COMS W4115
Programming Languages and Translators

Google Earth Customization Language
Final Project Report

12/19/2006

Carlos Icaza
cai2101@columbia.edu
Darren Tang
tt2191@columbia.edu
Wei Chung Hsu
wh2138@columbia.edu
Bhashinee Garg
bg2223@columbia.edu

Table of Contents

Chapter 1

Introduction

As a part of this project, we have designed and implemented a language called Google Earth Customization Language (GECL). It is meant for customizing information displayed on the Google Earth maps in a cost-free and user-friendly way.

1.1 Background

Google Earth is a 3D interface to the geography of the 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. All of this data, from the most basic – location of cities, borders – to the most complex – detailed elevation relief 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 square-shaped 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. So even joining the community does not give the users complete freedom in terms of the customization they are interested in.

1.2 Motivation

The KML (Keyhole Markup Language, based on the XML standard; Keyhole refers to the CIA's famous photo-reconnaissance satellites), used for managing three-dimensional geospatial data in Google Earth, 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. Although coding in KML seems to be a viable option, KML is verbose, like all tag-based languages, 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. Some examples of this extensive work are: determining the exact latitude, longitude, elevation of a place of their interest, etc. This becomes a major hurdle in no time.

Given these circumstances, we propose our language, GECL, a small effort towards making the job of customization easier. The output of a GECL program is a KML file that does the required customization on a Google Earth map.

1.3 Goals

GECL saves the user much of this work by generating a KML file from simple statements. More specifically, it was designed to have the following capabilities:

- computing distances in metric or Imperial measures into coordinates; saving the user from the tedious browsing of an atlas by retrieving the coordinates of relatively well known cities and towns all around the world;
- creating simple polygonal structures by giving as input just the most basic details, like length, width and height;
- allowing 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, which implement the necessary statements for control flow (conditionals, loops);

- quicker creation and manipulation of geographical place points, overlays and other Google Earth options;
- being platform independent, like KML itself;
- being efficient in producing a KML file as the output

1.4 Basic Features

1.4.1 Data Types

Apart from the basic data types, *int*, *float*, and *string*, we have several other data types: *dist*, *dir*, *coord*, *point*, *line*, *overlay*, *persp*, *place*, and *struct*. These data types are required to generate statements that ultimately form the KML file and get interpreted by the Google Earth maps.

1.4.2 Basic Operations

The basic operations that GECL does are: specifying the coordinates of a point using *coord*, draw a line between a number of specified points, draw an overlay on a few given coordinates, specify a particular perspective of viewing, show a place mark, and make a structure like a square, rectangular, and triangular building. These things can be done just with the statements containing the above data types.

1.4.3 Program Control Flow

Using control flow statements like while and do…while, a group of structures (square, rectangular, and triangular buildings) can be built at a given place.

1.4.4 Looking-up coordinates based on City/State/Country

Using a 10,000 city-name database, GECL has the ability to look for the exact coordinates (latitude, longitude, and elevation) of a city. All the user needs to do is to specify the name of the city and the state (or the country) it is located in. The compiler finds the coordinates, inserts them in the program, does the other calculations and returns the results in a way similar to as if the coordinates were specified directly in the program.

1.4.5 Coordinate Navigation Arithmetic

We have a special type of math that enables finding and fixing a determined geographical point and create a *coord* data type from this and other user defined data types.

coord <object_name> = *coord* <point> + (*dist*<distance>, *dir*<direction>);

This way, <object_name> would be a point defined by moving `distance feet` (all distances are converted to `feet` once processed by the compiler) in the direction of `direction` degrees by the compass, starting from `point`.

1.5 Possible Applications

Technical and Business Presentations

GECL can permit the quick creation of place points, paths and buildings that could help in making certain project presentations (i.e., civil engineering, retailing, tele-communications, etc) 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.

Chapter 2

Tutorial

This chapter represents a tutorial for a novice to get started with GECL and produce KML files. However, it already assumes that the reader is familiar with the basics of the geographical aspects of the globe, like latitudes, longitudes, elevation, perspective, etc. For more information on these, please see the references in the Appendix.

2.1 Getting Started

Every GECL file has the extension *.gecl*. Before executing a GECL file for the first time on a machine with Windows operating system, do the following tasks:

1. Set the environmental variable `PATH` to the folder containing the Java executables on the system. For example: `.;C:\Program Files\Java\jdk1.5.0_09\bin`
2. Set the system variable `CLASSPATH` to the again folder containing the Java executables and to the folder which contains all the project files. For example:
   ```
   .;C:\Program Files\Java\jdk1.5.0_09\bin;C:\Documents and
   Settings\bhasha\Desktop\tree_walker_v16
   ```
3. Include the ANTLR executable in the folder containing all the project files. This can give you relief from setting any more variables to the relevant folders. Else, add the path of the ANTLR executable/jar files to the `CLASSPATH` as well.
4. Run the Windows command prompt and change the current directory to the one containing the project files.
5. For a GECL file, say `trial.gecl`, enter the following commands in the command prompt:

   <name of ANTLR exe> `GECLgrammar.g`

   The following message tells that ANTLR has successfully completed its job and generated the required Java files:

```
ANTLR Parser Generator   Version 2.7.5 (20050201)
1989-2005 jGuru.com
```

6. To compile the java files, use: `javac *.java`

   If the command prompt returns, the files have been compiled successfully.
7. To run a GECL file, say, trial.gecl, use the following command:

   ```
   java GECLwalker trial.gecl
   ```

   This produces the `test.kml` file.
8. Copy `test.kml` to the folder containing Google Earth. Double-clicking the application picks up the *.kml* file from the folder directly and renders the data on the Google Earth browser.

2.2 A Simple Program

A GECL program has basically two parts: generic control flow statements (like if…else, do…while, and while) and specific GECL statements used for customizing the Google Earth maps. To see a complete list of all specific GECL statements, please refer to the Language Reference Manual (LRM) in the next chapter.

```
coord myCity;             // declare a variable of type coord
// specify the name of the city for coordinate look-up
AsigCity myCity IS ("New York City", "New York");
point myCtPoint IS myCity;             // make  point at the specified city
// put a placemark on the city, with some custom information
place myCtPlace IS myCtPoint, "Testing New York";
printk myCtPlace;     // print the city to KML file
```

This very simple example program illustrates the basic layout and programming style of GECL. All this program does is: it takes the name of a city, New York in this case, and makes a place mark at this city with some custom information (a text string here). The `AsignCity` statement is used to specify the name of a city to an object of type *coord*. The last line, `printk`, is very important. It instructs the GECL compiler to write this and the other relevant details to the output KML file. Unless the compiler sees this statement, it treats all the variables and other information as internal, i.e., to be used for calculations and book-keeping only and writes nothing to the KML file. On seeing this statement, it writes the identifier followed by `printk` and other relevant objects to the output file, making it understandable by the Google Earth browser.

The output of the above program is the following KML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">

<Folder>
<description>Blah blah</description>
<name>PLT_kml_test</name>
<open>0</open>
<Placemark>
<name> Testing New York </name>
     <Point>
          <coordinates>121.58,24.99,0.0</coordinates>
     </Point>
</Placemark>
</Folder>
</kml>
```

The `name` tag inside the `Placemark` tag holds the text data specified by the programmer and the `coordinates` tag inside the `Point` tag specifies the coordinates at which the place mark is placed. The coordinates have been found by looking-up Taipei in the city-name database.

2.3 A More Complex Program

In this example, we lay an image as an overlay on four coordinates specified by *coord*.

```
coord ovlyCoord1 IS 14.601, 37.919, 0ft;
coord ovlyCoord2 IS 15.358, 37.919, 0ft;
coord ovlyCoord3 IS 15.358, 37.465, 0ft;
coord ovlyCoord4 IS 14.601, 37.465, 0ft;
overlay myOverlay IS ovlyCoord1 ovlyCoord2 ovlyCoord3
ovlyCoord4 , "http://www.airplane-
world.addr.com/mpnasamain.jpg" , 0.0 ;
printk myOverlay;
```

The image/icon to be overlaid on the specified area (defined by *coord* statements) comes from the URL, `http://www.airplane-world.addr.com/mpnasamain.jpg`, in the `overlay` statement. The KML output of the program is:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">

<Folder>
<description>Blah blah</description>
<name>PLT_kml_test</name>
<open>0</open>
<GroundOverlay>
<name> myOverlay </name>
<visibility>1</visibility>
<Icon>
     <href>http://www.airplane-
world.addr.com/mpnasamain.jpg</href>
</Icon>
```

```
<LatLonBox>
     <north>37.919</north>
     <south>37.465</south>
     <east>15.358</east>
     <west>14.601</west>
</LatLonBox>
</GroundOverlay>
</Folder>
</kml>
```

2.4 Yet Another Example: Creating a group of Buildings

In this example, we create a group of rectangular buildings, all of them making a square.

```
int i = 1;
coord tmp_coord IS -15.0, -30.0, 0ft;
dist tmp_dist IS 200m;
int d = 90;
float f1 = 121.32856;
while(i <= 8) {
    tmp_coord IS tmp_coord + (tmp_dist, d);
     //when inside while loop, if-else need { } for inside scope
    if( i == 2 || i == 4 || i == 6 || i == 8) {
        d += 90;
}
    struct tmp_build IS square 20.0, 30.0, tmp_coord;
    printk tmp_build;
    i += 1;
}
```

The KML output of this program is a big list of tags, each making a building. The screen shots of the outputs of these programs can be seen in Appendix S.

Chapter 3

Language Reference Manual

1. Language Conventions

1.1. Comments

GECL allows single kind of comment, which starts from “//” and continues till the end of the line.

1.2. Identifiers (Variable, Function, and Object names)

Identifiers are used to name variables, functions and objects. The identifier is a terminal in our grammar. The following is our identifier rule:

letter: _|A|B|...|Z|a|...|z

digit: 0|1|...|9

identifier: letter (letter | digit)*

1.3. Numbers

A number can be an integer or floating point. It consists of digits and optional decimal point “.” as well as digits. In order to make our language simple, we don’t support exponent number now.

number: (digit)+ (. (digit)+)?

1.4. Strings

A string is a sequence of characters enclosed by double quotes. Basically, the character here means the 256 types of character in ASCII code.

String: “ (character)* “

1.5. Operators

=	%	--	%=	<=
+	||	+=	==	.
-	&&	-=	>	
*	!	*=	<	
/	++	/=	>=	

1.6. Keywords

dir	dist	Coord	point	line	Overlay	struct	persp
place	AsigCity	IS	if	else	While	do	break
IS	km	Ft	nm	m	Continue	square	rectangle
mi	N	S	E	W	Readfile	int	float
NE	NW	SE	SW	printk			

1.7. Tokens to terminate or separate blocks

{	}	(	)	,
[	]	;		

2. Types/Objects

In GECL, we use type followed by object name to allocate an object, and we process this object by this object name. There are several type keywords GECL supports list in chapter 1.7. These objects can be allocated like a standard type (i.e.: object myObject;), but require the IS keyword to be defined.

2.1. dist

Defines a length, which can be given in miles, meters, kilometers, feet, and nautical miles. Instantiated and manipulated like an int.

2.2. coord

The basic geographic object from which all others are built. Defines latitude, longitude an elevation of a specific geographical point.

2.3. point

Creates a basic ‘push-pin’ point at a certain coord, with title, comments and URL for links or for loading photos.

2.4. struct

This is the most important type in GECL used to construct a 3D object which could be a building, a car or a polygon.

2.5. place

This is the type to allocate a placemark object. The placemark consists of several struct objects, overlays and so on, grouped so as to be related to perspective objects

2.6 overlay

This is the type to allocate an overlay object so that we could specify the area we are going to look at with coordinates, photo URL and rotation.

2.7. persp

This type specifies where and how the viewer will look at a struct, coord, overlay, line and place object.

2.8. line

This is the type to allocate a line object. The reason to have this object is because we want to have some line mark in the Google Earth instead of using pure 3D objects.

3. Expressions

Our GECL is a language that consists of several expressions. The expression could be a list of integers, a boolean number, an identifier or a function call. The following is our expression grammar.

```
expr : int-lit
| identifier
| identifier = expr
| var post-op
| expr bin-op expr
| if expr expr else expr
| ( expr )

int-lit : ( digit )+
bin-op : arith-op | rel-op | bool-op
arith-op : + | - | * | / | %
rel-op : == | < | >
bool-op : ^ | |
post-op : ++ | -- | ~~
```

4. Statements

Semicolon “;” is a statement terminator in GECL. Basically, statements are executed in sequence except specified control-flow statements. Statements can separate into several categories listed below.

statement:

expression-statement

compound-statement

iteration-statement

conditional-statement

4.1 Expression statement

4.1.1 Assign statement

assign-statement:

primary-expression assignment-operator expression ;

Asssign-statements are the majority of expression statements. Basically, these statements assign the right side expression's final value to left side.

For example:

foo = bar + 10 ;

4.2 Compound statement

compound-statement:

{declarations statements}

A compound statement is composed of declarations and statements by put them into a pair of brackets. The body of a function definition is a compound statement.

4.3 Iteration statement

iteration-statement:

***do** {*

*statement**;***

…

*} **while** (expression);*

***while** (expression) {*

*statement**;***

***…**}*

Iteration statements use to loop a statement or a block of statements several times. In this for-loop statement, three expressions seperated by semicolon are all optional. The first expression which must have arithmetic type is evaluated once, and thus gives the initial state for the loop. The second expression is evaluated before each iteration, and if it becomes eqaul to zero (false), the iteration is terminate. The third expression is evalated after each iteration and thus gives a new state for the loop.

4.3.1 Break statement

break-statement:

***break**;*

Break statement only happens inside an iteration statement. It terminates inner-most iteration statement and gives the control to the next statement that following the interation statement.

4.3.2 Continue statement

continue-statement:

***continue**;*

Continue statement only happens inside an iteration statement. It causes control to pass through current loop (the smallest enclosing) and directly go to next loop's evaluation.

4.4 Conditional statement

conditional-statement:

***if** (expression) statement;*

***if** (expression) statement; **else** statement;*

The expression part of the conditional statement must have arithmetic type. If its value compares unequal to zero then the first statement is excuted. In the second form of conditional statement, the second statement is executed if the expression's value is zero.

5. Built-in functions

These are almost entirely the ones required to define the types, the equivalent of *constructor methods* in Java. The declaration and definition statement patterns shown include those for declaration alone, declaration of arrays, and assignment of objects to the array.

dist

dist <object_name> IS x[m | km | mi | ft | nm];

dist <object_name;

Defines a new dist object. The distance needs to be written like a float or int together with a modifier specifying in which measure the distance is being given: meters, kilometers, miles, feet or nautical miles.

Addition and multiplication can be performed between two dists objects. Example:

dist myNewDist IS 10m + myOldDist;

coord

coord <object_name> IS *float* <longitude>, *float* <latitude>, *dist* <elev>;

coord <object_name>;

Defines a coord object with longitude and latitude given in the GE standard floating point format. The elevation is given as a dist type. coord objects can be manipulated using an arithmetic based on navigation (see below).

point

point <object_name> IS coord <cord>;

Defines a point object, which appears in Google Earth as a pop-up square with title, coordinates and elevation, comments, and a URL that can connect to a photo or a website itself.

struct

struct <object_name> IS string <shape> .... (arguments vary according to shape given)

Defines a struct object using several in-built shapes. The arguments required by each shape are:

square float <length>, float <height>, coord <cord>;

rectangle dist <length>, dist <width>, dist <height>, coord <cord>;

overlay

overlay <object_name> IS coord <cord1> ,.... coord <cord4>, string <URL>, float <rotation>;

overlay <object_name>;

Defines an overlay object, a graphic file that is laid over the map according to the 4 coordinates and rotation passed to the function. The URL points to the graphic file.

line

line <object_name> IS coord <cord>.... cord <cord_n>;

Defines a line object whose points are given either by a list of coord objects, or by an array of such.

place

place <object_name> IS point <object>, string <name>, persp <perspective>;

place <object_name>;

Defines a place, or collection of point, struct, line or overlay objects that can be associated with a persp object, so as to look at the collection from the point of view defined by it. It can also be added/associated with a folder object, so as to group it together in the Google Earth navigation bar. A new instance of these objects can be added to an already existing place by using the 'add and assign' (+=) operator:

place <object_name> += point | struct | line | overlay <new_object>;

persp

persp <object_name> IS coord <camera_loc>, dist <range>, dir <tilt>, dir <heading>;

Defines a persp object, which can be related to place, line, overlay, struct or point objects, by sharing the same coord object. Once this is done, when clicking on them on the GoogleEarth navigation bar the GE browser look at these objects from the perspective defined by the persp object.

AsigCity

AsigCity <object_name> IS (“City”, “Country);

Returns the coordinates for a city, inserting them into <object_name>.

readfile

readfile ID scale translationX translationY translationZ Filename;

Returns the coordinates for a city, inserting them into <object_name>.

6. A special arithmetic for manipulating coord objects

To ease the finding and fixing of a determined geographical point, and to create a coord object from it, a special arithmetic allows the user to 'move' from one coord to a new point and create a define a new coord object out of it, using the following format:

coord <object_name> = coord <point> + (dist <x>, dir <y>);

In this way, <object_name> would be a point defined by moving x meters (all distances are converted to meters once processed by the compiler) in the direction of y degrees by the compass, starting from <point>.

Chapter 4

Project Plan

4.1 Processes Used

We had regular group meetings to discuss the project scenario, divide the next set of tasks amongst the team members, and set goals for every member. The initial weeks saw more frequent and longer meetings as the scope and the form of the language were to be laid. Later meetings were shorter and less frequent. However, at times, it was not possible for a member to attend a meeting. To keep every member updated, the major communication tools used were CubMail/HotMail/GMail for emails and MSN messenger for online communication.

Removing inter-person dependencies:

Keeping in mind that every individual has his/her own pace of working, we decided to remove inter-person dependencies to the maximum possible extent in the following way:

- We separated the whole project into two parts: language-specific part and generic control-flow part.
- Divided the work amongst the team members so that the two parts got developed in parallel. Each member was responsible for testing his/her part appropriately and sufficiently.
- In each phase, like front-end, back-end, etc, the parts were integrated in a timely manner to ensure consistency and steady progress.

For example, once the lexer was ready, the parser was divided so that the language-specific grammar could be written by one team member and the control-flow grammar by another. As soon as either one was ready, a team member started working towards the tree-walker for the ready part. Once the complete parser was ready, it was integrated into one grammar file and given to the members doing the tree-walker. In a long run, this approach worked very well.

4.2 Programming Style Guide

Each individual has his/her own coding style. Keeping this in mind, we did have a strict guide about typifying the code. But we did have the following rules set for ourselves:

- All the ANTLR grammar (lexer, parser, tree-walker, and static-semantic checker) should be in the same *.g* file.
- There should be clear documentation in each part of the code so that team members apart from the programmer understand it without any difficulty.
- Each new version should clearly accompany the changes/enhancements made from the previous version. This would later provide a good log for the project.
- Any major changes in design, etc ANTLR rules, should be first discussed with all the team members before making any change in the code.
- Names of files/classes/methods/variables should be consistent and meaningful throughout all programs. We tried to follow this rule to the maximum possible extent, although, there may be exceptions in the code.
- Each Java class should contain code and functions related to a particular functionality only. This helps making a robust and easy-to-change design.

4.3 Project Timeline

Here is the timeline for each specific task of the project.

Task	**Due Date**
White Paper	9/26/2006
Language Reference Manual	10/19/2006
Lexer and Parser	11/10/2006
Tree Walker and Code Generation	12/10/2006
Static-Semantic Checking	12/13/2006
Comprehensive Testing	12/17/2006
Freeze Code and Present	12/19/2006

4.4 Individual responsibilities

Project Phase	**Part of Phase**	**Individual**
LRM	-	Carlos Icaza-Estrada (Team Leader)
Lexer	-	Tz-Yang Tang
Parser	Control-flow part	WeiChung Hsu
	Language-specific part	Bhashinee Garg
	Combining	WeiChung Hsu
Tree Walker	Control-flow part	WeiChung Hsu
	Language-specific part	Tz-Yang Tang Carlos Icaza Estrada
	Combining	WeiChung Hsu
	Static-semantic analysis	Bhashinee Garg
Documentation and Testing	Documentation	Bhashinee Garg
	Chapter 3 of report (LRM)	Carlos Icaza-Estrada

Although the above table has been brought out very specifically, every team member was familiar with every phase of the project. Also, each member was responsible for testing his/her own part thoroughly. The final testing was done by WeiChung Hsu and Bhashinee Garg.

4.5 Tools and Languages

All the phases were carried out by the respective team-member on his/her laptop or CLIC lab machine. Since both GECL and KML are platform-independent, each member was free to use either Windows or Linux. No problems were encountered due to programs written in different platforms. Also, almost every team member worked from home, when Internet was not necessarily available. So, no specific version control system, like CVS repository on CLIC lab machines, was used. We used separate folders to maintain separate versions and a log of all subsequent changes from one version to another was carefully maintained. The same has been provided in the next section.

The front-end (lexer, parser, and tree-walker) were implemented using the ANTLR tool with Java 1.5. The back-end was done in JDK 1.5 solely. The output version of KML is 2.0. All of us had the latest version of Google Earth installed on our laptops, so each member went ahead with the testing of his/her part.

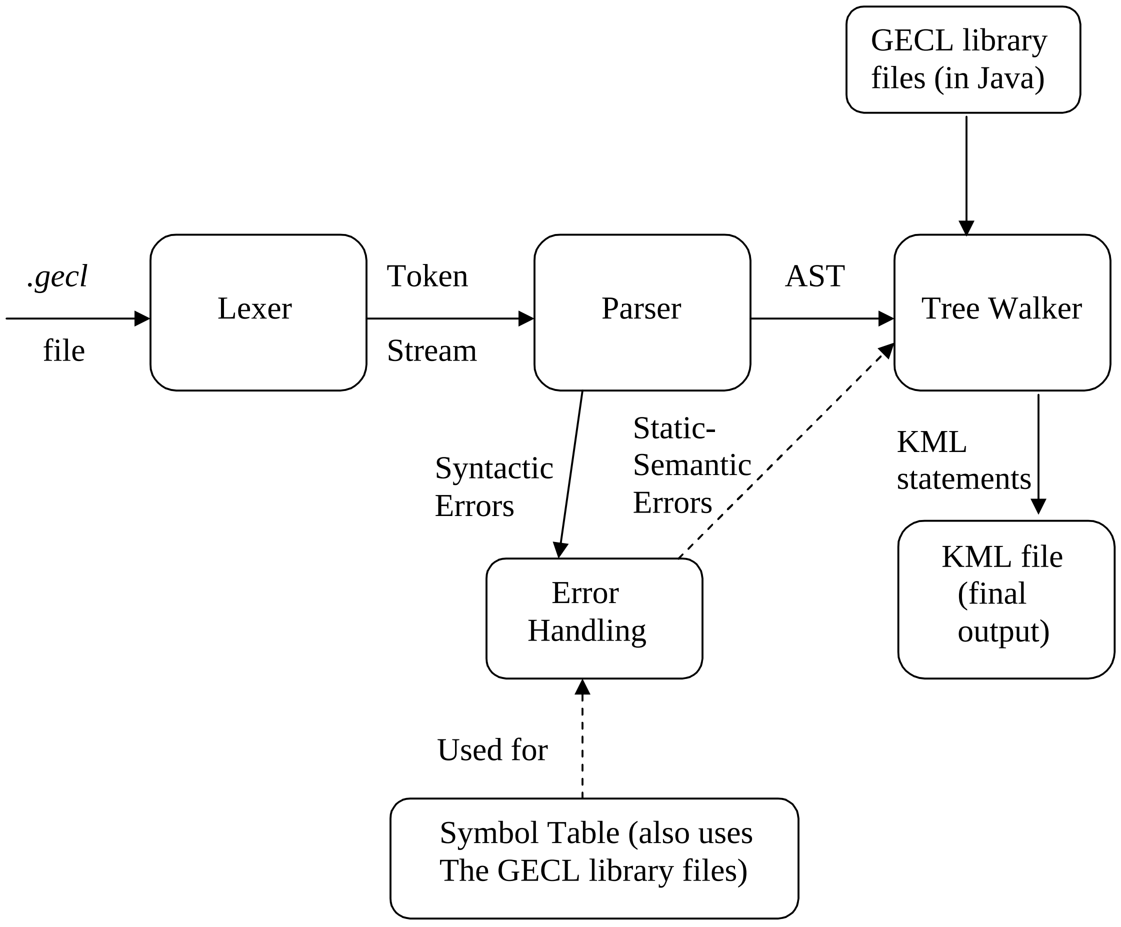
4.6 Project Log

Task	**Due Date**
White Paper initial draft	9/20/2006
White Paper final	9/26/2006
Language Reference Manual initial draft	10/10/2006
Language Reference Manual final draft	10/19/2006
Lexer	10/25/2006
Parser for GECL specific statements	11/01/2006
Parser for control flow statements	11/05/2006
Complete combined parser	11/10/2006
Tree Walker and Code Generation (v1) *struct*, *line*, *overlay*, *persp*	11/25/2006
Tree Walker and Code Generation (v2) city-name look-up function	11/28/2006
Tree Walker and Code Generation (v3) complete along with *coord*-navigation	12/10/2006
Static-Semantic Checking	12/13/2006
Comprehensive test-suites written	12/14/2006
Comprehensive Testing	12/17/2006
Freeze Code and Present	12/19/2006

Chapter 5

Architectural Design

5.1 Block Diagram of Translator



The GECL Translator – Flow of information from one component to next.

5.2 Description of the Architecture

GECL has mainly four components: lexer, parser, tree walker, and several library files. The first three are in the GECLgrammar.g. The error handling code is embedded in the tree walker part of this file. The flow of information between components is clear from the block-diagram above. Below given is the complete listing of all the GECL Translator files and the classes and methods each one contains.

GECLgrammar.g	Lexer, Parser, AST Walker, Static-Semantic Analyser	Tz-Yang Tang WeiChung Hsu Bhashinee Garg
Distance.java	Takes a string of type DIST, e.g., 10km, extracts the value and unit, processes them, returns a DIST again.	Carlos Icaza-Estrada
GECLdatatypes.java	Get and set methods for all GECL datatypes. Acts like a symbol table along with data storage. Drawing rectangle and square, and coordinate navigation logic.	Tz-Yang Tang Carlos Icaza-Estrada
GECLlibrary.java	Writes the output to a KML file upon `printk`. Coordinate look-up in city name database and error reporting.	Tz-Yang Tang Bhashinee Garg
PerspectType.java	Handling the perspective of viewing	Tz-Yang Tang
Ply_3D.java	Java library for rendering the PLY and Dragon on Google Earth	WeiChung Hsu
GECLrunner.java	Java file to launch GECLgrammar.g	-

5.3 GECL Symbol Tables

The symbol table of GECL is a rather complex data structure and deserves special mention. Given the large number of GECL datatypes, their hierarchical arrangement, and each one being composed of large number of primary datatypes (like *coord* consists of longitude, latitude, distance, etc), the symbol table has to do a lot of book-keeping task. The logic employed to effectively implement the symbol table is as follows: The standard java.util.Hashtable() is used to store the symbols as key-value pairs. The symbol name is the key and the value is the count of the symbol. Internally, a counter is maintained and incremented by one each time a new symbol is seen. This counter value acts as the value of the symbol (key) in the hash table. This counter is also the index of the symbol's record in the GECLdatatypes array. This array maintains all possible information about a particular symbol, for example, its datatype, its value (or component values), etc. To get a particular symbol's data, we first try to find it in the hash table based on key search and retrieve its corresponding value. The retrieved value is the index of the record of the symbol in the GECLdatatypes array.

5.4 Static-Semantic Analysis

In GECL, static-semantic analysis is very much in place. For all hierarchically connected datatypes, the AST walker walks the tree only as long as it sees previously declared variables of correct datatypes. Else, it reports an error and exits the program. For example: *place* expects an identifier of type *point*, which is in-turn a variable of type *coord*. So, the error handling routine checks for the following things in the order of mention:

1. Does the identifier exist in the symbol table? If no, exit.
2. Is it a variable of type *point*? If no, exit.
3. The moment point is seen, it checks whether it is a variable of type *coord*. If no, exit.
4. Accept and continue walking the tree.

In this way, the whole AST is walked and the output KML file is generated only if there is no static-semantic error in the GECL program.

Chapter 6

Test Plan

6.1 Some test suites

This section presents some test suites used for testing the GECL translator.

Test suite #1: (for testing GECL specific statements)

```
//------ Test Dist, Coord, Point, Place(place mark)--------
dist tsu IS 30km;
dist gg;
coord myCoord IS 28.0 , -48.099 , 0ft;
coord myCoordI IS 28 , -48.099 , 0ft;      // should give error
coord myC;
point myPoint IS myC1;   // should give error
place myPlace IS myPoint,"darren_test_title";
gg IS gg + 10km;
myC IS 28.0 , -48.099 , 0ft;
//--------------- Test Struct ------------------------------
coord myCoord1 IS 24.0 , 24.1 , 0ft;
coord myCoord2 IS 24.0005 , 24.1 , 0ft;
//should give error
struct mybuild2 IS reactangle 200.0, 50.0 , 100.0 , myCoord2;
//--------------- Test Overlay ------------------------------
coord ovlyCoord1 IS 14.601, 37.919, 0ft;
coord ovlyCoord2 IS 15.358, 37.919, 0ft;
coord ovlyCoord3 IS 15.358, 37.465, 0ft;
coord ovlyCoord4 IS 14.601, 37.465, 0ft;
overlay  myOverlay  IS  ovlyCoord1  ovlyCoord2  ovlyCoord3  ovlyCoord4  ,
"http://www.airplane-world.addr.com/mpnasamain.jpg" , 0.0 ;
//--------------- Test Perspective ------------------------------
coord myPpCoord IS 24.0 , 47.0 , 0ft;
```

```
point myPpPoint IS myPpCoord;
persp myPersp IS myPpCoord, 50ft, 10.0,  10.0;
place myPpPlace IS myPpPoint ,"darren_test_title_2" ,myPersp;
//--------------- Test Line -----------------------------
coord myLnCoord1 IS 24.30 , 47.30 , 0ft;
coord myLnCoord2 IS 24.60 , 46.70 , 0ft;
point myLnPoint1 IS myLnCoord1;
point myLnPoint2 IS myLnCoord2;
place myLnPlace1 IS myLnPoint1 ,"darren_test_title_3" ;
place myLnPlace2 IS myLnPoint22 ,"darren_test_title_4" ;       // should give error
line myLine IS myCoord myPpCoord myLnCoord1 myLnCoord2;
//--------------- Test Find City Coordinate ----------------------
coord mycity1 ;
AsigCity mycity1 IS ( "New York City", "New York" );
point myCtPoint1 IS mycity1;
place myCtPlace1 IS myCtPoint1 ,"darren_test_City";
coord mycity2;
AsigCity mycity2 IS ( "Taipei", "Taiwan" );
point myCtPoint2 IS mycity2;
place myCtPlace2 IS myCtPoint2 ,"darren_test_City2";
// Test Print out
printk myPlace;
printk mybuild1;
printk mybuild2;
printk myOverlay;
printk myPpPlace;
printk myLine;
printk myLnPlace1;
printk myLnPlace2;
printk myCtPlace1;
printk myCtPlace2;
```

Reasons for selecting this test suite: This test suite consists of all GECL specific statements. Some of the statements have parse errors and some have static-semantic errors. To detect both these kinds of errors, and to make sure that the static-semantic checker is reliable enough, this test-suite was a great help.

Test suite #2: (Control-flow statements)

```
if ( abc == 123 )
      abc = 456 ;
if ( abc != 123 )
      abc = 456 ;
if ( abc && 123 )
      abc = 456 ;
else
      abc = 789 ;
while ( kkk || 1 )
      def = 15 ;
do
      ddd = 123;
while ( bbb == 89 ) ;
```

Reasons for selecting this test suite: This test suite consists of some of the common possibilities in which control-flow statements and logical operators can occur together. Passing this test suite is a good sign, indicating that the generic language statements have the right grammar.

Test suite #3: (Complete combined GECL program)

```
// makes a group buildings in a square shape.
int i = 1;
coord tmp_coord IS -15.0, -30.0, 0ft;
dist tmp_dist IS 200m;
int d = 90;
float f1 = 121.32856;
while(i <= 8)     {
    tmp_coord IS tmp_coord + (tmp_dist, d);
    //when inside while loop, if-else need { } for inside scope
    if( i == 2 || i == 4 || i == 6 || i == 8)   {
        d += 90;  }
    struct tmp_build IS square 20.0, 30.0, tmp_coord;
    printk tmp_build;
    i += 1;}
readfile test_ply 100.0 f1 25.01259 100.0 "bun_zipper_res4.ply";
```

```
printk test_ply;
f1 += 0.01;
readfile test_ply2 100.0 f1 25.01259 200.0 "dragon_vrip_res4.ply";
printk test_ply2;
```

Reasons for selecting this test suite: This is a complete GECL program which combines the language specific statements as well as the control flow statements. It also makes good use of logical operators and nested scope like *if…else* within *while*. All these features of this program make it a good candidate for a test suite. The translator accepted this program correctly and generated a valid KML file as well.

The methodology used for static-semantic checking is mentioned in § 5.4 of Chapter 5, Architectural Design.

Chapter 7

Lessons Learnt

Carlos Icaza-Estrada

In this project I gathered much experience on team work, deepened my knowledge of the internal workings of a compiler and also of Google Earth's capabilities and limitations. Above all, I learned the great importance of disambiguating things when designing, coding and testing relatively large programs and of being precise in setting down function interfaces.

Tz-Yang Tang

While working on this project I went through all the procedure of front-end compiler. It was fun to know that how languages we use, like C and Java, work. When I used to write a *while* loop earlier, I felt it was *just* a *while* loop. But now, I know it much more deeply. It's really good to combine the theory we learned in the PLT class and the practical experience of building our own compiler. Also, when I built the tree walker, I got a chance to recall Java programming. My programming skills also improved after I learnt the various language concepts (scope, memory allocation, etc) in the PLT class.

From another point of view, due to this project, we also learned how to work in a team. We needed to separate work, communicate and negotiate during the whole project process. We also learned how to finish a project from design and specification to building and testing. It's good news that we finally finish it.

Bhashinee Garg

The very first thing that this project taught me is that one should start working early on projects like this. The second is that before starting to code directly, a robust framework should be developed. Hap-hazard coding style has many obvious sacrifices at later stages. Apart from these two, on the technical side, this project provided with an altogether new insight into the compiler world. While writing the parser for the GECL specific statements, I learnt a lot about the internal workings of a compiler, as well as of Google Earth and KML. While doing the static-semantic checking, I actually understood how terse it is to catch simple errors in a program. On a third front, I learnt the true importance of communicating effectively and being tolerant towards team members' variable paces of work.

WeiChung Hsu

Working on huge team project is not easy because we must have good plans, schedules, and communication. In addition, it is a huge challenge in how to split the job tasks efficiently so that everyone could have fair work loading and wouldn't do redundant task. We do learn how to teamwork from this project. The key point to make us success in teamwork is "library" and interface. According to the library, we could separate our job into several libraries without affecting each other. We could even test our program by calling our own library without others codes. In addition, both Google Earth and Compiler Design are interesting topics that I do learn a lot from these two areas.

Appendix C

Source Code

GECLgrammar.g

```
/*  Authors:    WeiChung Hsu
             Tz-Yang Tang
             Bhashinee Garg
*/
{
   import java.io.*;
   import java.util.*;
}
//===================LEXER======================================
class GECLLexer extends Lexer;
options
{
   k = 2;
   testLiterals = false;          // do not check matched tokens for (Parser) literals
   charVocabulary = '\3'..'\377'; // Using ASCII chars
            testLiterals = false;
}
// Identifier
ID  options {        testLiterals = true;    }
   :   ( 'a'..'z' | 'A'..'Z' | '_' )
       ( 'a'..'z' | 'A'..'Z' | '_' | '0'..'9' )*
   ;
// Comments
COMMENT
: "//" (~'\n')* '\n'
{ newline(); $setType(Token.SKIP); }
;
// String
// String constants must be contained on a single line and
// may contain double quotes, e.g.,
// "This is a constant with ""double quotes"""
StringConstant
: '"'!
( ~('"' | '\n')
| ('"'! '"')
)* '"'!
;
// White space
WS
: ( ' '
| '\t'
| '\f'
// handle newlines
| ( "\r\n" // Evil DOS
| '\r' // Macintosh
| '\n' // Unix (the right way)
)
{ newline(); }
)
{ $setType(Token.SKIP); }
;
// Numbers
// protected ==> protected lexer rules can be used by other lexer rules, but do not return tokens by themselves
protected DIGITS : ('0'..'9')+;
INTEGER
    : DIGITS ('.' DIGITS {$setType(FLOAT);})? (UNIT {$setType(DIST);} )?;
// Symbols
ASSIGN                                          : '='          ;
MOD             : '%'               ;
DEC             : "--" ;
MOD_ASSIGN      : "%="               ;
LTE             : "<=" ;
PLUS            : '+' ;
LOR             : "||" ;
PLUS_ASSIGN     : "+=" ;
EQUAL           : "==" ;
NOT_EQUAL       : "!=";
PERIOD          : '.'               ;
MINUS           : '-' ;
LAND            : "&&" ;
MINUS_ASSIGN    : "-=" ;
GT              : ">" ;
STAR            : '*' ;
LNOT            : '!' ;
STAR_ASSIGN     : "*=" ;
LT              : "<" ;
DIV             : '/' ;
INC             : "++" ;
DIV_ASSIGN      : "/=" ;
GTE             : ">=" ;
// Other symbols
SEMI: ';' ;
LPAREN : '(' ;
RPAREN : ')' ;
LCURLY : '{' ;
RCURLY : '}' ;
LBRACK : '[' ;
RBRACK : ']' ;
COMMA : ',' ;
protected
UNIT
    :   "km"
    |   "ft"
    |       "nm"
    |       'm'
    |       "mi"
    ;
//=================PARSER======================
class GECLParser extends Parser;
options
   {
      k = 2;
      buildAST = true;
   }
tokens
   {
      SHAPE;
      NStatementExpr;
                   NEmptyExpression;
                   NStringSeq;
                   NExpressionGroup;
                   NFunctionCallArgs;
                   NPostfixExpr;
                   NUnaryExpr;
                   NCommaExpr;
                   PROGRAM_AST;
   }
program
   :      stmts
      { #program = #([PROGRAM_AST], program); }
   ;
stmts
         :  (statement (WS)*)+ EOF!
         ;
//stmt
statement
   :  SEMI                // Empty statements
         |          distance SEMI!
   |  direction SEMI!
   |  coordinates SEMI!
   |  points SEMI!
         |          lines SEMI!
         |          overlays SEMI!
         |          perspective SEMI!
         |          places SEMI!
         |          structure SEMI!
   |  printout SEMI!
   |  assigncity SEMI!
         |  changeobj SEMI!
         |          declarations SEMI!
         |          read3dfile SEMI!
         |  expr SEMI!            { ## = #( #[NStatementExpr], ## ); } // Expressions
   |  "while"^ LPAREN! expr RPAREN! block_statement
         |  "do"^ block_statement "while" LPAREN! expr RPAREN! SEMI!
         |  "continue" SEMI!
   |  "break" SEMI!
   |  "return"^ ( expr )? SEMI!
         |  "if"^ LPAREN! expr RPAREN! block_statement
          ( //standard if-else ambiguity
            options { warnWhenFollowAmbig = false; } :
          "else" block_statement )?
   ;
```

```
block_statement
                                    :           statement
                                    |           LCURLY          ( statement )+ RCURLY
                                    ;
declarations
            : "int"^ ID ASSIGN! INTEGER
            | "float"^ ID ASSIGN! FLOAT
            ;
read3dfile
            : "readfile"^ ID (FLOAT | ID) (FLOAT | ID) (FLOAT | ID) (FLOAT | ID) StringConstant
            ;
distance
   :   "dist"^ ID ("IS"! DIST)?
   ;
direction
   :   "dir"^ ID "IS"! dir_spec
   ;
dir_spec
   :   INTEGER
   |   FLOAT
   |   compass_dir
   ;
compass_dir
   :   "N"
   |   "S"
   |   "W"
   |   "E"
   |   "NE"
   |   "NW"
   |   "SE"
   |   "SW"
   ;
coordinates
   :   "coord"^ ID ("IS"! coord_params)?
   ;
coord_params
   :   (FLOAT | (MINUS FLOAT)) COMMA! (FLOAT | (MINUS FLOAT)) (COMMA! DIST)?
   ;
points
   :        "point"^ (ID "IS"! ID)
            ;
lines
            :           "line"^ ID "IS"! (ID)+
            ;
overlays
            :           "overlay"^ ID "IS"! ID ID ID ID COMMA! StringConstant COMMA! dir_spec
            ;
perspective
            :           "persp"^ ID "IS"! ID COMMA! DIST COMMA! dir_spec COMMA! dir_spec
            ;
structure
            :           "struct"^ ID ("IS"! shape COMMA! ID)?
            ;
shape
```

```
            :           "square" FLOAT COMMA! FLOAT
            |           "rectangle" FLOAT COMMA! FLOAT COMMA! FLOAT
            |           "triangle" FLOAT COMMA! FLOAT
            |           "octagon" FLOAT COMMA! FLOAT
        { #shape = #([SHAPE], shape); }
            ;
places
            :           "place"^ ID "IS"! ID COMMA! StringConstant ( COMMA! ID )?
            ;
printout
            : "printk"^ ID
            ;
assigncity
            : "AsigCity"^ ID "IS"! LPAREN! StringConstant COMMA! StringConstant RPAREN!
            ;
changeobj
   :   ID "IS"^ ID ((STAR INTEGER) | (PLUS ((DIST)|(LPAREN! (ID |DIST ) COMMA! (ID |dir_spec )
RPAREN!))) )
   ;
expr
   :   assignExpr (options { warnWhenFollowAmbig = false; } :
                       c:COMMA^ { #c.setType(NCommaExpr); } assignExpr
                   )*
   ;
assignExpr
   :   conditionalExpr ( a:assignOperator! assignExpr { ## = #( #a, ## );} )?
   ;
assignOperator
   :   ASSIGN
   |   DIV_ASSIGN
   |   PLUS_ASSIGN
   |   MINUS_ASSIGN
   |   STAR_ASSIGN
   |   MOD_ASSIGN
   ;
conditionalExpr
   :      logicalOrExpr
          ( QUESTION^ expr COLON! conditionalExpr )?
   ;
constExpr
   :   conditionalExpr
   ;
logicalOrExpr
   :   logicalAndExpr ( LOR^ logicalAndExpr )*
   ;
logicalAndExpr
   :   equalityExpr ( LAND^ equalityExpr )*
   ;
equalityExpr
   :   relationalExpr ( ( EQUAL^ | NOT_EQUAL^ ) relationalExpr )*
   ;
relationalExpr
   :   additiveExpr ( ( LT^ | LTE^ | GT^ | GTE^ ) additiveExpr )*
   ;
```

```
additiveExpr
    :   multExpr ( ( PLUS^ | MINUS^ ) multExpr )*
    ;
multExpr
    :   unaryExpr ( ( STAR^ | DIV^ | MOD^ ) unaryExpr )*
    ;
unaryExpr
    :   postfixExpr
    |   INC^ unaryExpr
    |   DEC^ unaryExpr
    |   u:unaryOperator castExpr { ## = #( #[NUnaryExpr], ## ); }
    ;
castExpr
    :   unaryExpr
    ;
unaryOperator
        :       STAR
        |       PLUS
        |       MINUS
        |       LNOT
        ;

postfixExpr
    :   primaryExpr ( postfixSuffix {## = #( #[NPostfixExpr], ## );} )?
    ;
postfixSuffix
    :   (LBRACKET expr RBRACKET | INC | DEC )+
    ;
primaryExpr
    :   ID
    |   charConst
    |   INTEGER
    |   FLOAT
    |   StringConstant
    |   LPAREN! expr RPAREN!    { ## = #( #[NExpressionGroup, "("], ## ); }
    ;
argExprList
    :   ID ( COMMA! ID )*
    ;
protected
charConst
    :   CharLiteral
    ;
stringConst
    :   (StringLiteral)+        { ## = #(#[NStringSeq], ##); }
    ;


//==================TREE          WALKER          WITH          STATIC          SEMANTIC
CHECKER=========================
{
            import java.io.PrintWriter;
            import java.io.IOException;
            import java.lang.Object;
            import wdcor.WorldCoords;
}
class GECLWalker extends TreeParser;
{
    java.util.Hashtable dict = new java.util.Hashtable();
    java.util.Hashtable PerspTab = new java.util.Hashtable();
    int count=0;
    int countp=0;
    GECLdatatypes[] b = new GECLdatatypes[100];
    GECLdatatypes[] tm = new GECLdatatypes[100];
    PerspectType[] pr = new PerspectType[100];
    Distance sep_dist;
    GECLlibrary kk = new GECLlibrary();
            int tabs = 0;
            int if_break = 0;
            int if_continue = 0;
            int if_skip_rcurly = 0;
            int if_state = 1;
            float recursive_f(AST rec_root)
            {
                    AST e1, e2;
                    int entry_index;
                    //first check if this root is binary operator, if yes, then it will recursively go to next level
                    //if not, then just evaluate and return the value
                    if(rec_root == null)
                            return 0;
                    else if(rec_root.getText().length()==0)
                    {
                            return recursive_f(rec_root.getFirstChild());
                    }
                    else if(rec_root.getText().equals("-"))
                    {
                            e1 = (AST) rec_root.getFirstChild();
                            e2 = (AST) e1.getNextSibling();
                            return (recursive_f(e1) - recursive_f(e2));
                    }
                    else if(rec_root.getText().equals("*"))
                    {
                            e1 = (AST) rec_root.getFirstChild();
                            e2 = (AST) e1.getNextSibling();
                            return (recursive_f(e1) * recursive_f(e2));
                    }
                    else if(rec_root.getText().equals("+"))
                    {
                            e1 = (AST) rec_root.getFirstChild();
                            e2 = (AST) e1.getNextSibling();
                            return (recursive_f(e1) + recursive_f(e2));
                    }
                    else if(rec_root.getText().equals("/"))
                    {
                            e1 = (AST) rec_root.getFirstChild();
                            e2 = (AST) e1.getNextSibling();
                            return (recursive_f(e1) / recursive_f(e2));
                    }
		else if(rec_root.getText().equals("="))
		{
			// assign to left node, left node must be ID
			e1 = (AST) rec_root.getFirstChild();
			e2 = (AST) e1.getNextSibling();

			if( !(dict.containsKey(e1.getText()))) //this is number, make no sense to assign to
number, so skip assing operator
			{
				recursive_f(e2);
				return Float.parseFloat(e1.getText());
			}
			else // this is ID, could be assigned value
			{
				entry_index=(Integer)dict.get( e1.getText());
				if(b[entry_index].type == "int")
				{
					b[entry_index].int_value = (int) recursive_f(e2);
					return b[entry_index].int_value;
				}
				else if(b[entry_index].type == "float")
				{
					b[entry_index].float_value = recursive_f(e2);
					return b[entry_index].float_value;
				}
				return recursive_f(e2);
			}
		}
		else if(rec_root.getText().equals("+="))
		{
			// assign to left node, left node must be ID
			e1 = (AST) rec_root.getFirstChild();
			e2 = (AST) e1.getNextSibling();

			if( !(dict.containsKey(e1.getText()))) //this is number, make no sense to assign to
number, so skip assing operator
			{
				recursive_f(e2);
				return Float.parseFloat(e1.getText());
			}
			else // this is ID, could be assigned value
			{
				entry_index=(Integer)dict.get( e1.getText());

				if(b[entry_index].type == "int")
				{
					b[entry_index].int_value += (int) recursive_f(e2);
					return b[entry_index].int_value;
				}
				else if(b[entry_index].type == "float")
				{
					b[entry_index].float_value += recursive_f(e2);
					return b[entry_index].float_value;
				}
```

```
                return recursive_f(e2);
            }
        }
        else if(rec_root.getText().equals("-="))
        {
            // assign to left node, left node must be ID
            e1 = (AST) rec_root.getFirstChild();
            e2 = (AST) e1.getNextSibling();
            if( !(dict.containsKey(e1.getText()))) //this is number, make no sense to assign to
number, so skip assing operator
            {
                recursive_f(e2);
                return Float.parseFloat(e1.getText());
            }
            else // this is ID, could be assigned value
            {
                entry_index=(Integer)dict.get( e1.getText());
                if(b[entry_index].type == "int")
                {
                    b[entry_index].int_value -= (int) recursive_f(e2);
                    return b[entry_index].int_value;
                }
                else if(b[entry_index].type == "float")
                {
                    b[entry_index].float_value -= recursive_f(e2);
                    return b[entry_index].float_value;
                }
                return recursive_f(e2);
            }
        }
        else if(rec_root.getText().equals("*="))
        {
            // assign to left node, left node must be ID
            e1 = (AST) rec_root.getFirstChild();
            e2 = (AST) e1.getNextSibling();
            if( !(dict.containsKey(e1.getText()))) //this is number, make no sense to assign to
number, so skip assing operator
            {
                recursive_f(e2);
                return Float.parseFloat(e1.getText());
            }
            else // this is ID, could be assigned value
            {
                entry_index=(Integer)dict.get( e1.getText());

                if(b[entry_index].type == "int")
                {
                    b[entry_index].int_value *= (int) recursive_f(e2);
                    return b[entry_index].int_value;
                }
                else if(b[entry_index].type == "float")
                {
                    b[entry_index].float_value *= recursive_f(e2);
                    return b[entry_index].float_value;
```

```
				}
				return recursive_f(e2);
			}
		}
		else if(rec_root.getText().equals("/="))
		{
			// assign to left node, left node must be ID
			e1 = (AST) rec_root.getFirstChild();
			e2 = (AST) e1.getNextSibling();

			if( !(dict.containsKey(e1.getText()))) //this is number, make no sense to assign to
number, so skip assing operator
			{
				recursive_f(e2);
				return Float.parseFloat(e1.getText());
			}
			else // this is ID, could be assigned value
			{
				entry_index=(Integer)dict.get( e1.getText());

				if(b[entry_index].type == "int")
				{
					b[entry_index].int_value /= (int) recursive_f(e2);
					return b[entry_index].int_value;
				}
				else if(b[entry_index].type == "float")
				{
					b[entry_index].float_value /= recursive_f(e2);
					return b[entry_index].float_value;
				}
				return recursive_f(e2);
			}
		}
		else if(rec_root.getText().equals("=="))
		{
			e1 = (AST) rec_root.getFirstChild();
			e2 = (AST) e1.getNextSibling();
			if( recursive_f(e1) == recursive_f(e2))
				return 1;
			return -1;
		}

		else if(rec_root.getText().equals("!="))
		{
			e1 = (AST) rec_root.getFirstChild();
			e2 = (AST) e1.getNextSibling();
			if( recursive_f(e1) != recursive_f(e2))
				return 1;
			return 0;
		}
		else if(rec_root.getText().equals("<"))
		{
			e1 = (AST) rec_root.getFirstChild();
			e2 = (AST) e1.getNextSibling();
```

```
        if( recursive_f(e1) < recursive_f(e2))
                return 1;
        return -1;
}
else if(rec_root.getText().equals(">"))
{
        e1 = (AST) rec_root.getFirstChild();
        e2 = (AST) e1.getNextSibling();
        if( recursive_f(e1) > recursive_f(e2))
                return 1;
        return -1;
}
else if(rec_root.getText().equals("<="))
{
        e1 = (AST) rec_root.getFirstChild();
        e2 = (AST) e1.getNextSibling();
        if( recursive_f(e1) <= recursive_f(e2))
                return 1;
        return 0;
}
else if(rec_root.getText().equals(">="))
{
        e1 = (AST) rec_root.getFirstChild();
        e2 = (AST) e1.getNextSibling();
        if( recursive_f(e1) >= recursive_f(e2))
                return 1;
        return 0;
}
else if(rec_root.getText().equals("&&"))
{
        e1 = (AST) rec_root.getFirstChild();
        e2 = (AST) e1.getNextSibling();
        int tempi1 = (int) recursive_f(e1);
        int tempi2 = (int) recursive_f(e2);
        if(tempi1>0 && tempi2>0)
                return 1;
        return 0;
}
else if(rec_root.getText().equals("||"))
{
        e1 = (AST) rec_root.getFirstChild();
        e2 = (AST) e1.getNextSibling();
        int tempi1 = (int) recursive_f(e1);
        int tempi2 = (int) recursive_f(e2);
        if(tempi1>0 || tempi2>0)
                return 1;
        return 0;
}
else if(rec_root.getText().equals("break"))
{
        if_break = 1;
        return 1;
}
else if(rec_root.getText().equals("continue"))
```

```
{
        if_continue = 1;
        return 1;
}
else if(rec_root.getText().equals("++"))
{
        // assign to left node, left node must be ID
        e1 = (AST) rec_root.getFirstChild();
        entry_index=(Integer)dict.get( e1.getText());

        if(b[entry_index].type == "int")
        {
                return (b[entry_index].int_value++);
        }
        else if(b[entry_index].type == "float")
        {
                return (b[entry_index].float_value++);
        }
        return 0;
}
else if(rec_root.getText().equals("--"))
{
        // assign to left node, left node must be ID
        e1 = (AST) rec_root.getFirstChild();
        entry_index=(Integer)dict.get( e1.getText());

        if(b[entry_index].type == "int")
        {
                return (b[entry_index].int_value--);
        }
        else if(b[entry_index].type == "float")
        {
                return (b[entry_index].float_value--);
        }
        return 0;
}
else
{
        if( !(dict.containsKey(rec_root.getText()))) //this is number
        {
                return Float.parseFloat(rec_root.getText());
        }
else //this is id
        {
                entry_index=(Integer)dict.get( rec_root.getText());

                if(b[entry_index].type == "int")
                        return b[entry_index].int_value;
                else if(b[entry_index].type == "float")
                        return b[entry_index].float_value;
                else
                {
                        return 0;
                        //not in our consideration now
```

```
                                                }
                                        }
                                }
                }}
program {int dummy; }
: #(PROGRAM_AST {
                    kk.creatkml();
                    kk.sortCityCoords();
                }
    (dummy=stmt
          {
          System.out.println("obj number count = " + (count-1));
          })+
          {kk.endkml();}
         )
;
stmt returns [int dummy]
{dummy=0;float ii=0.0f; float jj=0.0f;float kf=0.0f;double dd=0.0d;
int tempi1 = 0;
float tempf1 = 0.0f;
}
: #("dist" {
                                    b[count] = new GECLdatatypes();
                                    b[count].settype("distance");              }
   ID     {b[count].setname(#ID.getText());
                        dict.put(#ID.getText(),count);}

   (DIST   {b[count].setdists( #DIST.getText()); })?
   { count++; }
   )

| #("coord" {
                                     b[count] = new GECLdatatypes();
                                     b[count].settype("coord");              }
   ID      {b[count].setname(#ID.getText());
                        dict.put(#ID.getText(),count);}
   {b[count].setcoord( ii,jj,"0ft");}
   (((MINUS  (m1:FLOAT   {ii=Float.parseFloat(#m1.getText()); ii=0-ii;}))|
   (f1:FLOAT   {ii=Float.parseFloat(#f1.getText());}))
   ( ( MINUS  (m2:FLOAT   {jj=Float.parseFloat(#m2.getText()); jj=0-jj;} ) ) |
   ( f2:FLOAT   {jj=Float.parseFloat(#f2.getText());}))
   DIST    {b[count].setcoord( ii,jj,#DIST.getText());})?
   {count++; }
   )

| #("AsigCity" {  }
   as1:ID  {   if(!(dict.containsKey(#as1.getText()))) kk.reportError("not a valid city", #as1.getText());
             else dummy=(Integer)dict.get(#as1.getText());
          }
   as2:StringConstant {}
   as3:StringConstant {
   WorldCoords tmp=kk.getCityCoords(#as2.getText(),#as3.getText());
   b[dummy].setCityCoords(tmp);}
   )
```

```
| #("point" {
                                    b[count] = new GECLdatatypes();
                                    b[count].settype("point");
          }
   s1:ID   {b[count].setname(#s1.getText());
                        dict.put(#s1.getText(),count);}
            s2:ID   {
          if(!(dict.containsKey(#s2.getText()))) kk.reportError("unrecognized", #s2.getText());
          else {
               dummy=(Integer)dict.get(#s2.getText());
               if(b[dummy].gettype()!="coord")
                 kk.reportError("not a coordinate", #s2.getText());
               else
                 b[count].cpy(b[dummy]);
          }
          count++;
                   }
   )
| #("place" {
                                    b[count] = new GECLdatatypes();
                                    b[count].settype("place");              }
   s3:ID      {b[count].setname(#s3.getText());
                        dict.put(#s3.getText(),count);}
            s4:ID   {
                                   if(!(dict.containsKey(#s4.getText())))                    kk.reportError("unrecognized",
#s4.getText());
                        else {
                             dummy=(Integer)dict.get(#s4.getText());
               if(b[dummy].gettype()!="point")
                 kk.reportError("not a point", #s4.getText());
               else
                 b[count].cpy(b[dummy]);
                             }
                   }
            s5:StringConstant {
                         b[count].settitle(#s5.getText());
                        }
            (pla1:ID   {
                                   if(!(PerspTab.containsKey(#pla1.getText())))          kk.reportError("unrecognized",
#pla1.getText());
                        else {
                             dummy=(Integer)PerspTab.get(#pla1.getText());
               b[count].setpersp(pr[dummy]);
                             }
                   })?
            { count++;  }
   )
| #("struct"{         b[count] = new GECLdatatypes();
                                                b[count].settype("struct");                         }
   s6:ID   {          b[count].setname(#s6.getText());
                                   dict.put(#s6.getText(),count);                                               }

   (((       "square"              {b[count].setshape("square");}
            f3:FLOAT            {ii=Float.parseFloat(#f3.getText());}
```

```
        f4:FLOAT        {jj=Float.parseFloat(#f4.getText());}
                s7:ID           {
                                        if( !(dict.containsKey(#s7.getText()))) {
                                        kk.reportError("unrecognized: ",#s7.getText());
                                    }else {
                                dummy=(Integer)dict.get(#s7.getText());
                                b[count].cpy(b[dummy]);
                                }
                                b[count].setsquare(ii,jj);
                        }
  )|("rectangle"  {b[count].setshape("rectangle");}
        f5:FLOAT        {ii=Float.parseFloat(#f5.getText());}
        f6:FLOAT        {jj=Float.parseFloat(#f6.getText());}
        f7:FLOAT        {kf=Float.parseFloat(#f7.getText());}
                s8:ID           {
                                        if( !(dict.containsKey(#s8.getText()))) {
                                        kk.reportError("unrecognized: ",#s8.getText());
                                    }else {
                                dummy=(Integer)dict.get(#s8.getText());
                                b[count].cpy(b[dummy]);
                                }
                                b[count].setrect(ii,jj,kf);
                        }

  )|("triangle" {b[count].setshape("triangle");})
  ))?
  { count++; }
  )
| #("overlay"      {          b[count] = new GECLdatatypes();
                                    b[count].settype("overlay");              }
  ovs1:ID          {          b[count].setname(#ovs1.getText());
                                dict.put(#ovs1.getText(),count);}
        ovs2:ID             {           if(   !(dict.containsKey(#ovs2.getText())))   kk.reportError("unrecognized",
#ovs2.getText());
                                else {
                                dummy=(Integer)dict.get(#ovs2.getText());
              if(b[dummy].gettype()!="coord")
                kk.reportError("not a coordinate", #ovs2.getText());
              else
                tm[0]=b[dummy];
                        }
                 }
        ovs3:ID             {           if(   !(dict.containsKey(#ovs3.getText())))   kk.reportError("unrecognized",
#ovs3.getText());
                                else {
                                dummy=(Integer)dict.get(#ovs3.getText());
              if(b[dummy].gettype()!="coord")
                kk.reportError("not a coordinate", #ovs3.getText());
              else
                                tm[1]=b[dummy];
                        }
                 }
        ovs4:ID             {           if(   !(dict.containsKey(#ovs4.getText())))   kk.reportError("unrecognized",
#ovs4.getText());
```

```
                                        else {
                                        dummy=(Integer)dict.get(#ovs4.getText());
                  if(b[dummy].gettype()!="coord")
                     kk.reportError("not a coordinate", #ovs3.getText());
                  else
                                        tm[2]=b[dummy];
                                }
                          }
            ovs5:ID             {       if(  !(dict.containsKey(#ovs5.getText())))  kk.reportError("unrecognized",
#ovs5.getText());
                                        else {
                                        dummy=(Integer)dict.get(#ovs5.getText());
                  if(b[dummy].gettype()!="coord")
                     kk.reportError("not a coordinate", #ovs5.getText());
                  else
                                        tm[3]=b[dummy];
                                }
                                        b[count].setoverlay(tm[0],tm[1],tm[2],tm[3]);
                                }
            StringConstant {b[count].seturl(#StringConstant.getText());  }
            ovs6:FLOAT          {   ii=Float.parseFloat(#ovs6.getText());
                                                b[count].setrotat(ii);
                                                count++;
                                        }
            )
| #("persp" { pr[countp] = new PerspectType(); }
   psp1:ID      {pr[countp].setname(#psp1.getText());
                PerspTab.put(#psp1.getText(),countp);}
            psp2:ID   {
                                if(       !(dict.containsKey(#psp2.getText())))       kk.reportError("unrecognized",
#psp2.getText());
                          else {
                              dummy=(Integer)dict.get(#psp2.getText());
               if(b[dummy].gettype()!="coord")
                 kk.reportError("not a coordinate", #psp2.getText());
               else
                 pr[countp].setcoord(b[dummy]);
                              }
                      }
            DIST    {               Distance Mydist = new Distance(#DIST.getText());
                                        pr[countp].setrange( Mydist.getDouble()/0.3048 );}
            psp3:FLOAT          {   ii=Float.parseFloat(#psp3.getText());
                                                pr[countp].settilt(ii);
                                        }
            psp4:FLOAT          {   jj=Float.parseFloat(#psp4.getText());
                                                pr[countp].setheading(jj);
                                                countp++;
                                        }
   )
| #("line" {
                                b[count] = new GECLdatatypes();
                                b[count].settype("line");             }
   ln1:ID      {b[count].setname(#ln1.getText());
                                dict.put(#ln1.getText(),count);}
            (ID   {
                                if(           !(dict.containsKey(#ID.getText())))           kk.reportError("unrecognized",
#ID.getText());
          else {
                         dummy=(Integer)dict.get(#ID.getText());
              if(b[dummy].gettype()!="coord")
                kk.reportError("not a coordinate", #ID.getText());
              else
                          b[count].insert(b[dummy]);
                        }
                })+
            { count++;  }
    )
|  #("IS"
    type:ID     {   if( !(dict.containsKey(#type.getText()))) kk.reportError("unrecognized", #type.getText());
                else {
                  dummy=(Integer)dict.get(#type.getText());
                  if(b[dummy].gettype()!="coord"            &&            b[dummy].gettype()!="distance"            &&
b[dummy].gettype()!="struct")
                      kk.reportError("invalid statement", "IS");
                  else if(((b[dummy].gettype()).toString()).equals("distance")) {
                      AST ast = type.getNextSibling();
                      String dstnc = ast.toString();
                      if(!(dict.containsKey(#ast.getText()))) kk.reportError("unrecognized", #ast.getText());
                      int dumm = (Integer)dict.get(#ast.getText());
                      if(b[dumm].gettype()!="distance") kk.reportError("invalid statement", "IS");
                      dstnc = b[dumm].getdists();
                      sep_dist = new Distance(dstnc);
                      int IntValue=0;
                      double tmpdouble=0.0;
                      float FloValue=0.0f;
                      if((ast.getNextSibling()).getNextSibling().getType()==DIST){
                         dstnc = ((ast.getNextSibling()).getNextSibling()).toString();
                       Distance new_dist = new Distance(dstnc);
                       tmpdouble= new_dist.getDouble(); // get meter
                       FloValue = (float)tmpdouble;

                       }else if ((ast.getNextSibling()).getNextSibling().getType()==INTEGER){
                                dstnc = ((ast.getNextSibling()).getNextSibling()).toString();
                       IntValue = Integer.parseInt(dstnc);
                       }
                    //  dstnc = ((ast.getNextSibling()).getNextSibling()).toString();
                   //   int value = Integer.parseInt(dstnc);
                      if(((ast.getNextSibling()).toString()).equals("*")) {
                         dd = sep_dist.getDouble() * IntValue / 0.3048;
                         dstnc = Double.toString(dd) + "ft";
                         b[dummy].setdists(dstnc);
                         System.err.println("scqwcqwceqc    "   +   dstnc   +   "      "   +   dummy   +   "         "   +
b[dummy].getdists());
                      }
                      else {
                         dd = (sep_dist.getDouble() + FloValue) / 0.3048;
                         dstnc = Double.toString(dd) + "ft";
                         b[dummy].setdists(dstnc);
                         System.err.println("scqwcqwceqc   "  +  dstnc  +  "     "  +  dummy  +  "        "  +
b[dummy].getdists());
                      }
                    }else if(((b[dummy].gettype()).toString()).equals("coord"))
                    {

                       System.out.println("Debug!! ");
                       float setdist=0.0f ;
                       int setdir=0;
                    // get first ID
                       AST ast = type.getNextSibling();
                      String dstnc = ast.toString();
                      if(!(dict.containsKey(#ast.getText()))) kk.reportError("unrecognized", #ast.getText());
                      int dumm = (Integer)dict.get(#ast.getText());
                      System.out.println("Debug!! 1 long = "+b[dumm].gettype());
                      if(!b[dumm].gettype().equals("coord")) kk.reportError("invalid statement", "IS");
                       boolean IsPlus = false;
                       if(((ast.getNextSibling()).toString()).equals("+")) {
                      IsPlus = true;
                      }
                      else {
                      kk.reportError("invalid operator", "*");
                      }
                    // get first parameter suppose a dist ID or number
                      AST ast1 = (ast.getNextSibling()).getNextSibling();
                      System.out.println("Debug!! 1.5 type = "+ast1.getType());
                      if(ast1.getType()==ID){
                    // get a dist ID
                      dstnc = ast1.toString();
                      if(!(dict.containsKey(#ast1.getText()))) kk.reportError("unrecognized", #ast1.getText());
                      int dummdist = (Integer)dict.get(#ast1.getText());
                      System.out.println("Debug!! 2 long = "+b[dummdist].gettype());
                      if(b[dummdist].gettype()!="distance") kk.reportError("invalid statement", "IS");
                                                      sep_dist              =                 new
Distance(b[dummdist].getdists());
                                                      setdist = (float)(sep_dist.getDouble()/ 0.3048);


                                                      }else if(ast1.getType()==DIST){
                                                 // get a dist number
                                                      sep_dist = new Distance(#ast1.getText());
                                                      setdist = (float)(sep_dist.getDouble()/ 0.3048);

                                                      }

                  // get second parameter suppose a direction
                      AST ast2 = ast1.getNextSibling();
                      System.out.println("Debug!! 2.5 type = "+ast2.getType());
                      if(ast2.getType()==ID){
                    // get a dir ID
                      dstnc = ast2.toString();
                      if(!(dict.containsKey(#ast2.getText()))) kk.reportError("unrecognized", #ast2.getText());
                      int dummdir = (Integer)dict.get(#ast2.getText());
                      System.out.println("Debug!! 3 type = "+b[dummdir].gettype());
```

```
                    if(b[dummdir].gettype()!="int") kk.reportError("invalid statement", "IS");
                                                                      setdir = b[dummdir].getIntValue();

                                                                      }else if(ast2.getType()==INTEGER){
                                                              // get a dir number
                                                                      setdir = Integer.parseInt(#ast2.getText());
                                                                      }


                   System.out.println("setdist = "+setdist);
                   b[dummy].navigate(b[dumm].getlon(),b[dumm].getlat(),setdist,setdir);
                       System.out.println("Debug!! long = "+b[dummy].getlon()+"lat = "+b[dummy].getlat());

                   }else
                   {   System.err.println("dcrv");}
                 }
               }
   )
|  #("printk"
   ID    {
                    if( !(dict.containsKey(#ID.getText()))) kk.reportError("unrecognized: ",#ID.getText());
                    else  {dummy=(Integer)dict.get(#ID.getText());  System.out.println("printk  find  index  =
"+dummy +"  type = "+b[dummy].getname()); }
                    if(b[dummy].gettype() == "distance") kk.printdis(b[dummy].getdists());
                    if(b[dummy].gettype() == "coord") kk.printcoord(b[dummy]);
                    if(b[dummy].gettype() == "point") kk.printPoint(b[dummy]);
                    if(b[dummy].gettype() == "place") kk.printPlace(b[dummy]);
                    if(b[dummy].gettype() == "struct"){
                    if(b[dummy].getshape() == "square") kk.printquadrangle(b[dummy]);
                    if(b[dummy].getshape() == "rectangle") kk.printquadrangle(b[dummy]);
                    if(b[dummy].getshape()              ==              "triangle")              /*{
tm.settriangle(b[dummy]);kk.printtriangle(b[dummy]);}*/;
                    }
                    if(b[dummy].gettype() == "overlay") kk.printOverlay(b[dummy]);
                    if(b[dummy].gettype() == "line") kk.printLine(b[dummy]);
                           if(b[dummy].gettype()                  ==                  "read3Dfile")
b[dummy].read3Dfile.renderKML(kk.outp);
               }
   )

|  #("int" {
                           b[count] = new GECLdatatypes();
                           b[count].settype("int");          }
   ID     {b[count].setname(#ID.getText());
                    dict.put(#ID.getText(),count);}

         i1:INTEGER {
                                          tempi1=Integer.parseInt(#i1.getText());
                     b[count].setIntValue(tempi1);
                      count++;
                     }
   )
|  #("float" {
                           b[count] = new GECLdatatypes();
```

```
                                b[count].settype("float");              }
   ID      {b[count].setname(#ID.getText());
                    dict.put(#ID.getText(),count);}

           f21:FLOAT {
                                                         tempf1=Float.parseFloat(#f21.getText());
                     b[count].setFloatValue(tempf1);
                      count++;
                     }
   )
 | #("readfile" {
                                b[count] = new GECLdatatypes();
                                b[count].settype("read3Dfile");
                                }
   ID      {b[count].setname(#ID.getText());
                    dict.put(#ID.getText(),count);
                              }

           ( scale1:FLOAT {
                    b[count].read3Dfile.scale = Float.parseFloat(#scale1.getText());
           }
           | ID {
                    tempi1 = (Integer)dict.get(#ID.getText());
                    b[count].read3Dfile.scale = b[tempi1].getFloatValue();
                    }
           )
           ( translate_x1:FLOAT {
                    b[count].read3Dfile.translate_x = Float.parseFloat(#translate_x1.getText());
           }
           | ID {
                    tempi1 = (Integer)dict.get(#ID.getText());
                    b[count].read3Dfile.translate_x = b[tempi1].getFloatValue();
                    }
           )
           ( translate_y1:FLOAT {
                    b[count].read3Dfile.translate_y = Float.parseFloat(#translate_y1.getText());
           }
           | ID {
                    tempi1 = (Integer)dict.get(#ID.getText());
                    b[count].read3Dfile.translate_y = b[tempi1].getFloatValue();
                    }
           )
           (translate_z1:FLOAT {
                    b[count].read3Dfile.translate_z = Float.parseFloat(#translate_z1.getText());
           }
           | ID {
                    tempi1 = (Integer)dict.get(#ID.getText());
                    b[count].read3Dfile.translate_z = b[tempi1].getFloatValue();
                    }
           )
           st21:StringConstant {
                                                         b[count].read3Dfile.openfile(#st21.getText());
                      count++;
                     }
   )
| statement
;

statement
{
            float exprReturn = 0;
            float ii = 0.0f;
            float jj = 0.0f;
            float kf = 0.0f;
            double dd = 0;
            AST e1, e2, e3, as2, as3;
            int dummy = 0;
}
       :     s:SEMI
                  |     #(NStatementExpr
             exprReturn = expr
             )              // Expressions
// Iteration statements:
       |     #( w:"while" {if_break = 0; if_skip_rcurly = 0;}
                                                exprReturn = exp1:expr
                                                (
                                                            { exprReturn > 0 }? block1:block_statement
                                                                        {
                                                                                    if(if_break == 1)
                                                                                                if_break = 0;
                                                                                    else
                                                                                    {

                                                                                                while( recursive_f(exp1) > 0)
                                                                                                {
                                                                                                            e1     =     (AST)
w.getFirstChild(); //expr
                                                                                                            e1     =     (AST)
e1.getNextSibling(); // {
                                                                                                            e1     =     (AST)
e1.getNextSibling(); // first statement now
                                                                                                            while(true)
                                                                                                            {

            if(e1.getText().length()>0)
                                                                                                                        {

            if(e1.getText().equals("break"))
                                                                                                                                    {

            if_break = 1;

            break;
                                                                                                                                    }

            else if(e1.getText().equals("if"))
                                                                                                                                    {
```

```
System.out.println("inside while loop and if");
e2 = (AST) e1.getFirstChild(); // expr => ==, ||, &&, != .......
System.out.println("first child node is " + e2.getText());
e3 = (AST) e2.getNextSibling(); // {
System.out.println("next node is " + e3.getText());
e3 = (AST) e3.getNextSibling(); // block statement
System.out.println("next node is " + e3.getText());
int tempi1 = (int) recursive_f(e2);
if(tempi1 > 0) // execut this block
{
        System.out.println("execut this block");
        while(!e3.getText().equals("}"))
        {
                recursive_f(e3);
                e3 = (AST) e3.getNextSibling();
                System.out.println("next node is " + e3.getText());
        }
}
else
{
        //skip first block
        System.out.println("skip first block");
        while(!e3.getText().equals("}"))
        {
                e3 = (AST) e3.getNextSibling();
        }
        e3 = (AST) e3.getNextSibling(); //check "else"
        System.out.println("check else");
        if(e3 != null) //this is else
        {
                System.out.println("this is else inside");
                e3 = (AST) e3.getNextSibling(); // "{"
                e3 = (AST) e3.getNextSibling(); // statement block
                while(!e3.getText().equals("}"))
                {
                        recursive_f(e3);
                        e3 = (AST) e3.getNextSibling();
                }
        }
}
if(if_break == 1 || if_continue == 1)
{
        break;
}
                                                                                        }
else if(e1.getText().equals("continue") || e1.getText().equals("}"))
                                                                                        {
break;
                                                                                        }
else if(e1.getText().equals("dist"))
                                                                                        {
//dist
b[count] = new GECLdatatypes();
b[count].settype("distance");
//ID
```

```
e2 = e1.getFirstChild();
b[count].setname(e1.getText());
dict.put(e2.getText(),count);
//(DIST )?
e2 = e2.getNextSibling();
if(e2!=null)
b[count].setdists( e2.getText());
count++;
                                                            }
else if(e1.getText().equals("coord"))
                                                            {
//coord
b[count] = new GECLdatatypes();
b[count].settype("coord");
//ID
e2 = e1.getFirstChild();
b[count].setname(e2.getText());
dict.put(e2.getText(),count);
b[count].setcoord( ii,jj,"0ft");

//FLOAT | (MINUS FLOAT)
e2 = e2.getNextSibling();

if(e2 == null) //finished, there is no assign value here
{
}
else // assign value
{
        if(e2.getText().equals("-")) //MINUS FLOAT case
```

```
{
        e2 = e2.getNextSibling();
        ii=Float.parseFloat(e2.getText());
        ii=0-ii;
}
else //FLOAT
{
        e2 = e2.getNextSibling();
        ii=Float.parseFloat(e2.getText());
}

//FLOAT | (MINUS FLOAT)
e2 = e2.getNextSibling();
if(e2.getText().equals("-")) //MINUS FLOAT case
{
        e2 = e2.getNextSibling();
        jj=Float.parseFloat(e2.getText());
        jj=0-jj;
}
else //FLOAT
{
        e2 = e2.getNextSibling();
        jj=Float.parseFloat(e2.getText());
}
//DIST
e2 = e2.getNextSibling();
b[count].setcoord( ii,jj, e2.getText());
```

```
}
count++;
                                                                      }
else if(e1.getText().equals("AsigCity"))
                                                                      {
//as1:ID
e2 = e1.getFirstChild();
if(!(dict.containsKey(e2.getText())))
{

     kk.reportError("not a valid city", e2.getText());
}
else
{

     dummy=(Integer)dict.get(e2.getText());


     //as2:StringConstant
     e2 = e2.getNextSibling();
     as2 = e2;


     //as3:StringConstant
     e2 = e2.getNextSibling();
     as3 = e2;


     WorldCoords tmp=kk.getCityCoords(as2.getText(), as3.getText());
     b[dummy].setCityCoords(tmp);
}
                                                                      }
```

```
else if(e1.getText().equals("point"))
                                                                                                    {
//point
b[count] = new GECLdatatypes();
b[count].settype("point");
//s1:ID
e2 = e1.getFirstChild();
b[count].setname(e2.getText());
dict.put(e2.getText(),count);
//s2:ID
e2 = e2.getNextSibling();
if(!(dict.containsKey(e2.getText())))
        kk.reportError("unrecognized", e2.getText());
else
{
        dummy=(Integer)dict.get(e2.getText());
        if(b[dummy].gettype()!="coord")
                kk.reportError("not a coordinate", e2.getText());
        else
                b[count].cpy(b[dummy]);
}
count++;
                                                                                                    }
else if(e1.getText().equals("place"))
                                                                                                    {
//place
b[count] = new GECLdatatypes();
b[count].settype("place");
```

```
//s3:ID
e2 = e1.getFirstChild();
b[count].setname(e2.getText());
dict.put(e2.getText(),count);
//s4:ID
e2 = e2.getNextSibling();
if(!(dict.containsKey(e2.getText())))
        kk.reportError("unrecognized", e2.getText());
else
{
        dummy=(Integer)dict.get(e2.getText());
        if(b[dummy].gettype()!="point")
                kk.reportError("not a point", e2.getText());
        else
                b[count].cpy(b[dummy]);
}
//s5:StringConstant
e2 = e2.getNextSibling();
b[count].settitle(e2.getText());
// (pla1:ID)?
e2 = e2.getNextSibling();
if(e2 == null)
{
}
else
{
        if(!(PerspTab.containsKey(e2.getText())))
```

```
                kk.reportError("unrecognized", e2.getText());
        else
        {
                dummy=(Integer)PerspTab.get(e2.getText());
                b[count].setpersp(pr[dummy]);
        }
}
count++;
                                                                                        }
else if(e1.getText().equals("struct"))
                                                                                        {
//struct
b[count] = new GECLdatatypes();
b[count].settype("struct");
//s6:ID
e2 = e1.getFirstChild();
b[count].setname(e2.getText());
dict.put(e2.getText(),count);
//( shape ID )?
e2 = e2.getNextSibling();
if(e2==null)
{
}
else
{
        if(e2.getText().equals("square"))
        {
                b[count].setshape("square");
                //f3:FLOAT
                e2 = e2.getNextSibling();
                ii=Float.parseFloat(e2.getText());
                //f4:FLOAT
                e2 = e2.getNextSibling();
                jj=Float.parseFloat(e2.getText());
                //s7:ID
                e2 = e2.getNextSibling();
                if(!(dict.containsKey(e2.getText())))
                {
                        kk.reportError("unrecognized: " , e2.getText());
                }
                else
                {
                        dummy=(Integer)dict.get(e2.getText());
                        b[count].cpy(b[dummy]);
                }
                b[count].setsquare(ii,jj);
        }
        else if(e2.getText().equals("rectangle"))
        {
                b[count].setshape("rectangle");
                //f5:FLOAT
                e2 = e2.getNextSibling();
                ii=Float.parseFloat(e2.getText());
                //f6:FLOAT
                e2 = e2.getNextSibling();
```

```
                jj=Float.parseFloat(e2.getText());

                //f7:FLOAT

                e2 = e2.getNextSibling();

                kf=Float.parseFloat(e2.getText());

                //s8:ID

                e2 = e2.getNextSibling();

                if( !(dict.containsKey(e2.getText())))

                {

                        kk.reportError("unrecognized: ", e2.getText());

                }

                else

                {

                        dummy=(Integer)dict.get(e2.getText());

                        b[count].cpy(b[dummy]);

                }

                b[count].setrect(ii,jj,kf);

        }

        else if(e2.getText().equals("triangle"))

        {

                b[count].setshape("triangle");

        }

        count++;

}
                                                                                        }

else if(e1.getText().equals("overlay"))
                                                                                        {
//overlay
b[count] = new GECLdatatypes();
b[count].settype("overlay");
```

```
          //ovs1:ID
          e2 = e1.getFirstChild();
          b[count].setname(e2.getText());
dict.put(e2.getText(),count);

          //ovs2:ID
          e2 = e2.getNextSibling();
          if( !(dict.containsKey(e2.getText())))
                    kk.reportError("unrecognized", e2.getText());
else
          {
          dummy=(Integer)dict.get(e2.getText());
   if(b[dummy].gettype()!="coord")
      kk.reportError("not a coordinate", e2.getText());
   else
      tm[0]=b[dummy];
}

          //ovs3:ID
          e2 = e2.getNextSibling();
          if( !(dict.containsKey(e2.getText())))
                    kk.reportError("unrecognized", e2.getText());
else
          {
          dummy=(Integer)dict.get(e2.getText());
   if(b[dummy].gettype()!="coord")
      kk.reportError("not a coordinate", e2.getText());
   else
                    tm[1]=b[dummy];
}

          //ovs4:ID
          e2 = e2.getNextSibling();
          if( !(dict.containsKey(e2.getText())))
                    kk.reportError("unrecognized", e2.getText());
else
          {
          dummy=(Integer)dict.get(e2.getText());
   if(b[dummy].gettype()!="coord")
      kk.reportError("not a coordinate", e2.getText());
   else
                    tm[2]=b[dummy];
}

          //ovs5:ID
          e2 = e2.getNextSibling();
          if( !(dict.containsKey(e2.getText())))
                    kk.reportError("unrecognized", e2.getText());
          else
          {
                    dummy=(Integer)dict.get(e2.getText());
   if(b[dummy].gettype()!="coord")
      kk.reportError("not a coordinate", e2.getText());
   else
```

```
            tm[3]=b[dummy];
    }
    b[count].setoverlay(tm[0],tm[1],tm[2],tm[3]);

            //StringConstant
            e2 = e2.getNextSibling();
            b[count].seturl(e2.getText());

            //ovs6:FLOAT
            e2 = e2.getNextSibling();
            ii=Float.parseFloat(e2.getText());
            b[count].setrotat(ii);
            count++;
}
else if(e1.getText().equals("persp"))
{
            //persp
            pr[countp] = new PerspectType();

            //psp1:ID
            e2 = e1.getFirstChild();
            pr[countp].setname(e2.getText());
    PerspTab.put(e2.getText(),countp);

            //psp2:ID
            e2 = e2.getNextSibling();
            if( !(dict.containsKey(e2.getText())))
                        kk.reportError("unrecognized", e2.getText());
    else
            {
        dummy=(Integer)dict.get(e2.getText());
        if(b[dummy].gettype()!="coord")
            kk.reportError("not a coordinate", e2.getText());
        else
            pr[countp].setcoord(b[dummy]);
    }

            //DIST
            e2 = e2.getNextSibling();
            Distance Mydist = new Distance(e2.getText());
            pr[countp].setrange( Mydist.getDouble()/0.3048 );

            //psp3:FLOAT
            e2 = e2.getNextSibling();
            ii=Float.parseFloat(e2.getText());
            pr[countp].settilt(ii);

            //psp4:FLOAT
            e2 = e2.getNextSibling();
            jj=Float.parseFloat(e2.getText());
            pr[countp].setheading(jj);
            countp++;
}
else if(e1.getText().equals("line"))
```

```
{
            //line
            b[count] = new GECLdatatypes();
            b[count].settype("line");

            //ln1:ID
            e2 = e1.getFirstChild();
            b[count].setname(e2.getText());
   dict.put(e2.getText(),count);

            // ( ID )+
            e2 = e2.getNextSibling();
            while(e2 != null)
            {
                        if( !(dict.containsKey(e2.getText())))
                                    kk.reportError("unrecognized", e2.getText());
      else
                        {
                dummy=(Integer)dict.get(e2.getText());
          if(b[dummy].gettype()!="coord")
             kk.reportError("not a coordinate", e2.getText());
          else
                                                b[count].insert(b[dummy]);
                        }
                        e2 = e2.getNextSibling();
            }
            count++;

}

else if(e1.getText().equals("IS"))
{
            //IS, type:ID
            e2 = e1.getFirstChild();

            if( !(dict.containsKey(e2.getText())))
                        kk.reportError("unrecognized", e2.getText());
                else {
                  dummy=(Integer)dict.get(e2.getText());
                  if(b[dummy].gettype()!="coord"            &&            b[dummy].gettype()!="distance"            &&
b[dummy].gettype()!="struct")
                      kk.reportError("invalid statement", "IS");
                  else if(((b[dummy].gettype()).toString()).equals("distance")) {
                      AST ast = e2.getNextSibling();
                      String dstnc = ast.toString();
                      if(!(dict.containsKey(ast.getText()))) kk.reportError("unrecognized", ast.getText());
                      int dumm = (Integer)dict.get(ast.getText());
                      if(b[dumm].gettype()!="distance") kk.reportError("invalid statement", "IS");
                      dstnc = b[dumm].getdists();
                      sep_dist = new Distance(dstnc);
                      int IntValue=0;
                      double tmpdouble=0.0;
                      float FloValue=0.0f;
                      if((ast.getNextSibling()).getNextSibling().getType()==DIST){
```

```
                                  dstnc = ((ast.getNextSibling()).getNextSibling()).toString();
                               Distance new_dist = new Distance(dstnc);
                               tmpdouble= new_dist.getDouble(); // get meter
                               FloValue = (float)tmpdouble;

                               }else if ((ast.getNextSibling()).getNextSibling().getType()==INTEGER){
                                              dstnc = ((ast.getNextSibling()).getNextSibling()).toString();
                               IntValue = Integer.parseInt(dstnc);

                               }
                            //  dstnc = ((ast.getNextSibling()).getNextSibling()).toString();
                           //   int value = Integer.parseInt(dstnc);
                              if(((ast.getNextSibling()).toString()).equals("*")) {
                                  dd = sep_dist.getDouble() * IntValue / 0.3048;
                                  dstnc = Double.toString(dd) + "ft";
                                  b[dummy].setdists(dstnc);
                                  System.err.println("scqwcqwceqc   "  +  dstnc  +  "     "  +  dummy  +  "        "  +
b[dummy].getdists());
                              }
                              else {
                                  dd = (sep_dist.getDouble() + FloValue) / 0.3048;
                                  dstnc = Double.toString(dd) + "ft";
                                  b[dummy].setdists(dstnc);
                                  System.err.println("scqwcqwceqc   "  +  dstnc  +  "     "  +  dummy  +  "        "  +
b[dummy].getdists());
                              }
                          }else if(((b[dummy].gettype()).toString()).equals("coord"))
                          {

                               System.out.println("Debug!! ");
                               float setdist=0.0f ;
                               int setdir=0;
                          // get first ID
                               AST ast = e2.getNextSibling();
                              String dstnc = ast.toString();
                              if(!(dict.containsKey(#ast.getText()))) kk.reportError("unrecognized", #ast.getText());
                              int dumm = (Integer)dict.get(#ast.getText());
                              System.out.println("Debug!! 1 long = "+b[dumm].gettype());
                              if(!b[dumm].gettype().equals("coord")) kk.reportError("invalid statement", "IS");
                               boolean IsPlus = false;
                               if(((ast.getNextSibling()).toString()).equals("+")) {
                              IsPlus = true;
                              }
                              else {
                              kk.reportError("invalid operator", "*");
                              }
                          // get first parameter suppose a dist ID or number
                              AST ast1 = (ast.getNextSibling()).getNextSibling();
                              System.out.println("Debug!! 1.5 type = "+ast1.getType());
                              if(ast1.getType()==ID){
                          // get a dist ID
                              dstnc = ast1.toString();
                              if(!(dict.containsKey(#ast1.getText()))) kk.reportError("unrecognized", #ast1.getText());
                              int dummdist = (Integer)dict.get(#ast1.getText());
```

```
                        System.out.println("Debug!! 2 long = "+b[dummdist].gettype());
                        if(b[dummdist].gettype()!="distance") kk.reportError("invalid statement", "IS");
                                                                        sep_dist              =              new
Distance(b[dummdist].getdists());
                                                                        setdist = (float)(sep_dist.getDouble()/ 0.3048);


                                                                        }else if(ast1.getType()==DIST){
                                                                // get a dist number
                                                                        sep_dist = new Distance(#ast1.getText());
                                                                        setdist = (float)(sep_dist.getDouble()/ 0.3048);

                                                                        }

                     // get second parameter suppose a direction
                        AST ast2 = ast1.getNextSibling();
                        System.out.println("Debug!! 2.5 type = "+ast2.getType());
                        if(ast2.getType()==ID){
                      // get a dir ID
                        dstnc = ast2.toString();
                        if(!(dict.containsKey(#ast2.getText()))) kk.reportError("unrecognized", #ast2.getText());
                        int dummdir = (Integer)dict.get(#ast2.getText());
                        System.out.println("Debug!! 3 type = "+b[dummdir].gettype());
                        if(b[dummdir].gettype()!="int") kk.reportError("invalid statement", "IS");
                                                                        setdir = b[dummdir].getIntValue();

                                                                        }else if(ast2.getType()==INTEGER){
                                                                // get a dir number
                                                                        setdir = Integer.parseInt(#ast2.getText());
                                                                        }



                       System.out.println("setdist = "+setdist);
                       b[dummy].navigate(b[dumm].getlon(),b[dumm].getlat(),setdist,setdir);
                           System.out.println("Debug!! long = "+b[dummy].getlon()+"lat = "+b[dummy].getlat());

                      }else
                      {   System.err.println("dcrv");}
                   }
}
else if(e1.getText().equals("printk"))
{
            //printk, ID

            e2 = e1.getFirstChild();
            if( !(dict.containsKey(e2.getText())))
                        kk.reportError( "unrecognized: ", e2.getText() );
   else {
                        dummy = (Integer)dict.get(e2.getText());
                        System.out.println("printk find index = "+dummy +"  type = "+b[dummy].getname());
            }

   if(b[dummy].gettype() == "distance")
                        kk.printdis(b[dummy].getdists());
   if(b[dummy].gettype() == "coord")
                  kk.printcoord(b[dummy]);
   if(b[dummy].gettype() == "point")
                  kk.printPoint(b[dummy]);
   if(b[dummy].gettype() == "place")
                  kk.printPlace(b[dummy]);
   if(b[dummy].gettype() == "struct")
           {
           if(b[dummy].getshape() == "square")
                          kk.printquadrangle(b[dummy]);
           if(b[dummy].getshape() == "rectangle")
                          kk.printquadrangle(b[dummy]);
           if(b[dummy].getshape() == "triangle") ;
   }
   if(b[dummy].gettype() == "overlay")
                  kk.printOverlay(b[dummy]);
   if(b[dummy].gettype() == "line")
                  kk.printLine(b[dummy]);
           if(b[dummy].gettype() == "read3Dfile")
                  b[dummy].read3Dfile.renderKML(kk.outp);

}
else if(e1.getText().equals("readfile"))
{
           //readfile
           b[count] = new GECLdatatypes();
           b[count].settype("read3Dfile");

           //ovs1:ID
           e2 = e1.getFirstChild();
           b[count].setname(e2.getText());
   dict.put(e2.getText(),count);

           //scale1:FLOAT
           e2 = e2.getNextSibling();
           if( !(dict.containsKey(e2.getText())))
                  b[count].read3Dfile.scale = Float.parseFloat(e2.getText());
   else {
                  dummy = (Integer)dict.get(e2.getText());
                  b[count].read3Dfile.scale = b[dummy].getFloatValue();
           }

           //translate_x1:FLOAT
           e2 = e2.getNextSibling();
           if( !(dict.containsKey(e2.getText())))
                  b[count].read3Dfile.translate_x = Float.parseFloat(e2.getText());
   else {
                  dummy = (Integer)dict.get(e2.getText());
                  b[count].read3Dfile.translate_x = b[dummy].getFloatValue();
           }

           //translate_y1:FLOAT
           e2 = e2.getNextSibling();
           if( !(dict.containsKey(e2.getText())))
```

```
                        b[count].read3Dfile.translate_y = Float.parseFloat(e2.getText());
   else {
                        dummy = (Integer)dict.get(e2.getText());
                        b[count].read3Dfile.translate_y = b[dummy].getFloatValue();
            }

            //translate_z1:FLOAT
            e2 = e2.getNextSibling();
            if( !(dict.containsKey(e2.getText())))
                        b[count].read3Dfile.translate_z = Float.parseFloat(e2.getText());
   else {
                        dummy = (Integer)dict.get(e2.getText());
                        b[count].read3Dfile.translate_z = b[dummy].getFloatValue();
            }

            //filename
            e2 = e2.getNextSibling();
            b[count].read3Dfile.openfile(e2.getText());
            count++;

}









                                                                                                    }



            recursive_f(e1.getFirstChild());
                                                                                                    e1 = (AST)
e1.getNextSibling();

            System.out.println( "inside while loop" );
                                                                                        }
                                                                                        if(if_break == 1)
                                                                                        {
                                                                                                    if_break   =
0;

                                                                                                    break;
                                                                                        }
                                                                                        if_continue = 0;
                                                                            }
                                                                }
                                                    }
                                            | skip_block_statement
                                )
                    )

     |      #( d:"do" {System.out.println( "after do" ); if_break = 0; if_skip_rcurly = 0;}
            block2:block_statement  {System.out.println( "after block statement if_break " + if_break );}
```

```
                                        "while" {System.out.println( "after while" );}

           exprReturn = exp2:expr

                                                                  {
                                                                        System.out.println( "after expr" );

                                                                        if(if_break == 1)
                                                                        {
                                                                              System.out.println(  "if_break
is 1" );
                                                                              if_break = 0;
                                                                        }
                                                                        else
                                                                        {
                                                                              while( exprReturn > 0)
                                                                              {
                                                                                    e1      =      (AST)
d.getFirstChild(); //{
                                                                                    e1      =      (AST)
e1.getNextSibling(); // first statement now
                                                                                    while(true)
                                                                                    {

      if(e1.getText().length()>0)
                                                                                          {

      if(e1.getText().equals("break"))
                                                                                                {

      if_break = 1;

      break;
                                                                                                }

      else if(e1.getText().equals("continue") || e1.getText().equals("}"))
                                                                                                {

      break;
                                                                                                }

                                                                                          }


      recursive_f(e1.getFirstChild());
                                                                                          e1 = (AST)
e1.getNextSibling();

      System.out.println( "inside while loop" );
                                                                                    }
                                                                                    if(if_break == 1)
                                                                                    {
```

```
                                                                                                    if_break   =
0;
                                                                                                    break;
                                                                                            }
                                                                                            exprReturn           =
recursive_f(exp2);

                                                                                    }
                                                                            }

                                                                    }


            )
// Jump statements:
        |       "continue" {if_skip_rcurly = 1 ;}
                                        //( skip_statement )+ RCURLY
        |       "break"             { if_break = 1 ; if_skip_rcurly = 1; System.out.println( "set if_break " + if_break);}
                                        //( skip_statement )+ RCURLY
        |       #( r:"return"          { System.out.println( r ); }
                ( exprReturn = expr )?
                                { System.out.println( ";" ); }
                )
// Selection statements:
        |       #( "if" exprReturn = expr
                                                (
                                                        {    exprReturn    >    0    }?    block_statement    (    "else"
skip_block_statement )?
                                                        |
                                                                skip_block_statement  ("else"  block_statement
)?

                                                )
                                        )
                        ;

skip_statement
        :       s:SEMI
                        |       #( NStatementExpr . )
// Iteration statements:
        |       #( "while" . skip_block_statement )
        |       #( "do" skip_block_statement .)
// Jump statements:
        |       "continue"
        |       "break"
        |       #( "return" ( . )? )
// Selection statements:
        |       #( "if" . skip_block_statement ("else" skip_block_statement )?
                                        )
                        ;

skip_block_statement
                        :           skip_statement
                        |           lc:LCURLY
                        ( skip_statement )+
                        rc:RCURLY
                ;



block_statement
{
        int tempi;
}
                :       tempi = stmt
                |       lc:LCURLY                               { tabs++; }
                        (
                        {if_skip_rcurly == 0}? tempi = stmt
                        | tempi = stmt
                        )+
                        rc:RCURLY                               { if_skip_rcurly = 0 ;tabs--; }
                ;


/*
block_statement
                :       statement
                |       lc:LCURLY                               { tabs++; }
                        (
                        {if_skip_rcurly == 0}? statement
                        | skip_statement
                        )+
                        rc:RCURLY                               { if_skip_rcurly = 0 ;tabs--; }
                        //({if_skip_rcurly == 0}? rc:RCURLY)
                        //{if_skip_rcurly = 0; tabs--; System.out.println( "finish block state" );}
                ;
                */

expr returns [float binReturn]
{
int exprReturn = 0;
binReturn = 0;
}
    :   binReturn = binaryExpr
    |    emptyExpr
    ;

emptyExpr
    :  NEmptyExpression
    ;

binaryOperator
    :     ASSIGN
    |     DIV_ASSIGN
    |     PLUS_ASSIGN
    |     MINUS_ASSIGN
    |     STAR_ASSIGN
    |     MOD_ASSIGN
```

```
	|	LOR
	|	LAND
	|	EQUAL
	|	NOT_EQUAL
	|	LT
	|	LTE
	|	GT
	|	GTE
	|	PLUS
	|	MINUS
	|	STAR
	|	DIV
	|	MOD
	|	NCommaExpr
			|			INC
			|			DEC
	;


binaryExpr returns [float binReturn]
{
		binReturn = 0;
		int entry_index1 = 0;
}
	:
					b:binaryOperator
	    // no rules allowed as roots, so here I manually get
	    // the first and second children of the binary operator
	    // and then print them out in the right order
	    {
						binReturn = recursive_f(b);
	    }
	;
```

Distance.java

```
/*
 Author:	Carlos Icaza
*/


import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class Distance{

		public Distance(String in){
			Pattern p = Pattern.compile("(\\d+(\\.\\d*)?)(km|mi|m|nm|ft)?");

			Matcher m = p.matcher(in);

			boolean match = m.matches();
```

```
                if (match){
                        num = Double.valueOf( m.group(1) ); // get number and turn into a float
                        String measure = m.group(3);

                if (measure == null) measure = "m"; //meters by defaut, to avoid compareTo null
                                //pointer exception when input has no measure
                if (measure.compareTo("km") == 0) num = num * 1000.000;
                if (measure.compareTo("mi") == 0) num = num * 1609.344;
                if (measure.compareTo("nm") == 0) num = num * 1852.000;
                if (measure.compareTo("ft") == 0) num = num * 0.3048;
                }
                else{
                 num = -1.00;}
        }

        public Double getDouble(){
                return num;

        }

        public String getString(){
                return Double.toString(num);
        }
        private Double num;
}
```

GECLdatatypes.java

```
/*
 Author:    Tz-Yang Tang
           Carlos Icaza-Estrada
 */

import wdcor.WorldCoords;


public class GECLdatatypes
{
   // member variables
   String name;   // used in hash table
   String type;
   String dists;

   float coord_longitude;
   float coord_latitude;
   String coord_elev;

   String place_title;

        String StructShape;
   float leftup_coord_lo;
   float leftup_coord_la;
   float leftdn_coord_lo;
   float leftdn_coord_la;
```

```
  float righup_coord_lo;
  float righup_coord_la;
  float righdn_coord_lo;
  float righdn_coord_la;
  float struct_hi;
  float east_lo, east_la, north_lo, north_la, west_lo, west_la, south_lo, south_la;

  String ovyurl;
  float ovyrotat;

  float[] linelon = new float[100];
  float[] linelat = new float[100];
  String[] lineelv = new String[100];
  int lineIndex;

        int int_value; //for int
        float float_value;
        String string_value;

  PerspectType minePersp = new PerspectType();

        Ply_3D read3Dfile = new Ply_3D();

// constructor
  public GECLdatatypes(){
        name="default";
               type="default";
               dists="0ft";
               coord_longitude = 0.0f;
               coord_latitude = 0.0f;
               coord_elev = "default";
               place_title = "default";
               lineIndex = 0;
               int_value = 0;
               float_value = 0.0f;
               string_value = "defualt";
        }

// member functions

  public String getname(){
        return name;
        }
  public void setname(String tempstr){
        name = tempstr;
        }

  public String gettype(){
        return type;
        }
  public void settype(String tempstr){
        type = tempstr;
        }
```

```
public String getdists(){
          return dists;
          }
public void setdists(String tempstr){
          dists = tempstr;
          }

public void setcoord(float lon,float lat,String el){
          coord_longitude = lon;
          coord_latitude = lat;
          coord_elev = el;
          }

          public float getlon(){
          return coord_longitude;
          }

          public float getlat(){
          return coord_latitude;
          }

          public String getelv(){
          return coord_elev;
          }

          public void cpy(GECLdatatypes tempgecl){
                    coord_longitude=tempgecl.getlon();
                    coord_latitude=tempgecl.getlat();
             coord_elev=tempgecl.getelv();
                    }

public void settitle(String tempstr){
          place_title = tempstr;
          }

public String gettitle(){
          return place_title;
          }

public void setshape(String tempstr){
          StructShape = tempstr;
          }

public String getshape(){
          return StructShape;
          }

 public void setsquare(float leng,float heig){

          double earthradius=3963.189 * 5280;
          float radius = (float) Math.asin((leng/2)/earthradius);
          float degree = (float) Math.toDegrees(radius);

                    leftup_coord_lo = coord_longitude -degree;
```

```
        leftup_coord_la = coord_latitude +degree;

        leftdn_coord_lo = coord_longitude -degree;
        leftdn_coord_la = coord_latitude -degree;

        righup_coord_lo = coord_longitude +degree;
        righup_coord_la = coord_latitude +degree;

        righdn_coord_lo = coord_longitude +degree;
                righdn_coord_la = coord_latitude -degree;

        struct_hi=heig;
        }

public void setrect(float leng,float width,float heig){
        // the default length is in feet
        double earthradius=3963.189  * 5280;  // 1 mile = 5280 ft
        float radiusL = (float) Math.asin((leng/2)/earthradius);
        float radiusW = (float) Math.asin((width/2)/earthradius);
        float degreeL = (float) Math.toDegrees(radiusL);
        float degreeW = (float) Math.toDegrees(radiusW);

        leftup_coord_lo = coord_longitude -degreeL;
        leftup_coord_la = coord_latitude +degreeW;

        leftdn_coord_lo = coord_longitude -degreeL;
        leftdn_coord_la = coord_latitude -degreeW;

        righup_coord_lo = coord_longitude +degreeL;
        righup_coord_la = coord_latitude +degreeW;

        righdn_coord_lo = coord_longitude +degreeL;
                righdn_coord_la = coord_latitude -degreeW;

        struct_hi=heig;
        }

        public float getlfuplo(){
        return leftup_coord_lo;
        }
        public float getlfupla(){
        return leftup_coord_la;
        }
        public float getlfdnlo(){
        return leftdn_coord_lo;
        }
        public float getlfdnla(){
        return leftdn_coord_la;
        }
        public float getrtuplo(){
        return righup_coord_lo;
        }
        public float getrtupla(){
        return righup_coord_la;
```

```
         }
         public float getrtdnlo(){
         return righdn_coord_lo;
         }
         public float getrtdnla(){
         return righdn_coord_la;
         }

         public float getstrhi(){
                  // Get structure height
         return struct_hi;
         }

 public void setoverlay(GECLdatatypes o1,GECLdatatypes o2,GECLdatatypes o3,GECLdatatypes o4){

         leftup_coord_lo = o1.getlon();
         leftup_coord_la = o1.getlat();

         righup_coord_lo = o2.getlon();
         righup_coord_la = o2.getlat();

         righdn_coord_lo = o3.getlon();
                  righdn_coord_la = o3.getlat();

         leftdn_coord_lo = o4.getlon();
         leftdn_coord_la = o4.getlat();

         }

// Set Triangle
/*public static void settriangle(double leng,int heig){

   double dir = Math.toRadians(120.00); //turn degrees into radians for use with trig functions

   double cond_factor= (69.1105 * 5280.00) ;
   double long_const = ( leng / cond_factor ) * ( Math.cos(dir) );
   double lat_const  = ( leng / cond_factor ) * ( Math.sin(dir) );
   north_lo  = (float) coord_longitude;
   north_la  = (float) (coord_latitude + ( ( leng / Math.sqrt(3.00) ) / cond_factor ));

   right_lo = (float)(north_lo + long_const);
   right_la = (float) (north_la - lat_const);

   left_lo = (float) (north_lo - long_const);
   left_la = (float) (north_la - lat_const);
   struct_hi=heig;
} */

         public void seturl(String tmp){
         ovyurl = tmp;
         }
public String geturl(){
         return ovyurl;
         }
```

```
public void setrotat(float tmp){
                        ovyrotat = tmp;
          }
public float getrotat(){
                        return ovyrotat;
          }

 public void setpersp(PerspectType tmpobj){
                        minePersp=tmpobj;
                        minePersp.SetPersp();
          }

public void insert(GECLdatatypes tmpobj){
          linelon[lineIndex] = tmpobj.getlon();
          linelat[lineIndex] = tmpobj.getlat();
          lineelv[lineIndex] = tmpobj.getelv();
          lineIndex++;
          }

public int getLineNum(){
return lineIndex;
}

public float getLineArraylon(int tmp){
return linelon[tmp];
}

public float getLineArraylat(int tmp){
return linelat[tmp];
}

public String getLineArrayelv(int tmp){
return lineelv[tmp];
}


          public void setCityCoords(WorldCoords tmp){

           coord_longitude=(float)    tmp.longitude;
coord_latitude = (float)  tmp.latitude;

}


          public void setIntValue(int tempint){
          int_value = tempint;
          }
          public int getIntValue(){
          return int_value;
          }
          public void setFloatValue(float tempfloat){
          float_value = tempfloat;
          }
          public float getFloatValue(){
```

```
		return float_value;
		}
		public void setStringValue(String tempstring){
		string_value = tempstring;
		}
		public String getStringValue(){
		return string_value;
		}
		public void navigate(float lon, float lat, float dist, int dir )
		{		//assuming distance is given in feet
       double tmp;
       tmp = Math.toRadians(dir); //turn degrees into radians for use with trig functions
       double whatever;
       whatever= 69.1105*5280; //360.0 / ( Math.PI * (3963.189 * 5280.00));
       float cond_factor = (float)whatever;
       coord_longitude = lon + ( (dist / cond_factor) * ((float) Math.cos(tmp) ));
       coord_latitude  = lat + ( (dist / cond_factor) * ((float) Math.sin(tmp) ));
   }
}
```

GECLlibrary.java

```
/* Author:  Tz-Yang Tang
	  Bhashinee Garg
 */


import wdcor.*;
import java.io.IOException;
import java.io.PrintWriter;
import java.text.NumberFormat;


import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintStream;
import java.util.InputMismatchException;
import java.lang.Math;


public class GECLlibrary{

final String outputFilename = "GECLoutput.kml"; // OUPTUT FILENAME
PrintWriter outp;

WorldCoords[] citiesByName = new WorldCoords[9117];
int cityNumber = 0;
int num = 8;

NumberFormat fmt = NumberFormat.getInstance();

// Print Point Object
		public void printPoint(GECLdatatypes geclobj)
```

```
            {

                        dist tempdist = new dist(geclobj.getelv());
                        double tempdb;
                        tempdb= tempdist.getDouble(); // covert to meter
                        tempdb= tempdb/0.3048;  // from meter convert to feet
       fmt.setMaximumFractionDigits(num);

                        outp.println("<Point>   " );
                        outp.println("<coordinates>"  +fmt.format(geclobj.getlon())+','  +fmt.format(geclobj.getlat())
+','+tempdb+"</coordinates>");
                        outp.println("</Point>   " );
                        }

// Print Distance Object
            public void printdis(String dists)
            {
                        outp.println("<Distance>   " );
                        outp.println("<coordinates>" +dists+"</coordinates>");
                        outp.println("</Distance>   " );
                        }

// Print Coordinate Object
            public void printcoord(GECLdatatypes geclobj)
            {
                        dist tempdist = new dist(geclobj.getelv());
                        double tempdb;
             fmt.setMaximumFractionDigits(num);
                        tempdb= tempdist.getDouble(); // covert to meter
                        tempdb= tempdb/0.3048;  // from meter convert to feet
                        outp.println("<coordinates>"  +fmt.format(geclobj.getlon())+','  +fmt.format(geclobj.getlat())
+','+tempdb+"</coordinates>");
                        }

// Print Square
public void printquadrangle(GECLdatatypes geclobj)
            {
       fmt.setMaximumFractionDigits(num);
               outp.println("<Placemark>");
               outp.println("<name> "+ geclobj.getname() +" </name>");
                        outp.println(" <Polygon>");
                        outp.println("<extrude>1</extrude>");
            outp.println("<altitudeMode>relativeToGround</altitudeMode>");
                        outp.println(" <outerBoundaryIs>");
                        outp.println("   <LinearRing>");
            outp.println("    <coordinates>");
            outp.println(""+fmt.format(geclobj.getlfuplo())                          +','+fmt.format(geclobj.getlfupla())
+','+geclobj.getstrhi());
            outp.println(""+fmt.format(geclobj.getlfdnlo())+','+fmt.format(geclobj.getlfdnla())
+','+geclobj.getstrhi());
            outp.println(""+fmt.format(geclobj.getrtdnlo())+','+fmt.format(geclobj.getrtdnla())
+','+geclobj.getstrhi());
```

```
            outp.println(""+fmt.format(geclobj.getrtuplo())+','+fmt.format(geclobj.getrtupla())+','+geclobj.getstrh
i());
                        outp.println(""+fmt.format(geclobj.getlfuplo())              +','+fmt.format(geclobj.getlfupla())
+','+geclobj.getstrhi());
                        outp.println("     </coordinates>");
            outp.println("  </LinearRing>");
                        outp.println(" </outerBoundaryIs>");
                        outp.println("</Polygon>");
                        outp.println("</Placemark> ");
                        }


// Print Triangle
  /*public static void printtriangle(GECLdatatypes geclobj) {

       fmt.setMaximumFractionDigits(num);
       outp.println("<Placemark>");
               outp.println("<name> Test TRIANGLE </name>");
                        outp.println(" <Polygon>");
                        outp.println("<extrude>1</extrude>");
            outp.println("<altitudeMode>relativeToGround</altitudeMode>");
                        outp.println(" <outerBoundaryIs>");
                        outp.println("   <LinearRing>");
            outp.println("    <coordinates>");
            outp.println(""+fmt.format(west_lo)+','+fmt.format(west_la)+','+struct_hi);
            outp.println(""+fmt.format(east_lo)+','+fmt.format(east_la)+','+struct_hi);
       outp.println(""+fmt.format(north_lo)+','+fmt.format(north_la)+','+struct_hi);
            outp.println(""+fmt.format(west_lo)+','+fmt.format(west_la)+','+struct_hi);

                        outp.println(" </coordinates>");
            outp.println("  </LinearRing>");
                        outp.println(" </outerBoundaryIs>");
                        outp.println("</Polygon>");
                        outp.println("</Placemark> ");
    }*/


// Print Overlay
    public void printOverlay(GECLdatatypes geclobj)
            {
       fmt.setMaximumFractionDigits(num);
               outp.println("<GroundOverlay>");
               outp.println("<name> "+ geclobj.getname() +" </name>");
               outp.println("<visibility>1</visibility>");

                        outp.println("<Icon>   " );
                        outp.println("       <href>" +geclobj.geturl()+"</href>");
                        outp.println("</Icon>   " );
                        outp.println("<LatLonBox>");
                        outp.println("       <north>"+fmt.format(geclobj.getlfupla())+"</north>");
            outp.println("       <south>"+fmt.format(geclobj.getrtdnla())+"</south>");
            outp.println("       <east>"+fmt.format(geclobj.getrtdnlo())+"</east>");
            outp.println("       <west>"+fmt.format(geclobj.getlfuplo())+"</west>");
```

```
                        outp.println("</LatLonBox>");
                        outp.println("</GroundOverlay>");
                        }

// Print Line
    public void printLine(GECLdatatypes geclobj)
            {
                        int SizeOfLine;
      fmt.setMaximumFractionDigits(num);
                        outp.println("<Placemark>");
                outp.println("<name> "+ geclobj.getname() +" </name>");
                        outp.println("<visibility>1</visibility>");
                        outp.println(" <open>0</open>");
                        outp.println("<Style>");
            outp.println("<LineStyle>");
            outp.println("<color>ff00ffff</color>");
            outp.println("<width>2</width>");
            outp.println("</LineStyle>");
                        outp.println("</Style>");
                        outp.println("<LineString>");
            outp.println("       <extrude>1</extrude>");
            outp.println("       <tessellate>1</tessellate>");
            outp.println("       <altitudeMode>absolute</altitudeMode>");
            outp.println("    <coordinates>");
            SizeOfLine = geclobj.getLineNum();
            for (int i = 0 ; i < SizeOfLine;i++){
            dist tempdist = new dist(geclobj.getLineArrayelv(i));
                        double tempdb;
                        tempdb= tempdist.getDouble(); // covert to meter
                        tempdb= tempdb/0.3048;  // from meter convert to feet
            outp.println(""+fmt.format(geclobj.getLineArraylon(i))     +','+fmt.format(geclobj.getLineArraylat(i))
+','+tempdb);
                        }
                        outp.println("     </coordinates>");
                        outp.println("</LineString>");
                        outp.println("</Placemark> ");
                        }

// Print Placemark --> not complete!!
    public void printPlace(GECLdatatypes geclobj)
            {
                        dist tempdist = new dist(geclobj.getelv());
                        double tempdb;
      fmt.setMaximumFractionDigits(num);
                        tempdb= tempdist.getDouble(); // covert to meter
                        tempdb= tempdb/0.3048;  // from meter convert to feet
                outp.println("<Placemark>");
                outp.println("<name> "+ geclobj.gettitle() +" </name>");
                        if(geclobj.minePersp.BeenSet())
                        {
                                    outp.println("       <LookAt> ");
                        outp.println("
            <longitude>"+fmt.format(geclobj.minePersp.getlon())+"</longitude>");
```

```
                    outp.println("
          <latitude>"+fmt.format(geclobj.minePersp.getlat())+"</latitude>");
                    outp.println("
          <range>"+fmt.format(geclobj.minePersp.getrange())+"</range>");
                    outp.println("                    <tilt>"+geclobj.minePersp.gettilt()+"</tilt>");
                    outp.println("                    <heading>"+geclobj.minePersp.getheading()+"</heading>");
                              outp.println("       </LookAt>");
                              }
                    outp.println("       <Point>   " );
                    outp.println("                    <coordinates>"                          +fmt.format(geclobj.getlon())+','
+fmt.format(geclobj.getlat()) +','+tempdb+"</coordinates>");
                    outp.println("       </Point>   " );
                    outp.println("</Placemark> ");
                    }

// Print kml file header
          public void printhr()
          {
                    outp.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
                    outp.println("<kml xmlns=\"http://earth.google.com/kml/2.0\">\n");

          // Add temprory
                    outp.println("<Folder>");
          outp.println("<description>Showing-off GECL</description>");
          outp.println("<name>GECL Output</name>");
          outp.println("<open>0</open>");

                    }

// Print kml file tail
          public void printtl()
          {
                    // Add temprory
                    outp.println("</Folder>");
                    outp.append("</kml>\n\n");
                    }

// Open kml file PrintWriter file description and print header
          public void creatkml()
                    {

                    System.out.print("Trying to create KML script file...\n");
                    try{
                              PrintWriter out = new PrintWriter(outputFilename);
                              outp = out;
                              printhr();
                    }
                    catch (IOException exception)
                    {
                              System.out.println("Error processing file:   " + exception);
                    }


          }
```

```
// Close kml file PrintWriter file description and print tail
        public void endkml()
                {
                printtl();
                outp.close();
                }


        public void sortCityCoords(){


                WorldCoords[] Cities = null;
                String needless;

                //try-catch block if the file could not be found
                try{
                                File inputFile = new File("worldcities.txt");
                                Scanner scanFile = new Scanner(inputFile);
                                cityNumber = scanFile.nextInt();

                                 Cities = new WorldCoords[cityNumber];

                                 for(int i = 0; i < cityNumber; i++){

                                         needless = scanFile.nextLine();
                                         String cityState = scanFile.nextLine();
                                         double longi = scanFile.nextDouble();
                                         double lati = scanFile.nextDouble();

                                         Cities[i] = new WorldCoords(longi, lati, cityState);
                                 }

         //have the other 2 arrays' references set to the original
                                 for(int j = 0; j < cityNumber; j++){
                                         citiesByName[j] = Cities[j];
                                 }

                                 //now we sort the citiesByName lexicographically
                                 WorldCoordsComparator NameCmp = new CmpByName();
                                 SortingAlgorithms.mySortingAlgorithm(citiesByName, NameCmp);
                }

                catch(FileNotFoundException FNFE){
                        System.out.println("File not found.");

                }
 }


  public WorldCoords getCityCoords(String City, String Country){

         //clean possible data remains from last time
                                WorldCoords result1 = null, result2 = null;
                                String city1, state1, fileName, c_s, cityForSearch;
                                int resultNum = 1;
                                result1 = null;
                                result2 = null;

                                city1 = City;
                                state1 = Country;
                                c_s = city1 + state1;

                                //now make the city and state into a form that facilitates the searching,
                                //namely making it into the same form as the String name in WorldCoords
                                StringBuilder tmp = new StringBuilder("");
                           char c;
                           for(int i=0, j=0; i< c_s.length(); i++){
                             c = Character.toUpperCase(c_s.charAt(i));
                             if( c >= 'A' && c <= 'Z' ) tmp.append(c);
                           }
                           cityForSearch = tmp.toString();

                                //now we perform a binary search on citiesByName and put the results in
                           //result1 and result2
                                int low = 0;
                                int high = cityNumber - 1;
                                int mid = 0;

                                while(low <= high){
                                        mid = (low + high) / 2;
                                        if(cityForSearch.compareTo(citiesByName[mid].name) < 0)
                                                high = mid - 1;
                                        else if(cityForSearch.compareTo(citiesByName[mid].name) > 0)
                                                low = mid + 1;
                                        else
                                                break;
                                }
                         System.out.println("found city " + citiesByName[mid]);

                         return citiesByName[mid];
  }

  public void reportError(String msg, String id)
  {
    System.err.println(msg + ": " + id);
    System.exit(0);
  }
}
```

GECLrunner.java

```
import java.io.*;

import antlr.CommonAST;
import antlr.collections.AST;
import antlr.debug.misc.ASTFrame;
import antlr.DumpASTVisitor;
import antlr.Token;

public class GECLrunner
{
   public static void main(String[] args)
   {
      for (int i=0; i<args.length; i++)
      {
                        try
                        {
                                String programName = args[i];
                                DataInputStream dis = null;
                                if (programName.equals("-")) {
                                        dis = new DataInputStream( System.in );
                                }
                                else {
                                        dis          =          new          DataInputStream(new
FileInputStream(programName));
                                }
                                GECLLexer lexer = new GECLLexer ( dis );
                                GECLParser parser = new GECLParser(lexer);

                                parser.program();

                                CommonAST tree = (CommonAST)parser.getAST();
                                System.out.println(tree.toStringTree());

                                GECLWalker walker = new GECLWalker();
                                walker.program(tree);

                                // Open a window in which the AST is displayed graphically
                                ASTFrame frame = new ASTFrame("AST from the Simp parser", tree);
                                frame.setVisible(true);
                }
                        catch (Exception e)
                        {
                                System.err.println ( "exception: " + e);
                                e.printStackTrace();
                        }

      }
   }
}
```

PerspectType.java

```
import java.io.*;
import antlr.CommonAST;
import antlr.collections.AST;
import antlr.debug.misc.ASTFrame;
import antlr.DumpASTVisitor;
import antlr.Token;

public class GECLrunner
```

```
{
   public static void main(String[] args)
   {
      for (int i=0; i<args.length; i++)
      {
                          try
                          {
                                    String programName = args[i];
                                    DataInputStream dis = null;
                                    if (programName.equals("-")) {
                                              dis = new DataInputStream( System.in );
                                    }
                                    else {
                                              dis              =              new              DataInputStream(new
FileInputStream(programName));
                                    }
                                    GECLLexer lexer = new GECLLexer ( dis );
                                    GECLParser parser = new GECLParser(lexer);

                                    parser.program();

                                    CommonAST tree = (CommonAST)parser.getAST();
                                    System.out.println(tree.toStringTree());

                                    GECLWalker walker = new GECLWalker();
                                    walker.program(tree);

                                    // Open a window in which the AST is displayed graphically
                                    ASTFrame frame = new ASTFrame("AST from the Simp parser", tree);
                                    frame.setVisible(true);
                }
                          catch (Exception e)
                          {
                                    System.err.println ( "exception: " + e);
                                    e.printStackTrace();
                          }

      }
   }
}
```

Ply_3D.java

```
/* Author:  WeiChung Hsu
 */

import java.io.*;

public class Ply_3D
{
          String filename;
   float[] myX;
   float[] myY;
          float[] myZ;
```

```
            int[][] myPly;

            double maxX=0;
            double maxY=0;
            double maxZ=0;
            double minX=0;
            double minY=0;
            double minZ=0;
            double dif_L=0;
            double dif_W=0;
            double dif_H=0;
            double scale = 100;

            double translate_x = 121.32856;
            double translate_y = 25.01259;
            double translate_z = 100.0;

            int total_v;
            int total_ply;
            int state = 0;
            int reading_v=0;
            int reading_ply=0;
            int list_num=0;


            //final String outputFilename = "test.kml"; // OUPTUT FILENAME
            //PrintWriter outp;


// constructor
   public Ply_3D(){
            }
            public void translateTo(double x, double y, double z){
                        translate_x = x;
                        translate_y = y;
                        translate_z = z;
            }
            public void scaleTo(double s){
                        scale = s;
            }

            public void openfile(String tempstr){
                        String record = null;
                        String tokens[] = new String[30];
        int recCount = 0;
                        filename = tempstr;
                        double earthradius=3963.189  * 5280;  // 1 mile = 5280 ft
                        float radiusL;
                        float radiusW;
                        float degreeL;
                        float degreeW;

                        try {
                                    FileReader fr     = new FileReader(tempstr);
```

```
BufferedReader br = new BufferedReader(fr);
record = new String();
while ((record = br.readLine()) != null) {
        tokens = record.split(" ");
        if(tokens[0].equals("element"))
        {
                //System.out.println("hit element");
                if(tokens[1].equals("vertex"))
                {
                        total_v = Integer.parseInt(tokens[2]);
                        myX =  new float[total_v];
                        myY =  new float[total_v];
                        myZ =  new float[total_v];

                        //System.out.println("hit vertex " + total_v);
                }
                else
                {
                        total_ply = Integer.parseInt(tokens[2]);
                        myPly = new int[total_ply][3];
                        //System.out.println("hit vertex " + total_ply);
                }
        }
        else if(tokens[0].equals("end_header"))
        {
                state = 1;
                reading_v=0;
        }
        else if(state == 1)
        {
                myX[reading_v] = Float.parseFloat(tokens[0]);
                myY[reading_v] = Float.parseFloat(tokens[1]);
                myZ[reading_v] = Float.parseFloat(tokens[2]);


                if(maxX < myX[reading_v])
                        maxX = myX[reading_v];
                if(maxY < myY[reading_v])
                        maxY = myY[reading_v];
                if(maxZ < myZ[reading_v])
                        maxZ = myZ[reading_v];
                if(minX > myX[reading_v])
                        minX = myX[reading_v];
                if(minY > myY[reading_v])
                        minY = myY[reading_v];
                if(minZ > myZ[reading_v])
                        minZ = myZ[reading_v];


                reading_v++;
                if(reading_v == total_v)
                {
                        state = 2;
                        reading_ply = 0;
```

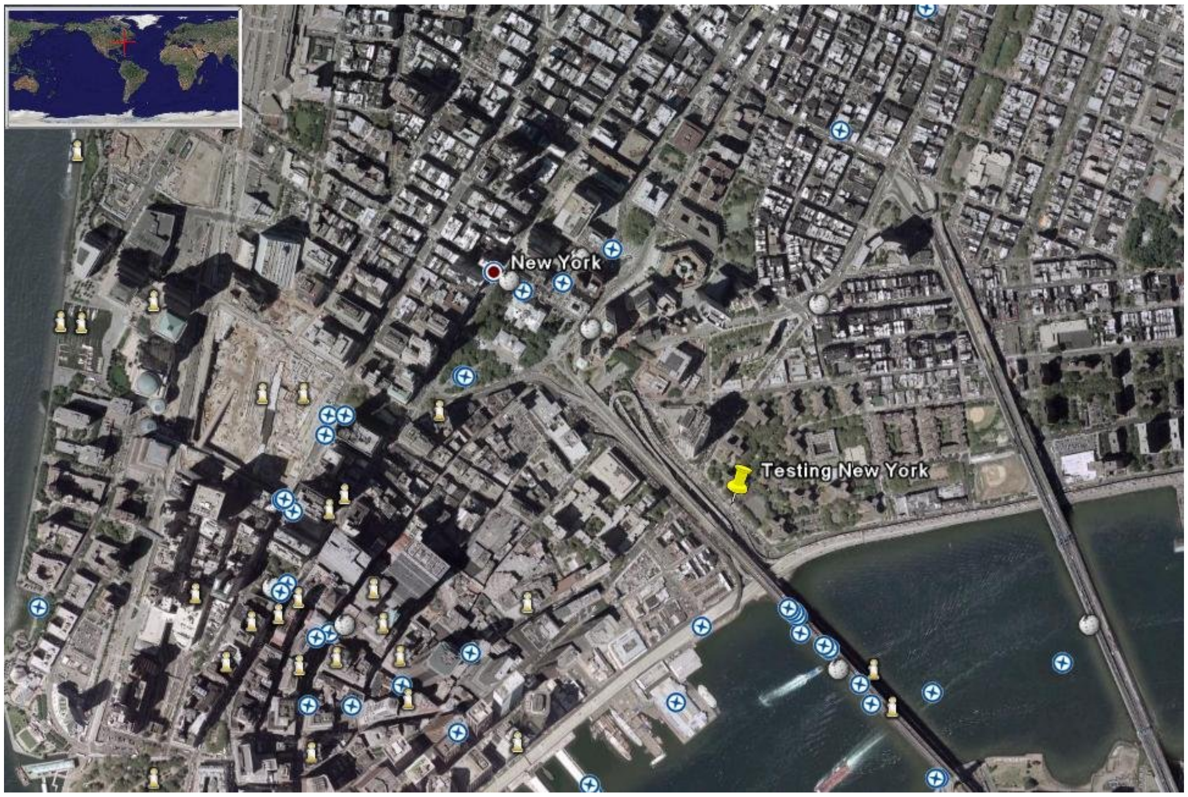
```
                                        }
                                }
                                else if(state == 2)
                                {
                                        list_num = Integer.parseInt(tokens[0]);
                                        myPly[reading_ply][0] = Integer.parseInt(tokens[1]);
                                        myPly[reading_ply][1] = Integer.parseInt(tokens[2]);
                                        myPly[reading_ply][2] = Integer.parseInt(tokens[3]);
                                        reading_ply++;
                                        if(reading_ply==total_ply)
                                                state = 3;
                                }
  }

                        dif_L=maxX - minX;
                        dif_W=maxY - minY;
                        dif_H=maxZ - minZ;
                        int i;
                        for(i=0;i<total_v;i++)
                        {

                                myX[i] = ((myX[i]-(float)minX)*(float)scale)/((float)dif_L);
                                myY[i] = ((myY[i]-(float)minY)*(float)scale)/((float)dif_W);
                                myZ[i] = ((myZ[i]-(float)minZ)*(float)scale)/((float)dif_H);



                                        radiusL = (float) Math.asin((myX[i]/2)/earthradius);
                                        radiusW = (float) Math.asin((myY[i]/2)/earthradius);
                                        degreeL = (float) Math.toDegrees(radiusL);
                                        degreeW = (float) Math.toDegrees(radiusW);
                                        myX[i] = degreeL;
                                        myY[i] = degreeW;


                        }




} catch (IOException e) {
  // catch possible io errors from readLine()
  System.out.println("Uh oh, got an IOException error!");
  e.printStackTrace();
}
    }




    public void renderKML(PrintWriter outp){
    int i;
                        for(i=0;i<total_ply;i++)     //                  for(i=0;i<10;i++)
                {
```

```
                outp.println("<Placemark>");
           outp.println("<name> a </name>");

                outp.println(" <Polygon>");
                outp.println("<extrude>0</extrude>");
        outp.println("<altitudeMode>relativeToGround</altitudeMode>");
                outp.println(" <outerBoundaryIs>");
                outp.println("   <LinearRing>");

                        outp.println("    <coordinates>");
                        outp.println(""+                    (myX[myPly[i][0]]+translate_x)              +','+
(myY[myPly[i][0]]+translate_y)  +','+ (myZ[myPly[i][0]]+translate_z));
                        outp.println(""+                    (myX[myPly[i][1]]+translate_x)              +','+
(myY[myPly[i][1]]+translate_y)  +','+ (myZ[myPly[i][1]]+translate_z));
                        outp.println(""+                    (myX[myPly[i][2]]+translate_x)              +','+
(myY[myPly[i][2]]+translate_y)  +','+ (myZ[myPly[i][2]]+translate_z));
                        outp.println(""+                    (myX[myPly[i][0]]+translate_x)              +','+
(myY[myPly[i][0]]+translate_y)  +','+ (myZ[myPly[i][0]]+translate_z));
                        outp.println("     </coordinates>");
                        outp.println("  </LinearRing>");
                outp.println(" </outerBoundaryIs>");
                outp.println("</Polygon>");
                outp.println("</Placemark> ");

                }
        }
```

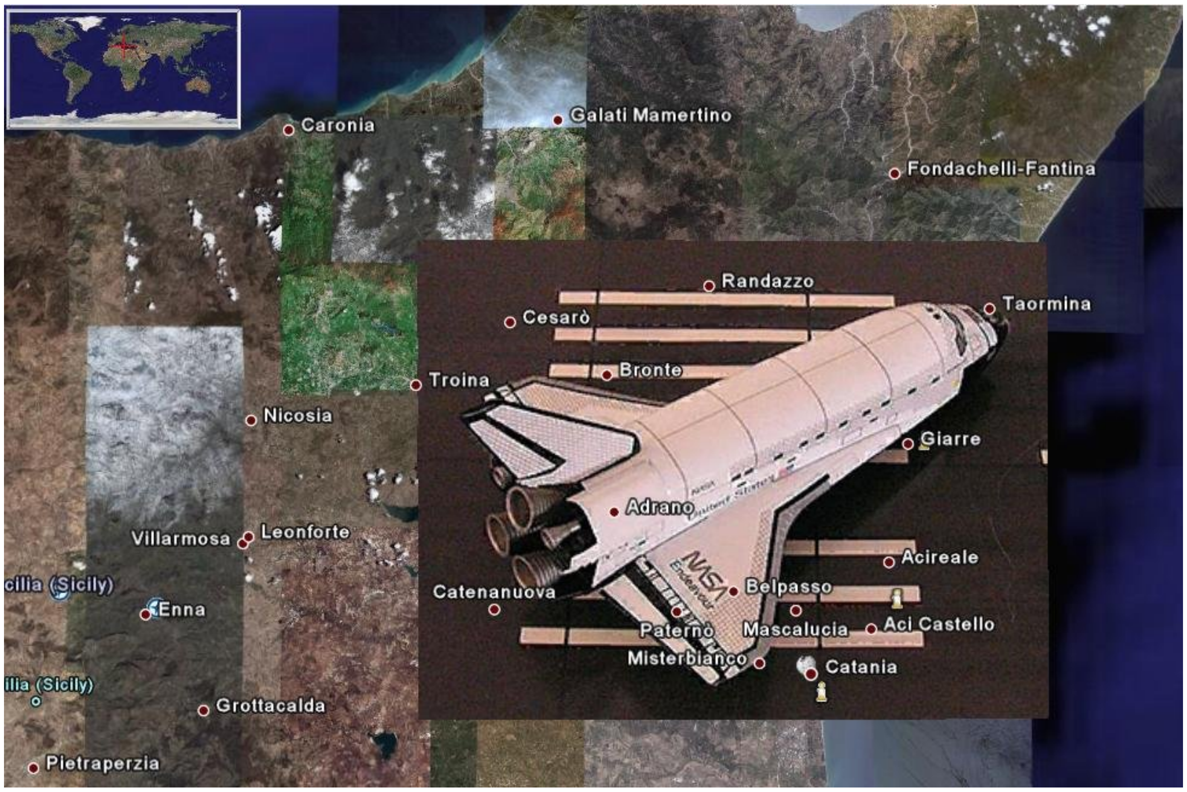
Appendix S

Screen Shots

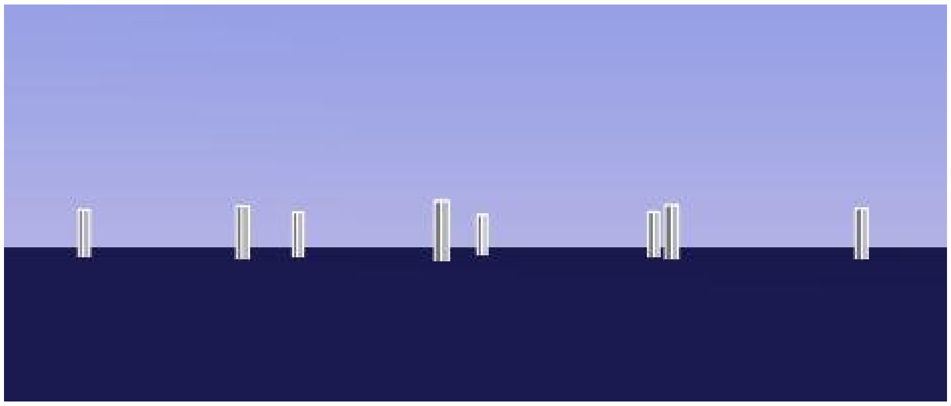
For example code 2.2:



For example code 2.3:



For example code 2.4:



Horizontal view

Another nifty feature