Augmented Reality, Google Earth and Tcl/Tk

Steve Landers and David Roseman
steve@digitalsmarties.com
dlr@eolas.com

Abstract

This paper will describe the implementation of an augmented reality simulation of a proposed wind farm, combining Google Earth (either standalone or via a browser plugin) with animated turbine models to give realistic impressions of the visual impact of a proposed offshore wind farm in Lake Michigan.

Several aspects of the simulation will be discussed, including the use of a Tcl/Tk application for the generation and layout of KML (geographic data) files, the use of Tcl CGI programs for generating dynamically sized images and data, and a Tcl/Tk program to control Google Earth.

Mention will also be made of some of the non-programming issues encountered, including visual perception issues such as Field of View and the Moon Illusion.

Background

This project was prompted by a proposal to place a 100-200 turbine wind farm near the shore in Lake Michigan. On hearing of the proposal, the question asked by local residents was “what would these things look like? “. And so a method of drawing accurate representations from arbitrary viewpoints was required.

After considering OpenGL[1] based approaches, it was decided to first investigate “off the shelf” or “consumer” solutions since one requirement was the ability to view the results on a typical desktop or notebook computer.

Google provides a couple of tools that seemed promising. Google Sketchup[2] is a drawing program that allows accurate, scaled renderings of three dimensional objects. Google Earth[3] allows 3D objects (including Google Sketchup models) to be placed in arbitrary locations. Any modeled object can be placed in any geographic location, at any orientation. Google Earth provides a "camera" which allows the rendered objects to be viewed from any location. The data may be passed to Google Earth using Keyhole Markup Language (KML) [4].

There were some desired features that Google Earth did not appear to support, especially animation and variable "focal lengths" of the camera. These would need to be implemented using work-arounds, but even so, it was decided to proceed with a Google Sketchup / Google Earth solution.

Two approaches were developed in parallel, a standalone Tcl/Tk program that controlled the Google Earth application, and a web-based version using Tcl and Javascript to control the Google Earth web browser plugin. The advantages and disadvantages of each will be discussed in this paper.

The rough sequence of development was:

- obtain and modify turbine model(s) and other models
- define turbine locations
- determine viewing parameters (location, elevation, viewing angles)
- build KML files
- define methods of passing data to Google Earth (both standalone and browser plugin version)
- validate visuals

Turbine Models

The first step in producing the wind farm simulation was to obtain suitable Google Sketchup models of approximately the same size as the proposed turbines. The Sketchup community has made available a large number of models, including wind turbines. A suitable model was downloaded, and re-scaled to the dimensions of the proposed turbines and the model was exported to a Collada Digital Asset Exchange (.dae) file [5], the format required by Google Earth.

The dimensions used were those of a 10 MW turbine being developed by Clipper Windpower [6]. The height to the top of a vertical blade is 574 feet.

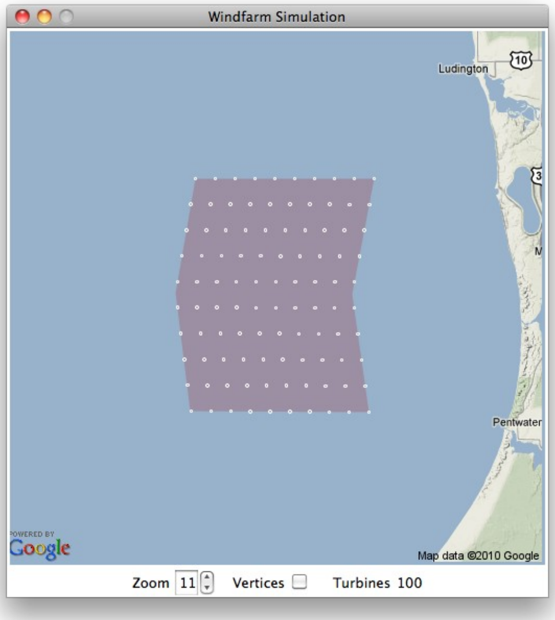
Laying out the wind farm using Google maps and Tcl/Tk

The next step in producing the wind farm simulation is to produce a Google Earth model with the required number of turbines equally spaced throughout the wind farm footprint.

Ideally, the geographic coordinates of the footprint would be provided by the wind farm developers. In this case, a polygon of the footprint was printed in the local newspapers and, by inserting a copy of this into a geographic information system (GIS) program, the latitude and longitude of the polygon vertices were determined.

A Tcl/Tk program was written that accepts two inputs: the number of turbines, and a series of rhomboid polygons describing the wind farm footprint (the vertices of each polygon are specified in latitude/longitude). The program produces a Google Earth model with the required number of turbines equally spaced throughout the wind farm footprint.

To allow visual inspection of the layout by the user, the program uses the Google Maps REST API [7] (via the Tcl http [8] package) to retrieve and display a static map containing the wind farm footprint. The map is returned as a 512x512 gif image of the local area, containing the wind farm footprint as a shaded polygon. The program then places the turbines equi-spaced within the wind farm footprint. The number of pixels per kilometre is calculated using the spherical geographical distance code from the Geodesy page on the Tclers' Wiki [9]. This is used to calculate the latitude and longitude of each turbine location, and the static map (displayed in a Tk canvas) is overlaid with a dot indicating the location of the turbine.



Zoom 11 Vertices ☐ Turbines 100

To generate the Google Earth model, the program substitutes the latitude and longitude of each turbine into an XML template, along with the bearing corresponding to the prevailing wind direction. The result is a single KML file describing the entire wind farm. To this is added the Sketchup model(s) for the turbines, and the resulting files archived as a zip (KMZ) file (the typical distribution format for a Google Earth model).

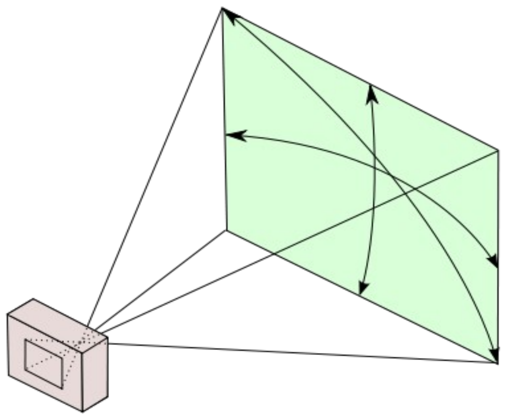
Viewing Parameters

The next step in producing the wind farm simulation was to produce Google Earth KML models for each of the desired viewpoints. Each viewing point comprises a latitude and longitude, altitude (above the ground level), heading (i.e. the direction the camera faces) and the tilt and roll of the camera. These are read from a text file, and substituted into an XML template, generating one KML file per viewing point.

This information is the simple part of the viewing parameters. The more difficult part to achieve is the Field of View.

The Field of View (FOV or Field of Vision or Angle of Vision) [10] is the angular extent of a given scene that can be viewed at a point in time. The FOV has two components, the horizontal and vertical FOV values (hFOV and vFOV respectively).

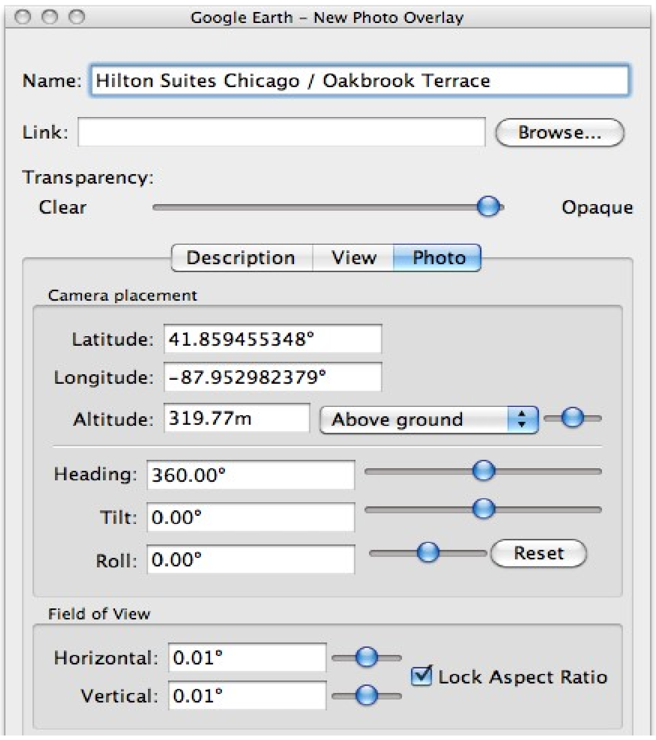
The Design Eye Point (or Design Eye Position) [11] is the position from which the user will optimally view the user interface (and therefore, achieve the most realistic visual perception of the simulation). In essence, this says the FOV value to use is a function of the size of the screen and the distance the viewer is from it.



But Google Earth has a fixed FOV of approximately 60 degrees horizontally and no way of changing it. The first simulations based on this resulted in the turbines appearing too small.

This appeared to be a show-stopper, but research revealed that the FOV can be approximated using Photo Overlays.

A Google Earth Photo Overlay [12] is a way of overlaying a photo on a Google Earth scene. When specifying the photo overlay, it is possible to specify values for the horizontal and vertical fields of view. If the image is specified as a transparent png, the FOV values are applied to the underlying Google Earth scene, resulting in an approximation of the required FOV values.


Google Earth – New Photo Overlay

Name: Hilton Suites Chicago / Oakbrook Terrace

Link:

Transparency: Clear — Opaque

Camera placement

Latitude: 41.859455348°

Longitude: -87.952982379°

Altitude: 319.77m Above ground

Heading: 360.00°

Tilt: 0.00°

Roll: 0.00°

Field of View

Horizontal: 0.01°

Vertical: 0.01°

[✓] Lock Aspect Ratio

So the decision was made to derive the FOV values based on an arbitrary viewing distance of 30 inches and the current screen size. The intention is to eventually provide several viewing distances, corresponding to a notebook, desktop, group, and auditorium settings.

The formulae for calculating the FOV are:

hFOV = atan(width / distance)*180/pi

vFOV = atan(height / distance)*180/pi

These can be derived by the simulation program (either standalone or web-based) at runtime using the width and height values for the current computer's display.

So the Google Earth KML files for each viewpoint were generated containing references to the FOV values (rather than the actual values).

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
     xmlns:gx="http://www.google.com/kml/ext/2.2"
     xmlns:kml="http://www.opengis.net/kml/2.2"
     xmlns:atom="http://www.w3.org/2005/Atom">
    <PhotoOverlay>
        <name>Ludington South Pierhead</name>
        <Camera>
            <longitude>-86.46923</longitude>
            <latitude>43.95198</latitude>
            <altitude>4</altitude>
            <heading>-146</heading>
            <tilt>85</tilt>
            <roll>0</roll>
            <altitudeMode>relativeToGround</altitudeMode>
        </Camera>
        <Icon>
            <href>http://$url/cgi-bin/transparent.png</href>
        </Icon>
        <ViewVolume>
            <leftFov>$leftFOV</leftFov>
            <rightFov>$rightFOV</rightFov>
            <bottomFov>$botFOV</bottomFov>
            <topFov>$topFOV</topFov>
            <near>0.1</near>
        </ViewVolume>
    </PhotoOverlay>
</kml>
```

These KML files are served at runtime from a Tcl CGI script [13] that accepts the width and height values as arguments (via the URL query string), generates values for leftFOV, rightFOV, botFOV and topFOV, substitutes them into a template then returns the resulting text with the correct content-type (application/vnd.google-earth.kml+xml) and Last-Modified / ETag value to avoid any caching.
At 30” viewing distance, the FOV horizontal values typically vary from 23.96° on a 13” display to 26.7° on a 27” display. By comparison, an 85mm focal length lens gives a hFOV of 23.9° and a 70mm focal length lens results in a hFOV of 28.8°.

The Tcl/Tk application to control Google Earth using COM

The stand-alone application requires Google Earth to be installed on the local computer. The application is deployed as a Starkit [14] and uses the Tcl Tcom package [15] to communicate with Google Earth.

The application GUI allows the user to locate the Google Earth executable and select options to build wind farm simulations, add comparison objects (such as the Mackinac Bridge [16] and

Statue of Liberty), define viewpoints ("camera" locations), and desired focal length (in 35 mm camera terms). Each of these actions causes a KML file to be generated, and that file is passed via Tcom to Google Earth.

The advantage of the stand-alone application is flexibility, in that a new KML file is generated for each combination of options. The disadvantage is that it only runs on Windows at this stage, and that it requires Google Earth to be installed on the local computer.

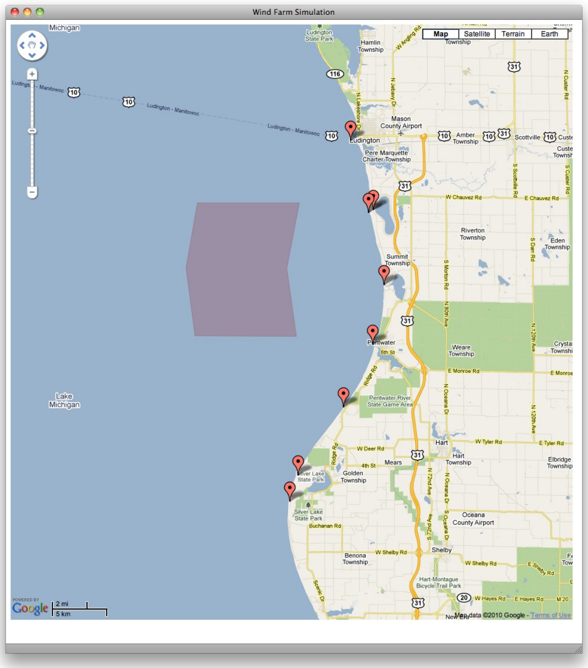
To address these issues a Google Earth plugin application was developed in parallel.

The Google Earth plugin application

The web-based browser simulation uses the Google Earth browser plugin [17] to display the wind farm simulation. The majority of the program code is Javascript running in the browser (~400 lines). This uses the Google Earth API [18] to control the plugin instance, allowing a dynamic interface to be presented to the user.

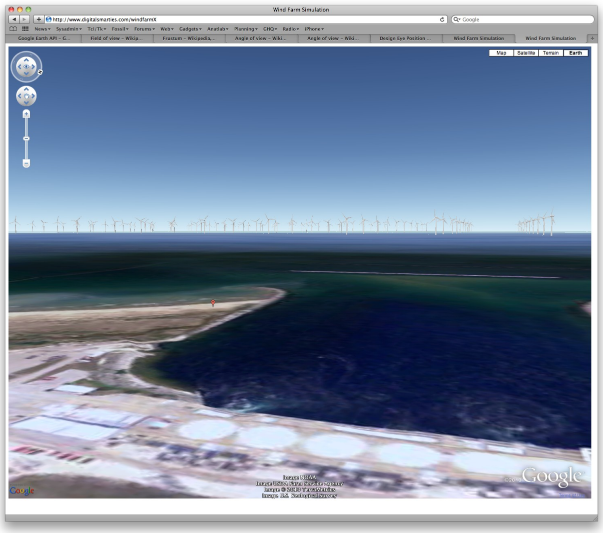
On starting, the Javascript code calculates the appropriate Field of View (FOV) values based on the size of the users screen and an arbitrary viewing distance of 30 inches.

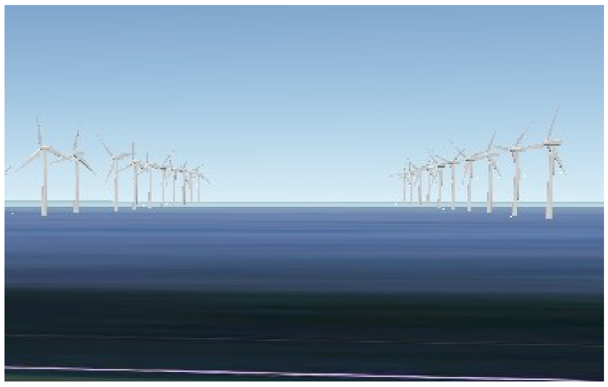
It then displays the wind farm footprint on a Google Earth map, along with placemarkers for each of the predefined viewing points. Finally, the Google Earth display is centered over the wind farm.



When the user selects a viewpoint, the display is changed to a Google Earth “Satellite 3D map” (a.k.a. aerial view) and the viewing point moved to the placemark. As described previously, the KML placemark corresponding to the viewpoint is extracted via a Tcl CGI script which substitutes the appropriate FOV values, and Google Earth “flies” the display to the viewing point.

The user can zoom or reposition the viewing point as usual using Google Earth features.

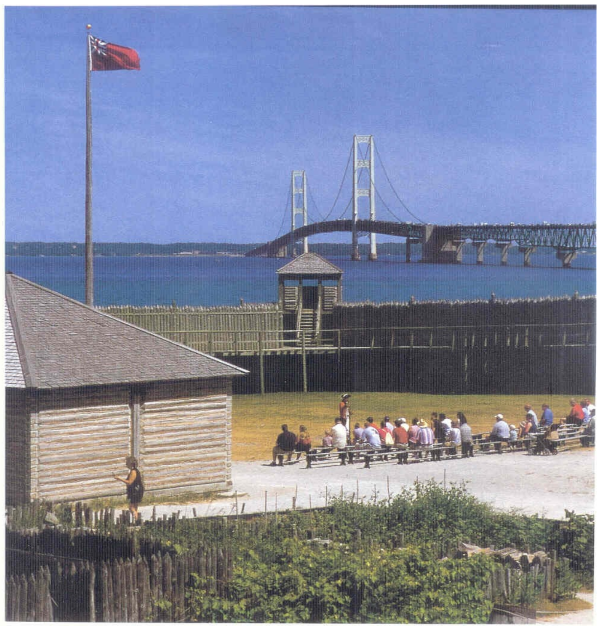




Subsequently, animation of the turbines was added. This involved two three modifications to the previously described scheme:

- multiple Sketchup models were defined, each a variant with the turbine blades at different angles
- the wind farm layout program was modified to randomly pick an initial blade position for each turbine
- the browser Javascript code was modified to update each turbine KML dynamically so that they stepped through the Sketchup models for each turbine blade position.

Validating Visuals



After incorporating Field of View support, it was observed that the turbines still didn't have the same visual impact as observed structures of a similar size.

For example, the towers on the Mackinac Bridge are 550' high, not quite as tall as the proposed turbines. From the south-west corner of the Mackinac Island the nearest tower is about 3.5 miles away, the same distance as the nearest turbines from shore. To the naked eye the towers appear to be huge, dominating the view.

The wind farm simulation was very close to how photographs of the turbines would look when taken with a normal lens (50mm in a 35mm camera). But the simulation seriously underestimated the visual impact on a human viewing the wind farm directly.

The reason for this is what is known as the Moon Illusion [19], an optical illusion in which to the naked eye the Moon appears larger near the horizon than it does when further up in the sky. This illusion also occurs with other celestial objects (e.g. the Sun looks much larger when rising or setting). It is known to be an illusion rather than an optical effect, because measurements of the angle that the full Moon subtends at an observer's eye show it remains constant as the moon rises or sinks.

The Moon Illusion has been known about since ancient times and across various cultures. No single explanation of the illusion is universally accepted, although one explanation (or is it a best guess?) arrived at after eliminating others theories is that the moon illusion is probably related to the way our brains are wired. Threats to primitive humans usually came from eye level not from overhead, therefore our brains perceiving eye level objects as bigger than overhead objects had adaptive (survival) value.

The solution used in the wind farm simulation was based on the observation that a larger focal length lens generally gives a more accurate representation of the visual perception of an object. A lens with a focal length of 85mm gives a reasonable approximation of the visual perception of the moon illusion) so the ratio of this focal length to the 50mm length was used to arrive at a scaling factor of 1.7, which results in a more typical visual perception. The scaling factor is applied within the KML for the turbines, as shown by this section of the turbine KML template:

```
<Placemark targetId="turbine$turbine">
    <name>Turbine $turbine</name>
    <Model id="turbine$turbine">
        <altitudeMode>relativeToGround</altitudeMode>
        <Location>
            <longitude>$lon</longitude>
            <latitude>$lat</latitude>
            <altitude>0</altitude>
        </Location>
        <Orientation>
            <heading>45</heading>
            <tilt>0</tilt>
            <roll>0</roll>
        </Orientation>
        <Scale>
            <x>1.7</x>
            <y>1.7</y>
            <z>1.7</z>
        </Scale>
        <Link>
            <href>windfarm.kmz/$model</href>
        </Link>
    </Model>
</Placemark>
```

This scaling factor has turned out to be a reasonable approximation for typical displays, but perhaps could be refined further by taking into account the characteristics of the current display.

An ongoing area of research is the "calibration" of the visual parameters by comparing images of the Mackinac Bridge created in Google Earth with actual photographs at varying focal lengths. At the moment the Moon Illusion correction is linear, but it is is anticipated the relationship will turn out to vary with the focal lengths.

Summary + conclusions

This project has once again demonstrated the value of Tcl/Tk as both a prototyping language, and as a glue language for combining other technologies. Further, the use of Tcl/Tk has extended to web-based technologies such as the Google Earth plugin and Google Maps.

But we knew this would be the case before we started tilting at windmills.

References

1 OpenGL - http://en.wikipedia.org/wiki/Opengl
2 Google Sketchup - http://sketchup.google.com/
3 Google Earth - http://www.google.com/earth
4 Keyhole Markup Language (KML) - http://en.wikipedia.org/wiki/Keyhole_Markup_Language
5 Digital Asset Exchange (DAE) - http://en.wikipedia.org/wiki/COLLADA
6 Clipper Windpower - http://www.clipperwind.com
7 Google Maps REST API - http://code.google.com/apis/maps/documentation/javascript/
8 The Tcl http package - http://wiki.tcl.tk/http
9 Geodesy - http://wiki.tcl.tk/geodesy
10 Field of View - http://en.wikipedia.org/wiki/Field_of_view
- http://en.wikipedia.org/wiki/Angle_of_view
11 Design Eye Position - http://en.wikipedia.org/wiki/Design_Eye_Position
12 Google Earth Photo Overlays - http://code.google.com/apis/kml/documentation/photos.html
13 Tcl CGI - http://wiki.tcl.tk/cgi
14 Starkit Deployment Technology - http://wiki.tcl.tk/starkit
15 Tcl Tcom package - http://wiki.tcl.tk/tcom
16 Mackinac Bridge - http://en.wikipedia.org/wiki/Mackinac_Bridge
17 Google Earth browser plugin - http://www.google.com/earth/explore/products/plugin.html
18 Google Earth API - http://code.google.com/apis/earth/documentation/
19 The Moon Illusion - http://en.wikipedia.org/wiki/Moon_illusion

All web references as at September 2010.