COMS W4115
Programming Languages and Translators

GECL
Google Earth Customization Language
White Paper

09/26/2006

Carlos Icaza
cai2101@columbia.edu
Darren Tang
tt2191@columbia.edu
Wei Chung Hsu
wh2138@columbia.edu

Abstract

GECL allows users access to capabilities of Google Earth that normally cost money or are too difficult to implement. These have mostly to do with creating 2D and 3D polygons on Google Earth maps, tasks that so far have been done through the rather expensive Google Earth Plus, or the tedious task of coding KML (Keyhole Markup Language – derived from the XML standard) files, which, like all XML, require the use of an extremely verbose tag-based language. Letting the compiler deal with the conversion of absolute dimensions into geographical points, the user can focus on designing the most complex polygonal structures or using predefined ones, and creating large collections of these in the Google Earth environment through algorithms. The user can also choose from a number of already specified perspectives for looking at these objects. These capabilities, together with quicker creation and manipulation of geographical place points, overlays and other Google Earth options, permit the user to produce a customized KML file for many uses in the shortest time possible – so making the powerful, and already-platform independent KML language a more useful tool for manipulating the Google Earth environment.

1. Introduction

Google Earth is a 3D interface to the geography of planet Earth, a browser that allows users to view all kinds of data that can be represented on a map, going from important census data on the income of certain counties, to trivial journal-like notes on places people have been to. All of this data, from the most basic – location of cities, borders – to the most complex – detailed elevation reliefs and geo-coded satellite photos- are stored on Google's servers, saving the user the huge trouble and expense of installing and manipulating a huge database.

The Google Earth browser is also able to render basic polygonal structures, going from simple squarish buildings to complex renderings of the skyscrapers of downtown Manhattan, and to paste pictures (any kind, from a satellite photo to a cartoon) into its maps, rendering them according to the elevation data stored on its databases. It also allows users to create place-points, which, when clicked, bring up customized data that can include any kind of URL, and also pictures.

But many of these capabilities are beyond the reach of most users, as they require either buying the more expensive commercial versions of the Google Earth browser, or taking the trouble of joining and maintaining a membership in the Google Earth Community, which keeps control over the data that is put on Google Earth's publicly available maps.

The KML (Keyhole Markup Language, based on the XML standard; Keyhole refers to the CIA's famous photo-reconnaissance satellites) gets around this limitation by letting users load their own customized geo-coded data directly onto the browser, so that it doesn't have to go through the Google Earth servers. This data can consist of place-points, customized and animated icons, overlays, and yes, polygonal structures. But KML, like all tag-based languages, is verbose, and, in its emphasis on customization, requires a lot of work on the user's part if he or she wants to use its most attractive features, like the design of polygonal structures.

GECL saves the user much of this work by generating a KML file from simple statements, computing distances in metric or Imperial measures into coordinates; saving him or her the tedious browsing of an atlas by retrieving the coordinates of relatively well known cities and towns all around the world; searching DEM (Data Elevation Model) databases in the Internet for the correct altitude of a determined point, allowing for better design of structures; and creating simple polygonal structures by inputting just the most basic details, like length, width and height. GECL will also allow the user to select predefined viewpoints from which to look at a point or object, and, to permit the creation of more complex structures and groups of them through algorithms, will implement the necessary statements for control flow (conditionals, loops).

2. Goals

Since the creation of overlays and place points for Google Earth is a rather simple task, our focus will be on the implementation of a script-like language that will allow users to navigate inside the Google Earth environment through the creation of way points, to help them locate the place on which they wish to place a determined Google Earth object - from a place point to a polygonal structure; and to produce complex instances of those structures *en masse* – as much and as far as KML and the Google Earth browser application will permit – through the algorithms that

our language will make possible. In short, our language will let the user create complex collections of structures (i.e.: towns, cities) on the Google Earth virtual world.

Ease of Use & Simplicity

GECL saves the user the tiring tasks of converting distances and building dimensions into coordinates, through the use of functions with easy to remember keywords like squareBuilding, and the use of well known operators like add and assign (=+) for modifying way points.

Portability

As the compiler will be built in Java, it will allow users in almost all existent OS platforms to use the language.

3. Possible Applications

Technical and Business Presentations

GECL will permit the quick creation of place points, paths and buildings that could help in making certain project presentations (i.e.: for civil engineering, retailing, telecommunications) more interesting and comprehensive in the data they represent.

Urban planning

Urban planners and architects could quickly create a draft of their projects before going through all the trouble and expense of putting the smallest details of their buildings into AutoCAD.

4. Sample Code

```
//Finding the coordinates of a city and making a new way point out
//of them.

coord myCity;
myCity = CityCoords(Sydney, Australia);
```

```
//Creating new way points based on the coordinates of already
//existing ones, or by a simple arithmetic based on navigation.
//Example "x + (distance(measure), direction (in compass letters or
//degrees)) to move a point.

coord newPoint;
newPoint = myCity + (50(km), NNW);
dist myDist = 60(km) + 10(m) + 10(nm) + 5(m);
coord newPoint2 = newPoint + (myDist, 270);
newPoint2 += (60(nm), S);

//Building a small block of 15 small, squarish buildings, the first
//one being a few meters off the ground ("rel" is a flag indicating
//that the structure's indicated altitude will be relative to the
//actual ground elevation at the given coordinates). The
//coordinates passed to the building function will be the center
//around which the new polygon's points will be created.
//By itself, the "new" operator is used to create many structures
//without keeping track of them.

structure smallBuilding =
               squareBuilding(newPoint2, 10(m), 20(ft), 5(m), rel);
dist total = 10;  // for each side of the block
dir newDir = E;
coord temPoint = newPoint2;
for (int i = 0; i < 15; i++)
     {
     if (total == 50){ newDir + 90; total = 10;} //Start on new side.
     temPoint += (10m, newDir); //Two halves of a building away to
                                    //avoid overlapping.
     new squareBuilding(temPoint, 10(m), 20(ft)); //At ground level
                                                  //by default
     total += 10(m);
     }
```