Might be old or deprecated) |
| js | Javascript files used for search function and jquery |
| old | Any old or previously used folders or files. Used as a backup system |
| py | Potential Python scripts for future use |
| Spatial\_Data | Spatial data that appears on the map. Vector(Section 3) and raster(tiles) datasets. |
| Files | |
| CampusMap.html | Main html document |
| README.md | Useful information regarding contents |

## 2.4 Recommended Programs and Software

Editing Data:

* **QGIS**
* **BlueSpray**
* **Notepad++**
* **GIMP**

WebMap:

* **CanvasMap.js**

Editing Html:

* **Notepad++**

Python:

* **Python 2.7**

## 2.5 Recommended Practices

**The `Old` Folder**

* Each folder should have a subfolder named `old`, to store unused files
* When a file becomes deprecated or outdated, place it in the `old` folder rather than permanently deleting it (this will allow you to retrieve any accidentally discarded files).

**Dating Spatial Data**

* Date any commonly re-edited files, such as spatial data, with a suffix.
  + Example: BuildingLabels\_Dec\_12.

**Create a Checkout System**

* When editing a commonly re-edited file, make a copy of it first, make your edits, then rename it with a new date suffix.
* If two people are working on the same file, you can compare the two files after the fact, rather than overwriting someone else's changes.

# 3 SPATIAL DATA

## 3.1 Overview

All labels on map and their coordinates are coming from BuildingLabels\_Dec20\_2016.js The clickable polygons where images and descriptions come from isBuildingOverlay\_Nov12.js Then each buildings search phrase comes from BuildingPhraseList.js

Points that appear on the map, when checked in the menu bar, come from Points\_Dec\_12.js

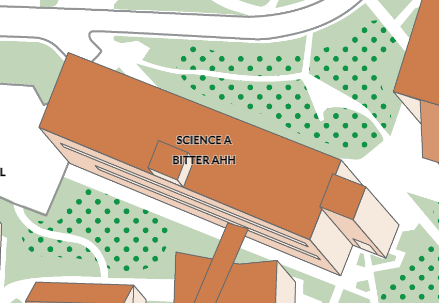
## 3.2 Datasets

-BuildingOverlay\_Nov12.js

Multipolygon Geojson, used for info boxes content such as the description and picture of each building. Can be easily edited in QGIS attribute table and when wanting to change something edit column “HTML”. <b></b> is bold text <p></p> is text <u></u> puts an underline <i></i> makes text italicized.

|  |
| --- |
| <div style="height:150x;width:150px">  <img src="Building\_Images/Little\_Apt.jpg" /></div> |

*This is an example of how to make images show up in the infoboxes just change the highlight to insert a new picture. They are retrieved from Building\_Images folder so be sure to add the images there first.*

-BuildingLabels\_Dec20\_2016.js Point geojson file used to label the buildings on the map. If a building label seems off, this is the dataset to change. In QGIS in the attribute table you can edit the building name in column “Labels\_2” and “Name”. 

“Labels\_2” is shown on the map when you are zoomed in while “Name” appears when you first open it. Use <br> to indent text such as SCIENCE A <br> BITTER AHH as shown in the figure.

-Points\_Dec\_12.js Used to project the icons on the map, within the attribute table in QGIS

the column “MenuIcon” has the photos of the icons for the drop down menu in the map

while “ICON” displays the photos of icons on the map itself. The photo icons are being

held in images folder so put any new icons there first to display it.

-Polylines\_Dec11\_2017.js Contains the polylines for the bus routes and access path, again if you want to change the menu icons just look to the images folder. Can be easily edited in QGIS.

## 3.3 [Collection Methods](https://docs.google.com/a/humboldt.edu/document/d/1xxBBST-aHVdcBMjaMJGA_tDCuxTaTxw4lFAlLMJHJv0/edit?usp=sharing) (For sustainability point data)

Field Data Collection Parameters

The GPS fieldwork was perform for the HSU Campus and Prof. James Graham. The performance was supervise by staff members that want to present types of features to be located, and recognize on the HSU map. To achieve the departments at HSU target accuracy, all collected GPS data were differentially corrected, in both present time and future process steps.

Naming Convention

The GPS data and attribute field names in said delivered to the departments of HSU that follow the same naming convention already applied to all existing data.

Identity



Compost Bin

Description

A portion of organic materials collected from two methods of composting on campus. The Facilities Management Department collects large quantities of compost per day on campus. They placing two different looking bins near the most recent dining locations where people would eat and relax. The department has a black metal (53,300 gallon) bin and a large green bin. Next, Waste-Reduction & Resource Awareness Program (WRRAP) is a student sustainability team that provides a small volume compost bin (5 gallons) in the department buildings that voluntarily request it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Shapefile Name | Type | Location | Description | Point Counts |
| Compost\_bins | Waypoint | (x,y) | Location | 32 |



Zip Car

Description

A car sharing system for HSU members to pick up a Zipcar vehicle at the provided location with a signpost stationed at the library circle, in front of the Student & Business Service Building and the Jolly Giant Commons parking lot. It is a membership program to allow accessible vehicle transportation for both HSU students and faculty on campus. To pick up a Zipcar is to first sign up at<http://www.zipcar.com/universities/humboldt-state-university> and get a $10 discount.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Shapefile Name | Type | Location | Description | Point Counts |
| Zip\_Car | Waypoint | (x,y) | Location | 3 |



Zagster Bike Share

Description

A bike sharing system to encourage bike riding at Humboldt State University. A full-service found on campus in front of the Jolly Giant Commons and the Harry Griffith Hall. To pick up a bike is to first sign up at <https://bike.zagster.com/arcata/> and reserve a bike.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Shapefile Name | Type | Location | Description | Point Counts |
| Zagster Bike Share | Waypoint | (x,y) | Number of Bike Racks | 2 |



Blue Lights

Description

Blue light emergency phones are located throughout campus. If you need assistance, press the large red round button on the front of the phone. All blue light emergency phones are connected directly to the University Police Department.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Shapefile Name | Type | Location | Description | Point Counts |
| Zip\_Car | Waypoint | (x,y) | Location | 3 |

# 4 CANVASMAP.JS

## 4.1 Overview

CanvasMap is a web mapping JavaScript library that allows you to create static and dynamic maps .The library allows for complete geospatial application support with scale bars, navigation, position information, and other features. CanvasMap was created and is maintained by Professor James Graham. For information and support, please contact him at [James.Graham@humboldt.edu](mailto:James.Graham@humboldt.edu) and for more information on CanvasMap.js, visit: [CanvasMap](http://gis-hub-test/cwis438/Includes/CanvasMap/Documentation/CanvasMap.html)

## 4.2 Contents

In the HSUMap folder you will find a subfolder that contains the web mapping library that powers the HSU Web Map. In the folder you’ll find:

**css -** style sheets for the elements of a CanvasMap

**Images -** icons used in the map (arrows)

**js -**the JavaScript files called from the main CampusMap.html document

**Lib -**additional libraries

## 4.3 CampusMap.html

Much of the CampusMap.html is writen in JavaScript, calling functions from a variety of .js documents held in the .../HSUMap/CanvasMap/js/ folder. All of which takes place in between the <head> tag. This helps code load faster in the beginning when the browser loads, rather than loading scripts halfway through the user’s experience. Example:

<head>

<script>

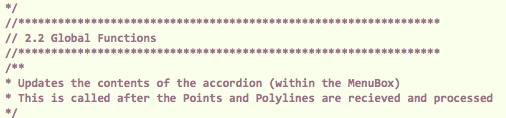
ALL OF THE JAVASCRIPT CODE...

</script>

</head>

This will explain what portions of the CampusMap.html document correspond to various important elements of the map/webpage. Look for these section headers:





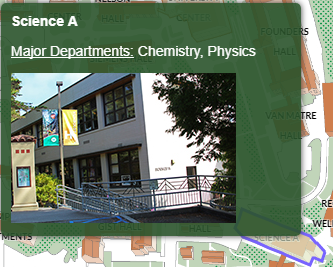


****

# 5 INFO BOXES

## 5.1 Overview

The infoboxes are the pop up descriptions and pictures of buildings when clicked in the map. As shown in the figure below, they describe the name and what each building offers. The data for this is coming from BuildingOverlay\_Nov12.js which is found in the Spatial\_Data folder. This file can be edited in ethier QGIS or Notepad but is more organized in QGIS. The Info Boxes are formatted in HTML (Hypertext Markup Language), so be sure to follow the specific formatting tips (**Section 6.4**)



## 6.2 Datasets

-BuildingOverlay.js

Found in Spatial\_Data folder. Used for info boxes content such as the description and pictures of building. Can be easily edited in QGIS or a text editor and when wanting to change something edit column “HTML”. If you want to add a new photo remember to resize the image first, more instructions on that in section **7.2**

|  |
| --- |
| <div style="height:150x;width:150px"><img src="Building\_Images/Little\_Apt.jpg" /></div> |

This is an example of how to make images show up in the info boxes just change the highlight to insert a new picture. They are retrieved from Building\_Images folder so be sure to add the images there first.

-SlideBoxes.css Is used for styling the info boxes, can be found in css folder.

## 5.3 How to edit

|  |
| --- |
| **With QGIS** |

**Step 1**. Open QGIS

**Step 2**. Drag and drop `BuildingOverlay.js` to QGIS then on the left hand side there is a layers panel that displays the filename, right click it.

**Step 3**. Click on properties and the layer properties window opens. On the left hand side clicks on Fields Tab.

**Step 4.** This opens the field columns of Building Overlay, to change the image associated with a building, click under Edit Widget for image column.

**Step 5**. This opens Edit Widget Properties window and on the left hand side select photo and leave it to default settings and hit ok.

**Step 6**. This should now close and show that the image column has photo labeled under Edit Widget. Click ok to close Layer properties.

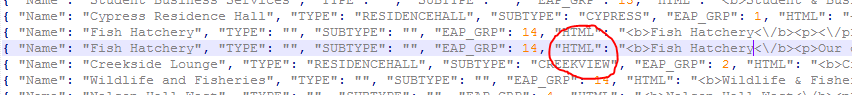
**Step 7**. Right click the filename again and open attribute table. To edit the content in the infobox click the pencil on the top left corner to toggle edit mode, look to the `HTML` column and there you will notice the content that is being displayed including the photos and description.

|  |
| --- |
| **With a Text Editor (Notepad++, DW, BBEdit …. *The list goes on*)** |

**Step 1**. Open Text editing software

**Step 2**. Click file > open > “.../Spatial\_Data/BuildingOverlay.js”

**Step 3**. Scroll down and you’ll notice each building being represented in each line. When you’ve located the desired building, scroll to the right to find the “HTML”: property:



**Step 4.** Everything after the “HTML”: property can be edited as long as it’s kept in “ ” quotation marks.

**Step 5**. If pasting HTML code into this property field, be sure to add an extra “\” before every “/”

*Example:*

**

**Step 6**.Since BuildingOverlay.js is stored as a GeoJSON file which is a language-independent data format and can often conflict with HTML formatting. Due to this, where normal HTML looks like this:

|  |
| --- |
| <b>Fish Hatchery</b><p>Our on-campus Fish Hatchery is a unique ... </p> |

Within a JSON/GeoJSON file it should look like this:

|  |
| --- |
| <b>Fish Hatchery<\/b><p>Our on-campus Fish Hatchery is a unique ... <\/p> |

Did you catch that?

**Step 7**. Save the file and be sure to test your changes

.

## 5.4 Formating

|  |  |
| --- | --- |
| <b>text</b> | **Bold text** |
| <p>text</p> | Regular text |
| <u>text</u> | Underlined text |
| <i>text</i> | *Italicized text* |
| <br> | Adds a break in text |

The `HTML` column should look something like this:

*Example:*

|  |
| --- |
| <b>Fish Hatchery</b><p>Our on-campus Fish Hatchery is a unique ... </p> |

## 5.5 Adding a Photo!

*Example:*

|  |
| --- |
| <div style="height:150x;width:150px"><img src="Building\_Images/fishhatchery3.jpg" /></div> |

And simply change the highlighted part to the new photo name. The photo should first be in Building\_Images folder before it will actually display.

## 5.6 Inserting Links

If you’d like to add links to the infoboxes copy and paste the text below into the HTML column of the BuildingOverlay.js:

<style>

a:link {

color: white;

background-color: transparent;

text-decoration: none;

}

a:visited {

color: white;

background-color: transparent;

text-decoration: none;

}

a:hover {

color: white;

background-color: transparent;

text-decoration: underline;

}

</style><a href="https://library.humboldt.edu/" target="\_blank"><b>Library</b></a>

Change the highlights to the corresponding website and whatever text you’d like shown.

# 6 IMAGES/ ICONS

## 6.1 Overview

-Building\_Images folder contains the photos used for BuildingOverlay\_Nov12.js this is where

you add any new photos to be shown on the info boxes

-images folder contains the photos for the icons in the map used for Points\_Dec\_12.js such as the basketball, rally point, printing kiosk, etc.

## 6.2 Building Image Sizing

When you want to add a new photo into the info box you must first resize the image before doing so. The easiest way to do this would be by using GIMP, scale the image down to 238x150 for landscape pictures and for portrait down to 150x225.

## 6.3 Icon Sizing

**Map Icons:**

Ex: `images/ZagsterBike\_Map.png`

Resolution: 72ppi

Size: 22x22 px or .35x.35inch

**Legend Icons:**

Ex: `images/ZagsterBike\_Legend.png`

Resolution: 72ppi

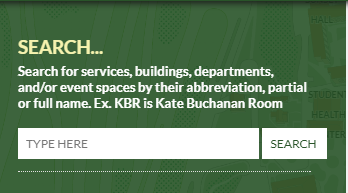
Size: 18x18px or .25x.25inch

\*\*Consider changing icons to .svg or .avg for better results

# 7 SEARCH

## 7.1 Overview

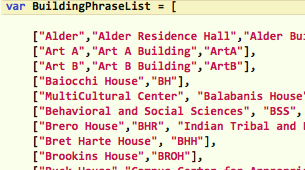
A search element that will accept user inputs to be searched for in a database to direct users to its respected building. (BSS, Science A,B,C,D, Founders Hall, etc.) The js folder is where the function and attributes are held.

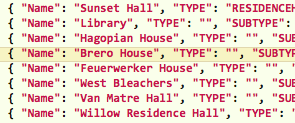


## 7.2 Datasets

-BuildingPhraseList.js Found in js folder, is used for finding the building location from the search bar. Can be opened in Notepad++ and edited there if there are any new phrases to input. Left hand side of the file is where you add new phrases, on the right is where it will be directed to.

\*\**The first item in each index represents the building it is associated with. If you plan to change the first item, it must also match the spelling of the ‘Name: ’ property in the BuildingOverlay.js file!!!*





-SearchUtil.js Is the search function code used to find the building locations. Uses BuildingPhraseList.js as the search terms.

-jquery-3.1.1.min.js This is a jquery that takes javascript code that would take up many lines and simplifies it to make it easier to use javascript on your website. It’s required for the SearchUtil.js file

-jquery-1.11.0.min.js This is also the same idea however it will be your job to figure out how it’s connected to the SearchUtil.js

# 8 MENU BAR

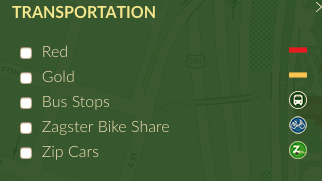
## 8.1 Overview

The menu bar is a graphic control element that provides access to functions like checkboxes and drop-down menus that activate various menu choices. Each item is divided up into various sections titled “TYPE” and “SUBTYPE” and “NAME”. These delineators are found in the attributes of each spatial data file. There are some rules that apply to these sections.

# 

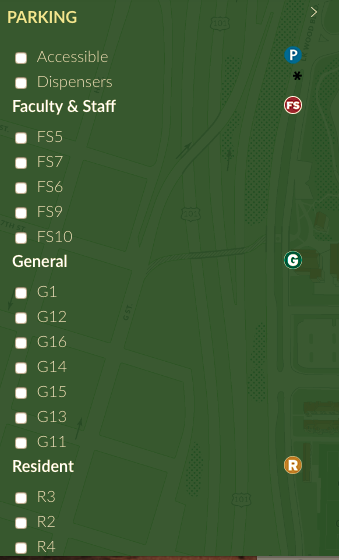
## 8.2 Type

A feature that is assigned a “TYPE:” property will appear in its corresponding section in the menu bar. Example: If a bus line shows “TRANSPORTATION” in the “TYPE” property, it will show up in the TRANSPORTATION section:



## 8.3 Subtype

Subtype refers to the sub heading/section that appears in some of the TYPE sections. Example: if a series of features are given the same SUBTYPE property, like “Faculty & Staff” in the PARKING section, then it will appear like this:



## 8.4 Rules

Seems easy right? However, when the “NAME” field is involved, it gets a little tricky. Let’s take the Bus Stops for example. Their “NAME” field is left blank, but their “SUBTYPE” is filled with the string “Bus Stops”:



Rather than placing a bunch of SUBTYPE headers (like you see with Faculty & Staff parking), one name is used for ALL of the Bus Stop point features.

If both “NAME” and “SUBTYPE” are filled, then you get something like the “Services” “SUBTYPE” seen in the “STUDENT SERVICES” “TYPE” this:



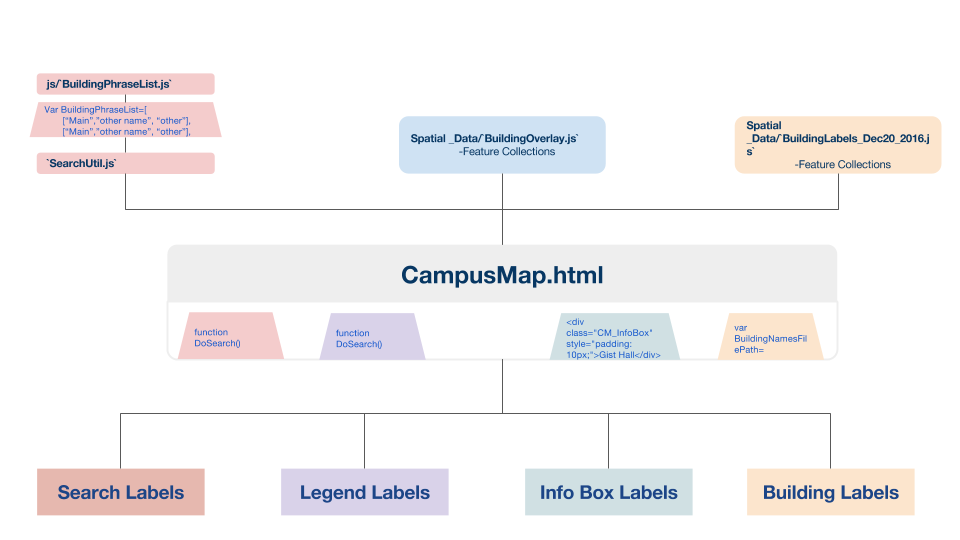
# 

# 9 PYTHON?

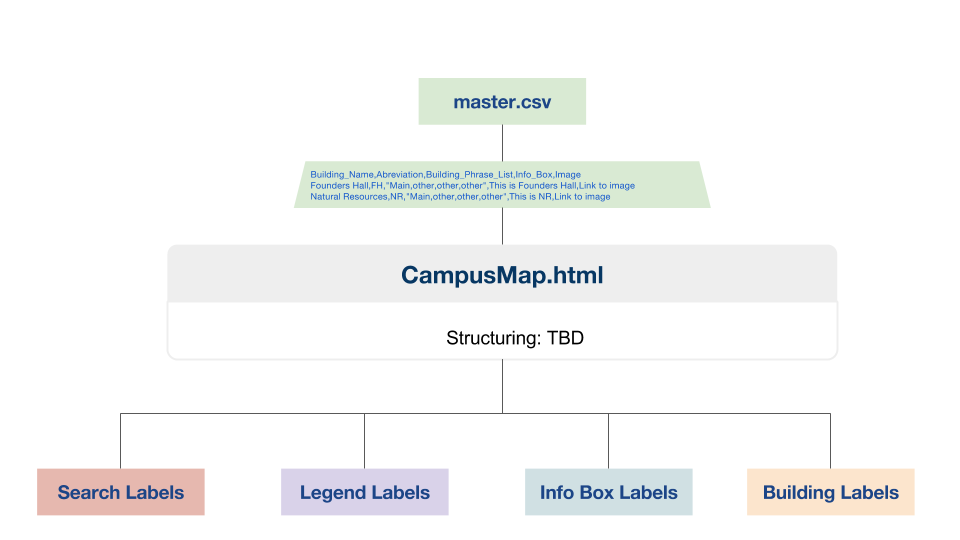
## 9.1 Overview

Due to time constraints, a fully functioning python based updating system was not able to be released by version 1.0 of the HSU Web Map. So we hope that future teams will take on this component. ***Here’s the gist’ of it:*** Since the method for updating labels, point features, and infobox descriptions requires at least a basic level of understanding in terms of GIS knowledge and or HTML/CSS, it would be nice to have a system that was a little simpler.

*How labels on the map, info text and search terms are currently being displayed:*

****

*An alternative:*

****

Building labels, infobox descriptions and search terms come from these datasets:

BuildingOverlay.js,

BuildingLabels.js,

BuildingPhraseList.js

Rather than editing these individually, a better method would be to edit a single .csv file (master.csv) which would have columns for all the necessary components, then a script that would take those values and write them to the appropriate dataset. Sounds great, right?! Unfortunately, there were a few hitches and wasn’t able to be released in time. So here’s what was started.

## 9.2 Requirement

The scripts found in the HSUMap/py folder contain a series of scripts that were developed on a MAC OS and tested in Bash(Unix Shell), therefore they won’t work immediately on a PC or in Command Line (Windows)

* Written using:
  + Python 2.7
* Uses these python libraries:
  + time
  + sys
  + logging
  + os.path
  + csv
  + pprint
  + geojson
  + Json

## 9.4 Contents

Scripts:

**feature\_id.py** - checks spatial data to see if each feature has a unique ID

**create\_csv.py** - creates an up to date Master.csv list with current info from data

**update\_js.py** - updates the appropriate spatial data according to the master.csv

Files:

**Id\_sheet.csv** - houses the unique ID’s that are currently used and still available

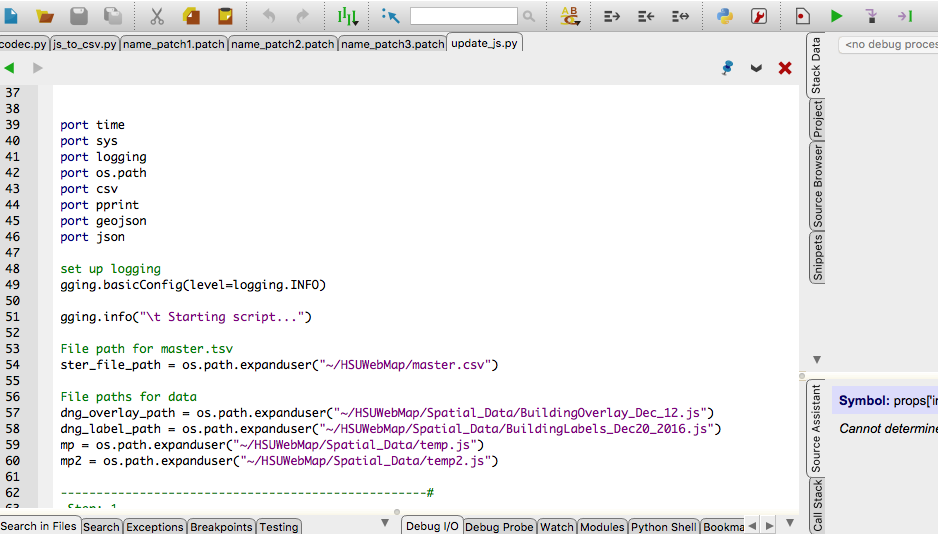
**Master.csv** - change feature names and infobox descriptions for update\_js.py

**README.md** - info about the scripts

## 9.3 How to run

|  |
| --- |
| **With WingIDE** |

1. **Open WingIDE**
2. **File > Open > .../HSUMap/py/feature\_id.py (or any other script)**
3. **Click Run!**

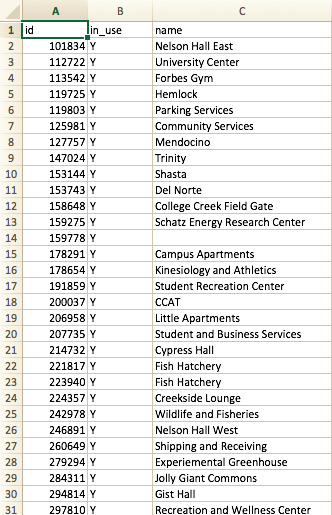
****

|  |
| --- |
| **With Terminal or Command line** |

1. **Open Terminal or Command Line**
2. **Type ‘python’**
3. **Add a space ‘ ’**
4. **Drag the desired script into the window**
5. **Click ‘Enter’!**

## 9.4 Properly utilizing an ID system

Since building names and features might change over time, it’s important to use an ID system to tag each feature. Rather than trying to join data from multiple files, based on their name, (which might be misspelled from one file to the next) giving each feature a unique ID number that is consistent across all datasets will eliminate many issues.

**Id\_sheet.csv: **

* This file has a total of 1,482 (484 currently in use) unique feature ID’s that were generated from [Random.org](https://www.random.org/strings/?mode=advanced)
* We doubt that HSU will grow expenotially and aquire 998 more buildings/features, but you never know! If that does happen and you run out of ID’s, you can always generate more in Excell
* The ID’s have no rhythm or rythm, they are completely random and are simply used to identify each feature uniquely
* If you notice that a feature doesn’t have an ID, add the next available one and update this sheet!
* OR use the **feature\_id.py** script to update it automatically

## 9.4 How the scripts work

|  |
| --- |
| **Feature\_id.py:**    **Step 1.**   * checks id\_sheet.csv for free or 'available' ID's   **Step 2.**   * loads selected .js file * checks for ID property * If there is no ID property, or the ID property is null, then it assigns it the next available ID   **Step 3.**   * Update id\_sheet.csv to reflect id usage |
| **create\_csv.py**  Since the two most important datasets, in terms of labeling data, are BuildingOverlay.js and BuildingLabels.js, the script runs through each data set and collects information from these properties:    The script then creates a .csv file that houses this data, all organized by the feature ID |
| **Update\_js.py (UNFINISHED):**  Takes the information from master.csv script and re-maps it back to the appropriate geojson files. The script is not fully working, and needs extra attention. Going through by each ID should yield the correct information, but re-writing a new .js file is the key. |
|  |
|  |
|  |

# 10 BASEMAP (RASTER TILES)

## 10.1 Overview

Raster tiling is a method used to take large raster datasets and convert them into small manageable squares, that are then pieced together programmatically when the viewer zooms in and out. These squares (tiles) often are stored as .png files and are stored in the range of 1,000 to 10,000+ images. For the HSU web map, the initial layers were given by the Marketing QIS software, then brought into Adobe illustrator using [**Mapublisher**](https://www.avenza.com/mapublisher/) **(**a spatial data plugin for adobe products).

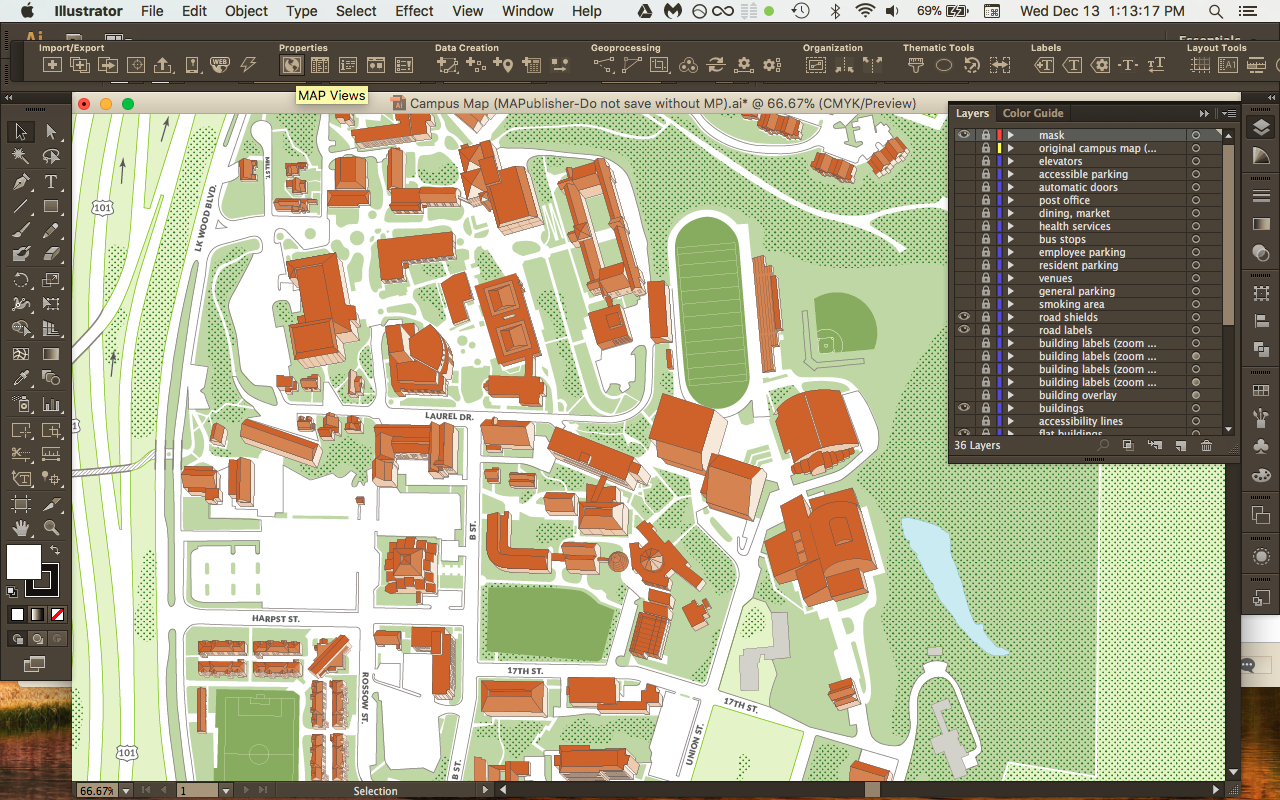
## 10.2 Tiling Method from Mapublisher to BlueSpray

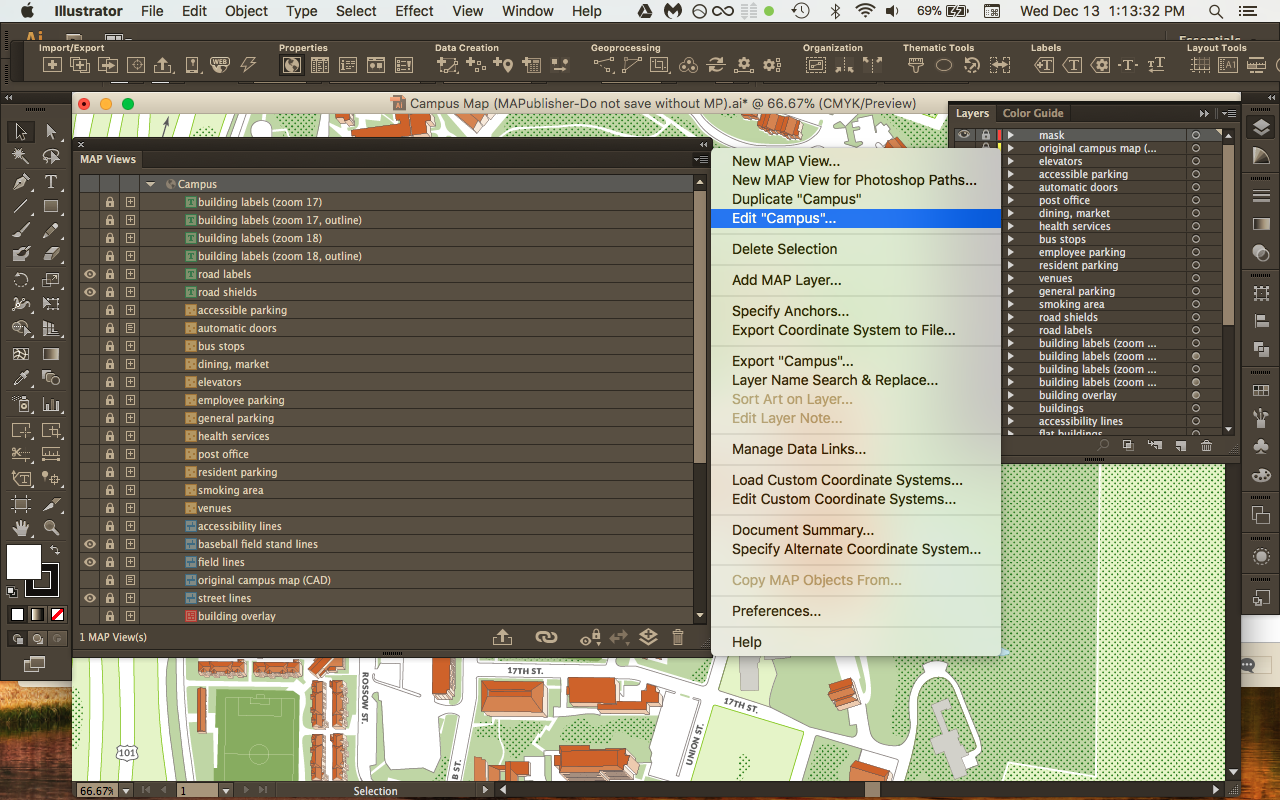
**In Mapublisher:**

Open the AI file found here:

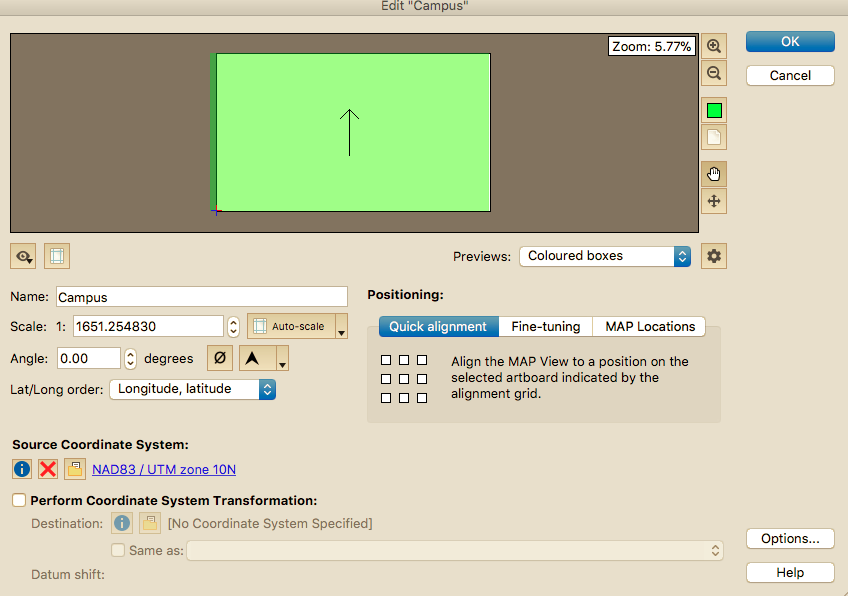
[**https://drive.google.com/open?id=0B5cUc0u0qgj\_WWU3VXR2VUtpaVk**](https://drive.google.com/open?id=0B5cUc0u0qgj_WWU3VXR2VUtpaVk)

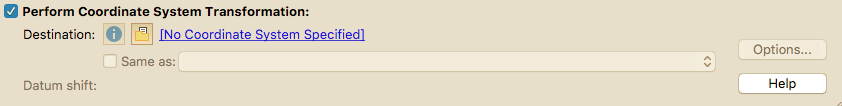
Be sure that the spatial reference is set to Nad83\_UTM\_Zone\_10N, by going to the Map Views tab. Then clicking “edit” in the top right menu button in the Map View window:



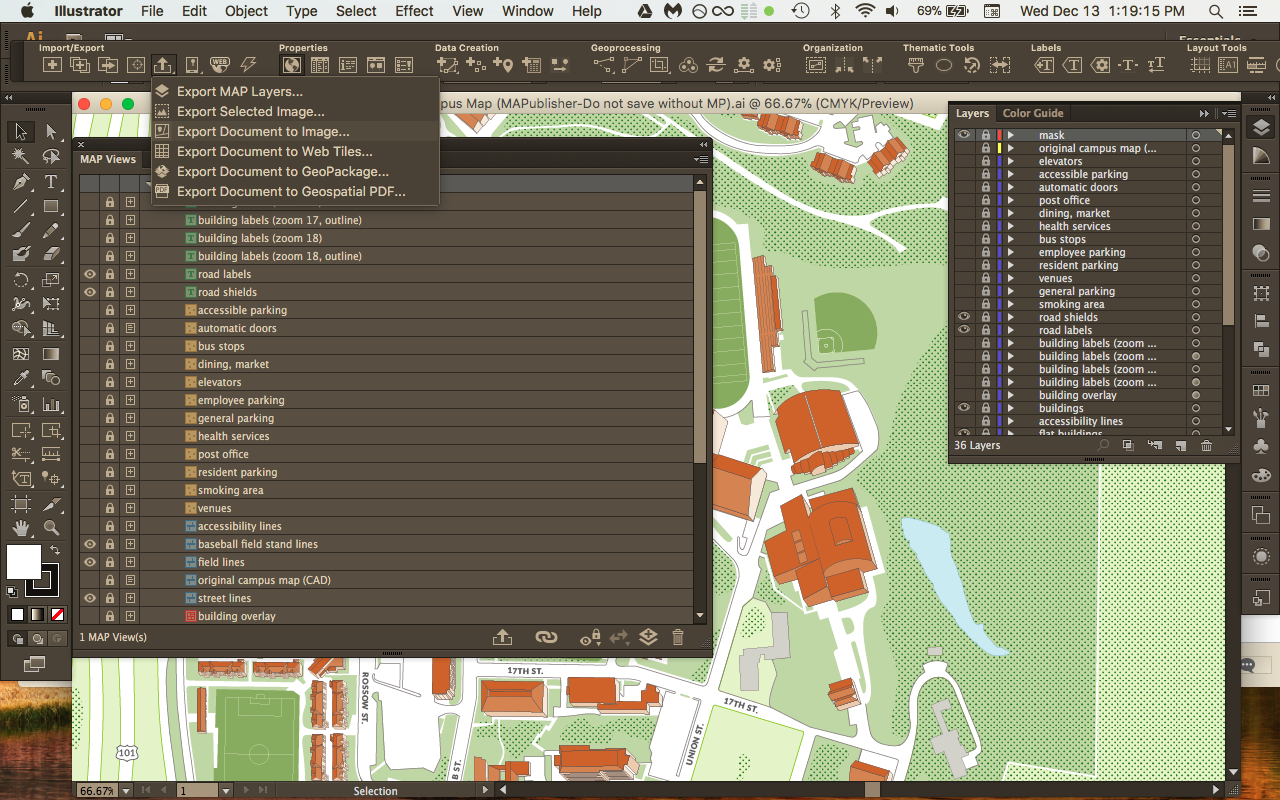
****

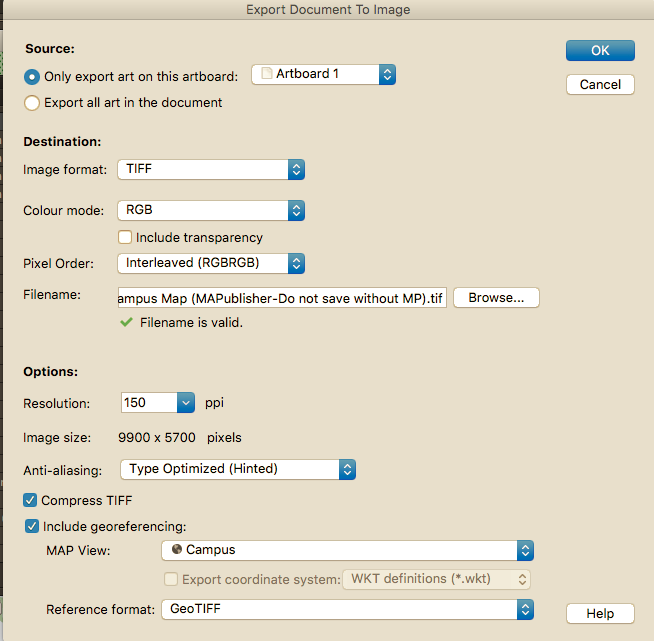
**You can then edit the spatial reference if needed:**

****

****

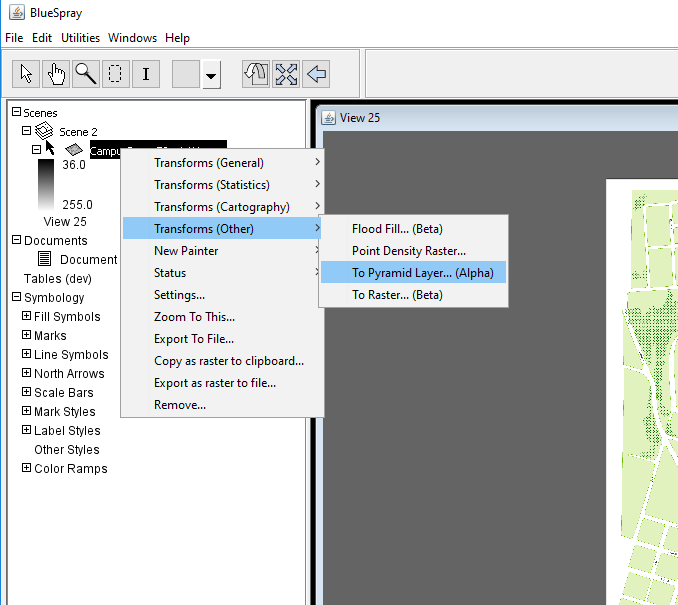
Export the map as a .tif using the export as image button. Be sure to export three versions with three different resolution sizes(72ppi, 150ppi, and 300ppi):



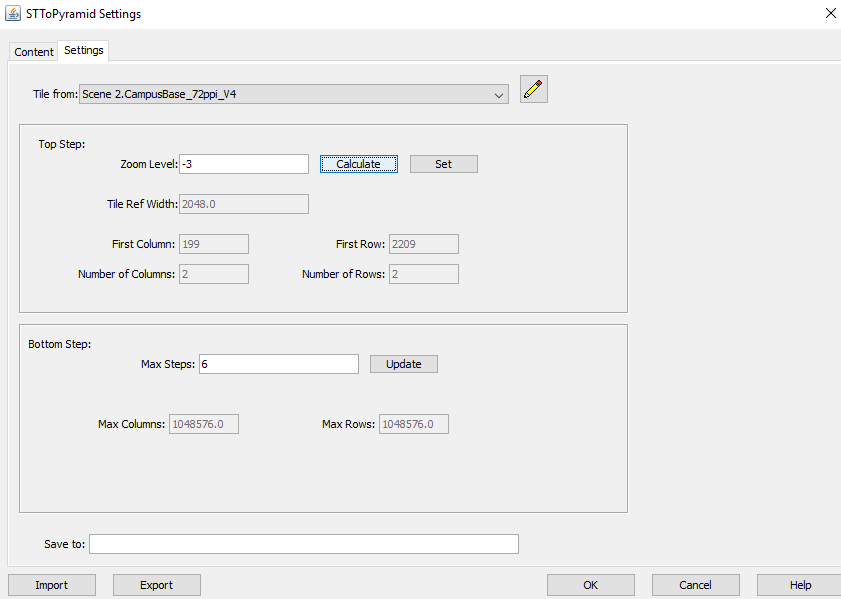


**Blue Spray (creating tiles):**

Drag and drop each image into the BlueSpray window. Right click on the Layer on the left hand side > Transforms (Other) > To Pyramid Layer



In the To Pyramid window, be sure to follow the top level and bottom level zoom ranges:



**Follow these settings for each raster:**

300ppi

* + Top Level Zoom level -3
  + Bottom Level Max steps 6

150ppi

* + Top Level Zoom: Level -3
  + Bottom: 5

72ppi

* + Top Level Zoom Level -3
  + Bottom: 4

# 11 SUGGESTIONS FOR FUTURE RELEASES

## 11.1 Overview

The campus web map is a student oriented project focused on building coding skills for the Advanced GIS class. Therefore, the webmap is an ongoing effort to improve the next version with students help and will hopefully be released soon to the public. Below are some suggestions to upgrade the map. Keep in mind that, the more that is added, the more that must be maintained. With so many hands in the project, it’s important to have a set of guidelines and systems in place to best update the map in the future

## 11.2 Suggestions

* Regarding the infoboxes, we were only able to insert links to a fewbuilding descriptions but the next group can return to this and add missing links to department websites and programs that each building references. It would certainly help make the website more useful and this step is under InfoBoxes section **6.3 How to Edit**.
* If any more professional photos come in for buildings those can be changed. Some of the photos were taken by this group with a simple cell phone camera, again to edit this, it’s found under InfoBoxes section.
* The search bar currently doesn’t display searches identified in the tabs, for instance when you search for coffee or compost bins it doesn’t display anything. This is because the search zooms to a single building. Adding extra code to allow for non building specific items could be considered
* Clean up the way BuildingOverlay.js is formatted, currently all text and pictures are jumbled onto one column which is “HTML” so separating it to be more easily legible would certainly help.
  + One tip would be to have seperate .html documents for each infobox content in the `HTML` attributes. Then having a script that reads the contents of each separate .html document.
* Finishing a script to utilize a master .cvs that would organize all the important spatial data and make it easier for administration to change items.

*We hope this document has been helpful. Please use this as a template for future releases.*

# 14 CREDITS AND ACKNOWLEDGEMENTS

#### Development:

James Graham

Jordan Adir

Monique Gill

Aaron Taveras

Joshua Rodriguez

Dylan Hill

Nathaniel Douglass

Kassandra Rodriguez

John Cortenbach

Colby Peffer

#### Special Thanks to:

CCAT

Phone: (707) 826-3551 I [ccathsu@gmail.com](mailto:ccathsu@gmail.com)

Dining Catering Services

Phone: (707) 826-3451 I [rlr4@humboldt.edu](mailto:rlr4@humboldt.edu)

Facilitates Management:

Morgan King I Sustainability & Climate Action Analyst I [Morgan.King@humboldt.edu](mailto:Morgan.King@humboldt.edu)

Katie Koscielak I Sustainability Analyst I 707-826-5945 I [kmk928@humboldt.edu](mailto:kmk928@humboldt.edu)

Marketing and Communications:

Kristen Gould | Creative Director | 707.826.4177 | kristen@humboldt.edu

Oh-SNAP

Phone : (707) 826-4556 I [ohsnap@humboldt.edu](mailto:ohsnap@humboldt.edu)

PSC

[powersavehsu@gmail.com](mailto:powersavehsu@gmail.com)

WRRAP

Phone: I [wrrap@humboldt.edu](mailto:wrrap@humboldt.edu)

ZIP Car

<http://www.zipcar.com/universities/humboldt-state-university>