

Table of Contents

What is Mapjunction?.....	1
History.....	1
Conversion to Open Source.....	2
Overall Architecture.....	2
Modifications to Mapserver.....	3
Multi-server Support.....	4
Storage of map data.....	4
Transapplet.....	4
Printing.....	5
Searching in the framed Java client	5
Address Searching.....	5
3D.....	6
QuadBlob Format.....	6
Structure of tree.....	7
Integration into mapserver.....	9
Command Line Tools.....	9

Mapjunction Developer Manual

This document is intended for a developer that is going to change the mapjunction software. Read the admin manual first.

Greg Cockroft. greg@agog.com 9/27/2004. First draft.

What is Mapjunction?

MapJunction is a system for developing web-based GIS applications. It utilizes a modified MapServer for its opengis interface support as the server of map layers. It includes multiple clients for viewing of the data. A java applet based client, a simple javascript client (inlineWMS), and coming soon a flash client.

It relies on geocoder.us for geocoding support.

There are html administration pages to ease the administration of the system and a java applet for uploading content and for georeferencing of raster layers.

History

The start of mapjunction was a contract for futureboston.org. Bill Warner the founder of futureboston wanted a website for the display and comparison of historical maps. This was in June 2000. At this time the majority of users were on slow dialup connections. Java was popular and I had never heard of opengis.org. Grass was studied but was not close enough to what was needed. Some Dxf and shapefile code was copied from grass as a starting point, but the majority of code was written from scratch. The results of this work can be viewed at http://www.mapsovertime.com/Boston_MA/index.html

In 2001 Bill introduced me to Martin Von Wyss of the Boston Redevelopment Authority. The BRA needed a way to print maps for the public at their map counter. I added printing to the mapjunction applet to allow them to print out their GIS layers. This is a

live system and can be viewed at http://www.mapjunction.com/places/Boston_BRA

For the last two years I have been off contract programming in a variety of non-GIS areas. My son Joshua Pettus worked with me for the summer of 2002 and since then has been employed by futureboston to write the Flash client and the server side changes to support flash.

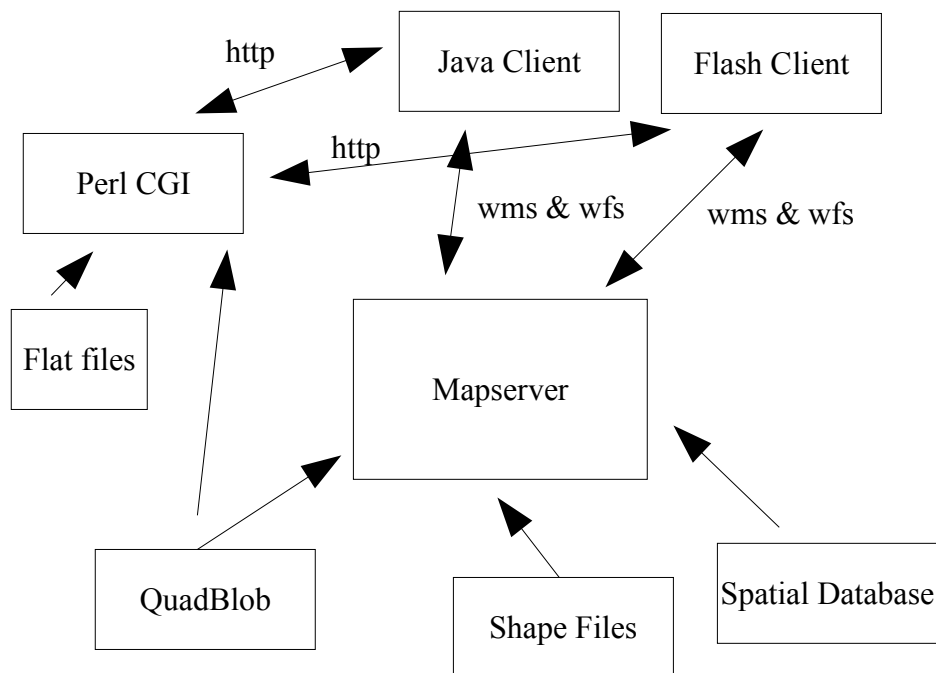
Conversion to Open Source

Mapjunction is being open sourced. In doing this I looked at the other open source web mapping projects. <http://mapserver.gis.umn.edu/> mapserver and <http://geoserver.sourceforge.net> geoserver. Both provide opengis wms and wfs functionality. See <http://opengis.org> for these acronyms.

I wanted to share as much as code as possible with other open source projects instead of providing duplicate functionality. Therefore the original raster and vector layer serving of mapjunction are being eliminated and mapserver modified and adopted as the code base for serving layers via wms/wfs interfaces.

The clients are being converted to use only opengis interfaces for retrieval of map layers.

Overall Architecture



To date the system has only been implemented on a redhat linux server. A new raster format has been added to mapserver called QuadBlob. Mapserver is the main workhorse program for serving up raster and vector layers via an opengis interface. The vector files

can be stored as they normally are in mapserver. As either shapefiles or in a spatial database such as PostGIS.

All map layers are served via wms or wfs. Untransformed raster maps can also be retrieved via wms. Example URLs.

<http://agog.com/places/mysite/dstmaps/wms> transformed rasters and vector layers

<http://agog.com/places/mysite/srcmaps/wms> untransformed rasters

<http://agog.com/places/mysite/dstmaps/wfs> vector layers

Apache rewrite rules are used to call mapserver with the appropriate arguments.

Modifications to Mapserver

The most significant change was the creating of a new raster format. See the quadblob section.

In most cases mapjunction works alongside mapserver, but there were changes that were needed. There is a very simple security system built into mapjunction to control who can view map layers. To support this a new tag was added in the mapfile. For each LAYER there can be a VIEWOWNER tag.

When mapfile is loaded, if there is an http cookie called user then that string will be compared to the VIEWOWNER tag to turn on or off a layer.

Multi-server Support

The legacy system was designed to support multiple map sources combining to appear as a single site. For instance you could have a government agency providing GIS layers that are combined with a non-profit that is hosting historical maps. This was implemented in the open source version. It would be possible to do this with the wms client support in mapserver. You will still see references to multiple sites in the findmap.pl.

Storage of map data

Lets look at one the maps from the sample site. The “1775 Boston” map was a sid file from the library of congress that was added to the site. The root path for the original layer is srcmaps/1775_Boston.

srcmaps/1775_Boston.rasmap This is mapfile for referencing the untransformed map.

srcmaps/1775_Boston.txt This contains the points picked in transapplet.

srcmaps/1775_Boston This is the quadblob directory for the source map.

If the map has been transformed twice, once with a piecewise polynomial and once with a first order polynomial, you will see the following files.

dstmaps/1775_Boston.P0.rasmap This is the mapfile for referencing piecewise map.

dstmaps/1775_Boston.P1.rasmap This is the mapfile for referencing the 1st order map.

dstmaps/1775_Boston.P0 Quadblob directory for the piecewise map.
dstmaps/1775_Boston.P1 Quadblob directory for the 1st order map .

Other extensions that may exist.

.html The java client will look here for a layer's meta information.

*.vecmap For a vector layer this has the mapfile for displaying the layer.

The separate mapfiles for each layer are combined to make dstmaps/dstlist.map and srcmaps/srclist.map. If dircount is set to a non-zero number in confit/site.txt the map layers will be stored in numbered subdirectories of dstmaps.

Transapplet

Historical maps were a big part of the “maps over time” project for futureboston. It was important to make it easy to add scanned maps to the site. A special java applet was written to perform this task. It is signed so that it can read from the local disk. You can upload an unreferenced Tiff, jpeg, or sid file into mapjunction. After the file is installed you can pick points to geo-reference the file. Each layer in mapjunction has a txt file that stores reference points. After picking points in the applet you can start a perl script on the server to transform a raster image so that it can be displayed in one of the clients. bin/dottransform.pl will find the matching points of a source and reference map. It creates input files and then forks off a bin/rubber to do the actual transform. The transform can either be a piecewise polynomial or a polynomial. For the piecewise polynomial transform a triangle mesh is created from the reference points. 3 points from each matching triangle are used to compute an affine transformation. Each destination pixel is transformed with the transform for its corresponding triangle. The destination pixel is the near-neighbor of the corresponding src pixel. Different methods such as bi-linear filtering could be added. The piecewise polynomial transform works well for old maps that are not accurate and they need to be stretched to match new maps. For newer maps or photos it is often best to use a polynomial transform. In this case all the reference points are used together to compute a global transform for the entire image. 1st/2nd/3rd order transforms can be selected.

Printing

Printing in java 1.1 was extremely limited. It was a requirement for the BRA that they have quality large paper prints. They also were primarily a windows shop. The win32 interface available with the microsoft java vm was used to call the windows win32 printing functions from the java applet. The demise of the MS java vm and the need to support other platforms forced another printing solution.

In the open source version mapserver is used to generate printouts. cgi script cgi-view/serverprint.pl. It displays a message for the client browser and passing the arguments on to workserverprint.pl to do the real work. A temporary mapfile is created and then mapserv is executed to build an output file. The mime header is pulled off the output and the file is put in the files directory for download from the client browser.

A periodic cron job is needed to delete the old print output files.

Searching in the framed Java client

Vector layers with information have a corresponding .flat file which is a pipe delimited file. This is a dump of a dbf file using dbfump. All of the files to search are listed in config/searchlist.txt. A perl program(bin/localsearch.pl) sequentially runs grep against this list of files and presents the results in a table. For the column headings there is a corresponding .schema file for each layer. It gives the column headings from a dbf file and marks columns to ignore or have hot links to graphical locations.

Address Searching

We are using the geocode services from geocoder.us. For the applet a perl cgi in cgi-view/geocode.pl will find the lat/long by using the SOAP interface available from geocoder.us. The applet is instructed to move to the location by some javascript in the onload argument to the body tag that shows the search results in the bottom frame of the browser.

3D

Elevation Data. Not yet finished.

Elevation is treated a 16 bit value that is stored in QuadBlob. Before converting to quadblob it needs to be in a pnm format. I added a new pnm format P7. The raw data is treated as 16 bit values. I previously used modified open source code to convert from USGS DEMS. I need to review this is try to take advantage of GDAL. This support has not yet been finished.

VRML Not yet finished

cgi-view/imagevrml.pl will run a C program bin/mapvrml that will generated patch of VRML for a viewing area. For the first open source release Mapserver will be queried to combine a list of map layers into a single image and this will be draped over elevation data. Eventually it would be nice to use wfs to query vector information and draw this as vectors. Also extruded building footprints need to be supported like in the original mapjunction site. Ideally these would be stored as 3D objects in a spatial database.

drawn on top. Rectangle areas can be replaced with detailed VRML models. This is done by placing their information in config/models.txt.

3d Rendered Raster Images Not yet finished

It is also possible to have a 3D scene rendered on the server with opengl and return a jpeg raster. Ideally both this functionality and the VRML functionality would be converted to

use the opengis wts interface. For now this is an http interface to a perl cgi.

QuadBlob Format

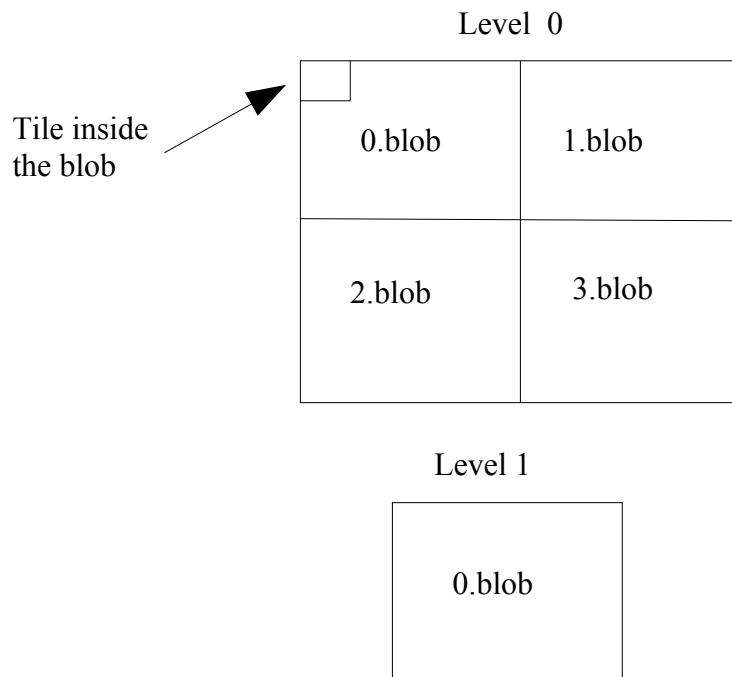
The quadblob format was created for a mapjunction.com site to serve up raster layers using mapserver.

I did not desire to write yet another raster format, but a planar organization is a fundamental assumption of GDAL and this does not allow for color space transformations when compressing as JPEG at the lower layers.

The design goals were fast retrieval of images for web serving, small footprint, support of large aeriels, efficient compression. The name blob comes from the assumption that this is not a format designed for transferring images between computer systems but a system dependent format. Byte endian handling is completely ignored. Quad comes from the fact that a quad tree is constructed for every image until it is smaller than 256x256.

The only input format that is supported is pnm. It leverages off the existing large set of open source tools to convert to pnm.

Structure of tree



Each level consists of a number of blob files. Each of these unless they are on the left or bottom must have the same width and height. The name of the blob files determines its

location in the level. 0.blob is always top left file in a level. The rest are numbered top to bottom left to right. The reason for multiple blob files per level is to avoid large file size issues. Each blob file itself is also tiled internally. This is to limit the amount of decompression that needs to happen to retrieve a given image area.

Format of blob file.

Fixed C Header

Optional compression specific information

Tile offset table

Tile data

Blobs must be either 1,2, or 3 bytes per pixel. The data can be uncompressed, or compressed with gzip or jpeg. Transparency is encoded by a special color value. If the transparency flag is set then when extracting for mapserver all pixel values of #010000 are considered clear. The optional compression specific information is only needed for JPEG. The jpeg header information is stored here so that it does not need to be replicated for each tile.

Each QuadBlob tree is a directory.

```
[gregtest]$ find srcmaps/8/Downtown_Aerial -print
srcmaps/8/Downtown_Aerial
srcmaps/8/Downtown_Aerial/0
srcmaps/8/Downtown_Aerial/0/0.blob
srcmaps/8/Downtown_Aerial/0/1.blob
srcmaps/8/Downtown_Aerial/0/2.blob
srcmaps/8/Downtown_Aerial/0/3.blob
srcmaps/8/Downtown_Aerial/2
srcmaps/8/Downtown_Aerial/2/0.blob
srcmaps/8/Downtown_Aerial/1
srcmaps/8/Downtown_Aerial/1/0.blob
srcmaps/8/Downtown_Aerial/3
srcmaps/8/Downtown_Aerial/3/0.blob
srcmaps/8/Downtown_Aerial/4
srcmaps/8/Downtown_Aerial/4/0.blob
srcmaps/8/Downtown_Aerial/5
srcmaps/8/Downtown_Aerial/5/0.blob
srcmaps/8/Downtown_Aerial/quadindex
```

Each level in the quad tree is also in a separate directory.
The quadindex file is the index for the whole tree.

```
[gregtest]$ cat srcmaps/8/Downtown_Aerial/quadindex
```

```
# levels width height bpp tilew tileh ulx uly cellsize trans
6 7877 6866 3 4096 4096 233651.875 902372.25 0.5 0
```

The first line of the file is an optional comment. For this file.
There are 6 levels. The highest resolution image is 7877x6866.
It is 3 bytes per pixel. The width and height of each blob file is 4096x4096.
The geographic location of the upper left pixel of the image is given.
Each pixel is 0.5 of the units of this geographic coordinate system.
In this case meters. The image is not transparent.

Integration into mapserver

For a mapjunction site all raster images are assumed to be in the quadblob format.

In mapraster.c the following call is inserted

```
return(quaddrawlayer(layer->data, &map->extent,img));
```

Two new files are added to the distribution.

mapquadblob.c wrapper to call quadextract.

quadread.c C needed to extract an image for mapserver.

Command Line Tools

Usage: **quadsingle**: Convert a single blob to and from pnm format

[-i] inputFileNames - Input image filename

[-o] outputFileNames - Output image filename

[-tilex 256] tilex - Width of Tiles

[-tiley 256] tiley - Height of Tiles

[-c none] compression - Compression type. (none,jpeg,gzip)

This tool is used to create the next level in a tree.

Usage: **quaddownsize**: Take 4 blobs and downsize to one pnm

[-i] inputDirectory - Input directory for the blobs

[-o] outputFileNames - Output pnm filename

[-1 -1] blob1 - Upper Left Blob

[-2 -1] blob2 - Upper Right Blob

[-3 -1] blob3 - Lower Left Blob
[-4 -1] blob4 - Lower Right Blob

Usage: **quadextract**: Extract a rectangle from a quadblob tree to pnm file.

[-i] inputFileNames - Input QuadBlob tree
[-o] outputFileNames - Output image filename
[-minx 0.000000] minx - Left of Rect
[-maxx 0.000000] maxx - Right of Rect
[-maxy 0.000000] maxy - Top of Rect
[-miny 0.000000] miny - Bottom of Rect
[-w 0] w - Width of resulting image

Usage: **quadinfo** dir

Dumps the index of a quadblob tree into an easy to parse format.

Usage: **quaddecompress**: Extract a complete quadblob tree to pnm file.

[-i] inputFileNames - Input QuadBlob tree
[-o] outputFileNames - Output pnm filename

There are several Perl scripts that used to build trees using the above C executables.

./quadbuildeasy.pl

Build a quadblob tree from a single pnm.

Options:

-x X coordinate of upperleft corner
-y Y coordinate of upperleft corner
-s Size of pixel at the first level
-c Compression [none,jpeg,gzip]
-i path to input pnm file
-o output directory for the tree

quadbuildfromtiles.pl

Build a quadblob tree from a set of tiles.

Options:

- c Final Compression [none,jpeg,gzip]
- f [0,1] compress as we go
- i list of world files. sdw,jfw, or tfw
- o output directory for the tree
- q quiet mode

./quadnextlevel.pl

Create a new level half the size of this one

Options:

- w width of source level
- h height of source level
- t tilesize of source and destination level
- c Compression [none,jpeg,gzip]
- i source level
- o output directory for the tree

./quadcomplevel.pl

Compress all the blobs in this level.

Options:

- c Compression [none,jpeg,gzip]
- i source level
- o output directory for the tree**