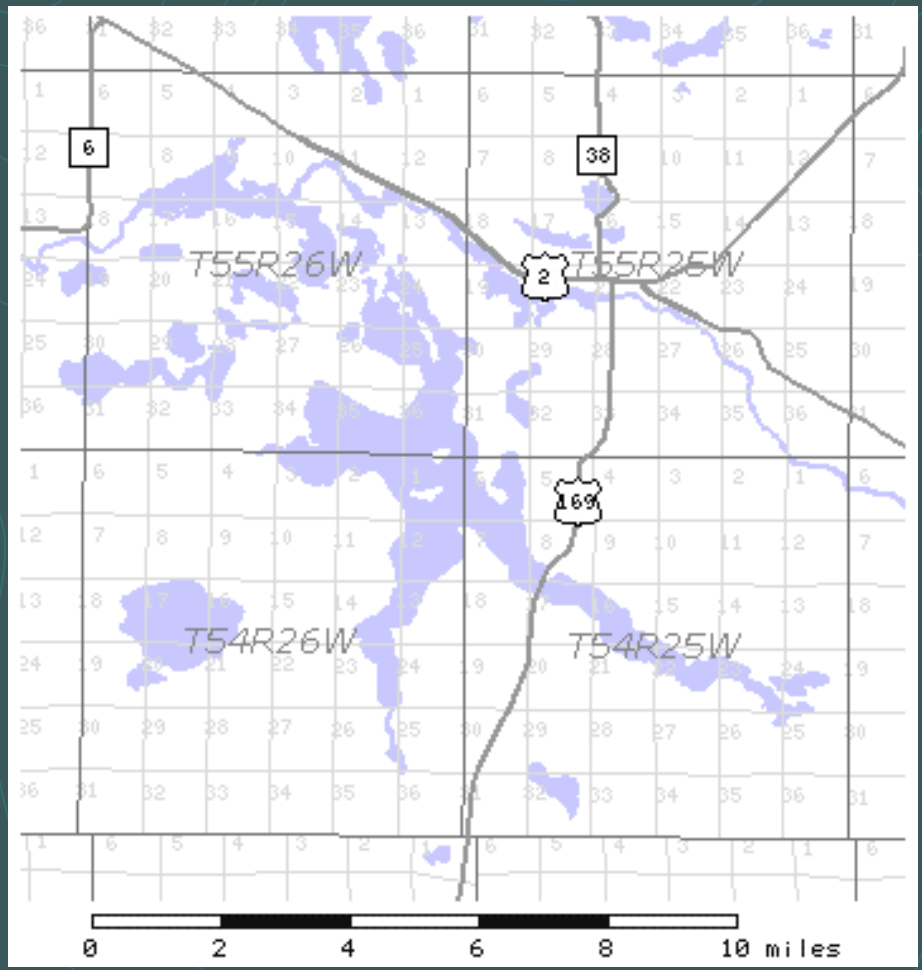
MapServer/MapScript: The Web and Beyond

Minnesota DNR Forestry Resource Assessment MapServer Applications on and off the Web

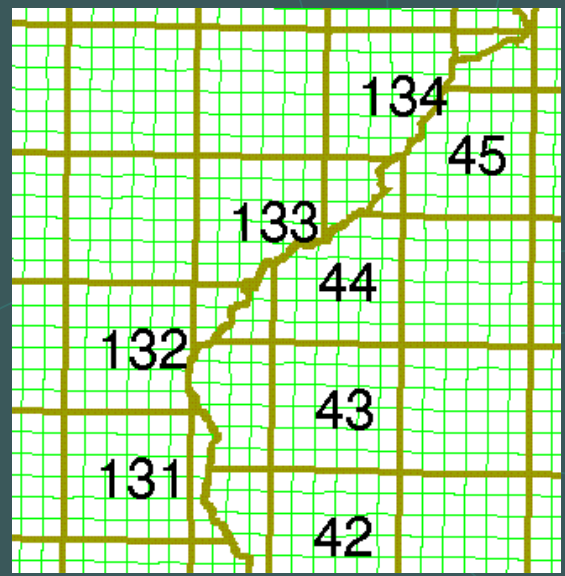
Fire Reporting System

- A java/jsp distributed data entry system written to replace aging COBOL system
- Fire locations recorded using PLS descriptors
- MapScript is used to:
  - Convert PLS descriptors to UTM coordinates
  - Perform validation checks on locational data
    - Does the administrative area entered by the user match the administrative area returned by MapScript?

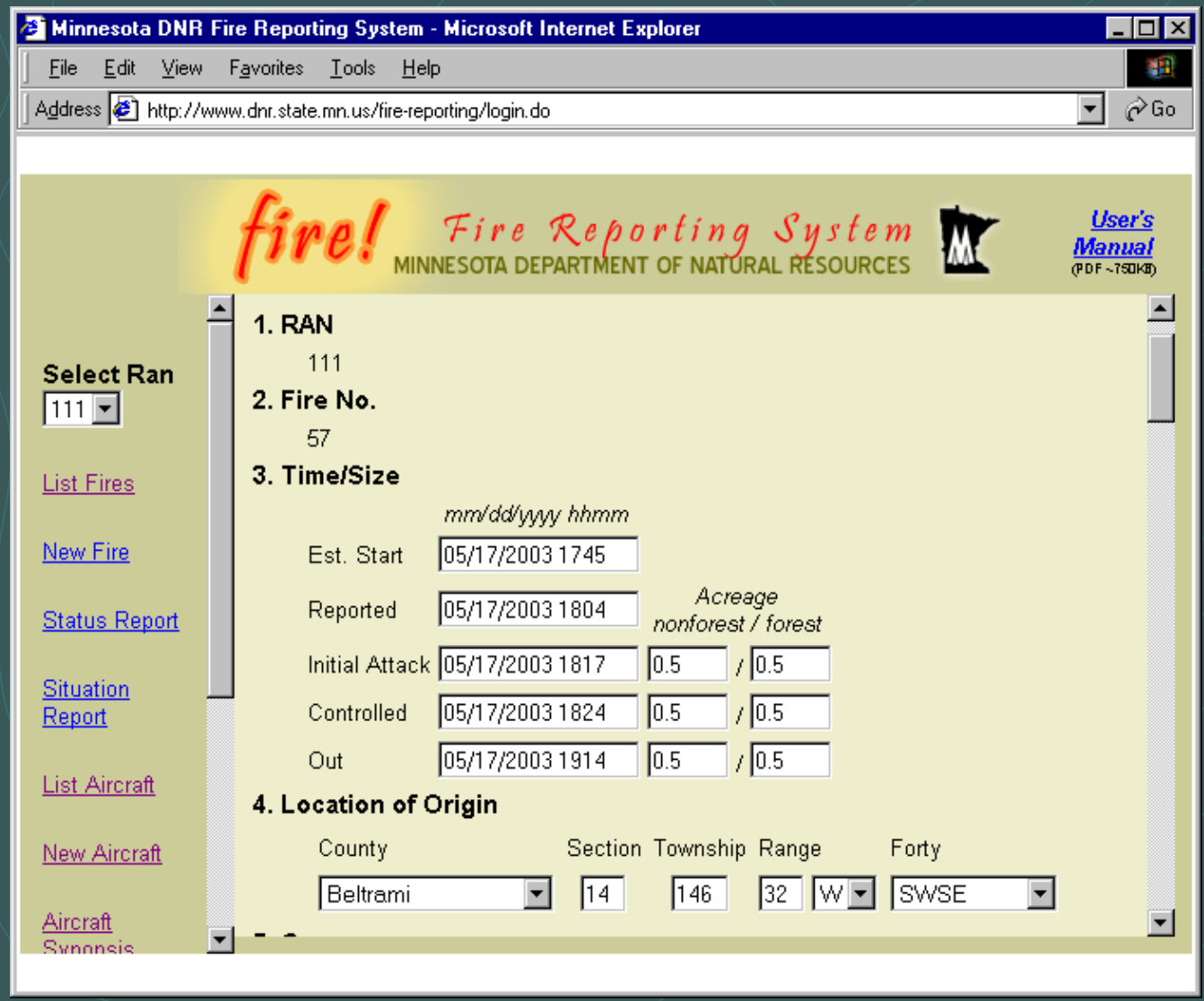
Public Land Survey (PLS) Background



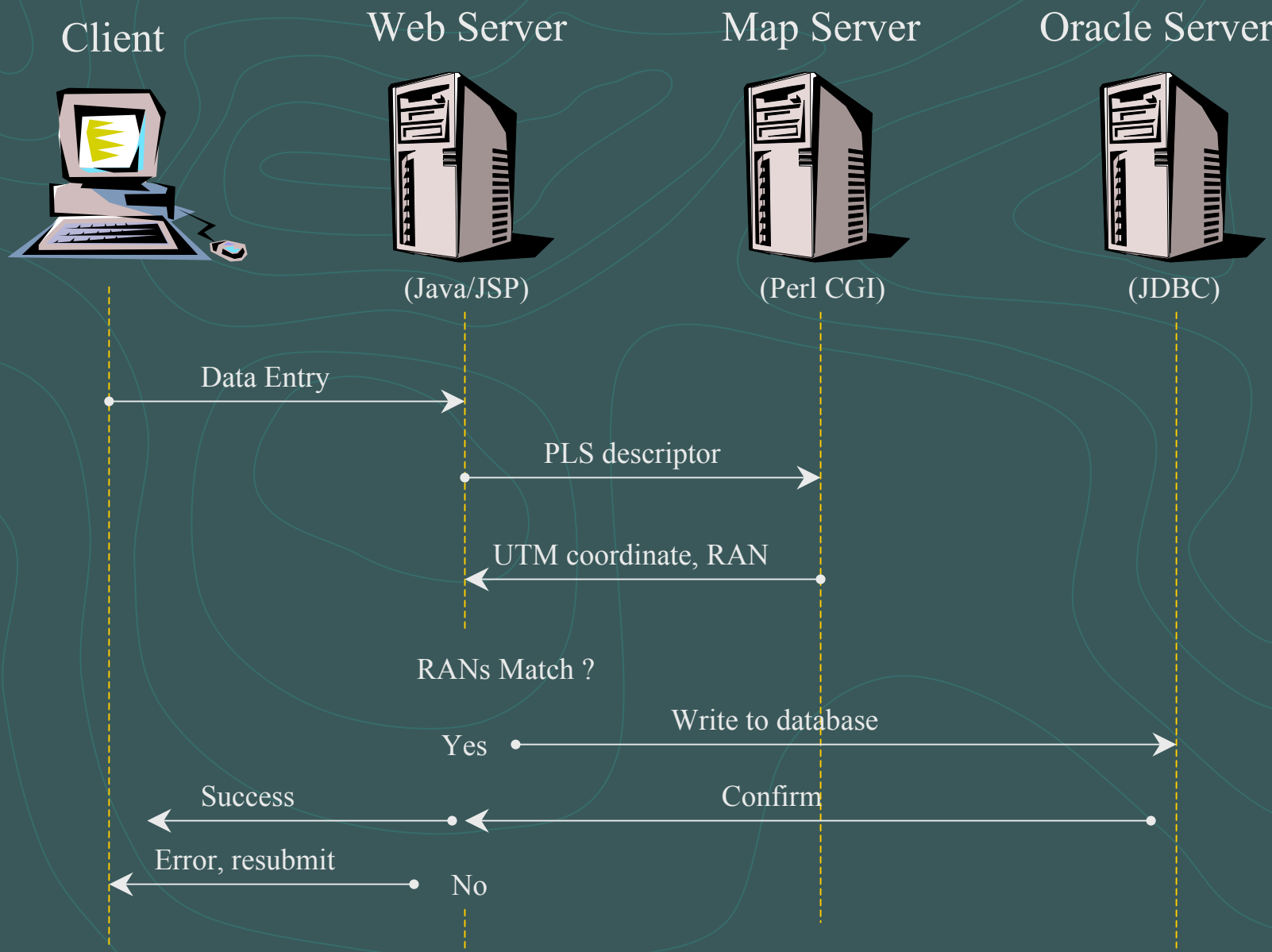
- Standard US cadastral system
- Townships consisting of 36 one square mile sections
- Each section subdivided into 16 forty-acre parcels, or “forties”



Data Entry Screen



Data Processing Flow



PLS-to-UTM Conversion

- Location is specified to the PLS forty level
- 1.3 million PLS forties in MN
- Use staged query:
  - Lookup township attribute in township layer
  - Set map extent to resultcache bounds from previous query
  - Lookup section attribute in section layer
  - Set map extent to resultcache bounds from previous query
  - Lookup forty attribute in forty layer
  - Retrieve forty shape, nominal center ($shape->getLabelPoint($point)) is our UTM coordinate

Locational Validation

- UTM coordinate for fire has been determined
- Simple queryByPoint against administrative layer tells us what administrative unit (RAN) that coordinate belongs to
- Return coordinate and RAN data to web server
- Check that returned RAN matches user-entered RAN.
  - Yes, write to database
  - No, return error to user and request resubmission

Setting up MapScript Queries

```
$map = new mapscript::mapObj(undef);
$queryLayer = new mapscript::layerObj($map);
$queryLayer->{data} = $shpToQuery;
$queryLayer->{status} = $mapscript::MS_ON;
$queryLayer->{template} = 'dummy';
$qclass = new mapscript::classObj($queryLayer);
$rv = $queryLayer->queryByPoint
        ($map,$point,$mapscript::MS_SINGLE,0);
$rv == $mapscript::MS_SUCCESS or warn "Unsuccessful
          queryByPoint" and return;
```

MapScript as a Spatial Data Processing Tool

- If you have a favorite scripting language that MapScript supports, you can leverage it for (lightweight) spatial data processing tasks
- Save your expensive proprietary software packages for where you really need them
- Avoid limitations of proprietary Archaic Macro Languages
- Integrate with other available scripting environment libraries (eg. Xbase in Perl)

Orthorectification Setup

Aiming for production processing of 4-5,000 photos annually, or 60-70 “blocks”.

In order to orthorectify a block of photography, we need:

- A shapefile showing estimated photo boundaries
- A shapefile showing estimated photo effective areas
- Digital scans of the relevant photos
- A textfile with “exterior orientation” parameters
- A mosaic of DOQ images for the area of interest

all stored according to a predefined directory structure and file naming system.

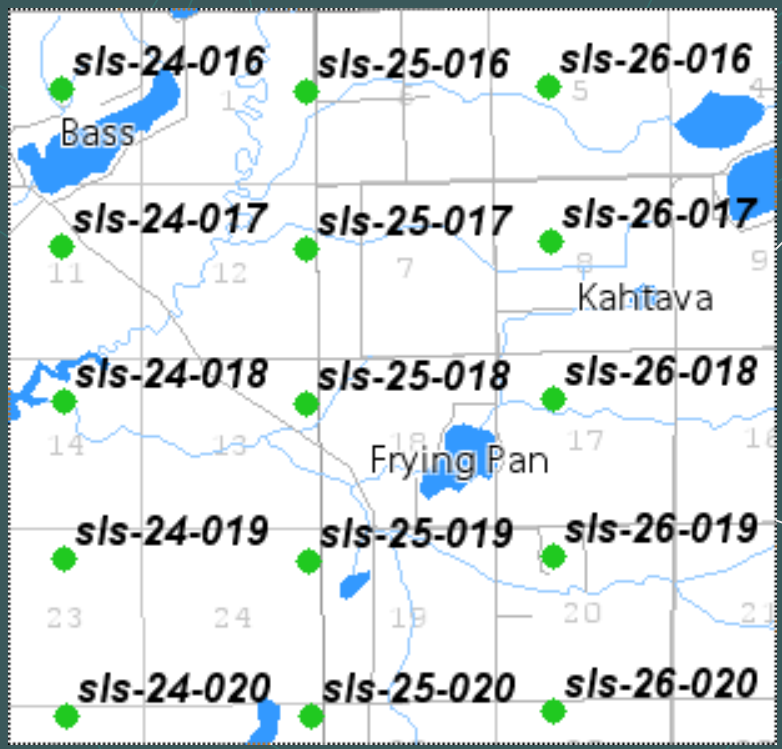
Given:

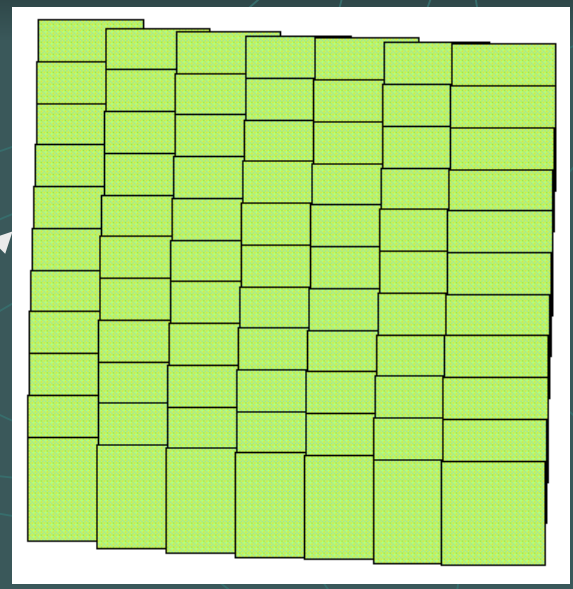
- A digital flightline index (shapefile) with:
  - an ID attribute matching photo filenames
  - a block attribute identifying which block each photo belongs to
- A spatial index to the DOQs
- The scale and format (eg., 9x9 inch) of the photography
- A well-structured photo scan and DOQ library

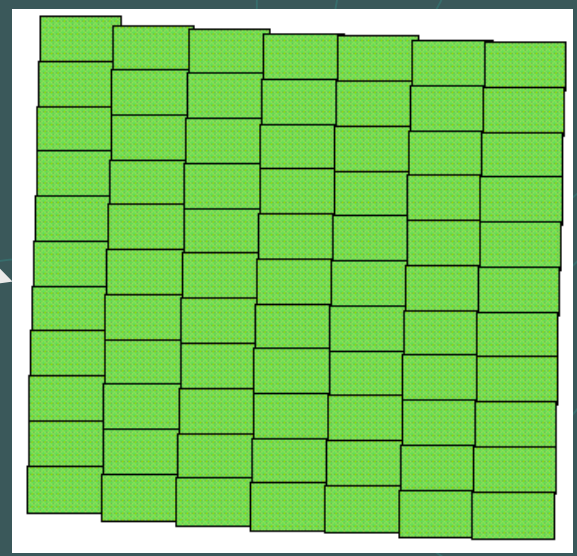
Then we can…

MapScript It!

Convert photo centroids to boundaries

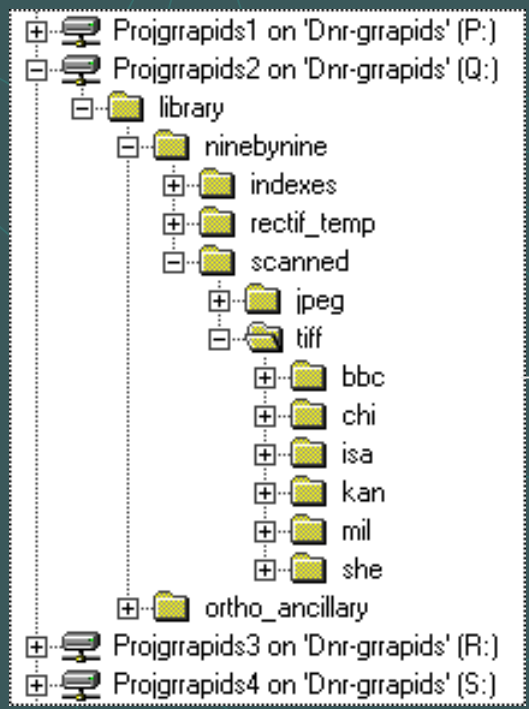


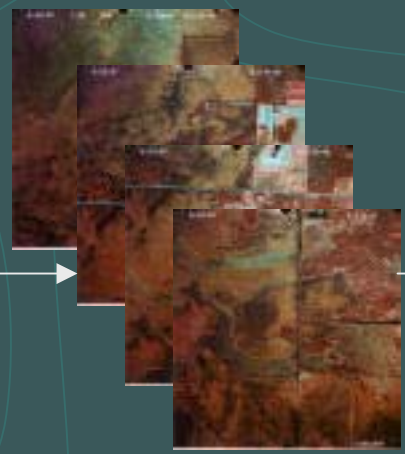


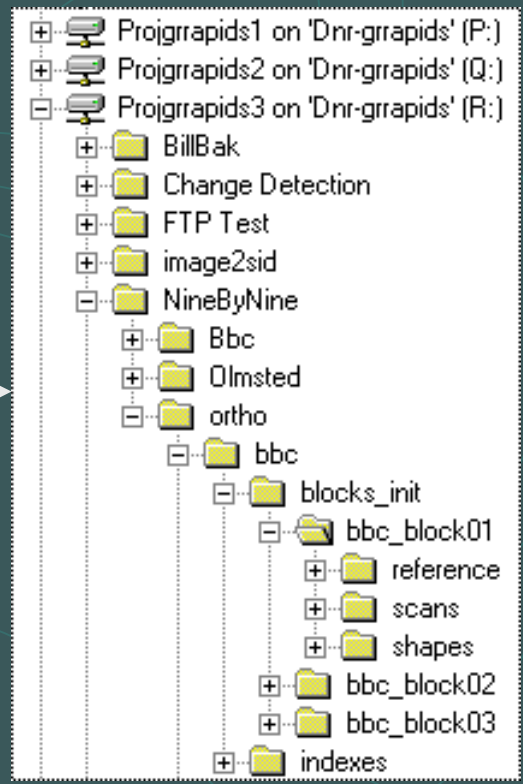


MapScript It!

Retrieve appropriate photo scans

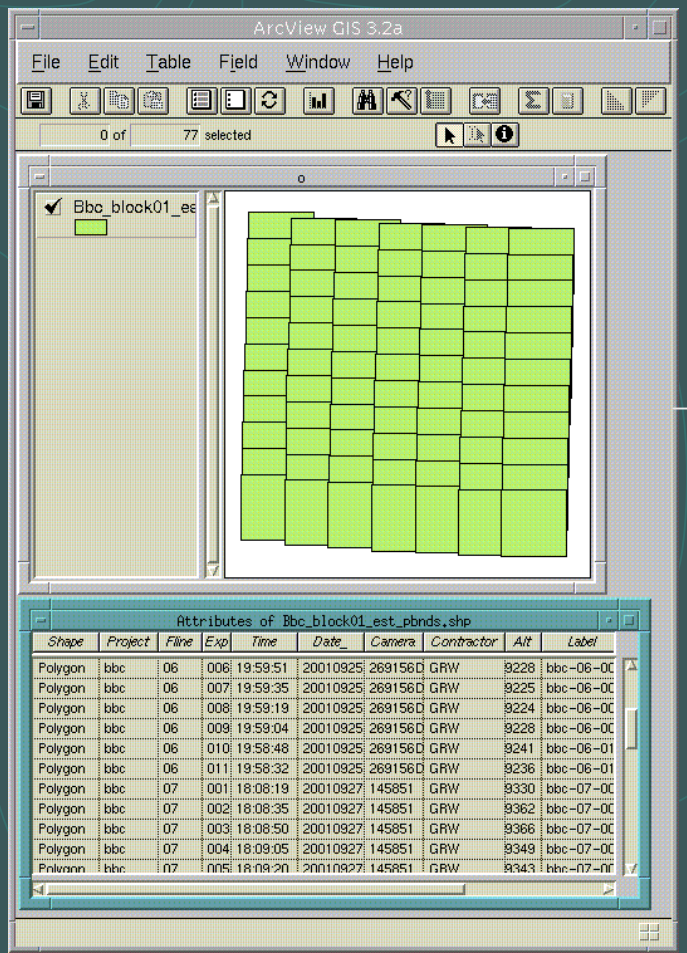


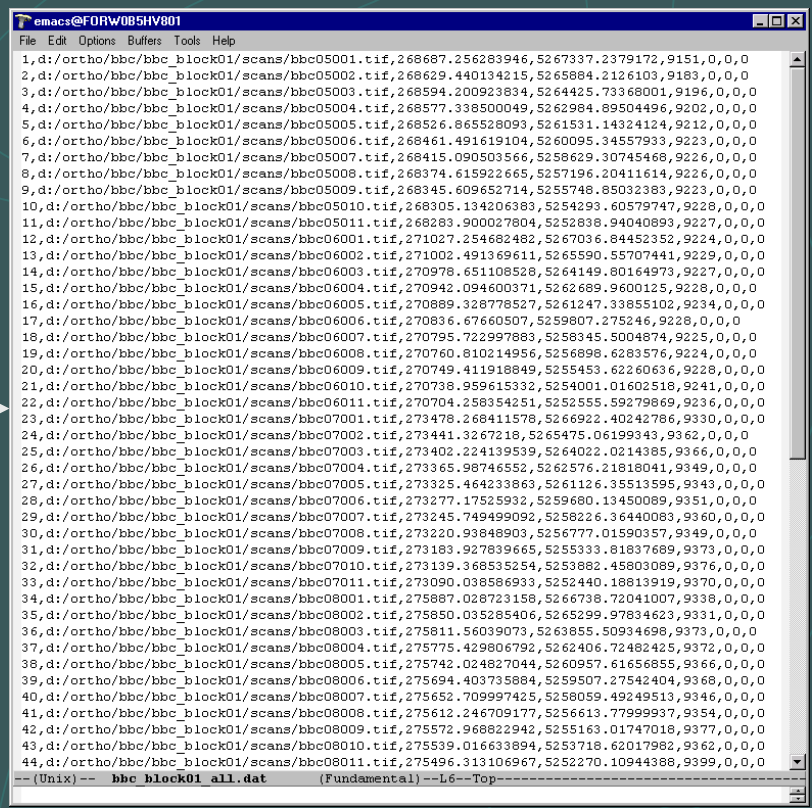




MapScript It!

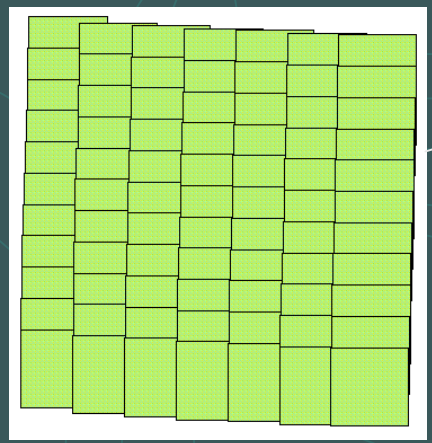
Dump required attributes to ASCII text



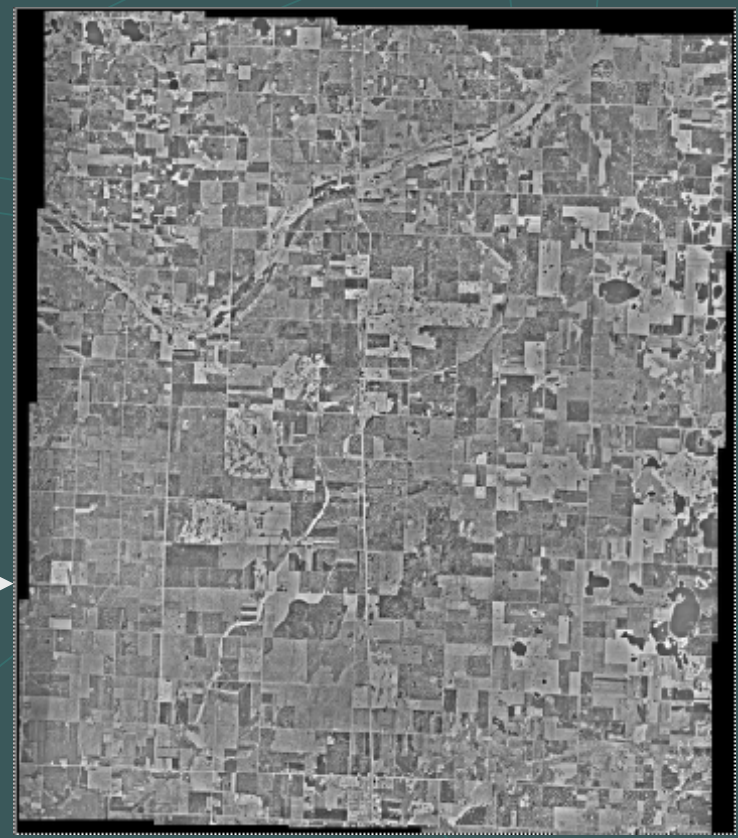


MapScript It!

Find and mosaic required DOQs



queryByFeatures



"When the only tool you have is a hammer, every problem starts to look like a nail."