Designing a Web Mapping Prototype for Land Use / Land Cover Forecast Animations: A webGIS for USGS White Paper

by Bill Lorand

Contents at a Glance:

- - -

0- Executive Summary

To evaluate options for publishing USGS land use and land cover forecasting data on the Internet, various client-side technologies were evaluated and a web mapping prototype for raster data animation was produced during a webGIS consulting project with the Western Geographic Science Center.

The (free) Google Earth web browser plugin proved the most useful evaluated method for displaying and animating raster image overlays. KML of the overlays was exported from ArcMap, and a web mapping prototype using the plugin with a user interface of XHTML, CSS and jQuery UI JavaScript widgets was produced.

For web visualization of the ecoregions at different levels/scales, a Google Maps API prototype using Google Fusion Tables initially looked promising, but proved to have technical limitations that rendered it unsuitable as an interactive visualization tool for polygons at multiple scales, so a full web prototype was not produced.
(Other technologies such as the ArcGIS Flex API may prove more robust for this type of data visualization, and would likely involve a server-based or cloud component).

Finally, Google Charts appears to be a promising (free) online tool to provide some accompanying analytics to the raster data. This would be a likely area for future technical evaluation and prototyping, but is out of scope for the initial project phase covered by this report.

1- Evaluating Animation Implementations

1.1 ArcMap Export to AVI

This known method served as a starting point for evaluating raster animation methods, and while it is serviceable, part of the intention of this webGIS consulting project was to find a more attractive and usable method to provide user-friendly animations of raster data through time.

1.2 jQuery Slider

Taking a cue from the cal-adapt.org web mapping application, a web prototype was produced for a jQuery UI "slider widget" that provided a usable animation for a series of rasters for the 9.2 level two ecoregion exported as .jpg files from ArcMap (see screenshot below) . While a user could scroll thru time using the interactive slider, even with the proposed addition of play and pause controls, these .jpgs showing land cover changes remained "flat" images, lacking georeferenced
basemap data behind them to provide context for any meaningful analysis.

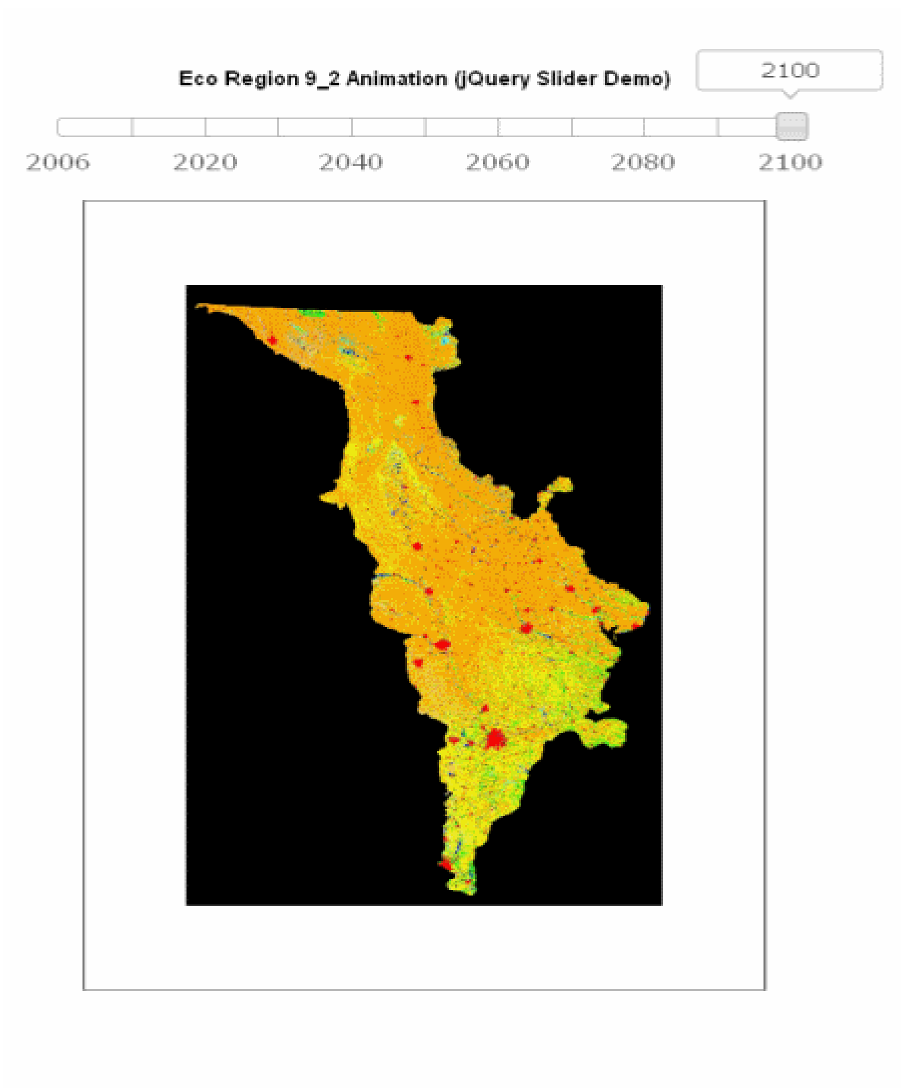

Figure 1: jQuery slider for raster animation

1.3 KML Overlay in Google Earth / Google Earth plugin

After seeing that Google Maps had limited support for image overlays (especially animated ones), an option that generates KML files from ArcMap as one continuous image was used to create .png files that served as the input to a KML animation for Google Earth. KMZ archives that ran these animations were produced using the steps outlined in *Appendix A: Creating Google Earth KML Animations from KMZ files with LULC ground overlay images exported from ArcMap*. After testing the files in Google Earth, these were ported over for use in the Google Earth web browser plugin through the dynamic creation of a <NetworkLink> KML tag.

These raster animations proved successful as they place the land cover changes over georeferenced satellite imagery provided by Google Earth, and also include the native Google Earth time slider. While some open issues remain (an ArcMap bug about the transparency of exported images, programmatic control of the speed of the time slider, and the lack of an on/off toggle for the raster image layers among them), these KML overlays in the web browser proved to be the most useful evaluated method and are featured in the webGIS prototype.

2- Webmap Prototyping with XHTML, CSS and jQuery

2.1 XHTML and CSS Column Layout

To house the raster animations in a web user interface, a standards-based XHTML structure was coded along with CSS for the column layout and styles. Layout considerations followed newly established conventions for designing web mapping applications and took cues from the layouts of ArcMap, Google Earth and sites such as parkinfo.org: a top banner for logo and title, a left column for legend, layers and here a download section, and a main "map portal" area on the middle right of the screen for map and data display.

2.2. jQuery UI JavaScript widgets

The jQuery UI JavaScript framework was used to great effect in the webGIS prototype for conserving screen real estate and providing a seamless and responsive user interface with its various widgets: jQuery "tabs" were used for toggling between animations and contextual content on the different emission scenarios, an "accordion widget" provided legend information and KMZ download links in a changeable vertical column like the familiar "Layers" and "Places" in the Google Earth client, and a "tooltip" UI plugin gave full definitions for the land use classification legend items on mouseover. jQuery proved to be very flexible, easy and quick to code with it's workhorse JavaScript files hosted in the Google cloud. It feels like a great and natural fit for web mapping applications. API details are available at http://jqueryui.com/ .

3- Areas for Further Technical Evaluation and Prototyping

3.1: Google Charts

Providing charts and analysis of the land use / land cover changes reflected in the raster animations was another desired outcome of the webGIS prototype project. While it is out of scope for the initial phase covered by this report, Google Charts looks like a promising (free) online tool to provide some accompanying analytics to the raster data, This would be a likely area for future technical evaluation and prototyping. Details about Google Chart Tools are available at **http://code.google.com/apis/chart/** .

3.2: Ecoregion Visualization

Initially, the webGIS prototype project tried to find an overarching way to provide a web interface that would allow users to view, analyze and download data at all three ecoregion scales/levels. This was a big challenge for data visualization and would require a flexible, zoomable, and intelligent web application that among other things could display visual and textual clues on the current region and region level in view (perhaps via color-shaded polygons and text "website breadcrumbs").

3.2.1 Google Maps and Fusion Tables

Trying to utilize Google Maps with the new Fusion Tables API to serve basic spatial data for the polygons was partially prototyped (see screen shot below), but a number of limitations of this model quickly surfaced.

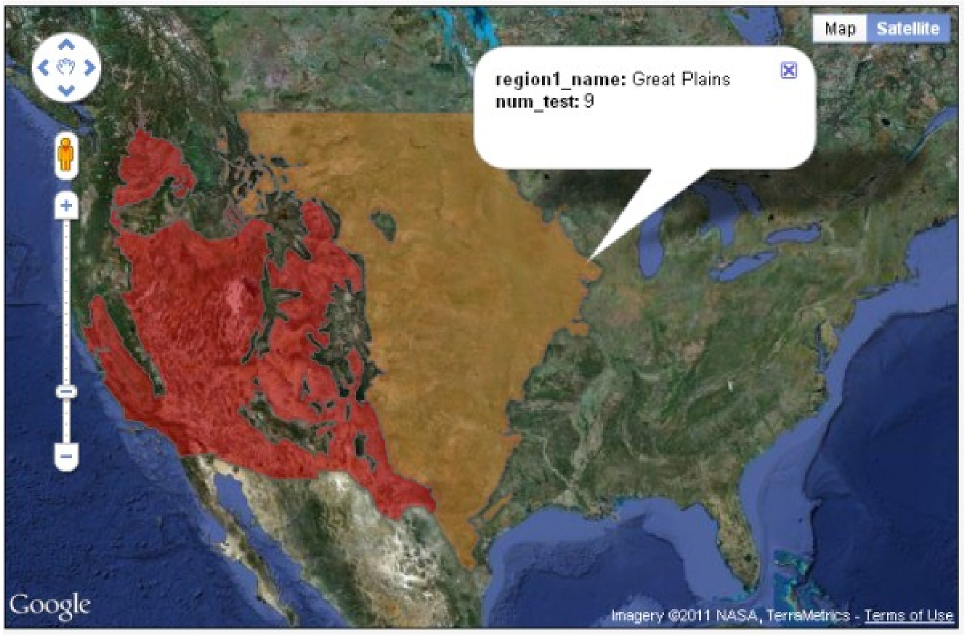
Map
Satellite
region1_name: Great Plains
num_test: 9
Google
Imagery ©2011 NASA, TerraMetrics - Terms of Use

Figure 2: Google Maps / Fusion Tables UI for data visualization of level 1 ecoregions

Fusion tables is very limited in the number of "styles" (read: colors) you can place on a Google Map. This would allow for symbolization of the four complex polygons at the ecoregion level one scale, but nothing more. Also, Fusion Tables is a new "experimental" Google Code feature whose support can be ended at any time. But the real issue with this initial approach is that it is just the wrong tool for this web application. The Google Maps / Fusion Table API is more appropriate for simple vector datasets and seems best suited for citizen cartographers making basic thematic maps (think ArcGIS Explorer), not to provide a multi-layered data visualization tool to get at complex temporal raster datasets.

3.2.2 ArcGIS Flex API

One technology that may prove to be a robust choice for creating this type of complex data visualization is the ArcGIS Flex API. A flash-based user interface might offer more control for the custom zoom levels and thematic overlays needed to communicate the boundaries of the land use ecoregions at the various levels. This would be another area for further technical investigation and prototyping in a future project phase. ArcGIS Flex API details are available at **http://help.arcgis.com/en/webapi/flex/** .

Appendix A:
Creating Google Earth KML Animations from KMZ files with LULC ground overlay images exported from ArcMap, step by step.

0- Starting point: Overlays folder with 10 kmz files/10 images posted on ftp site.
example: ftp://ftpext.usgs.gov/pub/wr/ca/menlo.park/jjones/A1B_overlays/

1- **Unzip/extract** 10 **kmz files** and rename generic doc.kml and composite.png files based on the represented year.

example: eco9_2_A1B_kansasCity_2010.kmz that contains doc.kml, composite.png becomes individual files eco92_A1B_kc_2010.kml, eco92_A1B_kc_2010.png
rationale: .png files need to be renamed by year to differentiate them; .kml files need to be isolated to be compiled into a larger kml file. (all files are generically named composite.png, doc.kml)

2- **Start compilation kml file**
Use one of the snapshot kml files as the start of your compilation kml file to hold the code that will become your animation (or start with a fresh kml file in a text editor). Make sure your file is valid kml and starts and ends with <xml><kml> </kml></xml> tags with the required namespace and encoding attributes).

3- **Edit KML: Add <Folder>**
Replace the <Document> tag with a <Folder> tag to hold what will be a series of <GroundOverlay> tags. This will create a mini directory structure in the Google Earth window for toggling on and off different year-specific image layers. Also provide an appropriate <Folder><name> tag.

example: <Folder><name>Kansas City Land Cover Change Animation 2010-2100 (A1B scenario)</name>...</Folder>

4- **Edit KML: link image, provide <GroundOverlay><name>**
Make sure the renamed composite image is linked correctly in the kml. Also, provide appropriate text for <GroundOverlay><name>

example: <Icon><href>images/eco92_A1B_kc_2010.png</href></Icon>
<GroundOverlay><name>KC 2010 composite image</name></GroundOverlay>

5- **Edit KML: Add <Timespan>**
Add <Timespan> tags nested inside <GroundOverlay> to power the animation. (There should be no gaps or overlaps between date/timestamps - for this animation we only require begin and end years).

example: <TimeSpan><begin>2010</begin><end>2020</end></TimeSpan>
&& the next <GroundOverlay> would contain
<TimeSpan><begin>2020</begin><end>2030</end></TimeSpan>

6- **Edit KML: Add <color> to make the overlay image have 50% opacity**
**Note*: there are some open issues with this approach - on first load the opacity works for the first image or two, but then it appears to "stack" the images, which wipes out the 50% opacity and makes them 0% opaque - this could be a bug in Google Earth - alternate approaches could include using "shared styles" in the kml, or adding opacity to .png files with image editing software. (open issue)

example: <color>7fffffff</color> (the first two digits '7f' control opacity in kml)

7- **Save / test compilation kml file**
Now the compilation kml file should have one complete GroundOverlay entry for 2010, including links to a composite image and an appropriate timespan; It would be a good idea to save this file at this point and test it by opening it in Google Earth.

example filename: lc_change_KC_2010-2100_A1B.kml

8- **Add all GroundOverlays for the animation**
The next step is to "cut and paste" the <GroundOverlay> from the 2020 file into your compilation file, then make the necessary kml edits from steps 3-6 above; Do this for all other .kml files up to the 2100 file. (10 files total)

9- **Confirm consistent <LatLonBox> entries**
After there are 10 <GroundOverlay> tags in your compilation kml file, pick a representative <LatLonBox> and associated N,S,E,W sub-elements and paste this in for all 10 <GroundOverlay> entries;

All bounding box entries may look fairly similar, (or identical up to 3 decimal places), but may have small decimal degree differences [perhaps] based on the visual extent when they were exported from ArcMap. Making all <LatLonBox> entries exactly the same should prevent any "image jumping" when the animation plays.

10- **Create a .kmz archive for serving the animation**
Now you should have a fully functional animation showing Land Cover change for the selected region for 2010 - 2011.

Next, create a .kmz archive for serving the animation. Zip up the file and the associated directory structure for supporting files using a program like WinZip (for Windows) to create a new archive. In this case, this consists of an /images/ subdirectory. After a new archive is created, manually change the file extension from .zip to .kmz (Please note that each .kmz archive can only include one .kml file - this is usually titled doc.kml but descriptive kml filenames are allowed and more useful).

Finally, test your .kmz by opening it in Google Earth. It should "fly to" the zoomed-in region, a time-slider should appear in the Google Earth window and it should run the animation. (You can adjust your animation settings in google earth to control speed and number of stops)

Appendix A Useful Links:

KML Reference:
http://code.google.com/apis/kml/documentation/kmlreference.html
KMZ Files:
http://code.google.com/apis/kml/documentation/kmzarchives.html
KML Time and Animation:
http://code.google.com/apis/kml/documentation/time.html