

**DEVELOPING A NEW ONLINE DISTRIBUTION METHOD FOR MULTIBEAM  
DATA**

by

James Muggah

**B.Sc. Biology, St. Francis Xavier University, 2007**

**Advanced Diploma in GIS, NSCC's Centre of Geographic Sciences, 2008**

A Report Submitted in Partial Fulfillment of  
the Requirements for the Degree of

**Master of Engineering**

**In the Graduate Academic Unit of Geodesy and Geomatics Engineering**

Supervisor: John E. Hughes Clarke, Ph.D., Geodesy and Geomatics Engineering

**THE UNIVERSITY OF NEW BRUNSWICK**

**September, 2010**

©James Muggah, 2010

## **ABSTRACT**

Since 2003, all underway multibeam and sub-bottom data from the Canadian Coast Guard Ship Amundsen has been posted online within approximately six months of the end of each cruise. Two custom interfaces were developed to allow users to view the data. The first was stripmaps, showing 25 by 5 kilometre mapsheets, with two different sun-illuminations for bathymetry, backscatter, and properly referenced sub-bottom data. The second interface, providing access to 15' latitude by 30' longitude mapsheets, was implemented in 2006. This interface allowed users to download the bathymetric and backscatter data at 10 metre resolution. While this interface matched the underlying data management scheme implemented at the University of New Brunswick, the zoom and pan capability was at a fixed scale with limited contextual data.

In the past few years, with the introduction of web-based geographic information systems (GIS) (e.g. Google Maps, Yahoo Maps, Bing Maps), there have been thousands of maps published online. These online GIS programs are a suitable platform to display the seven years of Amundsen coverage within the context of the GIS-served satellite imagery and allow the user to freely browse all data in a familiar interface. The challenge, however, for serving up third party data through these map engines is to efficiently cope with the multiple zoom levels and changing resolutions.

Custom tiling software was developed to take all the raw data from the seven years of Amundsen (and others') multibeam coverage and convert it into multiple scale resolution images suitable for interpretation by Google Maps. The images were stored in a pyramid structure utilizing Google's map projection and uniquely named to reflect their georeferencing and resolution. This image pyramid is then accessed by Google Maps according to the user's current zoom level to optimize visualization. This multi-resolution data is served up on demand from the University of New Brunswick for dynamic overlay on Google's satellite data. Point overlays were developed to show each stripmap, adding to the functionality of the website by providing users the full picture of the seafloor (topography and underlying sediments).

This web interface allows any interested parties to easily view multibeam and sub-bottom data from the Pacific Ocean through the Canadian Arctic Archipelago and into the Atlantic Ocean.

The broad overview helps to understand regional trends and then focus on areas of interest at high resolutions to see particular features. The web interface also provides a link to the 15' by 30' mapsheet model with full source traceability and download capability.

## **ACKNOWLEDGEMENTS**

I would like to thank Ian Church for his continued help with understanding the Ocean Mapping Group's code and developing the custom tile creation program. Also John Hughes Clarke and Jonathan Beaudoin for their input.

Members of the Ocean Mapping Group who collected and processed the data. ArcticNet and the Amundsen provided a platform for data collection and research. Prime funding for this research was from Imperial Oil Limited. The sponsors of the Chair in Ocean Mapping: U.S. Geological Survey, Kongsberg Maritime, Route Survey, Canadian Navy, Rijkswaterstaat. Additional funding from Geological Survey of Canada, NRCan, Canadian Hydrographic Service, and DFO.

# TABLE OF CONTENTS

ABSTRACT.....	ii
ACKNOWLEDGEMENTS.....	iv
TABLE OF CONTENTS.....	v
LIST OF FIGURES .....	vii
CHAPTER 1. INTRODUCTION .....	1
CHAPTER 2. BACKGROUND .....	5
2.1 CURRENT DISTRIBUTION METHOD .....	5
2.2 PROPOSED DISTRIBUTION METHOD .....	8
CHAPTER 3. METHODS .....	16
3.1 TYPES OF OVERLAYS .....	16
3.1.1 – Ground Overlay Method .....	17
3.1.2 – Tile Overlay Method.....	17
3.2 CREATION OF TILES .....	19
3.2.1 – Tile Bounds Calculations .....	20
3.2.2 – Creation of Tile Images.....	23
3.2.3 – Filling the Gap in the Tiles .....	25
3.2.4 – Collapsing of Tiles .....	27
3.3 POINT OVERLAYS .....	29
3.4 WEBSITE .....	31
CHAPTER 4. RESULTS .....	37
4.1 SPEED OF WEBSITE AND TILE CREATION .....	37
4.2 SIMPLIFICATIONS AND FUTURE DIRECTIONS.....	39
4.2.1 - Creation of Backscatter Tiles.....	40
4.2.2 - Creation of Tiles Using the 100 Ping Bounds .....	40

4.2.3 - Implementing weigh_grid Options .....	41
4.2.4 - Developing an Update Procedure After Each Field Season .....	43
4.2.5 - Developing a New Colour Scale.....	43
4.2.6 - Collapsing r4 Files .....	44
4.2.7 - Download Option .....	45
4.2.8 - Using the Marker Manager for Stripmap Overlays .....	45
4.2.9 - Maintain and Update the Website.....	46
CHAPTER 5. CONCLUSIONS .....	47
BIBLIOGRAPHY .....	48
APPENDIX I – WEBSITE CODE .....	51
APPENDIX II – TILE CREATION CODE .....	60
APPENDIX III – FILL GAP CODE.....	64
APPENDIX IV – COLLAPSE CODE.....	65
APPENDIX V – STRIPMAP CREATION CODE .....	68

## LIST OF FIGURES

Figure 2.1 - The ArcticNet Strip map website that was developed in 2003 to show multibeam and sub-bottom data.....	7
Figure 2.2 - The ArcticNet Basemap Series website that was developed in 2006 to show bathymetry and backscatter data.....	8
Figure 2.3 – The four main free web-GIS programs used by developers to create custom maps online.....	12
Figure 2.4 – The three different background images used by Google Maps. The top left is the Map View, the top right is the Terrain View, and the bottom is the Satellite View.....	13
Figure 2.5 – The array of overlays, white tiles, Google’s satellite tiles, and Ocean Mapping Group tiles as they appear in Google Maps. Google’s Ocean imagery exists on the right (blue background), but disappears on the left (white outlined tiles). The array of overlays was used to show the Ocean Mapping Group’s tiles in the areas where Google did not have Ocean imagery.....	15
Figure 3.1 - Tile creation for Google Maps, each tile is subsequently divided into four new tiles.....	18
Figure 3.2 - The pyramid structure used by Google Maps which varies the resolution of the image according to the zoom level and latitude.....	19
Figure 3.3 – Defining the tile bounds used in the calculations discussed in section 3.2.1.....	21
Figure 3.4 - The naming (column, row, and zoom level) system for the Google Maps tiles (e.g. the top left tile name would be 0_0_1.png).....	24
Figure 3.5 – The one pixel gap created many vertical lines when the images were tiled in Google Maps. These lines were fixed with the GM_fillGap program.....	26

Figure 3.6 – The tile overlay after the blank pixels were filled using the GM_fillGap program.....	26
Figure 3.7 - The four tiles from the higher zoom level (left) are joined to make the new tile in the lower zoom level (right).....	28
Figure 3.8 - A sample of the xml file used to generate the stripmap point overlays.....	29
Figure 3.9 - The stripmap image that pops-up when the point marker is clicked. The pop-up also has a link to the stripmap website.....	30
Figure 3.10 - Generating the API key for the domain name <a href="http://omg.unb.ca">http://omg.unb.ca</a> .....	31
Figure 3.11 - The API key generated by Google.....	31
Figure 3.12 - Standard map controls and custom map controls that were added to customize the map.....	34
Figure 3.13 – Error handling for the link to download ArcticNet Basemaps. In the left image, the user clicked on a valid ArcticNet Basemap. In the right image, the user clicked on an area where there was no multibeam data collected, therefore the link was not shown.....	36



## **CHAPTER 1. INTRODUCTION**

The Ocean Mapping Group at the University of New Brunswick has been collecting data in the Arctic since 2003. Spending seven years in the Arctic, during which the multibeam and sub-bottom profiler were continuously logging data, allowed the Ocean Mapping Group to continually expand the multibeam and sub-bottom coverage with each new field season. This has resulted in collecting over 102,000 km<sup>2</sup> of bathymetric data, or having roughly mapped 2 percent of the Canadian Arctic region. The Ocean Mapping Group currently distributes all bathymetric and sub-bottom data online to industry and science members. The primary focus of this project was to develop a new online distribution method for multibeam and sub-bottom data collected in the Arctic.

The Ocean Mapping Group had been working in the Canadian Arctic as a member of ArcticNet's Network of Centres of Excellence of Canada (NCE) program. ArcticNet is a collection of Universities and research institutes, made up of students, researchers and managers that work together with government, industry and northern communities to study climate change in the coastal Canadian Arctic [ArcticNet, 2010]. It provides funding, and the CCGS Amundsen as a platform to perform marine research in the coastal Canadian Arctic. The ongoing mapping of the Amundsen has been divided into both seabed science investigation and geomatics engineering research, which inspired multiple thesis topics for members of the Ocean Mapping Group. The role of the Ocean Mapping Group in ArcticNet is to “map the bottom topography and geological structure

of the Northwest Passage and other regions of the Canadian Archipelago as a first step towards the management of increased intercontinental ship traffic and resource exploration as ice conditions improve, and will contribute invaluable information to assess the economic, sovereignty and security implications of an ice-free NW Passage.” [Fortier, 2003].

The Ocean Mapping Group’s mapping platforms in the Arctic have been the CCGS Amundsen a 98 metre, 1200 class medium size icebreaker, CCGS Nahidik a 53 metre, special navais vessel and the CSL Heron a 10 metre survey launch, which was on loan to the University from the Canadian Hydrographic Service. As discussed by Bartlett et al. [2004], the Amundsen mapping instruments included:

- Kongsberg EM 302 (recently upgraded from an EM 300) 30 kHz multibeam echosounder
- Knudsen 320R 3.5 kHz sub-bottom profiler
- C&C Technologies CNav differential GPS receiver
- Applanix POS/MV 320 inertial navigation system
- ODIM Brooke-Ocean (Rolls-Royce) MVP 300 moving vessel profiler
- Seabird 911 CTD
- Honeywell Barometer
- Applied Microsystems surface sound speed probe

The Heron mapping instruments included:

- Kongsberg EM 3002 multibeam echosounder
- Knudsen 320B 3.5 and 28 kHz sub-bottom profiler
- Knudsen 320B 200 kHz single beam and sidescan echosounders
- ODIM Brooke-Ocean (Rolls-Royce) MVP 30 moving vessel profiler
- C&C Technologies CNav differential GPS receiver
- CODA F-180 motion sensor
- AML surface sound speed probe

The Nahidik mapping instruments, as part of a portable multibeam system installation which could be set-up on any capable vessel, included:

- A pole mounted Kongsberg EM 3002 300 kHz multibeam echosounder
- CODA F-180 motion sensor
- Seatex MRU-6
- AML surface sound speed probe
- C&C Technologies CNav differential GPS receiver

This project focuses on the distribution of multibeam (bathymetry and backscatter) and sub-bottom (seismic) data, collected in the Arctic. It should also be noted that most of the above systems on the mapping vessels are capable of logging raw data. The raw data was stored on the Ocean Mapping Group's servers and with enough interest, the data could be made available online. This data included: CTD (about the salinity,

temperature, and density of the water for oceanographic purposes), water column backscatter, POS Pac format INS data, atmospheric pressure and GPS pseudo-ranges.

## **CHAPTER 2. BACKGROUND**

The collection of multibeam and sub-bottom data in the Arctic has driven a strong demand for viewing and downloading the data online. The current users required this data for navigation, engineering, natural resources, and benthic habitat applications. Providing multibeam and sub-bottom data to the users, other than least depths, involved processing to create a product and a method to distribute each product. This distribution was beneficial to the users in terms of avoiding redundant data collection and prioritizing areas which should be re-mapped [Beaudoin et al., 2008].

### **2.1 CURRENT DISTRIBUTION METHOD**

The Ocean Mapping Group had two methods in place for distributing the data collected in the Arctic: Arctic Stripmaps and the ArcticNet Basemap Series. Both of these distribution methods served their intended purpose; however, with evolving technology, improvements to these distribution methods could be made.

In 2003, the Ocean Mapping Group was collecting multibeam and sub-bottom data during opportunistic transits and short, dedicated site surveys. The data collected during the first few Arctic field seasons was sparse, causing the data distribution method for multibeam and sub-bottom data to focus on corridors of data. As discussed by Beaudoin

et al. [2008], in 2003 a Stripmap website (see figure 2.1) was developed which contained two different sun-illuminated bathymetry (across track and along track) images, a backscatter image and a correspondingly georeferenced sub-bottom image (2-dimensional seismic plot). To compliment these images, overview and location maps were also added, providing the necessary contextual information.

The stripmap website provided users with the full picture of the sea floor (bottom topography and its underlying sediment layers) in 25 by 5 kilometre mapsheets. For each year, users could walk through each 25 by 5 km stripmap, viewing the multibeam and sub-bottom images together, as if they were following the Amundsen ship track. While the website was well received by ArcticNet users for providing all the important information, improvements in web-GIS programs have provided tools to serve up the data with more detailed contextual information.

In 2006, after four years of building up coverage in the Arctic, the multibeam distribution method was shifted to areas of coverage. As discussed by Beaudoin et al. [2008], in 2006 the ArcticNet Basemap Series (see figure 2.2) was developed which contained a set of tiled bathymetry and backscatter images. Each basemap covered 15' of latitude and 30' of longitude, at a resolution of 10 metres and in the Lambert conformal conic projection. These basemaps also contained an overview and location map to help place the data in geographic context.

The basemap website was a custom design, allowing users to view and download (as ESRI grid files) all of the bathymetry and backscatter information collected in the Arctic since 2003. For those concerned with data management and quality, it provided full source traceability, including details on all lines of all years that contributed to each mapsheet. Once again, new developments in web-GIS programs presented opportunities to improve upon this method and serve up multi-resolution imagery seamlessly, with more detailed contextual information.

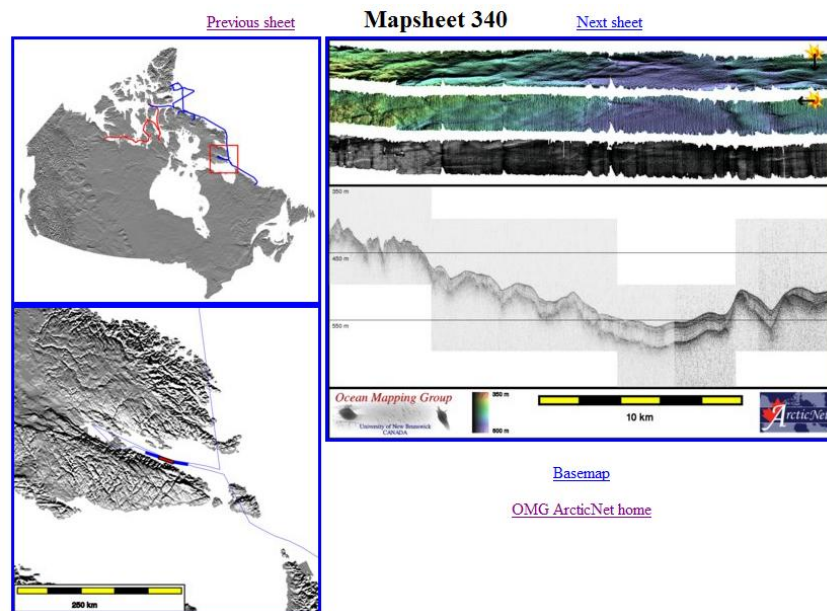


Figure 2.1 - The ArcticNet Stripmap website that was developed in 2003 to show multibeam and sub-bottom data.

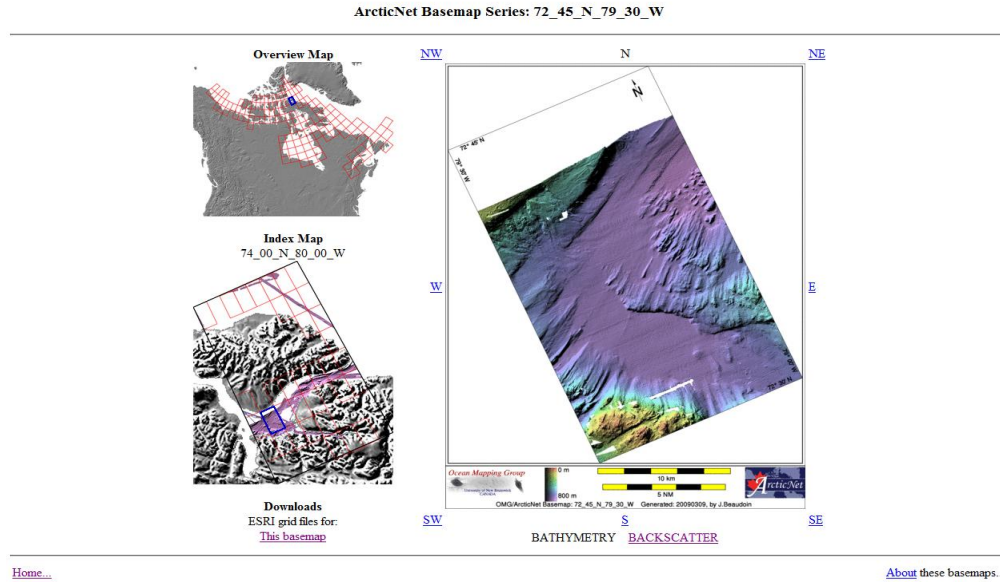


Figure 2.2 - The ArcticNet Basemap Series website that was developed in 2006 to show bathymetry and backscatter data.

## 2.2 PROPOSED DISTRIBUTION METHOD

The stripmap and basemap websites provided the outline for the development of the new distribution method. The new distribution method used a web-GIS interface to display the Ocean Mapping Group's Arctic dataset. The web-GIS interface allowed the multibeam imagery to be served up seamlessly, on top of high resolution satellite images, aerial photographs and Google's ocean bathymetry. It also integrated the stripmap website in the form of point overlays. The reasons behind choosing a web-GIS interface for the new data distribution model will be discussed in this section.



Geographic information systems (GIS) have been around since the 1960s, although they were typically expensive, difficult to use, and proprietary, which limited their use. A general definition of a GIS is a “computer systems for capturing, storing, querying, analyzing, and displaying geospatial data.” [Chang, 2008]. Geospatial data refers to the location and characteristics of spatial features. GIS systems have the ability to display the geographic area, at a user specified scale, viewed from above and also relate this geographic data with other information types [Geller, 2007]. Relating different sources of geographic data makes GIS a suitable platform to display and distribute the Ocean Mapping Group’s Arctic dataset.

Since 2004, there has been an increase in the development of free web-based geographic information systems, with Google Maps, Yahoo Maps, Bing Maps and MapQuest being the most common [Geller, 2007]. Developers have taken advantage of this and published hundreds of thousands of maps online [Google Geo Blog, 2010]. A few examples of these maps with bathymetry overlaid have been published by the University of Hawaii at Manoa [Hawaii Mapping Research Group, 2009] and in the listings found on the Magic Instinct Software website [2010]. These web-based systems have a small set of GIS tools, however many third-party “mash-ups” have been related to these GIS tools, allowing developers to add more functionality to their maps [Elias et al., 2008]. A mash-up is mixing two or more services from websites or web programs to create a new service. Developers and companies can use these systems to display their data to anyone with access to the internet and an internet browser.

The four main competitors (Google Maps, Yahoo Maps, Bing Maps and MapQuest) all have similar mash-ups implemented through JavaScript, and similar user interfaces (see figure 2.3). Google Maps was chosen as the web-based GIS system for this project because it had a popular Application Programming Interface (API), suitable satellite imagery and Google Ocean imagery. Google Maps was also a familiar interface to most internet users. Using a familiar interface would help users without GIS experience, to easily view and use the website.

The GIS tools available in the Google Maps API allowed developers to add points, lines, polygons and custom overlays to the map, as well as geocode addresses [Google Maps v2, 2010]. Developers could add standard Google controls and create custom controls for the map, customizing the appearance of the map for the users.

The Google Maps interface used the Mercator projection and had three different layers: map, satellite and terrain view (see figure 2.4). Each layer provides the user with different contextual information.

The map layer showed the terrestrial surface of the Earth as a white background, the aquatic surface as a blue background and National Parks as a green background. The map layer also labelled the names of Countries, Cities, Towns, Roads, Rivers, Oceans, etc... The shoreline from the map layer had a coarser resolution than the satellite layer.

The terrain layer showed a relief map of the terrain, providing users with elevations and general ground cover type. This layer also labelled the names of Countries, Cities, Towns, Roads, Rivers, Oceans, etc... The shoreline from the terrain layer was the same as the map layer; a coarser resolution than the satellite layer.

The satellite layer combined high resolution aerial photography and satellite imagery to produce detailed images of the Earth. This satellite layer had varying resolutions of imagery depending on the viewing location. Urban areas were typically covered with higher resolution imagery and rural or unpopulated areas were typically covered with lower resolution imagery. The satellite data was further complimented with Google ocean imagery, which showed imagery of the ocean basins, at low resolution (~10 km [Sandwell and Smith, 2010]). The ocean imagery was mainly created from the sea surface undulations, caused by sub-seabed gravity differences, which were recorded by satellite altimetry [Stewart, 1985]. Recently, Google has added new higher resolution ocean imagery in selected areas. This new imagery was from ship echosounders, collected by many organizations, including the Center for Coastal and Ocean Mapping – Joint Hydrographic Center [Google LatLong Blog, 2008]. This view also had a toggle box to show or hide labels with the names of Countries, Cities, Roads, Rivers, Oceans, etc...

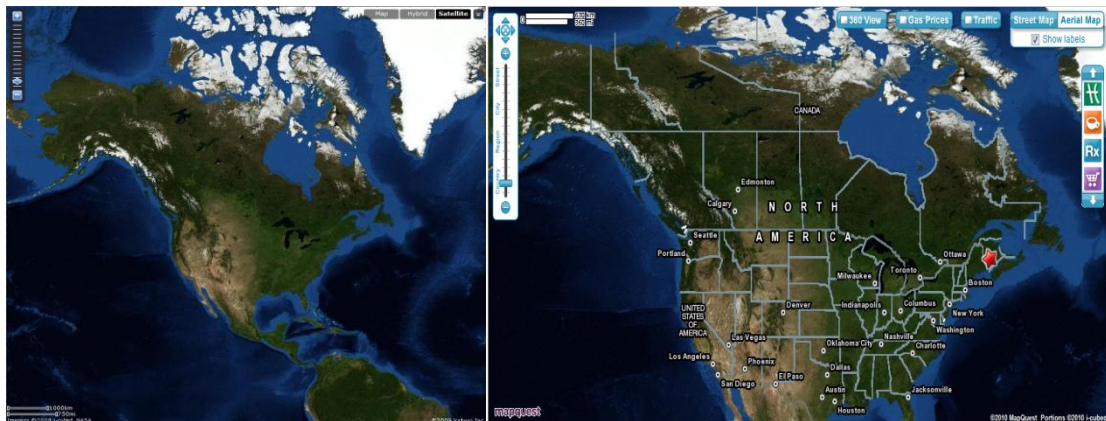
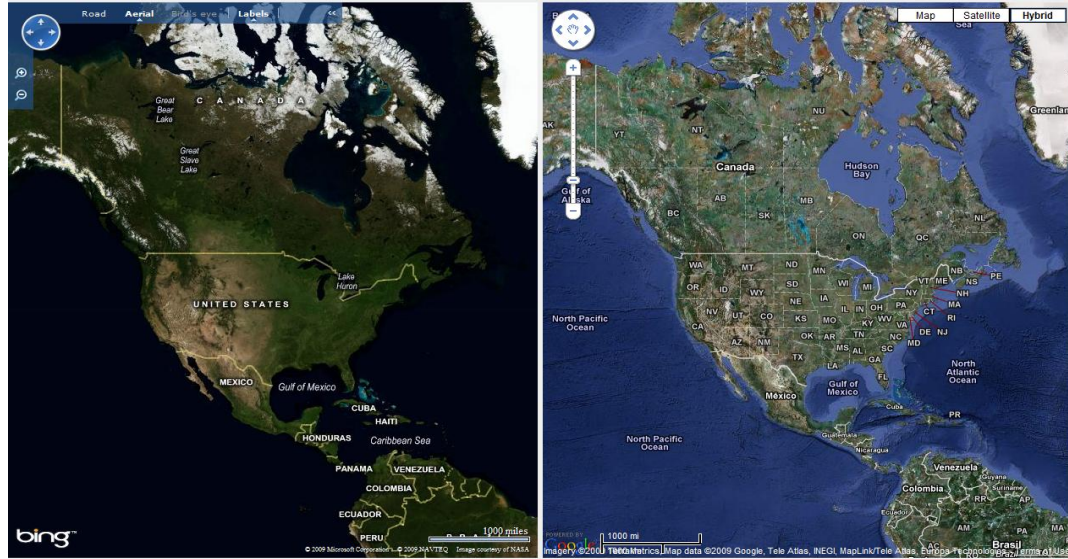


Figure 2.3 – The four main free web-GIS programs used by developers to create custom maps online.

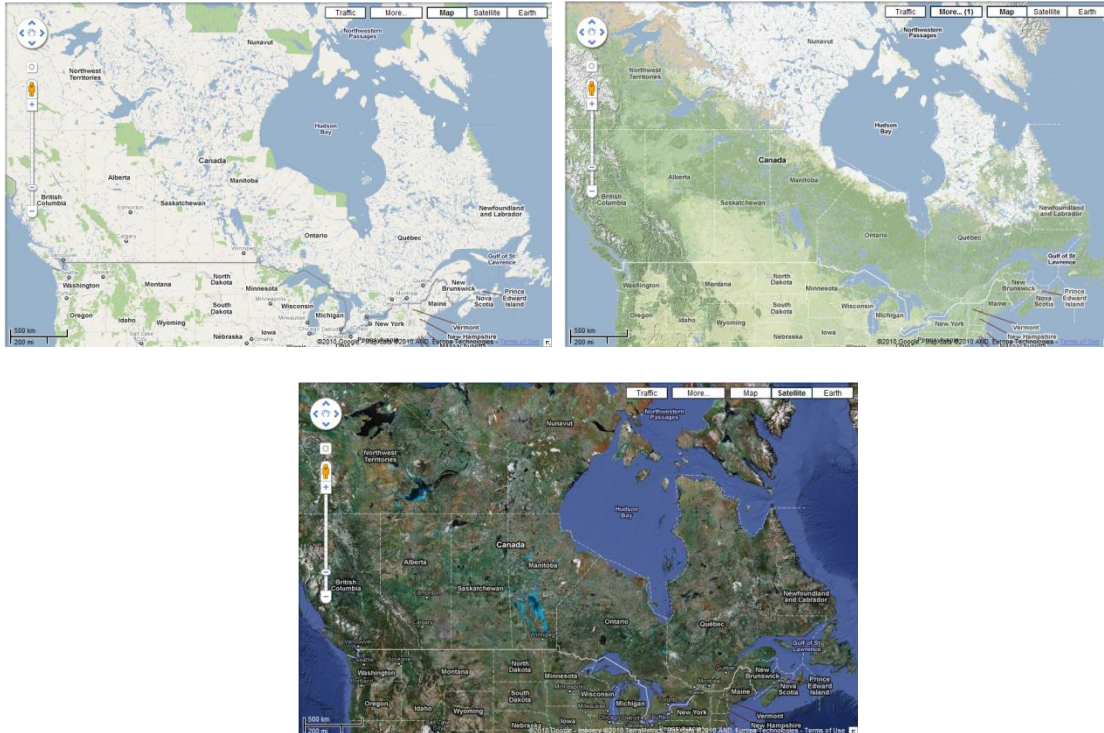


Figure 2.4 – The three different background images used by Google Maps. The top left is the Map View, the top right is the Terrain View, and the bottom is the Satellite View.

Google Maps was a suitable web-GIS platform, however it was still not the perfect solution to online data distribution. One of the disadvantages to using Google Maps was the use of the Mercator projection. The Mercator projection was a standard map projection for nautical charting because it presented lines of a constant bearing as straight lines [Pearson, 1990]. The disadvantage of Google Maps Mercator Projection was that it did not show any imagery or overlays of the Polar Regions. The reason being that a Mercator projection distorted the imagery as the latitude increased or decreased away from the Equator. As the imagery approached the poles, the distortion became infinite, which was why Greenland appeared to be larger than Africa. The bounds for Google's

Mercator Map were set at 85.011° N, 180° W, 85.011°S, and 180°E. The reason Google's map does not go above 85.011°N or below 85.011°S was because Google wanted to create a square map, which simplified the tiling scheme (discussed further in section 3.2). Not showing the poles was not a concern because all the data collected in the Arctic was below 80 degrees north.

Another disadvantage to using Google Maps was the high resolution satellite imagery and aerial photography covered mainly the urban areas (sparse in the Arctic). High resolution satellite imagery and aerial photography was not commercially available in all areas of the world and it was expensive to purchase. This was not a major concern because the low resolution satellite imagery and aerial photography was comparable to most of the shorelines on electronic charts in the Arctic. Google's imagery was also being updated more frequently as new imagery became available [Google Blog, 2010].

The last disadvantage was Google's ocean imagery was only available at lower resolutions and zoom levels. In areas away from the coast and at higher zoom levels (>10), the ocean imagery disappeared and was replaced with a grey background. This caused a few problems for the overlay of Ocean Mapping Group tiles, as they would not be displayed over areas with no ocean imagery. Displaying the Ocean Mapping Group tiles in these areas involved creating an array of overlays. The base layer was made up of white tiles. On top of the base layer was Google's satellite tiles, and the top most layer was the Ocean Mapping Group's custom tiles (see figure 2.5).



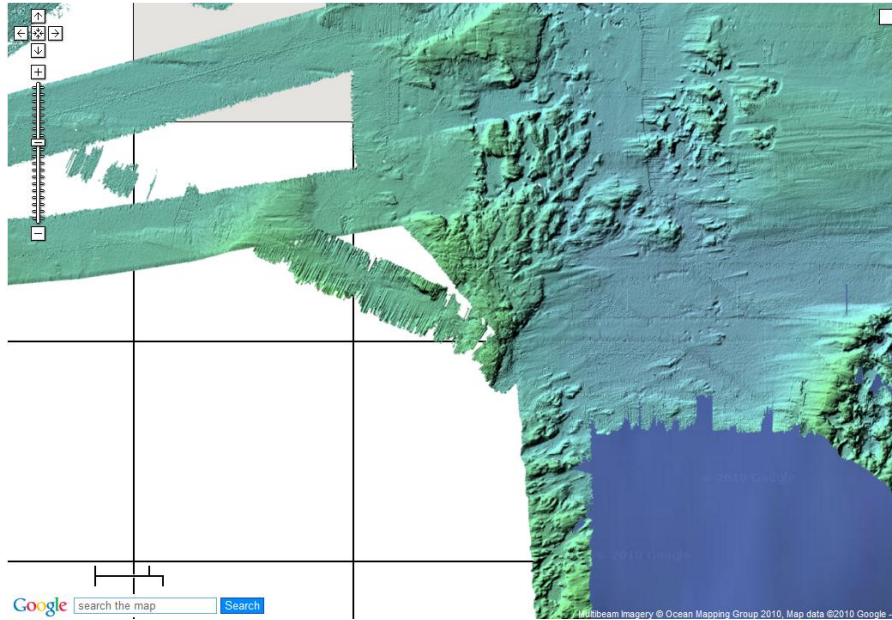


Figure 2.5 – The array of overlays, white tiles, Google’s satellite tiles, and Ocean Mapping Group tiles as they appear in Google Maps. Google’s Ocean imagery exists on the right (blue background), but disappears on the left (white outlined tiles). The array of overlays was used to show the Ocean Mapping Group’s tiles in the areas where Google did not have Ocean imagery.

## **CHAPTER 3. METHODS**

To fulfil the objectives outlined in this project, the Ocean Mapping Group's data needed to be overlaid on a contextual background image. Users could pan and zoom to any area or resolution they choose while still having the ability to download the data in the 15' latitude by 30' longitude, 10 metre resolution data, thereby preserving the source traceability.

### **3.1 TYPES OF OVERLAYS**

The scale of the Ocean Mapping Group's dataset meant that an efficient method must be used to create, store, and display the data online. Google Maps provided two types of custom overlays: ground overlays and tile overlays. While both types of overlays displayed images on top of Google's background layers, they were very different in how they were implemented. The main difference between the tile overlay and the ground overlay was the tile overlay used a fixed image size, with varying resolutions, for each zoom level. The ground overlay used a single image of any size, with a fixed resolution for all zoom levels. These two overlays are discussed in more detail here.



### **3.1.1 – Ground Overlay Method**

The ground overlay method was suitable for a small dataset covering a small area. It was relatively simple to implement, where developers only needed to have an image, with a transparent background, and the Southwest and Northeast latitude / longitude coordinates for georeferencing. This overlay displayed the image at the original resolution, regardless of zoom level.

### **3.1.2 – Tile Overlay Method**

The tile overlay method was an efficient way to display large datasets, however, it was more complex in its design and implementation than the ground overlay method. The tile overlay method involved creating a 256 by 256 pixel image, for each zoom level, everywhere multibeam data was collected. The number of potential tiles that needed to be created increased with the zoom level. For each subsequent zoom level, the image was divided into four tiles (see figure 3.1), thus giving the equation for the number of tiles necessary to cover the world, for that particular zoom level, as:

$$\text{Number of Tiles} = 4^n \quad (\text{where } n = \text{zoom level}).$$

Google only required tiles to be created where there was multibeam data, rather than creating blank tiles for most of the world. This drastically reduced the space necessary to store the tiles (~20 KB each). For example, the latest generation of tiles for all arctic data

collected between 2003 and 2009, produced just over 240,000 tiles for zoom level 14. Whereas, the amount of tiles necessary to cover the world at zoom level 14 was just over 268,000,000.



Figure 3.1 - Tile creation for Google Maps, each tile is subsequently divided into four new tiles. [Google Maps v2, 2010]

Tiling the data drastically speeds up the load time of the online map because it only loaded the tiles, at an appropriate resolution, within the user's map bounds. Depending on the user's screen resolution, this usually translated to approximately 8 (256 x 256 pixel) tiles for a typical map window size of 500 x 800 pixels.

The tiles used a pyramid structure (see figure 3.2) that varied the resolution of the tile according to the zoom level and latitude. At the lowest zoom level (zoom level 0), where the whole world was visible, the resolution of the multibeam data could be low because of the map scale. At the highest zoom level (zoom level 19), where details such as vehicles and houses were visible, the resolution of the multibeam data needed to be high so the overlay would not become pixelated.

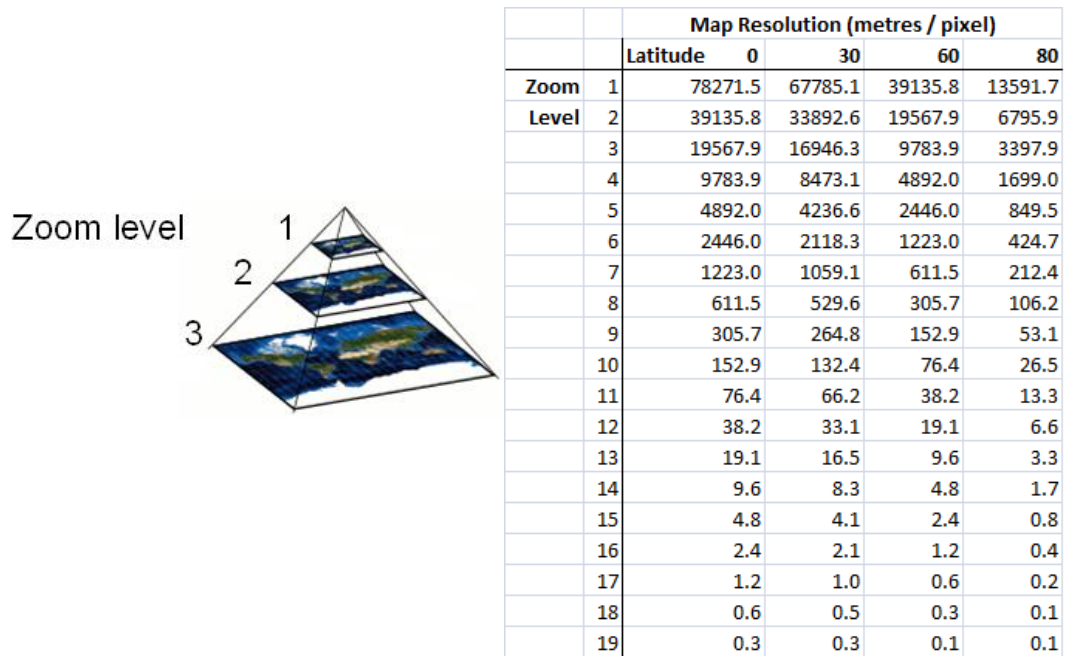


Figure 3.2 - The pyramid structure used by Google Maps which varies the resolution of the image according to the zoom level and latitude. [Pridal, 2008]

### 3.2 CREATION OF TILES

To create the tiles for overlay on Google Maps, the Ocean Mapping Group developed a custom software program. The program used the navigation tracks from the CCGS Amundsen, the CSL Heron and other vessels to determine which tiles the navigation track intersected at the highest zoom level (zoom level 14). Using the navigation tracks to determine which tiles needed to be created avoided creating unnecessary tiles, speeding up the program and saving disk space. At the highest zoom level, and in deep water (>800 metres), it was possible for the swath from the multibeam file to intersect up to

five tiles. To ensure there were no gaps in the data, 24 additional tiles (in an expanding square) were created around each navigation track tile. For each tile that needed to be created, a list of the cleaned multibeam files that contributed to that tile was also created.

### 3.2.1 – Tile Bounds Calculations

Google Maps requires each tile to have a Spherical Mercator projection with a fixed size of 256 pixels by 256 pixels. In order to create each tile, the latitude / longitude bounds of each 256 x 256 pixel tile, needed to be calculated. This was done as follows:

The Earth is divided into 360° of longitude and 180° of latitude. This represents a sphere, where the circumference at the equator is represented by  $2\pi$  multiplied by the radius of the Earth at the Equator ( $R = 6378137$ ). Since a Mercator map did not show the poles, Google simplified their map by cutting off the poles to make the map square. To create the square map, Google used the circumference at the equator of  $2\pi R$  to represent 360° of longitude and the total range of latitude. To calculate the absolute upper latitude of the Google map, divide  $2\pi R$  by 2 to give  $\pi R$ . Multiplying  $\pi$  by  $R$  equals 20037508.34 metres. Converting 20037508.34 metres to decimal degrees from the Spherical Mercator projection formula found in Maling [1973, p.153]:

$$y = \log_e \tan \left( \frac{\pi}{4} + \frac{\varphi}{2} \right)$$

rearranging to solve for  $\varphi$ :

$$\varphi = 2 \tan^{-1}(e^y) - \frac{\pi}{2}$$

$$\varphi = 2 \tan^{-1}(e^{20037508.34 \text{ m} / R}) - \frac{\pi}{2}$$

$$\varphi = 85.0511^\circ$$

This meant that the top latitude of Google's map was  $85.0511^\circ$ .

Figure 3.3 defines the terms used in the next calculations. The terms are: upper latitude tile bound, lower latitude tile bound, left longitude tile bound, and right longitude tile bound.



Figure 3.3 – Defining the tile bounds used in the calculations discussed in section 3.2.1.  
[Google Maps v2, 2010]

To calculate the lower latitude bound of the north most tile, the pole to pole latitude must be known,  $2\pi R$  for Spherical Mercator, as well as the zoom level of the tile. The formula for calculating the lower bound:

$$\text{Lower latitude tile bound} = -2\pi R / 2^{\text{zoom level}} + (\text{upper latitude tile bound in metres})$$

The lower latitude tile bound was then converted to decimal degrees using the Spherical Mercator formula:

$$\varphi = 2 \tan^{-1}(e^y) - \frac{\pi}{2}$$

The upper latitude was then set to the lower latitude bound for the calculation of the next tile. The latitude calculations can be done specifically for each tile if the row information was known.

$$\text{Upper Latitude} = -1 * \text{row} * 2\pi R / 2^{\text{zoom level}} + \pi R$$

$$\text{Lower Latitude} = -2\pi R / 2^{\text{zoom level}} + \text{Upper Latitude}$$

The Latitude bounds were then converted from spherical metres to decimal degrees.

The projection latitude was calculated by adding the Upper latitude bound to the lower latitude bound and then dividing by two. The projection latitude was used only for the `make_blank` program, which created a georeferenced blank mapsheet with the least distortion at the projection latitude.

The left longitude tile bound was first set to the most Westerly longitude of the Mercator projection (-180°). To calculate the right longitude tile bound, the circumference of the Earth must be known, 360° of longitude, as well as the zoom level of the tile. The formula for calculating the right bound:

$$360^\circ / 2^{\text{zoom level}} + \text{left longitude tile bound}$$

The left longitude tile bound was then set to the right longitude bound for the calculation of the next tile. The latitude calculations can be done specifically for each tile if the column information was known.

$$\textit{Left longitude} = \textit{column} * 360 / 2^{\textit{zoom level}} - 180$$

$$\textit{Right longitude} = 360 / 2^{\textit{zoom level}} + \textit{Left longitude}$$

### 3.2.2 – Creation of Tile Images

Google Maps required images for overlay in the map. The first step was to create a blank mapsheet in the spherical Mercator projection, using the latitude / longitude bounds, projection latitude (calculated above) and a custom resolution. The resolution for each tile was determined using the following formula:

$$\textit{Map resolution} = (\textit{resolution at Equator for zoom level 0}) * \cos(\textit{latitude}) / 2^{\textit{zoom level}}$$

$$\textit{*where resolution at Equator} = (2\pi R) / 256 \textit{ pixels per tile}$$

$$\textit{*where } R = 6378137$$

These mapsheets were created for each relevant (filled) tile, at the highest zoom level selected. The blank mapsheets were named according to the row, column and zoom level (see figure 3.4). They were then populated with multibeam soundings by assigning

floating point values to grid cells, using a weighting function (weigh\_grid, from the Ocean Mapping Group Swathed software toolkit).

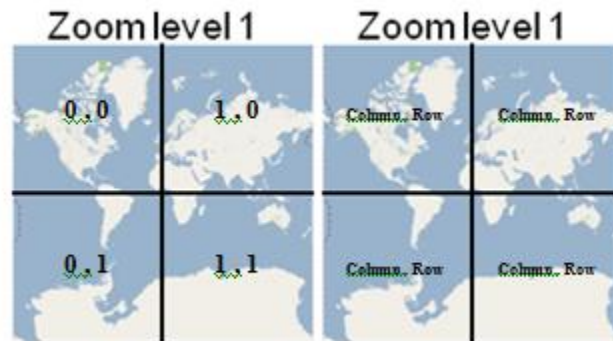


Figure 3.4 - The naming (column, row, and zoom level) system for the Google Maps tiles (e.g. the top left tile name would be 0\_0\_1.png). [Google Maps v2, 2010]

After the multibeam data was gridded into the tiles, each tile was sun-illuminated (addSUN, from the Ocean Mapping Group Swathed software toolkit) to give the multibeam data a 3-dimensional appearance. The data was then colour shaded according to depth using the Ocean Mapping Group's custom bathymetric colour scheme. The colour scheme was deliberately chosen to be the same throughout the entire Arctic dataset, with colour coded depths ranging from zero metres to 1000 metres (everything below 1000 metres was a constant colour). The colour coded and sun shaded tiles were mixed together to produce an image (mix\_ci, from the Ocean Mapping Group Swathed software toolkit). The new image was converted to a PNG image, which was supported by Google, with a transparent background using the "imagemagick" program. Imagemagick was a free software program that could read, write and convert images in a variety of image formats [ImageMagick Studio, 1999].



### **3.2.3 – Filling the Gap in the Tiles**

The programs used to create the multibeam tiles were not able to fill one pixel on the right side of each tile. This was due to the Ocean Mapping Group's custom program "addSUN". During the sun-illumination, the program's default settings used the pixel values from above and to the right to assign a value to the current pixel. In the last column of the image, there was no pixel to the right, so no values were assigned to the last pixel. When all the images were tiled together in Google Maps, many vertical lines, on the right side of each tile, were visible in the map (see figure 3.5).

To fill the pixels in the last column of the images, a new program called "GM\_fillGap" was created. This program was used after the sun-illumination and colour shading to patch the value from the pixel to the left, into the blank pixel (see figure 3.6). This method was not the perfect fix, because the sun-illumination in the last two pixels was the same.

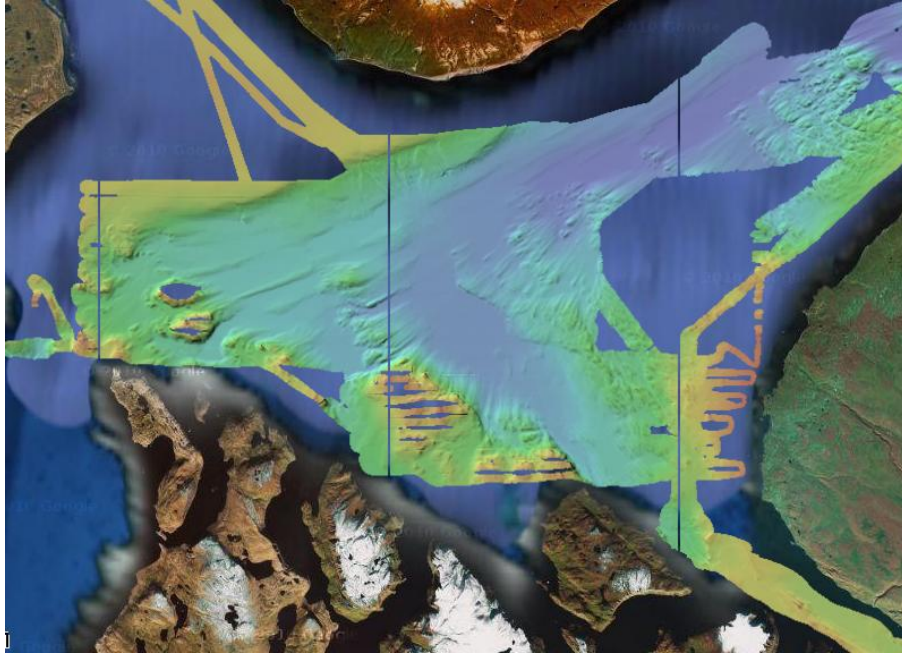


Figure 3.5 – The one pixel gap created many vertical lines when the images were tiled in Google Maps. These lines were fixed with the GM\_fillGap program.

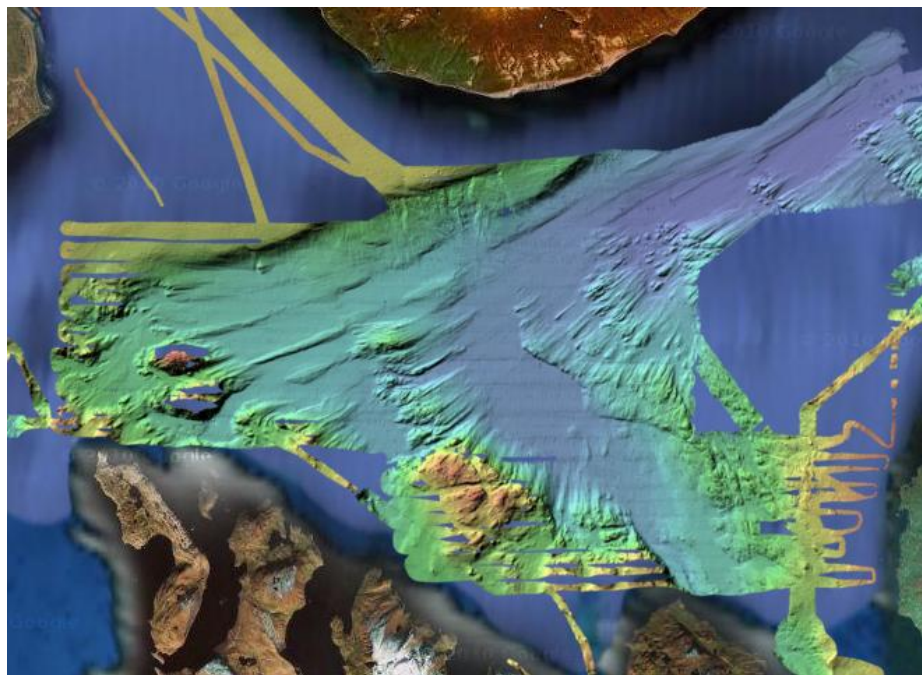


Figure 3.6 – The tile overlay after the blank pixels were filled using the GM\_fillGap program.

### 3.2.4 – Collapsing of Tiles

Following the creation of the highest zoom level (level 14), the tile images were collapsed to create the lower resolution zoom levels (zoom level 13, 12, 11... 3). For each lower zoom level (e.g. level 13), four tiles in the original zoom level (level 14) were used to generate one tile in level 13. The collapsing was done using the name (row, column and zoom level) of each tile in zoom level 14 to determine the name of the tile in level 13. This was done using the following calculations:

$$\text{Level 13 column} = (\text{Level 14 column} + 1 / 2^{\text{zoom level 14}}) * 2^{\text{zoom level 14} - 1}$$

If the level 14 column was odd (before adding 1), add 0.5 to the level 13 column and then subtract 1. If the level 14 column was even, subtract 1 from the level 13 column.

$$\text{Level 13 row} = (\text{Level 14 row} + 1 / 2^{\text{zoom level 14}}) * 2^{\text{zoom level 14} - 1}$$

If the level 14 row was odd, add 0.5 to the level 13 column and then subtract 1. If the level 14 row was even, subtract 1 from the level 13 row.

For each tile in level 13, the order (top right, top left, bottom right and bottom left) of the four tiles in the level 14 were calculated using the following steps:

$$\text{Top left tile name} = \text{level 13 tile column} * 2, \text{level 13 tile row} * 2, \text{tile zoom level 13} + 1$$

$$\text{Top right tile name} = \text{top left tile column} + 1, \text{top left tile row}, \text{zoom level} + 1$$

$$\text{Bottom left tile name} = \text{top left tile column}, \text{top left tile row} + 1, \text{zoom level} + 1$$

$$\text{Bottom right tile name} = \text{top left tile column} + 1, \text{top left tile row} + 1, \text{zoom level} + 1$$

After the order of the four tiles in level 14 was determined, the program “imagemagick” was used to join the four images according to their order (see figure 3.7). If there was a tile that did not exist in the previous zoom level, a blank, transparent 256 x 256 pixel tile was used in its place. If none of the four tiles existed, the new tile was skipped. The program created a new 512 x 512 pixel image which was then compressed to a 256 x 256 pixel image. This same method was used to collapse each subsequent level; level 13 was collapsed to level 12, and so on until level 3 was achieved.

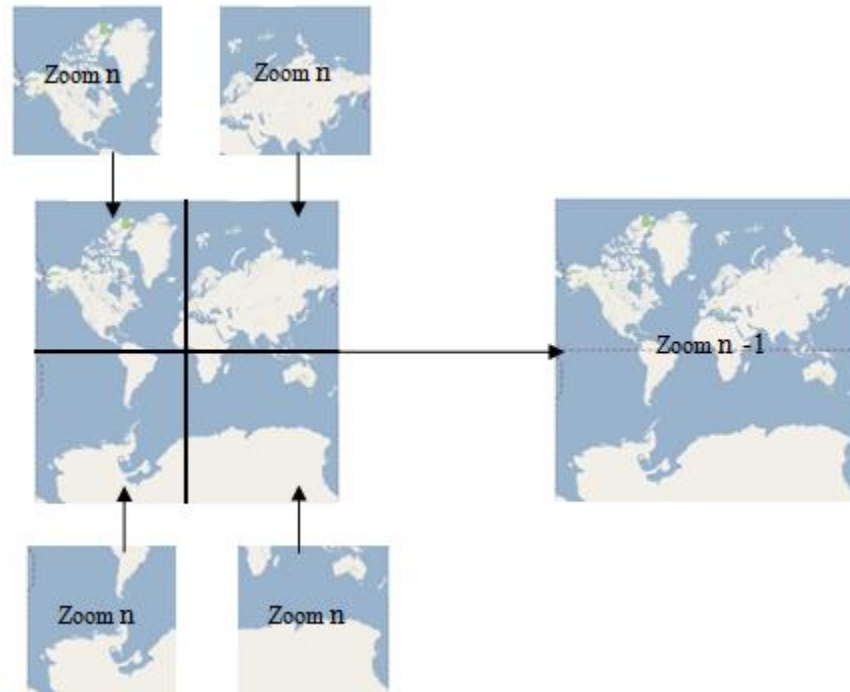


Figure 3.7 - The four tiles from the higher zoom level (left) are joined to make the new tile in the lower zoom level (right). [Google Maps v2, 2010]

### 3.3 POINT OVERLAYS

The point overlays in this project were created from the stripmap website. Stripmaps gave users the full picture of the seafloor by displaying information about the bottom topography, seafloor sediments, and the sub-seafloor sediments. The methods used to generate the point overlays are discussed here.

The stripmap point overlays were created using an xml file. The xml file had custom tags, generated by parsing the header file and path name for each stripmap, and placing the correct values into the corresponding stripmap marker tag. The custom tag values contained the latitude and longitude of the centre of each 25 x 5 km mapsheet, the year the stripmap data was collected, a link to the stripmap image, a link to the stripmap website, the name of the mapsheet and the colour of the custom icon (see figure 3.8).

```
<markers>
<marker name="Mapsheet 0" address="http://www.omg.unb.ca/Projects/Arctic/stripmaps/nwp2008/img/strip_0.gif" lat="67.925121" lng="-114.973642" year="2008" icon="purple"/>
<marker name="Mapsheet 1" address="http://www.omg.unb.ca/Projects/Arctic/stripmaps/nwp2008/img/strip_1.gif" lat="68.020356" lng="-114.569748" year="2008" icon="purple"/>
<marker name="Mapsheet 10" address="http://www.omg.unb.ca/Projects/Arctic/stripmaps/nwp2008/img/strip_10.gif" lat="68.546710" lng="-109.808829" year="2008" icon="purple"/>
</markers>
```

Figure 3.8 - A sample of the xml file used to generate the stripmap point overlays.

The website added the point overlays to Google Maps by parsing the xml file. The values from the xml file were used to display the points according to the latitude / longitude in the xml file. The other xml tag values were used to create the colour coded markers, and populate the balloon pop-up.

A toggle box was created on the website, allowing users to toggle the stripmap markers on and off. When the stripmaps were toggled on, users could click on the point marker and a balloon would pop-up showing the name of the stripmap, the stripmap image and a link to the stripmap website (see figure 3.9).

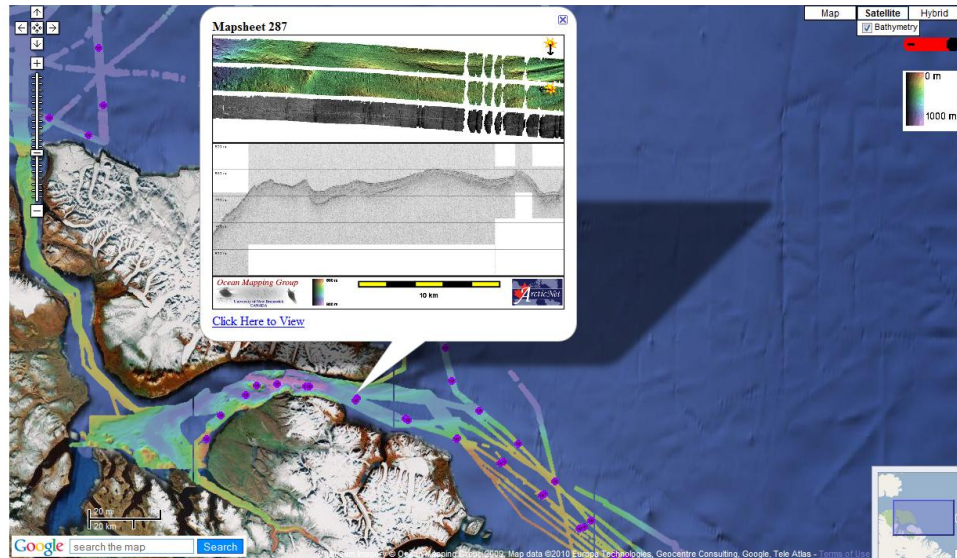
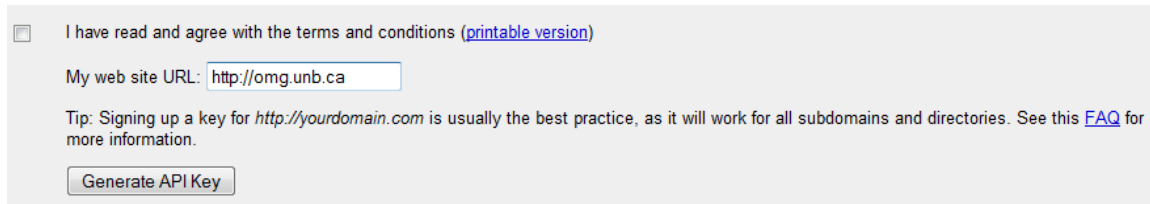


Figure 3.9 - The stripmap image that pops-up when the point marker is clicked. The pop-up also has a link to the stripmap website.

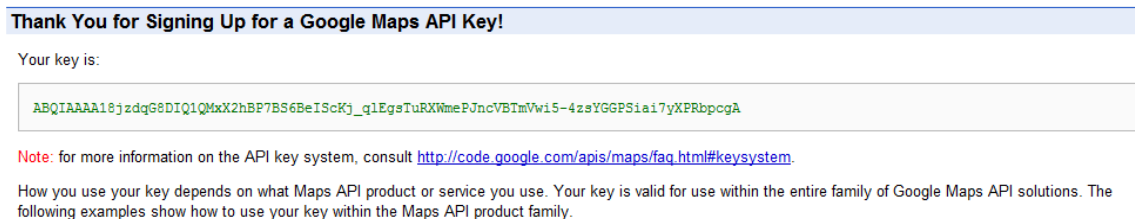
### 3.4 WEBSITE

The creation of the website with Google Maps used Google's Application Programming Interface (API). The API is an interface that links Google's code to the web programming language Javascript. In order to add Google Maps to a website, an API key was needed to register the website with Google. The API key can be obtained by registering and opening a Google account. There were an unlimited number of API keys a Google account could utilize. When registering the API key, it was recommended to sign up the domain name, so all subdomains and directories could use the same API key [Google Maps v2, 2010]. For this project, the domain name: <http://omg.unb.ca> was used to generate the API key (see figure 3.10 & 3.11).



A screenshot of the Google Maps API key generation form. At the top, there is a checkbox labeled "I have read and agree with the terms and conditions" with a link to a "printable version". Below this, the text "My web site URL:" is followed by a text input field containing "http://omg.unb.ca". A tip below the input field states: "Tip: Signing up a key for http://yourdomain.com is usually the best practice, as it will work for all subdomains and directories. See this FAQ for more information." At the bottom of the form is a button labeled "Generate API Key".

Figure 3.10 - Generating the API key for the domain name <http://omg.unb.ca>.



A screenshot of the Google Maps API key confirmation page. At the top, a blue header bar contains the text "Thank You for Signing Up for a Google Maps API Key!". Below this, the text "Your key is:" is followed by a text input field containing the API key: "ABQIAAAA18jzdgG8DIQ1QMxX2hBP7BS6BeIScKj\_q1EgsTuRXWmePJncVBtmVwi5-4zsYGGPSia17yXPRbpcgA". Below the input field, a red "Note" states: "Note: for more information on the API key system, consult <http://code.google.com/apis/maps/faq.html#keysystem>". At the bottom, a paragraph explains: "How you use your key depends on what Maps API product or service you use. Your key is valid for use within the entire family of Google Maps API solutions. The following examples show how to use your key within the Maps API product family."

Figure 3.11 - The API key generated by Google.

The website was created using html and Javascript languages. Google's API is written in Javascript which can be embedded into the html code used to create the website. On Google's website [Google Maps v2, 2010] they state there was no fee for embedding Google Maps into a website as long as the website was freely available to the end user. There was also no limit to the number of website views per day.

The website with Google Maps compatible seafloor imagery was hosted on the Ocean Mapping Group's web-server at the University of New Brunswick. When the website was loaded onto a user's computer, satellite imagery tiles were loaded from Google's server to the website and the multibeam tiles, stored on the Ocean Mapping Group's web-server, were served up on top of Google's satellite imagery.

Google Maps allowed the embedded map to be customized. In this project, the map properties (map center and zoom level) were set to Centre on Northern Canada, at zoom level three, when the website was loaded. This allowed any users on the website for the first time to see the extent of the Ocean Mapping Group's dataset. The satellite map type was chosen as the background imagery because Google's new low resolution bathymetry provided the best shoreline resolution.

To customize the appearance of the map, a combination of standard Google Map controls and custom controls were added (see figure 3.12). The standard Google controls helped keep the map familiar to most users.



The standard Google controls were:

- Zoom and pan controls
- Map type control
- Scale bar
- Search bar
- Overview map
- Copyright information.

The following custom controls were added:

- Legend for the bathymetry (colour bar)
- Opacity control to make the satellite data transparent (to see the tile boundaries for debugging)
- Checkbox to toggle the bathymetry overlay on and off
- Map properties: the Southwest and Northeast bounds of the map, the zoom level and the centre of the map
- The mouse: position on the map in latitude / longitude, position in pixels, the name of the tile the mouse is in, and the tile resolution
- Left click anywhere on the map for a balloon pop-up with a link to the corresponding ArcticNet Basemap, allowing users to download the multibeam grid files
- Checkbox to toggle the strip maps on and off

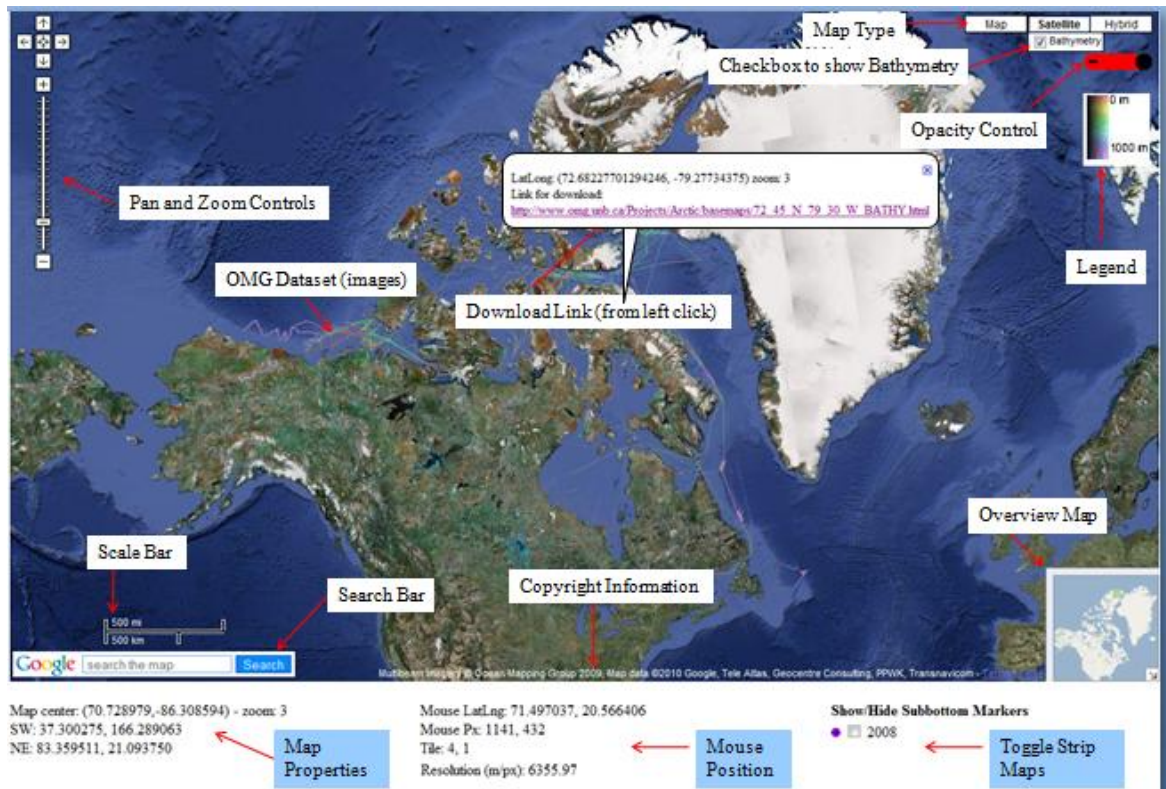


Figure 3.12 - Standard map controls and custom map controls that were added to customize the map.

This website was designed to allow users to continue to download the multibeam data in ESRI grid format. This was accomplished by linking the website to the ArcticNet Basemap series, which allowed users to download the 15' latitude by 30' longitude mapsheets. This link was generated when the user left clicked anywhere on the map. The location of the mouse at the time of the click was passed to a function that calculated which ArcticNet basemap the mouse was in. The mouse location was in decimal degree format with brackets surrounding the argument and a comma separating the values. The function first removed the brackets and then converted the values from decimal degrees to degrees and minutes. The function then calculated which 15 minute increment of

latitude and 30 minute increment of longitude the mouse was in. The latitude / longitude argument was then converted to a web link for the corresponding ArcticNet basemap.

The web link generated from a left click could be for any area in the world, so a function to handle invalid links was created. To determine if there was a valid basemap where the user clicked, an XMLHttpRequest head request was sent in Javascript to determine if the basemap webpage existed. This request returned a 200 status code if the website existed or another status code if it did not, for example a 404 if the webpage was not found [Wikipedia, 2010]. In areas where the Ocean Mapping Group had not collected multibeam data, or if the user clicked on the terrain, no link to download the data was shown (see figure 3.13). The XMLHttpRequest head request required “www” in the URL of the Google Maps website to work properly. To ensure that it was always present, the Ocean Mapping Group’s server was configured to redirect the URL <http://omg.unb.ca> to <http://www.omg.unb.ca>.

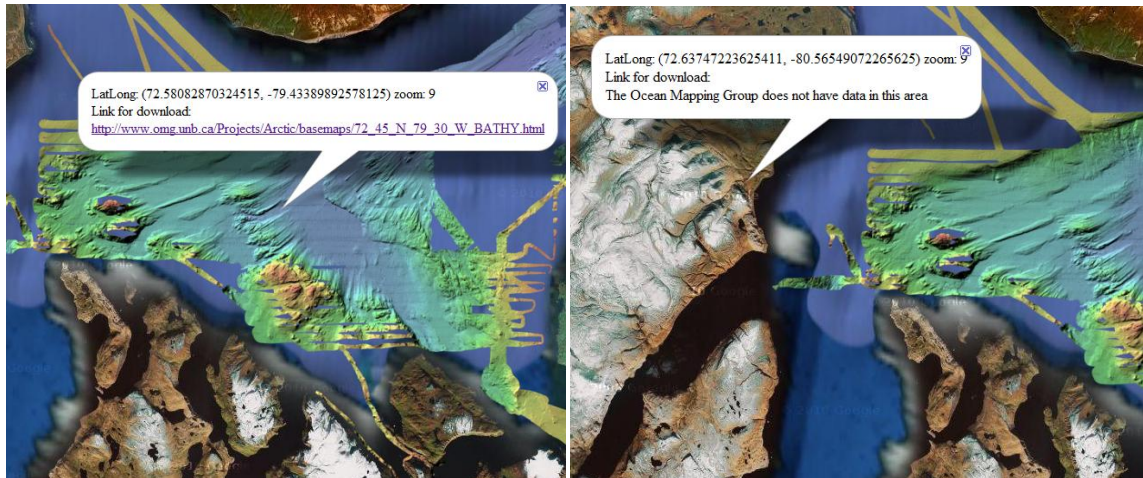


Figure 3.13 – Error handling for the link to download ArcticNet Basemaps. In the left image, the user clicked on a valid ArcticNet Basemap. In the right image, the user clicked on an area where there was no multibeam data collected, therefore the link was not shown.

## **CHAPTER 4. RESULTS**

This section describes how effective each method was for creating and displaying the multibeam overlays. It proceeds to discuss some of the simplifications used to create the tiles and point overlays. Finally, there is a discussion about some features and methods which will be implemented in the future.

### **4.1 SPEED OF WEBSITE AND TILE CREATION**

Throughout this project there have been several iterations of how to overlay the multibeam and sub-bottom data in Google Maps. The first method used to overlay multibeam data was the ground overlay method. As was discussed in section 3.1.1, the ground overlay method was not suitable for large datasets. This was proven after a quick test of a small multibeam dataset around Bylot Island, where the webpage load time for approximately 50 images, was between 10 and 15 seconds. The single resolution image, regardless of zoom level, also increased the load time of the website because it was not necessary to have high resolution images at low zoom levels.

The second method used was the tile overlay method (see section 3.1.2). This method had several iterations for creating the tiles. The first method involved the use of Microsoft's MapCruncher for Virtual Earth [Elson et al., 2007]. Creating the tiles with MapCruncher proved that the tile overlay method was suitable to overlay the large

multibeam dataset because there was no delay loading the tiles into the website. MapCruncher's tile overlay had a few areas where the tiles did not line up correctly. This was determined to be a projection problem between the Ocean Mapping Group's format and the format used by MapCruncher. The solution was to create a custom Ocean Mapping Group program to create the tiles.

The custom program, developed by the Ocean Mapping Group, to create the tiles also had several iterations. The first method was to create all tiles between 81° N, 42° N, 179° W and 41° W. This method gridded each tile within the bounds, for each zoom level (level 3 to 13). After gridding each tile, using all multibeam lines for each tile, the program would delete all the empty tiles to save disk space. This method was not very efficient as the tiles that were created covered all of Canada and part of the United States, land mass. Using a list of all multibeam lines, slowed down the gridding program because it had to search through each line to see which lines intersected each tile. The first update to this method, checked to see if the tile in the previous zoom level existed, before it would grid the data. When gridding each new tile, the program used a list of multibeam lines that intersected the previous zoom level tile, rather than all multibeam lines. This method was also slow in producing the tiles; the time it took to create levels 3-12 was approximately 2 weeks, compared to MapCruncher which created levels 3-11 in approximately 1.5 days.

The next method used to create the tiles was discussed in section 3.2. This method gridded multibeam data at the highest zoom level first, and then collapsed each subsequent level without the need to re-grid the data. This method was faster, taking approximately 5 days to grid the highest level (level 14) and approximately 1.5 days to collapse all other zoom levels (13-3). While this method was faster than all other methods used thus far, it still had room for improvement (discussed in next section).

## **4.2 SIMPLIFICATIONS AND FUTURE DIRECTIONS**

The fundamental objectives of creating a new online distribution method for the Ocean Mapping Group's data, has been completed. However, there are a few functions that will improve the functionality of the website for the end user. These are as follows:

- Creation of backscatter tiles
- Creation of tiles using the 100 ping bounds
- Implementing weigh\_grid options
- Developing a new colour scale
- Collapsing r4 files
- Developing an update procedure after each field season
- Using the marker manager for stripmap overlays
- Maintain and update the website
- Download option

#### **4.2.1 - Creation of Backscatter Tiles**

The current online distribution method, ArcticNet Basemaps, provided multibeam bathymetry and backscatter images to users. Currently, this project only provides the multibeam bathymetry. Adding the backscatter tiles to Google Maps would help users interpret the quality of the bathymetry and provide some rough seafloor classification. Creating the backscatter tiles should be implemented without any serious issues because all the necessary calculations have been completed for the bathymetry.

#### **4.2.2 - Creation of Tiles Using the 100 Ping Bounds**

The process of creating 24 tiles around each tile with the navigation track was a simple solution to the problem of having a wider swath than the single tile created from the navigation track. This solution should be replaced in future iterations by using the 100 ping bounds of each merged file. The 100 ping bounds are the latitude and longitude bounds in 100 ping increments of the merged file. This would allow for dynamic creation of the tiles, regardless of zoom level, saving time and space by not creating empty tiles.



### **4.2.3 - Implementing weigh\_grid Options**

Gridding the multibeam data into the tiles was done without any specific options. The specific options give higher quality surveys and newer echosounders stronger weights when the data was gridded into each tile. Two different options which should be used in the gridding process are discussed here.

One option was to use a custom weighting function, which was a linear scale by beam number that had the strongest weights at nadir, and tapered down to the weakest weights for the outer beams (This function is built into the weigh\_grid software from the Ocean Mapping Group Swathed software toolkit). When running a survey with 200 percent coverage (covering the seafloor twice), the nadir beams from one line were weighted stronger than the outer beams of the adjacent lines. Since the nadir beams typically had less noise than the outer beams, a more accurate representation of the sea floor was achieved.

The beamwidth option in weigh\_grid defined the size of the beam footprint and also helped with weighting each beam. Weighting each beam meant the centre of each beam had more weight than the surrounding beam footprint. This created a smoother transition between each beam, with different values populating each pixel (if the beam footprint was larger than the pixel).

There should also be a method in place to cope with different sonars, and different data qualities (ie. in large sea state, ice or the sonars ability to perform in a certain water depth range). Having different weigh\_grid commands for different sonars would be the simpler of the two methods to implement. This method involved using different custom weights, for each sonar, in the weigh\_grid command. The different float file outputs from weigh\_grid could then be combined, treating all sonars equally, or weighting each sonar according to its quality.

Coping with different data qualities, in large seas or ice would have to be a more manual procedure, perhaps looking at the magnitude of the roll and pitch for the specific weighting command. The large amount of data collected throughout the seven years in the arctic, with seemingly random areas of poor quality soundings, makes it more difficult to perfect these weighting commands.

To avoid a lengthy regridding process, patchArea from the Ocean Mapping Group Swathed software toolkit could be used. The patchArea program removes the necessity to regrid each individual mapsheet, instead it resamples existing grids, inheriting all the weigh\_grid options already built into the existing ArcticNet 15'x30' basemaps. This could be done for both bathymetry and backscatter.

#### **4.2.4 - Developing an Update Procedure After Each Field Season**

The ArcticNet Basemap series had a similar problem when it came to updating after each field season. The basemap series currently creates a new gridded, floating point file for each 15 x 30 mapsheet traversed in each new field season and then combines all years of each specific 15 x 30 mapsheets together. The combination utilizes the summed weighting calculated for each grid node. It is therefore equivalent to having gridded all the data into a single mapsheet. Combining each float file to make the one tile would not only speed up the update process by not having to grid all of the data, but it would also allow for custom weighting for different sonar types (currently built into the addWG program from the Ocean Mapping Group Swathed software toolkit).

#### **4.2.5 - Developing a New Colour Scale**

The fixed colour scale for the Arctic (0 to 1000 metres) was necessary when viewing the whole arctic dataset. This colour scheme had limitations because it only produced an 8-bit image. An 8-bit image contained  $2^8$  colours, which was 256 colours. It may be more useful to develop a 16 or 24-bit colour scheme, which would contain either 65,536 or 16,777,216 colours. This would sharpen the current colour scale, thereby improving the bathymetric image in all depth ranges. The second option would be to create a second 8-bit colour scheme and stack the two colour schemes together, creating a new 16-bit

colour scheme. The first 8-bit colour scheme could be used for shallow waters and the second 8-bit colour scheme could be used for deeper waters. This second option would be more appropriate because the colours would be stretched over a smaller depth range, allowing for easier interpretation.

#### **4.2.6 - Collapsing r4 Files**

Collapsing the tiles to make each lower zoom level was done with the program `imagemagick`. While the math used to calculate which tiles contribute to the lower zoom level was completed, the program presently only collapses the PNG images. The program should create new r4 files for each zoom level, which can be done using some of the logic already built into the `decr4` program. An r4 file was a floating point file used to store gridded bathymetric data (latitude, longitude and depth). The `decr4` program was a program that collapsed by integer steps (in this case 2 steps) floating point files. In this program, the number of pixels to be combined was specified and these pixels were then collapsed into one new pixel. For use in this project, a two by two pixel array (four pixels) from the higher zoom level would be combined into one pixel in the new r4 file. Collapsing the r4 files would be beneficial for speeding up the time it takes to collapse each zoom level, as well as having r4's created for each level. Having r4 files for each tile at each zoom level would make it easier to implement a new download option, discussed next.

#### **4.2.7 - Download Option**

The Ocean Mapping Group periodically gets requests for data, which forces the manual creation of ESRI grids, geotiffs, ascii xyz files, etc... for the user. The Google Maps interface could possibly automate these requests for data. The website could provide an interface where users enter information about the area they want data (specific bounds or map extent), and what format they want the output to be (ie. raw, merged, keb, segy, r4, geotif, tif world file, ASCII, or ESRI flt file). The website could then send a script to a processing machine with the float files already created, for each zoom level, at an appropriate resolution. These float files could be combined together to fill the requested area and then converted to the appropriate file format. This would effectively remove the need to update the original Arctic Basemap website.

#### **4.2.8 - Using the Marker Manager for Stripmap Overlays**

The implementation of the stripmaps into the website as point files from an xml file worked well for one to two years of stripmap data, approximately 1000 points. Toggling on and off these 1000 points, did not affect the load time of the website. When four years of stripmaps (2006 to 2009 stripmaps produced over 8000 points) were added to the website in the xml format, the load time of the website was very slow. This method of displaying the stripmap data needed to be improved. Although it had yet to be

implemented, Google has a marker manager which is used to manage points (markers) added to the map. This marker manager compared as to the xml file was similar to the tile overlay versus the ground overlay. The xml file was fixed to display all the points on the map regardless of the map extent, whereas the marker manager had the ability to group markers together and only load points within the map extent. Switching from parsing an xml file to using the marker manager will be an efficient method for displaying the stripmaps.

#### **4.2.9 - Maintain and Update the Website**

The website needed to be maintained and updated as new versions of Google Maps API and new Google Maps features become available. Recently, Google updated their maps API from version 2.x to version 3. In this new version, Google had added new tools to target the mobile maps market as well as update the regular desktop browsing tools [Google Maps v3, 2010]. This meant that the Ocean Mapping Group's website could be customized to perform better on mobile devices.

## CHAPTER 5. CONCLUSIONS

The Ocean Mapping Group collects many different sources of data: multibeam bathymetry, backscatter, water column backscatter, seismic (sub-bottom), CTD and MVP casts for oceanography, GPS positioning and tidal information. Google Maps was a particularly agile platform for developing a new method to display the Ocean Mapping Group's large multibeam dataset. It also has the potential to provide a platform to link all other sources of data the Ocean Mapping Group collected. Rather than searching for data in many different locations, users can now go to one location (Google Maps) to search and link data both spatially and contextually. Users are able to get a broad overview of an Arctic region, understand the regional trends and then focus on areas of interest at high resolutions to see particular features.

Presenting this topic at both the ArcticNet and Canadian Hydrographic conferences has increased the demand for using the website and downloading the gridded bathymetry. Feedback from users shows that this method of displaying multibeam data online was moving in the right direction. The interface was easy for everyone to use, even if they did not have a background in GIS or hydrography.

## BIBLIOGRAPHY

- ArcticNet (2010). *ArcticNet Description*. ArcticNet Website. [on-line] 9 September 2010.  
<http://www.arcticnet.ulaval.ca/index.php>
- Bartlett, J., Beaudoin, J and Hughes Clarke, J.E. (2004). *CCGS Amundsen: A New Mapping Platform for Canada's North*. Lighthouse, Journal of the Canadian Hydrographic Association; Edition No. 65.
- Beaudoin, J., Hughes Clarke, J.E., Bartlett, J., Blasco, S., and Bennett, R. (2008). *Mapping Canada's Arctic Seabed: Collaborative Survey Processing and Distribution Strategies*. Proceedings of the Canadian Hydrographic Conference and National Surveyors Conference 2008.
- Chang, K-T., (2008). *Introduction to Geographic Information Systems*. 4<sup>th</sup> edition. New York, NY: McGraw-Hill.
- Elias, M., Elson, J., Fisher, D., Howell, J. (2008). “*Do I Live in a Flood Basin?*” *Synthesizing Ten Thousand Maps*. CHI 2008 Proceedings, April 5-10, 2008, Florence Italy. pp. 255-264.
- Elson, J., J. Howell, D. Fisher, J. Douceur. (2007). *MSR MapCruncher for Virtual Earth*. [on-line] 23 January 2009.  
<http://research.microsoft.com/en-us/um/redmond/projects/mapcruncher/>
- Fortier, M. (2003). *Project 1.6 The Opening NW Passage: Resources, Navigation, Sovereignty & Security*. ArcticNet Phase 1 (2004-2008). [on-line] 5 June 2010.  
<http://www.arcticnet.ulaval.ca/research/theme1.php>
- Geller, T. (2007). *Imaging the World: The State of Online Mapping*. IEEE Computer Graphics and Applications, Vol. 27, No. 2, pp. 8-13.
- Google Maps v2. (2010). *Google Maps API*. Google Code [on-line] 20 June 2010.  
<http://code.google.com/apis/maps/documentation/introduction.html>  
<http://code.google.com/apis/maps/documentation/javascript/v2/overlays.html>



- Google Maps v3. (2010). *Google Maps API*. Google Code [on-line] 20 September 2010.  
<http://code.google.com/apis/maps/documentation/javascript/>
- Google Blog. (2008). *Google Lat Long Blog: News and Notes by the Google Earth and Maps Team*. [on-line] 20 September 2010.  
<http://google-latlong.blogspot.com/search/label/imagery>
- Google Geo Blog. (2010). *Google Geo Developers Blog: Big Birthday... Google Maps API turns 5!* [on-line] 20 September 2010.  
<http://googlegeodevelopers.blogspot.com/2010/06/big-birthday-google-maps-api-turns-5.html>
- Google LatLong Blog. (2008). *Google Lat Long Blog: Wander the Seafloor like Never Before*. [on-line] 20 September 2010.  
<http://google-latlong.blogspot.com/2010/02/wander-seafloor-like-never-before.html>
- Hawaii Mapping Research Group (2009). *Main Hawaiian Islands Multibeam Synthesis*. School of Ocean and Earth Science and Technology, University of Hawaii at Manoa. [on-line] 25 June 2010.  
<http://www.soest.hawaii.edu/HMRG/Multibeam/explorer.php#Virtual>
- ImageMagick Studio (1999). *Introduction to ImageMagick*. [on-line] 20 September 2010.  
<http://www.imagemagick.org/script/index.php>
- Magic Instinct Software. *Google Ocean: Google Maps and Google Earth as visualization tools for Marine Data*. [on-line] 27 June 2010.  
<http://www.justmagic.com/GM-GE.html>
- Maling, D.H. (1973). *Coordinate Systems and Map Projections*. London, England: George Philip and Son Limited.
- Pearson, F. (1990). *Map Projections: Theory and Applications*. Boca Raton, Florida: CRC Press.
- Pridal, K.P. (2008). *Tiles à la Google Maps: Coordinates, Tile Bounds and Projection*. [on-line] 6 June 2010.  
<http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/>

- Sandwell, D.T., Smith, W.H.F. (2010) *Exploring the Ocean Basins with Satellite Altimeter Data*. NOAA: National Geophysical Data Center. [on-line] 6 June 2010.  
<http://www.ngdc.noaa.gov/mgg/bathymetry/predicted/explore.HTML>
- Stewart, R.H. (1985). *Methods of Satellite Oceanography*. Berkley, California: University of California Press.
- Wikipedia (2010). *List of HTTP Status Codes*. [on-line] 27 June 2010.  
[http://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes](http://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

## APPENDIX I – WEBSITE CODE

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" style="height:100%">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
    <title>Arctic Basemaps on Google Maps</title>
    <!--last updated June 1, 2010-->
    <script
src="http://maps.google.com/maps?file=api&v=2&sensor=false&key=ABQIAAAA18jzdg
G8DIQ1QMxX2hBP7BS6BeIScKj_qLEgStuRXWmePJncVBTmVwi5-4zsYGGPSiai7yXPRbpcgA"
type="text/javascript"></script>
  </head>
  <body onload="GUnload()" onload="load()" style="height:100%">
    <table style="text-align: left; width: 100%;" align="center" border="0"
cellspacing="1" cellpadding="0">
      <tr>
        <td></td>
        <td><center><h2><font face="Comic Sans MS" size=6 color="#000066">Google Maps
with Bathymetry Overlaid</font></h2></center></td>
        <td></td>
      </tr>
    </table>
    <pre>Single left click gives you a link to download the data</pre>
    <div id="map" style="width: 98%; height: 90%;"></div> <!--Display map on the web
and set its dimentions, div id="mapDiv"-->

    <noscript><b>JavaScript must be enabled in order for you to use Google Maps.</b>
      However, it seems JavaScript is either disabled or not supported by your browser.
      To view Google Maps, enable JavaScript by changing your browser options, and then
      try again.
    </noscript>
    <br>

    <!--Code to Create Google Maps-->

    <!--set the script type to javascript and the src is used to enter the API key and
points to the Google Maps location-->
    <script type="text/javascript">
      //

      //Declare variables
      var map;
      var bathyLayer;
      var bathyHybridLayer;
      var bathySatMap;
      var bathyMap;

      // Opacity control stuff

      // A global variable
      var XSLIDERLENGTH = 55;          // maximum width that the knob can move (slide width
minus knob width)

      // Create a Custom GControl
      function XSliderControl(i) {
        this.init = i;//initial slider position
      }
      XSliderControl.prototype = new GControl();

      // This function positions the slider to match the specified opacity
      XSliderControl.prototype.setSlider = function(opacity) {
        var left = Math.round((XSLIDERLENGTH*opacity));
        this.slide.left = left;</pre>
</div>
<div data-bbox="514 871 539 885" data-label="Page-Footer">
<p>51</p>
</div>
```

```

        this.knob.style.left = left+"px";
    }

    // This function reads the slider and sets the overlay opacity level
    XSliderControl.prototype.setOpacity = function() {
        var o = this.slide.left/XSLIDERLENGTH;
        bathyHybridLayer[1].getOpacity = function () {return o;};
        if (this.map.getCurrentMapType() == bathySatMap)
        {
            this.map.setMapType(G_SATELLITE_MAP); //toggle map type to refresh opacity
            this.map.setMapType(bathySatMap);
        }
    }

    // This gets called by the API when addControl(new XSlider()) is used
    XSliderControl.prototype.initialize = function(map) {
        // obtain Function Closure on a reference to "this"
        var that=this;
        // store a reference to the map so that we can make calls on it
        this.map = map;

        // Is this MSIE, if so we need to use AlphaImageLoader
        var agent = navigator.userAgent.toLowerCase();
        if ((agent.indexOf("msie") > -1) && (agent.indexOf("opera") < 1)){this.ie = true}
    else {this.ie = false}

        // create the background graphic as a <div> containing an image
        var container = document.createElement("div");
        container.style.width="74px";
        container.style.height="19px";

        // Handle transparent PNG files in MSIE
        if (this.ie) {
            var loader =
"filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='SlideControl.png',
sizingMethod='scale');";
            container.innerHTML = '<div style="height:19px; width:74px; ' +loader+ ' '
></div>';
        } else {
            container.innerHTML = '';
        }

        // create the knob as a GDraggableObject
        // Handle transparent PNG files in MSIE
        if (this.ie) {
            var loader =
"progid:DXImageTransform.Microsoft.AlphaImageLoader(src='Slider.png',
sizingMethod='scale');";
            this.knob = document.createElement("div");
            this.knob.style.height="19px";
            this.knob.style.width="19px";
            this.knob.style.filter=loader;
        } else {
            this.knob = document.createElement("img");
            this.knob.src = "Slider.png";
            this.knob.height = "19";
            this.knob.width = "19";
        }
        container.appendChild(this.knob);
        this.slide=new GDraggableObject(this.knob, {container:container});
        this.container = container;

        // attach the control to the map
        map.getContainer().appendChild(container);

        // init slider
        this.setSlider(this.init);

```

```

// Listen for the slider being moved and set the opacity
GEvent.addListener(this.slide, "dragend", function() {that.setOpacity()});

// Listen for map being changed to show / hide slider
GEvent.addListener(this.map, "maptypechanged", function() {
    if(that.map.getCurrentMapType() == bathySatMap)
    {
        that.knob.style.display="";
        that.container.style.display="";
    }
    else
    {
        that.knob.style.display="none";
        that.container.style.display="none";
    }
});

return container;
}

// Set the default position for the control
XSliderControl.prototype.getDefaultPosition = function() {
    return new GControlPosition(G_ANCHOR_TOP_RIGHT, new GSize(7, 47));
}

// Add legend (Bathy colour bar)
function Legend() {}
Legend.prototype = new GControl;
Legend.prototype.initialize = function(map) {
    var me = this;
    me.panel = document.createElement("div");
    me.panel.style.background = "white";
    me.panel.innerHTML = '';
    map.getContainer().appendChild(me.panel);
    return me.panel;
};

Legend.prototype.getDefaultPosition = function() {
    return new GControlPosition(
        G_ANCHOR_TOP_RIGHT, new GSize(10, 90));
};
Legend.prototype.getPanel = function() {
    return me.panel;}

//Time to try and get markers working using an xml file.
var SB_icon = new GIcon();
SB_icon.image = 'http://www.omg.unb.ca/people/jmuggah/SB_marker_red_trans.png';
SB_icon.iconSize = new GSize(10, 10);
SB_icon.iconAnchor = new GPoint(5, 5);
SB_icon.infoWindowAnchor = new GPoint(3, 1);

var markerGroups = { "red": [], "blue": [], "purple": [], "green": [], "yellow":
[], "orange": []};
function toggleGroup(icon) {
    for (var i = 0; i < markerGroups[icon].length; i++) {
        var marker = markerGroups[icon][i];
        if (marker.isHidden()) {
            marker.show();
        } else {
            marker.hide();
        }
    }
}

// Call this function when the page has been loaded

```

```

function load(){
    if (GBrowserIsCompatible()){

        //resizeMap();
        //map = new
GMap2(document.getElementById("mapDiv"),{draggableCursor:'crosshair'});
        map = new GMap2(document.getElementById("map"),{draggableCursor:'crosshair'});

        //Point to directory where tiles are stored
        var bathyTiles = function (a,b) {
            return
"http://www.omg.unb.ca/~jmuggah/tiles/"+a.x+"_"+a.y+"_"+b+".png";
        }

        //Satellite imagery with bathy tiles overlaid

        //Declare array to hold different tile layers
        bathyHybridLayer = new Array();
        //First layer[0] is white background (always exist) tile layer
        bathyHybridLayer[0] = new GTileLayer(new GCopyrightCollection('') , 5);
        bathyHybridLayer[0].getTileUrl = function(tile, zoom) {
            return
"http://www.omg.unb.ca/~jmuggah/white_map_tile_outlined.gif";
        };
        bathyHybridLayer[0].getOpacity = function() {return 1.0;};
        //Next layer [1] is Google satellite tiles
        bathyHybridLayer[1] = G_SATELLITE_MAP.getTileLayers()[0];
        //Third layer [2] is OMG bathy tiles
        bathyHybridLayer[2] = new GTileLayer(new GCopyrightCollection('') , 5);
        bathyHybridLayer[2].getTileUrl = bathyTiles;
        bathyHybridLayer[2].getCopyright = function(a,b) {return "Multibeam
Imagery &#169; Ocean Mapping Group 2010";};
        bathyHybridLayer[2].getOpacity = function () {return 1.0;};//opacity of
the non transparent part
        if(navigator.userAgent.indexOf("MSIE") == -1)
            bathyHybridLayer[1].isPng = function() {return true;};

        bathySatMap = new GMapType(bathyHybridLayer,
            G_SATELLITE_MAP.getProjection(), 'Imagery with
Bathymetry',{errorMessage:"", alt:"Show imagery with Bathymetry"});
        bathySatMap.getTextColor = function() {return "#FFFFFF";};
        map.addMapType(bathySatMap);

        var hc = new GHierarchicalMapTypeControl();
        hc.addRelationship(G_SATELLITE_MAP, bathySatMap , "Bathymetry");
        hc.addRelationship(G_HYBRID_MAP, bathyMap , "Show Bathymetry");

        map.addControl(new GLargeMapControl());
        map.addControl(hc);

        //map.addControl(new GScaleControl());
        map.addControl(new XSliderControl(bathyHybridLayer[1].getOpacity()));
        //change [1] to reflect which layer you are making transparent.

        map.setCenter(new GLatLng(72.659588,-77.871094), 3);
        map.setMapType(bathySatMap);
        // map.addOverlay(new GTileLayerOverlay(bathyLayer));
        var bottomLeft = new GControlPosition(G_ANCHOR_BOTTOM_LEFT, new
GSize(100,40));
        map.addControl(new GScaleControl,bottomLeft);
        map.addControl(new GOverviewMapControl());
        // Add a collapsible overview map in the corner
        map.enableScrollWheelZoom();
        // Enable scroll wheel zoom
        map.enableGoogleBar();
    }
}

```

```

        //var topRight = new GControlPosition(G_ANCHOR_TOP_RIGHT, new
GSize(25,150));
        //map.addControl(new HtmlControl(''),topRight);
        map.addControl(new Legend());
        GEvent.addListener(map, 'mousemove', mouseMove);
        GEvent.addListener(map, "moveend", moveEnd);
        GEvent.addListener(map, "zoomend", zoomEnd);
        //GEvent.addListener(map, "singlerightclick", click);
        GEvent.addListener(map, "click", click);
        updateStatusBar();

        GDownloadUrl("stripmaps.xml", function(data) {
            var xml = GXml.parse(data);
            var markers = xml.documentElement.getElementsByTagName("marker");
            for (var i = 0; i < markers.length; i++) {
                var name = markers[i].getAttribute("name");
                var address = markers[i].getAttribute("address");
                var link = markers[i].getAttribute("link");
                var year = markers[i].getAttribute("year");
                var icon = markers[i].getAttribute("icon");
                //markerGroups[icon].push(tmarker);
                var point = new GLatLng(parseFloat(markers[i].getAttribute("lat")),
                    parseFloat(markers[i].getAttribute("lng")));
                //var marker = createMarker(point, name, address, type, color);
                var marker = createMarker(point, name, address, link, year, icon);
                map.addOverlay(marker);
                marker.hide()
            }
        });
        // display a warning if the browser was not compatible
        else {
            alert("Sorry, the Google Maps API is not compatible with this browser");
        }
    }
    //}

    var normalProj = G_SATELLITE_MAP.getProjection();

    function mouseMove(mousePt) {
        var zoom = map.getZoom();
        var oStatusDiv = document.getElementById("mouseTrack");
        var mousePx = normalProj.fromLatLngToPixel(mousePt, zoom);
        var Resolution =
(156543.04*Math.cos(mousePt.y.toFixed(6)*(Math.PI/180)))/Math.pow(2,map.getZoom());
        oStatusDiv.innerHTML = 'Mouse LatLng: ' + mousePt.y.toFixed(6) + ', ' +
mousePt.x.toFixed(6) ;
        oStatusDiv.innerHTML += '<br>';
        oStatusDiv.innerHTML += 'Mouse Px: ' + mousePx.x + ', ' + mousePx.y;
        oStatusDiv.innerHTML += '<br>';
        oStatusDiv.innerHTML += 'Tile: ' + Math.floor(mousePx.x / 256) + ', ' +
Math.floor(mousePx.y / 256);
        oStatusDiv.innerHTML += '<br>';
        oStatusDiv.innerHTML += 'Resolution (m/px): ' + Resolution.toFixed(2) ;
        //oStatusDiv.innerHTML += 'Resolution (m/px): ' +
(156543.04*Math.cos(mousePt.y.toFixed(6)*(Math.PI/180)))/Math.pow(2,map.getZoom()) ;
    }

    function moveEnd() {
        updateStatusBar();
    }

    function zoomEnd(oldZ,zoom) {
        updateStatusBar();
    }

    function updateStatusBar() {

```

```

        var center = map.getCenter();
        var zoom = map.getZoom();
        var bounds = map.getBounds();
        var SW = bounds.getSouthWest();
        var NE = bounds.getNorthEast();

        var oCoords = document.getElementById("coords");
        oCoords.innerHTML = 'Map center: (' + center.y.toFixed(6) + ', ' +
center.x.toFixed(6) + ') - zoom: ' + zoom;
        oCoords.innerHTML += '<br> ';
        oCoords.innerHTML += 'SW: ' + SW.y.toFixed(6) + ', ' + SW.x.toFixed(6);
        oCoords.innerHTML += '<br> ';
        oCoords.innerHTML += 'NE: ' + NE.y.toFixed(6) + ', ' + NE.x.toFixed(6);

    }

    //now we create the sub-bottom markers..
    function createMarker(point, name, address, link, year, icon) {
        //var icon = coloredIcon(iconStr);
        var imarker = new GIcon(SB_icon);
        if (icon == "blue") {
            imarker.image = "http://www.omg.unb.ca/people/jmuggah/SB_marker_blue_trans.png";
        }
        else if (icon == "purple") {
            imarker.image =
"http://www.omg.unb.ca/people/jmuggah/SB_marker_purple_trans.png";
        }
        else if (icon == "green") {
            imarker.image = "http://www.omg.unb.ca/people/jmuggah/SB_marker_green_trans.png";
        }
        else if (icon == "yellow") {
            imarker.image =
"http://www.omg.unb.ca/people/jmuggah/SB_marker_yellow_trans.png";
        }
        else if (icon == "orange") {
            imarker.image =
"http://www.omg.unb.ca/people/jmuggah/SB_marker_orange_trans.png";
        }
        else {
            imarker.image = "http://www.omg.unb.ca/people/jmuggah/SB_marker_red_trans.png";
        }
        var marker = new GMarker(point, imarker);
        markerGroups[icon].push(marker);
        //var marker = new GMarker(point, new customIcons[icon]);
        //var html = '<b>' + name + '</b> <br/> ';
        GEvent.addListener(marker, 'click', function() {
            // marker.openInfoWindowHtml(html);
            //marker.openInfoWindowHtml('<b>' + name + '</b> <br> <a href="' + address + '"
target="_blank">' + address + '</a>');
            //marker.openInfoWindowHtml('<b>' + name + '</b> <br> <a href="' + address + '"
target="_blank">Click Here to View</a>');
            marker.openInfoWindowHtml('<b>' + name + '</b> <br>  <br><a href="' + link + '"
target="_blank">Click Here to View</a>');
            //marker.openInfoWindowHtml('<b>' + name + '</b> <br>' + address );
        });
        return marker;
    }

    /*function click() {
        alert("You clicked the map.. stay tuned for link to download data.");
    }*/
    function click(overlay,latlng) {
        if (latlng) {

```



```

var latlon = latlng.toString();
var temp=latlon.replace(",","");
var temp2=temp.replace(" ","");
var ll_array = temp2.split(",");

var ladege=parseInt(ll_array[0], 10);
var laminn=ll_array[0]-ladege;
var lamins=laminn*60;
var lamin=parseInt(lamins, 10);

var ll_array1=ll_array[1].replace("-","");

var lodege=parseInt(ll_array1, 10);
var lominn=ll_array1-lodege;
var lomins=lominn*60;
var lomin=parseInt(lomins, 10);

if (lamin > 0 && lamin <= 15){
    lamin = 15;
}
else if (lamin > 15 && lamin <= 30){
    lamin = 30;
}
else if (lamin > 30 && lamin <= 45){
    lamin = 45;
}
else{
    lamin = "00";
    ladege = ladege + 1;
}

if (lomin > 0 && lomin <= 30){
    lomin = 30;
}
else{
    lomin = "00";
    lodege = lodege + 1;
}

//      var datafile =
"http://www.omg.unb.ca/Projects/Arctic/basemaps/"+ladege+"_"+lamin+"_N_"+lodege+"_"+lomin
+"_W_BATHY.html";

function checkUrl(url) {
    var req= new XMLHttpRequest(); // XMLHttpRequest object
    try {
        req.open("HEAD", url, false);
        req.send(null);
        return req.status== 200 ? true : false;
        //alert("Page status: " + req.status);
    }
    catch (er) {
        return false;
    }
}

if
(checkUrl("http://www.omg.unb.ca/Projects/Arctic/basemaps/"+ladege+"_"+lamin+"_N_"+lodege
+"_"+lomin+"_W_BATHY.html")==true){
    map.openInfoWindowHtml(latlng, "LatLong: " + latlng.toString() + "
zoom: " + map.getZoom() + "<br>Link for download: <br> <a
href='http://www.omg.unb.ca/Projects/Arctic/basemaps/"+ladege+"_"+lamin+"_N_"+lodege+"_"+

```

```

lomin+"_W_BATHY.html'
target='_blank'>http://www.omg.unb.ca/Projects/Arctic/basemaps/"+ladeqs+"_"+lamin+"_N_"+l
odeqs+"_"+lomin+"_W_BATHY.html</a>");
    }
    else{
        map.openInfoWindowHtml(latlng, "LatLong: " + latlng.toString() + "
zoom: " + map.getZoom() + "<br>Link for download: <br> The Ocean Mapping Group does not
have data in this area");
    }

}

}

/*function resizeMap() {
    //contain.style.width = document.body.clientWidth - 330 + "px";
    //contain.style.height = document.body.clientHeight - 300 + "px";
    document.getElementById("mapDiv").style.width = document.body.clientWidth - 100 +
"px";
    document.getElementById("mapDiv").style.height = document.body.clientHeight - 275
+ "px";
    //var oBox = document.getElementById('cBoxes');
    //oBox.style.height = document.body.clientHeight - 250 + 'px';

    if (map) {
        map.checkResize();
    }
}*/

//]]>
</script>

<div class="statusBar">
<table cellpadding="0" cellspacing="0" width="100%">
    <tr>
        <td valign="top" width="35%">
            <div class="statusDiv" id="coords">Map center:</div>
        </td>
        <td valign="top" width="35%">
            <div class="statusDiv" id="mouseTrack">Mouse:</div>
        </td>
        <td valign="top" width="30%"><form name="form1"
action=""><strong>Show/Hide Subbottom Markers</strong><br />
            <!-- 
            <input type="checkbox" name="red" id="red" onClick="toggleGroup('red')">
checked="checked" /> Red Markers<br />
            
            <input type="checkbox" name="blue" id="blue"
onClick="toggleGroup('blue')"> checked="checked" /> Blue Markers<br /> -->
            <!-- 
            <input type="checkbox" name="green" id="green"
onClick="toggleGroup('green')"> /> 2006<br />
            
            <input type="checkbox" name="yellow" id="yellow"
onClick="toggleGroup('yellow')"> /> 2007<br /> -->
            

```

```

☐

```

## APPENDIX II – TILE CREATION CODE

```
#!/bin/tcsh
#
# Assumes that Arctic Data resides in /drives/viscount/disk1/data/
# Zoom levels 0 - 14
# Created by James Muggah April 2010 *updated June 2010

#usage of program
#GoogleMapTileCreation -basemap /drives/Heron1/disk1/jmuggah/NewTiles/GoogleWorld14.blank
-nav
/drives/viscount/disk1/data/2009_Amundsen/002_Vancouver_Island/EM302/decnav/JD188/0055_20
090707_103443.decnav -out /drives/Heron1/disk1/jmuggah/NewTiles/test3.txt

if ( -e TileList.txt ) then
    rm -f TileList.txt
endif
if ( -e Tiles.txt ) then
    rm -f Tiles.txt
endif
if ( -e Output.tmp ) then
    rm -f Output.tmp
endif
if ( -e navlist.txt ) then
    rm -f navlist.txt
endif
if ( -e NewTileList.txt ) then
    rm -f NewTileList.txt
endif
if ( -e Temp.txt ) then
    rm -f Temp.txt
endif
if ( -e SortedTiles.txt ) then
    rm -f SortedTiles.txt
endif

#create nav file list
ls /drives/viscount/disk1/data/2000_Healy/decnav/JD*/*.decnav >> navlist.txt
ls /drives/viscount/disk1/data/2002_Marai/SB2100/decnav/jd*/*.decnav >> navlist.txt
ls /drives/viscount/disk1/data/2003_Amundsen/*/EM300/decnav/*.decnav >> navlist.txt
ls /drives/viscount/disk1/data/2003_Healy/decnav/*.decnav >> navlist.txt
ls /drives/viscount/disk1/data/2003_Healy_Eastern_Arctic/decnav/2*/*.decnav >>
navlist.txt
ls /drives/viscount/disk1/data/2004_Amundsen/ArcticNet_Leg1/EM300/decnav/*.decnav >>
navlist.txt
ls /drives/viscount/disk1/data/2004_Amundsen/GSC_Survey/EM300/decnav/*.decnav >>
navlist.txt
ls /drives/viscount/disk1/data/2004_Amundsen/Labrador_Transit/EM300/decnav/*.decnav >>
navlist.txt
ls /drives/viscount/disk1/data/2004_Amundsen/Leg9/EM300/decnav/*.decnav >> navlist.txt
ls /drives/viscount/disk1/data/2004_Amundsen/Leg8/EM300/decnav/JD*/*.decnav >>
navlist.txt
ls /drives/viscount/disk1/data/2005_Amundsen/0*/EM300/decnav/JD*/*.decnav >> navlist.txt
ls /drives/viscount/disk1/data/2006_Amundsen/0*/EM300/decnav/JD*/*.decnav >> navlist.txt
ls /drives/viscount/disk1/data/2006_Heron_Leg1/0*/EM3002/decnav/JD*/*.decnav >>
navlist.txt
ls /drives/viscount/disk1/data/2006_Heron_Leg2/0*/EM3002/decnav/JD*/*.decnav >>
navlist.txt
ls /drives/viscount/disk1/data/2007_Amundsen/0*/EM300/decnav/JD*/*.decnav >> navlist.txt
ls /drives/viscount/disk1/data/2008_Amundsen/0*/EM300/decnav/JD*/*.decnav >> navlist.txt
ls /drives/viscount/disk1/data/2008_Heron_Arctic/0*/EM3002/decnav/JD*/*.decnav >>
navlist.txt

foreach LIST (`cat /drives/viscount/disk1/data/2009_Amundsen/merged_input_NOLeg3.list`)
```

```

        set NAVI = `echo $LIST | awk 'BEGIN { FS = "/" } ; {print
"/drives/viscount/disk1/data/2009_Amundsen/"$1/"$2"/decnav/"$4"/"$5}'`
        set DNAVI = `echo $NAVI | awk 'BEGIN { FS = "." } ; {print $1".decnav"}'`
        echo $DNAVI >> navlist.txt
    end

    foreach FILE (`cat navlist.txt`)

        #echo "Doing" $Navlist
        /home/jmuggah/local/linux/bin/GoogleMapTileCreation -basemap
/drives/Heron1/disk1/jmuggah/NewTiles/GoogleWorldTest.blank -nav $FILE -skip 10 -out
/drives/Heron1/disk1/jmuggah/NewTiles/TileList.txt

    end

#Going to try and create tiles around tile from decnav files..
cat TileList.txt | awk '{print $1,$2,$3"\n"$1+1,$2-
1,$3"\n"$1+1,$2,$3"\n"$1+1,$2+1,$3"\n"$1,$2-1,$3"\n"$1,$2+1,$3"\n"$1-1,$2-1,$3"\n"$1-
1,$2,$3"\n"$1-1,$2+1,$3"\n"$1+2,$2-2,$3"\n"$1+1,$2-2,$3"\n"$1,$2-2,$3"\n"$1-1,$2-
2,$3"\n"$1-2,$2-2,$3"\n"$1-2,$2-1,$3"\n"$1-2,$2,$3"\n"$1-2,$2+1,$3"\n"$1-2,$2+2,$3"\n"$1-
1,$2+2,$3"\n"$1,$2+2,$3"\n"$1+1,$2+2,$3"\n"$1+2,$2+2,$3"\n"$1+2,$2+1,$3"\n"$1+2,$2,$3"\n"
$1+2,$2-1,$3}' >| NewTileList.txt

awk < NewTileList.txt '{ print $1, $2, $3 }'| sort -n| uniq >| SortedTiles.txt

awk < SortedTiles.txt '{ print $1, $2 }'| sort -n| uniq >| Tiles.txt

#Trying to create an array of column and row tiles.
set countc = (`cat Tiles.txt | awk '{print $1}'`)
set countr = (`cat Tiles.txt | awk '{print $2}'`)

#Now trying to find the size of the arrays.
set arraysiz = `echo $#countc`
echo $arraysiz

#-----Now for tile creation-----
@ level = 14

set TEMP_LIST_FILE = `mktemp`

while ($level == 14)
    echo "Working on level "$level >> Output.tmp
    set DIR = `echo $level`
    set Folder = `echo $level`
    if ( ! -e $Folder ) then
        mkdir $Folder
    endif
    @ row = `echo $level | awk '{print 2^$1}'`
    @ column = $row
    echo "Row & Columns "$row >> Output.tmp

    set i = 1
    while ( $i < $arraysiz )
        echo "Column: "$countc[$i] "Row: "$countr[$i] # >> Output.tmp

        # 1 => Top Left
        # 2 => Bottom Right
        set lon1 = `echo $countc[$i] "*" 360 "/" $column "- 180" | bc -l`
        #echo "Lon 1 "$lon1
        set lon2 = `echo "360 "/" $column "+" $lon1 | bc -l`
        #echo "Lon 2 "$lon2

        set lat1 = `echo "-1 *" $countr[$i] "*" 40075016.685578488 / "$row" +
20037508.342789244" | bc -l`
        set lat2 = `echo "-40075016.685578488 /" $row" +" $lat1 | bc -l`
        set projlat = `echo ("($lat1+"$lat2") / 2" | bc -l`

```

```

        set lat3 = `echo "0" $lat1 | invproj +proj=merc +a=6378137 +b=6378137
+lon_0=0 +lat_ts=0 -f '%.6f' | awk '{print $2}`
        set lat4 = `echo "0" $lat2 | invproj +proj=merc +a=6378137 +b=6378137
+lon_0=0 +lat_ts=0 -f '%.6f' | awk '{print $2}`
        set projlat2 = `echo "0" $projlat | invproj +proj=merc +a=6378137
+b=6378137 +lon_0=0 +lat_ts=0 -f '%.6f' | awk '{print $2}`
        #echo "lat 1 "$lat3 " lat 2 "$lat4 " projlat "$projlat2
        #attempt 1 at finding resolution according to zoom
        #set res1 = `echo "((1853 * 60 * (" $lon1 "-" $lon2 "))) / c(" $projlat2 "*"
( 3.141592654 / 180 ))) / 256" | bc -l`
        set res1 = `echo "156543.04 * c(" $projlat2 "*" ( 3.141592654 / 180 )) /
2^" $level | bc -l`
        #echo $res1
        @ res2 = `echo $res1 | awk '{ if ($1 < 10 ) print "1" }'`
        if ( $res2 == 1 ) then
            set res1 = 10
            #echo $res1
        endif
        #echo $projlat2
        # Give make_blank file name
        set filename = `echo $countc[$i]"_"$countr[$i]"_"$level`
        echo $filename >> Output.tmp

        echo "trying tile "$filename

        echo "Doing the MakeBlank -- can take some time..." >> Output.tmp
        ./do_make_blank $filename $lat3 $lon1 $lat4 $lon2 $projlat2 $res1 >
/dev/null

        echo "Done Make_Blank !!!" >> Output.tmp

        # make the r4

        tor4 $filename

        echo "Made the R4's !!" >> Output.tmp

        # Do the weigh Grid

        echo "Gridding!" >> Output.tmp

        #-----
        -----
        #Here we need to give it the list of merged files (in array)
        cat SortedTiles.txt | awk '{if ($1 == "$countc[$i]") if ($2 ==
"$countr[$i]") print $3 }' >| Temp.txt

        weigh_grid $filename `cat Temp.txt`

        echo "Done Gridding!" >> Output.tmp

        # r4to8bit

        r4to8bit -low -1000 -high 0 $filename.r4 $filename.8bit

        # Check to see in the file is empty, if it is continue & delete existing
files

        if (`checkEmpty $filename.8bit` == Empty) then
            echo "Empty Mapsheet!!! Removing empty files" >> Output.tmp
            rm -f $filename.*
            @ i++
            continue
        else if (`checkEmpty $filename.8bit` == NotEmpty) then
            echo "Not Empty! Keep Going" >> Output.tmp
        endif

```

```

# AddSun

addSUN -range 130 205 $filename

# mix_ci

mix_ci -mask $filename.sun_315 -ignore 255 -c $filename.8bit -i
$filename.sun_315 -ppm -m $filename.ppm

# convert to png 256x256

convert -transparent white $filename.ppm temp.png
convert -resize 256x256! temp.png $filename.png
rm -f temp.png

#cp $filename.png /homes/jmuggah/public_html/tiles/
#If want to gzip files do it here..
#gzip -f $filename.*
mv *$DIR.* $DIR

#@ countc = $countc + 1
echo "Increment Colomn!" >> Output.tmp

    @ i++
    echo $i
end

    @ level = $level + 1
end

```

## APPENDIX III – FILL GAP CODE

```
#!/bin/tcsh

foreach FILE (*)
#   gunzip $FILE
#   set FILE_PREFIX = `echo $FILE | sed -e 's/.r4.gz//g' | awk '{print $1}' `
#   set FILE_PREFIX = `echo $FILE | sed -e 's/.r4//g' -e 's/14\\///g' | awk '{print $1}' `
#   set FILE_PREFIX = `echo $FILE | sed -e 's/.r4//g' | awk '{print $1}' `
  echo "using prefix : $FILE_PREFIX"

  if (! (-e $FILE_PREFIX.orig.png) ) then
    cp $FILE_PREFIX.png $FILE_PREFIX.orig.png
  endif

  r4to8bit -low -1000 -high 0 $FILE_PREFIX.r4 $FILE_PREFIX.8bit
  GM_fillGap -edgeit -ignore 0 $FILE_PREFIX.8bit $FILE_PREFIX.edge.8bit
  addSUN -range 130 205 $FILE_PREFIX
  GM_fillGap -edgeit -ignore 255 $FILE_PREFIX.sun_315 $FILE_PREFIX.edge.sun_315
  mix_ci -mask $FILE_PREFIX.edge.sun_315 -ignore 255 -c $FILE_PREFIX.edge.8bit -i
  $FILE_PREFIX.edge.sun_315 -ppm -m $FILE_PREFIX.edge.ppm
  convert -transparent white $FILE_PREFIX.edge.ppm temp.png
  convert -resize 256x256! temp.png $FILE_PREFIX.png

  rm temp.png
  #rm $FILE_PREFIX.fill.sun_315
  rm $FILE_PREFIX.edge.sun_315
  #rm $FILE_PREFIX.fill.8bit
  rm $FILE_PREFIX.edge.8bit
  rm $FILE_PREFIX.edge.ppm

  #display $FILE_PREFIX.png &

  #gzip $FILE_PREFIX.r4

end

exit
```



## APPENDIX IV – COLLAPSE CODE

```
#!/bin/tcsh
#
# Trying to collapse the tiles using adjoin program.
# Created by James Muggah May 2010

@ level = 14

#if there is already a PreZoom file, delete it.
if ( -e PreZoom.txt) then
    rm -f PreZoom.txt
endif
if ( -e SortedPre.txt) then
    rm -f SortedPre.txt
endif

#to do all zoom levels, first copy Tiles.txt to a new file. then
#create a loop while ($level > 3). then we have a temp list for each zoom.

#Trying to create an array of column and row tiles.
set countc = (`cat Tiles.txt | awk '{print $1}'`)
set countr = (`cat Tiles.txt | awk '{print $2}'`)

#Now trying to find the size of the arrays.
set arraysiz = `echo $#countc`
echo $arraysiz
set UpDIR = `echo $level - 1 | bc -l`
if ( ! -e UpDIR.txt) then
    mkdir $UpDIR
endif

set i = 1
while ( $i < $arraysiz )
#while ($level > 2)
    #echo "Working on level "$level >> Output.tmp
    #@ zoom = $level - 1
    set DIR = `echo $level`
    set zoom = `echo $level "- 1" | bc -l`
    set filename = `echo $countc[$i]"_"$countr[$i]"_"$level`

    # do math to find file name of mapsheet in above zoom level

    set MATHc = `echo $countc[$i] "+ 1" | bc -l`
    set MATHr = `echo $countr[$i] "+ 1" | bc -l`
    set MATHz = `echo $level`

    #Calculate the tile you are in. column/2^zoom * 2^zoom-1
    set PreZoomC = `echo "(" $MATHc"/2^"$MATHz" )" * 2^(" $MATHz"-1 )" | bc -l`
    set PreZoomR = `echo "(" $MATHr"/2^"$MATHz" )" * 2^(" $MATHz"-1 )" | bc -l`

    # if column/row is odd add 0.5 to get to whole # and subtract 1 to get
    # column/row starting at 0
    if ( $countc[$i] % 2 != 1 ) then
        set PreZoomC = `echo "(" $PreZoomC" + 0.5 )-1" | bc -l`
    else
        set PreZoomC = `echo $PreZoomC - 1" | bc -l`
    endif

    if ( $countr[$i] % 2 != 1 ) then
        set PreZoomR = `echo "(" $PreZoomR" + 0.5 )-1" | bc -l`
    else
        set PreZoomR = `echo $PreZoomR - 1" | bc -l`
    endif

    set PreZoomCI = `echo $PreZoomC | awk -F \. '{print $1}'`
```

```

set PreZoomRI = `echo $PreZoomR | awk -F \. '{print $1}'`

set Pre_filename = `echo $PreZoomCI_"$PreZoomRI_"$zoom`

#here we need to echo out the filename and the prezoom filename to a file
#so we get a unique list. Then we know which tiles to put where.

echo $Pre_filename" " $filename >> PreZoom.txt

@ i++
#echo $i
end
#set j = 1
#while ( $j < $arraysize )

#To go down a level from previous zoom just multiply row column by 2 and you get top left
#tile then just need to add 1 to row, then column, then row & column to get other 3
tiles.

#need to have.. if exist (1 row down, place it in, otherwise just put in generic 256x256
blank
#image.

#cat PreZoom.txt | awk '{if ($1 == "$Pre_filename") print $2 }' >! Temp.txt
awk < PreZoom.txt '{ print $1 }'| sort -n| uniq >! SortedPre.txt

if ( -e Htemp.png ) then
    rm -f Htemp.png
endif
if ( -e H2temp.png ) then
    rm -f H2temp.png
endif

foreach FILE (`cat SortedPre.txt`)

    set UpperLeftc = `echo $FILE | awk 'BEGIN { FS = "_" } ; {print $1*2}'`
    set UpperLefttr = `echo $FILE | awk 'BEGIN { FS = "_" } ; {print $2*2}'`
    set PZoom = `echo $FILE | awk 'BEGIN { FS = "_" } ; {print $3+1}'`
    set UpperRightc = `echo $UpperLeftc + 1 | bc -l`
    set UpperRighttr = `echo $UpperLefttr`
    set LowerLeftc = `echo $UpperLeftc`
    set LowerLefttr = `echo $UpperLefttr + 1 | bc -l`
    set LowerRightc = `echo $UpperLeftc + 1 | bc -l`
    set LowerRighttr = `echo $UpperLefttr + 1 | bc -l`

    set UpperLeft = `echo $DIR/"$UpperLeftc_"$UpperLefttr_"$PZoom".png`
    #echo $UpperLeft
    set UpperRight = `echo $DIR/"$UpperRightc_"$UpperRighttr_"$PZoom".png`
    set LowerLeft = `echo $DIR/"$LowerLeftc_"$LowerLefttr_"$PZoom".png`
    set LowerRight = `echo $DIR/"$LowerRightc_"$LowerRighttr_"$PZoom".png`

    if ( -e $UpperLeft ) then
        set UL = 1
    else
        set UL = 0
    endif
    if ( -e $UpperRight ) then
        set UR = 1
    else
        set UR = 0
    endif
    if ( -e $LowerLeft ) then
        set LL = 1
    else
        set LL = 0
    endif
    if ( -e $LowerRight ) then

```

```

        set LR = 1
    else
        set LR = 0
    endif

    if ( $UL == 1 && $UR == 1 ) then
        adjoin -m H -b none $UpperLeft $UpperRight Htemp.png
    endif
    if ( $LL == 1 && $LR == 1 ) then
        adjoin -m H -b none $LowerLeft $LowerRight H2temp.png
    endif
    if ( $UL == 1 && $UR != 1 ) then
        adjoin -m H -b none $UpperLeft generic_tile.png Htemp.png
    endif
    if ( $UL != 1 && $UR == 1 ) then
        adjoin -m H -b none generic_tile.png $UpperRight Htemp.png
    endif
    if ( $UL != 1 && $UR != 1 ) then
        adjoin -m H -b none generic_tile.png generic_tile.png Htemp.png
    endif
    if ( $LL == 1 && $LR != 1 ) then
        adjoin -m H -b none $LowerLeft generic_tile.png H2temp.png
    endif
    if ( $LL != 1 && $LR == 1 ) then
        adjoin -m H -b none generic_tile.png $LowerRight H2temp.png
    endif
    if ( $LL != 1 && $LR != 1 ) then
        adjoin -m H -b none generic_tile.png generic_tile.png H2temp.png
    endif

    adjoin -m V -b none Htemp.png H2temp.png $UpDIR/"$FILE".png"
    convert -resize 256x256 $UpDIR/"$FILE".png" $UpDIR/"$FILE".png"

    if ( -e Htemp.png ) then
        rm -f Htemp.png
    endif
    if ( -e H2temp.png ) then
        rm -f H2temp.png
    endif
    echo "working.."

end

#set outpng = (`cat PreZoom.txt | awk '{print $1}'`)

# @ level = $level - 1

#end

#adjoin -m H -b none 5220_5570_14.png 5221_5570_14.png Htemp.png
#adjoin -m H -b none 5220_5571_14.png 5221_5571_14.png H2temp.png
#adjoin -m V -b none Htemp.png H2temp.png 2610_2785_13.png
#convert -transparent white -resize 256x256 2610_2785_13.png 2610_2785_13.png
#convert -resize 256x256 2610_2785_13.png 2610_2785_13.png

#end

```

## APPENDIX V – STRIPMAP CREATION CODE

```
#!/bin/tcsh
#
# Creates a xml file with all the goodies needed to put the stripmap image
# into Google Maps
# Created by James Muggah May 2010

#ls /drives/viscount/disk1/data/2008_Amundsen/maps_25x5/box_headers/Box.header* >>
maps.txt

if ( -e dirs.txt ) then
    rm -f dirs.txt
endif
if ( -e maps.txt ) then
    rm -f maps.txt
endif
if ( -e stripmaps.xml ) then
    rm -f stripmaps.xml
endif

#echo "/drives/viscount/disk1/data/2006_Amundsen/maps_25x5/" >> dirs.txt
#echo "/drives/viscount/disk1/data/2007_Amundsen/maps_25x5/" >> dirs.txt
echo "/drives/viscount/disk1/data/2008_Amundsen/maps_25x5/" >> dirs.txt
#echo "/drives/viscount/disk1/data/2009_Amundsen/maps_25x5/" >> dirs.txt

echo "<markers>" >> stripmaps.xml
foreach DIR (`cat dirs.txt`)

    set project = `echo $DIR | awk -F / '{print "nwp"$6}' | sed 's/_Amundsen/'`
    set year = `echo $DIR | awk -F / '{print $6}' | sed 's/_Amundsen/'`
    if ( $year == 2003 ) then
        set icon = `echo "red"`
    else if ( $year == 2004 ) then
        set icon = `echo "red"`
    else if ( $year == 2005 ) then
        set icon = `echo "blue"`
    else if ( $year == 2006 ) then
        set icon = `echo "green"`
    else if ( $year == 2007 ) then
        set icon = `echo "yellow"`
    else if ( $year == 2008 ) then
        set icon = `echo "purple"`
    else if ( $year == 2009 ) then
        set icon = `echo "orange"`
    endif

    ls `echo $DIR"box_headers/Box.header*"` >> maps.txt

    #echo "<markers>" >> stripmaps.xml
    foreach FILE (`cat maps.txt`)
        set lat = `edhead -show $FILE | awk 'NR==2{print $11}'`
        set lon = `edhead -show $FILE | awk 'NR==2{print $9}'`
        set map_num = `echo $FILE | awk -F / '{print $9}' | sed 's/Box.header/'`
        set address = `echo
"http://www.omg.unb.ca/Projects/Arctic/stripmaps/"$project"/img/strip_"$map_num".gif"`
        set link = `echo
"http://www.omg.unb.ca/Projects/Arctic/stripmaps/"$project"/"$map_num".html"`
        echo '    <marker name="Mapsheet '$map_num'" address="'$address'" link="'$link'"
lat="'$lat'" lng="'$lon'" year="'$year'" icon="'$icon'"/>' >> stripmaps.xml
    end
    #echo "</markers>" >> stripmaps.xml

end
echo "</markers>" >> stripmaps.xml
```