Geo

A geometric solution language

Project Report

Group Members
PM:
Qi Wang (qw2197)
Language Guru:
Yuechen Zhao (yz2877)
System Architect:
Zichen Chao (zc2321)
Tester:
Ziyi Luo (zl2471)

(COMS W4115) Programming Languages and Translators
Fall 2015
Dec. 22, 2015

Contents

Chapter 1 – Introduction

Geo is a geometric solution language that enables students, physicists, mathematicians, as well as other geometry-interested professionals to solve spatial problems efficiently. It involves functionalities that computes relationships among geometric figures, including lines, circles, and rectangles, etc. Users are able to define figures, set their moving patterns, and perform various analysis (including static and dynamic analyses) of the interacting figures. If desired, a user can also change the moving pattern or shape of objects.

1.1 Motivation

Many complex problems can be solved with graph beautifully, however, most standard programming languages do not have convenient tools for creating and manipulating graphs, especially dynamic graphs. So we decide to design a simple while powerful graph oriented programming language, Geo. It provides built-in support for graph, so users can define graphs in an intuitive way. In addition, the most important part about Geo is that graphs can move and interact, allowing dynamic analysis with a user-friendly interface.

1.2 Key Features

1.2.1 Geometric Specific

We provide an intuitive way for declaring graphs, defining relationships and make calculations. Users can declare a line by specifying the start point and end point. Users can also calculate the intersecting point of a line and a circle by a single command.

1.2.2 Dynamic Movement

Users can define moving objects in Geo, and see the animations directly through our panel interface. Dynamic analysis can be done easily with run statement, which will be explained in detail in the Language Tutorial part.

1.2.3 Complete Programming Functionality

Geo also has complete functionality as a normal programming language, simple algorithms such as Fibonacci number can be expressed in Geo in a recursive or iterative fashion; more complicated algorithms such as Dijkstra's shortest path algorithm can also be expressed.

Chapter 2 - Language Tutorial

2.1 Basic Structure of a Geo Program

The hello world example of Geo is provided below, which contains a simplest function which takes in no arguments and only executes print. Then we simply call function hello() under the definition of the function hello().

```
// hello word
@panel helloworld
function hello(): void:
  print("Hello World!");
end
hello();
@end
```

2.2 Built-in Data Types and Variable Assignment

Geo supports the following primitive types: int, float, bool, char and string. The code below illustrates on how to assign values for built-in data types. One thing to notice is that we do not need to specify the types of variables because Geo knows types implicitly.

```
x = 10;
c = 'a';
s = "hello";
flag = true;
```

2.3 Control Structures

The code below shoes the usage of if-else statement, for loop and while loop. It should print -1, h, e, l, l, o, 0 successively.

```
a = 0;
if(a>0):
    a = a + 1;
else:
    a = a - 1;
end
print(a);

for i in "hello":
    print(i);
end

while(a<0):
    a = a + 1;
    print(a);
end
```

2.4 Writing and Calling a Function

The function declaration consists of the function keyword, function name, input arguments and their types, and finally a return type followed by a colon. A Geo program has no entry function, so main() is not necessary. The function body is all the statements above the end keyword. Below is an example of a recursive gcd algorithm that illustrates the usage of function.

```
function gcd(a:int, b:int): int:
    while(a != b):
        if(a>b):
            a = a - b;
        else:
            b = b - a;
        end
    end
    return a;
end
```

2.5 Advanced Date Types

For all primitive types, Geo has a corresponding list type. To declare an list, the elements of the list are explicitly written in "{ }" on the left side, and it has "append" and "pop" method. The following example illustrates the usage.

```
a = {0, 0, 0};
a.append(1);
a.pop();
```

2.6 Geometric data types

Geo supports geometric types including dot, line, polygons and circle. For geometric type, the declaration is quite different from the basic type variable declaration. For example, the declaration and initialization of a dot is shown as follow:

```
dot1 = [1.5,4.0];
dot2 = [3.0,4.0];
line1 = line(dot1, dot2);
```

A dot has a x coordinate value and a y coordinate value which should be defined simultaneously, and a line is defined by two points. More on this are in chapter 3.

2.7 Defining graphs

A typical Geo program of graphs includes:
panel presets;
`function` declaration and definition;
geometric shape declaration and initialization;
`runset` declaration and initialization;
`run` statement description.

An example follows the above structure description is shown:

```
//panel presets
@panel panel_demo

//function declaration and definition
function print_dot_list(dots:list):void:
    for d in dots:
        print(d.getX(),' ',d.getY(),'\n');
    end
end

//geometric shape declaration and initialization
line1 = line(2.0,3.0);
circle1 = circle([3,4], 5);

//runset declaration and initialization
line1.setRunstep(-0.5,'a');
circle1.setRunstep(0.1,'b');
rs = runset(50);
rs.addPara(line1, 'a');
rs.addPara(circle1, 'b');

//run statement description
run rs:
set = line1.intersect(circle1);
if (len(set)>0)
    print_dot_list(set);
end

@end
```

This example defines a function called print_dot_list to print every dot’s coordinates in a dot set using for loop. Then it declares a line and a circle, and they are set to move dynamically through setRunstep which lets objects move one step per second.

Chapter 3 - Language Reference Manual

3.1 Lexical Elements

This section specifies the lexical elements of Geo programming language.

3.1.1 Seperators

Geo compiler regards the following char as the characters to separate tokens:

```
' '
'\t'
'\n'
```

At least one of these characters is required to separate otherwise adjacent identifiers, constants, and certain operator-pairs. Arbitrary combination of such characters to separate tokens is permitted for the Geo compiler will automatically ignored such characters when analyzing the program.

3.1.2 Comments

There are two kinds of comments in Geo:
Multiple-line comment: all the text in /* comments */ is ignored (as in C and Java):

```
/* text */
```

Single-line comment: all the text after // to the end of the line is ignored (as in Java):

```
// text
```

3.1.3 Identifiers

An identifier consists of a sequence of letters, digits and '_', where the first character must be a letter. Identifiers are case-sensitive in Geo.
The following identifiers are legal:

```
abc
a7G
Foo
a_b
```

The following identifiers are illegal:

```
 7a
_a
```

3.1.4 Keywords

The following identifiers are reserved as the keywords and cannot be used as identifiers.

```
bool
break
char
circle
const
else
end
float
for
function
if
in
int
line
list
polygon
return
run
string
void
while
```

3.1.5 Constants

3.1.5.1 Integer Constants

An integer constant is a sequence of digits starts with an optional negative sign.

```
0
1
20
210
-15
```

3.1.5.2 Float constants

A float constant consists of an integer part, a decimal part, a fraction part, an e, and an optionally signed integer exponent. The integer and fraction parts both consist of a sequence of digits. Either the integer part, or the fraction part (not both) may be missing; either the decimal point or the e and the exponent (not both) may be missing.

```
1.0
.5
```

```
2e3
5e-5
2.5e+10
```

3.1.5.3 String Constants:
A string constant consists of a sequence of characters enclosed in double quotes.

```
"Hello world!"
"1234567890"
"Name\tID\tScore\n"
```

3.1.5.4 Mathematical Constants
Mathematical constants include some frequently used parameters such as:

```
PI = 3.141592653589793;
```

3.1.6 Separators

Several characters are used as the separators:

```
( ) { } [ ] ; , .
```

Note that `';'` denotes the end of a sentence. `'.'` is the access operator.

3.1.7 Operators

Following characters are regarded as the operators by Geo compiler:

Operator	Usage	Associativity
=	Assignment	Right
==	Equal to	-
!=	Unequal to	-
>	Greater	-
>=	Greater or equal to	-
<	Less	-
<=	Less or equal to	-

&	Logic AND	-
|	Logic OR	-
!	Logic NOT	Right
+	Addition	Left
-	Subtraction	Left
*	Multiplication	Left
/	Division	Left
.	Access	Left
^	Exponentiation	Left
%	Modulo	Left

The precedence of operators is as follows:

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

3.2 Data Types

This chapter introduces all standard data types in Geo.

3.2.1 Basic Types

Basic Types includes `int`, `float`, `bool`, `char` and `string`.
For integer type `int`, the data ranges from -2147483648 to 2147483648.
For floating point type `float`, the data ranges from about -3.4E+38 to +3.4E+38.
For boolean type bool, it has two optional value `true` and `false`.
For character type `char`, the data ranges from `NUL` to `DEL`, that is, from 0 to 127.
For character string type `string`, the data consists of zero or more characters enclosed in

double quotes. `string` is a powerful data type with several build-in functions in Geo STD library. Please refer to Geo STD library in 3.6 for more information about `string`.

3.2.2 Geometric Types

Geometric types includes geometric control type `runset` and geometric shapes include `dot`, `line`, `polygon` and `circle`.

3.2.2.1 `runset`

Geometric control type `runset` is a data type to be recognized by the keyword `run`. In Geo STD library, runset is defined as:

```
runset: runset(times_of_run:int, optional time_gap:float);
```

`times_of_run` denotes the times that the run statement will execute. time_gap is an optional input which controls the time gap (in second) of each run. For more build-in functions of `runset`, please refer to Geo STD library in 3.6.

3.2.2.2 `dot`

Geometric shape `dot` represents a dot in panel. In Geo STD library, `dot` is defined as:

```
dot: dot(x:float, y:float);
```

`dot` is a very special geometric type. The instantiation of a `dot` is quite different from other geometric shapes. Please refer to Section 4.1 for more information. For more build-in functions of `dot`, please refer to Geo STD library in 3.6.

3.2.2.3 `line`

Geometric shape `line` represents a line (either finite or infinite long) in panel. In Geo STD library, `line` is defined as:

```
line(a:float,b:float, optional endpointx1:float, optional endpointx2:float);
line(d1:dot,d2:dot, optional endpointx1:float, optional endpointx2:float);
```

Any of the above four definitions can be used to initialize a `line` object. `endpointx1` and `endpointx2` are optional parameter that indicate the x-coordinates of two endpoints of a `line`. If `endpointx1` and `endpointx2` are used, the `line` object is set as a segment. For more build-in functions of `line`, please refer to Geo STD library in 3.6.

3.2.2.4 `polygon`

Geometric shape `polygon` is a set of different shapes. Polygons is defined as:

```
polygons polygon(apex:list);
```

Note that apex[] is a `list` of `dots`. For `list`, please refer to Section 3.3. Note that the apexes in the list should be in a reasonable order (i.e., either clockwise or counterclockwise). For more build-in functions of `polygons`, please refer to Geo STD library in 3.6.

3.2.2.5 `circle`

Geometric shape `circle` represents a circle in panel. In Geo STD library, `circle` is defined as:

```
circle: circle(center:dot,radius:float);
```

A circle is defined with a center coordinate and a radius. For more build-in functions of `circle`, please refer to Geo STD library in 3.6.

3.2.3 list

`list` is a set of a certain type of objects. A `list` contains a number of variables. The number of variables may be zero, in which case the `list` is said to be empty.
`A list` named `listdemo` is defined as:

```
listdemo={t1:tp1,t2:tp2, … , t_length_of_list:tpn};
```

Syntax to get the nth element in listdemo is simple: `listdemo#[n-1];`
For more build-in functions of `list`. For all lists, Geo offers several build-in functions shown as bellow:

```
function append(:int):void;
function pop():void
function len(a:list):int;
```

3.3 Expressions

3.3.1 Variables

Variable consists of two parts: name and value. Name of a variable must be a legal identifier mentioned in Section 2.2. value of a variable depends on the data type of the variable.

3.3.1.1 Basic type variable declaration and initialization

For basic types, the variable declaration and initialization is quite straightforward:

```
//Initialize an int named i with value equals 2
i = 2;
//Initialize a float named afloat with value equals 3e-5
afloat = 3e-5;
//Initialize a bool named judge with value equals true
judge = true;
//Initialize a char named c with value equals 'c'
c = 'c';
//Initialize a string named str with value equals "str"
str = "str";
```

3.3.1.2 Geometric type variable declaration and initialization

For geometric type, the declaration is quite different from the basic type variable declaration. To declare a dot, the declaration and initialization are shown as follow:

```
dot1 = [1.5,2.2];
```

A dot is declared with two float parameters within a pair of square brackets. For other geometric types, the declaration and initialization examples are shown as follow:

```
//Declare a line y=3x+2 with x in [0,5.5].
line1 = line(3.0,2.0,0,5.5);
//Declare a circle (x-5)^2+(y-10)^2 = 6.2^2
circle1 = circle([5,10],6.2);
//Declare a pentagon with apex [0,0][2,0][2,2][1,5][0,3]
apex = {[0,0],[2,0],[2,2],[1,5],[0,3]};
pentagon = polygon(5,apex);
```

3.3.2 Presets

Presets occur at the beginning of the Geo program (except `@end`). Basic analyzing environment is set in presets. Each preset begins with an `@` sign.

`@panel panelname` (essential) defines a panel with a legal identifier `panelname`.

`@end` (essential) indicates the boundary of a specific panel.

3.4 Functions

This chapter introduces functions in Geo. Functions are the key of extending the usage of models and solving geometric problems. Geo provides a powerful STD library for geometric problem solution while most of the methods in STD library are encapsulation in functions.

3.4.1 Function Declarations

A function is declared as the following rule:

```
function func_name(parameter:parameter_type,...):return_type:
    statements;
 end
```

Here function is the keyword for function declaration. `func_name` is the identifier for this function. input parameters must be in the form of `identifier:data_type`. return_type is the type of the return value. If the function does not have a return value, `:return_type` part should be `:void`.

3.4.2 Function Definitions

A typical function definition is shown as below. Note that `return value(s);` is not needed when the function do not have a return value(s) (i.e., `:return_type` is defined as `:void`);

```
function func_name(para:para_type const,...):return_val_type:
  local_variable declaration and initialization;
  function operations;
  return value(s);
end
```

Note that, local variables are available only in the domain of a function. All local variables will be terminated as the function ends. All parameters of functions are value parameters in default,

3.4.3 Calling Functions

A function is called as the following rule:

```
val = func_name(para1,para2, ...);
```

As for the functions in models. The function is called as:

```
val = model_instant_identifier.func_name(para1,para2, ...);  |
```

3.5 Compound Statements

3.5.1 The `if` Statement

A if-else control flow follows the following rule:

```
if (condition1):
  statement1;
else:
   if (condition2):
      statement2;
    else:
      statement3;
    end
end
```

if, else and end are keywords for the control flow. Condition 1 through condition 2 defines conditions to enter the if/else cases, and statements 1 through 3 defines the statements to execute in three corresponding cases.

3.5.2 The `while` Statement

A while loop follows the following rule:

```
while (condition):
  statement;
end
```

while and end are keywords for the while-loop statement. The condition defines the condition when one continues entering the while-loop. The statement defines the code to execute inside the loop.

3.5.3 The `for` Statement

A for loop follows the following rule:

```
for element in alist:
  statement;
end
```

for and in are keywords in the for-loop statement. The element denotes an element in the set, and after the execution of a for-loop statement, all elements in set should be iterated exactly once. The statement denotes the statement to execute when one enters the loop.

3.5.4 The `run` Statement

`run` is a keyword for dynamic geometric analytics. Each run must correspond to an instant of `runset`. Each geometric shape that will be dynamically analyzed in the `run` statement must be added into the runset instant at first. A typical run statement is shown as follows:

```
run runset_name:
  dynamic analytics sentences;
  change parameters for the next time run;
end
```

3.6 STD Library Reference

This chapter introduces the STD library of Geo. STD library (`std.glib`) is a file preloaded by Geo compiler before compiling the programs. STD library includes system functions declaration, geometric types (including build-in functions) declaration.
Part A: System constant declaration.

```
PI = 3.141592653;
```

Part B: System function declaration.

```
function print(info:int):void;
function print(info:float):void;
function print(info:char):void;
function print(info:bool):void;
function print(info:string):void;
```

Part C: Geometric types declaration.

```
//controltype runset
runset:
  runset(times_of_run:int, optional time_gap:float);
  function addPara(g:geometricShape, para:char):bool;
  function removePara(g:geometricShape,para:char):bool;
  function enableRun():void;
  function disableRun():void;
  function mark(d:dot):void;

end


//geometricShape dot
//dot parameter name: 'x' 'y'
dot:
  dot(x:float, y:float);

  function getX():float;
  function getY():float;

  function distance(dot1:dot):float;
end

//geometricShape line
//line parameter name: 'a' 'b'
//line formula y = ax + b
line:
  line(a:float,b:float, optional endpointx1:float, optional
  endpointx2:float);
  line(d1:dot,d2:dot, optional endpointx1:float, optional endpointx2:float);

   function getPara(pos:char):float;
   function distance(d:dot):float;
   function getPara(para:char):float;
   function setPara(para:char, v:float):void;
   function getRunstep(para:char):float;
   function setRunstep(para:char, v:float):void;
   function getY(x:float):float;
   function getX(y:float):float;
   function contains(d:dot):bool;
   function getMidpoint():dot;
   function getEndpoints():list;
   function length():float;
   function pointAway(d:dot, v:float):dot;
   function isParallel(l:line):bool;
   function intersect(g:geometricShape):list;

//geometricShape circle
circle:
  circle(center:dot,radius:float);

  function setRunstep(val:float,pos:char):void;
  function getRunstep(pos:char):float;
  function getCenter():dot;
  function getRadius(r:float):void;
  function setCenter(d:dot):void;
  function setRadius():float;
  function intersect(g:geometricShape):list;
  function getPointbyarc(arc:float):dot;



//geometricShape polygons
//polygons parameter name: 'a' 'b' ...
//polygons formula: A set of dots
polygons:
  polygons(num_of_apex:int,apex:list);
  function setRunstep(val:float,pos:char):void;
  function getRunstep(pos:char):float;
  function getPoints():list;
  function getArea():float;
  function getAngle():list;
  function getParimeter():float;
  function intersect(g:geometricShape):list;
  function getCentroid():dot;
  function getSides():list;
```

Chapter 4 - Project Plan

4.1 Project Management

4.1.1 Planning

Our group members met every Friday afternoon since we were all available at that time. The whole project was divided to 4 stages, proposal, language design, basic functions implementation and graph functions implementation. Initially we worked individually on small problems, but as the project progressed, we found it more efficient to work together on major challenges. We chose CLIC as our main workspace, most part of this project was finished there.

4.1.2 Specification

We intended to implement everything exactly as stated in the LRM, however, later we realized that it was impossible to stick to it since many problems were not considered in the LRM. So we decide to update the reference manual as we develop our language. For example, in the first version of LRM we required user to specify the data type of iterator and set in the for loop, but since the iterator must belong to set there was not need to specify its type. And some features in the first LRM are not implemented because they are too hard or not necessary. For example, we planned to let the panel capable of showing two images at one time, however, in python it was too hard to realize, so we had to skip this part.

4.1.3 Development

To coordinat works of different members and keep track of our progress, we use Git as our version control system with Github as the host. When members have completed and tested their works, they push them to the master branch of our Geo project. Then we will look over the codes together, make improvements and combine different parts. Most time we work together in CLIC, especially the final stage of the project, because we can discuss together if any problem occurs.

4.1.4 Testing

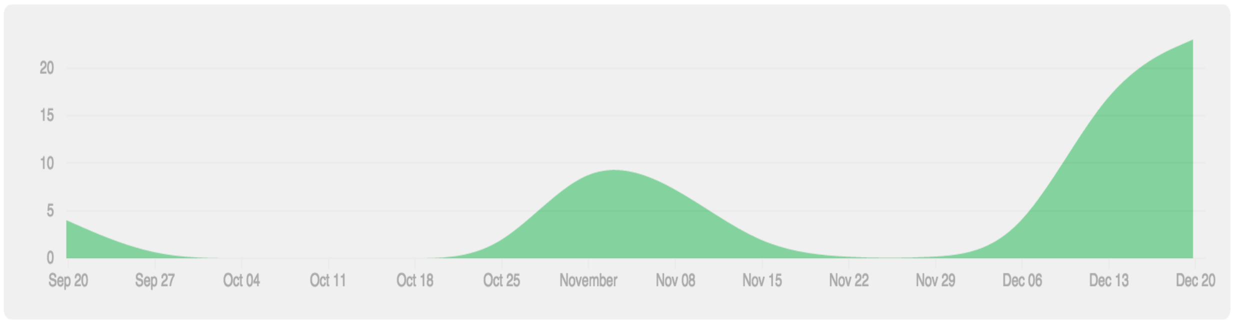
A first simple test case was written for the hello world demo. Then we added many individual test cases, each focusing on some small functions of the language. Since the project is quite big in size, we have to test it this way otherwise we may easily get confused about which part is wrong. Also each test case is prepared with an expected result, if the actual output of the source code is not the same as our expected output then the compiler needs correction of this function.

4.2 Programming Style Guide

• Initial lower case for function and variable names.
• Codes after the “let” statement and pattern matching are indented.
• Avoid breaking expressions over multiple lines.
• Commonly reused code are written as a helper function.

4.3 Project Timeline

The following graph shows our commits to Github during the whole semester.



Sep 28, 2015
Finished proposal
Oct 15, 2015
Started with LRM
Oct 25, 2015
Finished LRM
Oct 30, 2015
Started with scanner and parser, AST
Nov 02, 2015
Finished scanner, parser and AST, started with compiler
Nov 07, 2015
Continued on compiler, modified parser and AST
Nov 07, 2015
Finished compiler, prepared simple test cases
Nov 16, 2015
Hello world demo
Nov 20, 2015
Started with semantic check, code generation and testing
Nov 30, 2015
Continued on semantic check, code generation, and testing
Dec 06, 2015
Updated SAST, code generator, added test cases

Dec 16, 2015
Finished with SAST and code generator, added more test cases
Dec 17, 2015
Started with final report, prepared for demo
Dec 20, 2015
Finished final report, demo worked well
Dec 21, 2015
Final presentation

4.4 Roles and Responsibilities

Zichen Chao - System Architect (worked on paser, ast, compiler, the major contributor of sast)
Qi Wang – Product Manager (worked on parser, ast, copiler, wrote final report and slides)
Yuechen Zhao - Language Guru (designed language features and helped with python functions)
Ziyi Luo – Tester (worked on scanner, created test suites and Geo python libraries)

4.5 Software Development Environment

The Geo project was built on OS X and as stated above, Git was used as a distributed version control system and Github was the host. We used sublime to write the compiler, including the scanner, parser, ast, sast, compiler and test cases. The language we used was mainly OCaml, and python for augmenting the graph visualization and animation. Lastly, makefiles were used to automate the process of compiling, shell scripts were used for testing.

4.6 Project Log

```
commit 010405f8865a579145149643400ffd5e7f566fa5
Author: Qi Wang <wang.qi@columbia.edu>
Date:   Mon Dec 21 18:23:47 2015 -0500

    add final final ppt

commit 5ce5019d9b2c2358d298349a61863194bc2e776b
Author: Qi Wang <wang.qi@columbia.edu>
Date:   Mon Dec 21 18:20:28 2015 -0500

    add ppt pdf

commit 3821402449cedaa8926b8cdbd7f12ff681cb21c8
Author: Ziyi Luo <zl2471@columbia.edu>
```

```
Date:   Mon Dec 21 17:35:32 2015 -0500

    FINAL FINAL ppt

commit f1faebc79e6ad49f6073fbc309fe9160ff57a305
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Mon Dec 21 17:17:17 2015 -0500

    BUGBUGBUG

commit f1b41d62f0576948851425790b54d80b3612b747
Author: Yuechen-Zhao <zhaoyuechen@dyn-160-39-179-18.dyn.columbia.edu>
Date:   Mon Dec 21 17:07:11 2015 -0500

    ppt final version

    hehe

commit d2cbd5637eed029ccf5932ddca37689526be0e82
Author: Yuechen-Zhao <zhaoyuechen@dyn-160-39-179-18.dyn.columbia.edu>
Date:   Mon Dec 21 16:51:14 2015 -0500

    ppt

commit 46d7898ecbd6fbff132223870c7b1b947892a679
Author: Yuechen-Zhao <zhaoyuechen@dyn-160-39-179-18.dyn.columbia.edu>
Date:   Mon Dec 21 16:36:33 2015 -0500

    ppt update

    hehehe

commit 354fd045ba786b2d0488c98fe82aa21ed2871090
Author: Yuechen-Zhao <zhaoyuechen@dyn-160-39-179-18.dyn.columbia.edu>
Date:   Mon Dec 21 15:56:26 2015 -0500

    Add code graph-ppt

commit eb2a5322704424f5270884073ff2c3907bcb8a42
Author: Ziyi Luo <zl2471@columbia.edu>
```

```
Date:   Mon Dec 21 15:28:39 2015 -0500

    Update geo shell for producing independent executable python code

commit 8fbaa4ad0d37347b2e8d3f3af6a3a3b86f355468
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Mon Dec 21 14:40:55 2015 -0500

    Clean older files and for better presentation

commit 50a9b858ea5a2248f4de1d13e2b3d5d54de6b4a9
Author: Qi Wang <wang.qi@columbia.edu>
Date:   Mon Dec 21 12:17:33 2015 -0500

    Update report and ppt

commit 8d98d44fd4a5d0c228bfd58af9cbf9aa0566c781
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Mon Dec 21 07:50:29 2015 -0500

    Updates all test sets

commit 082076749ec07bf02c146f5346c50df84da45f64
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Mon Dec 21 05:56:29 2015 -0500

    Update sysgeo.py

commit ac3317afb04e67bc9ba3eafb866991b73060f96f
Author: chaozc <zichen.chao@gmail.com>
Date:   Mon Dec 21 05:46:45 2015 -0500

    polygon shape

commit e9de5defcffd735b0dd6923243feba701ccfe058
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Mon Dec 21 04:59:50 2015 -0500

    Demo 1 Done
```

```
commit 5ee3c9d8b43f1f333c20995fbf4a98341171a891
Author: chaozc <zichen.chao@gmail.com>
Date:   Mon Dec 21 04:08:19 2015 -0500

    add rule

commit 77cf3b40d3a471dc0f72d5cfe275602dfea95f5e
Author: chaozc <zichen.chao@gmail.com>
Date:   Mon Dec 21 04:04:05 2015 -0500

    insert

commit 506494575f9cb4bdb19bbdef884de6d0bf5ef173
Author: Qi Wang <wang.qi@columbia.edu>
Date:   Mon Dec 21 03:49:56 2015 -0500

    Presentation PPT

commit 6d61734c0af2b37405ece4c33d3161bf62103c1c
Author: Yuechen-Zhao <zhaoyuechen@dyn-160-39-179-18.dyn.columbia.edu>
Date:   Mon Dec 21 02:38:46 2015 -0500

    error checkings

commit 0dcc80a158c22894ce4d0186dbf23cc4554c1e9a
Author: Yuechen-Zhao <zhaoyuechen@dyn-160-39-179-18.dyn.columbia.edu>
Date:   Mon Dec 21 02:06:23 2015 -0500

    gui demo 1

commit 90188e54712e0a97e9a0b72fab8015065f2b09e2
Author: chaozc <zichen.chao@gmail.com>
Date:   Mon Dec 21 02:15:30 2015 -0500

    rewrite run, break

commit e4588f00a4d5c7659c8b13dc37a429368759a2b4
Author: chaozc <zichen.chao@gmail.com>
Date:   Mon Dec 21 01:38:48 2015 -0500
```

```
    polygon added

commit c14f60d417d78c3fbf1c694805cc2ea572fc1fe6
Merge: 72a57d8 8fd94ba
Author: chaozc <zichen.chao@gmail.com>
Date:   Mon Dec 21 01:32:14 2015 -0500

    Merge branch 'master' of github.com:chaozc/Geo

commit 72a57d888d628262576c17bd7fc0d97f4698349c
Author: chaozc <zichen.chao@gmail.com>
Date:   Mon Dec 21 01:31:40 2015 -0500

    list element fixed

commit 8fd94baad3e63c235c444d2639308f31895aefe7
Author: Yuechen-Zhao <zhaoyuechen@dyn-160-39-179-18.dyn.columbia.edu>
Date:   Mon Dec 21 00:22:57 2015 -0500

    Gui

    hahaha

commit 4b6faf936d95a80de91fd8b19c3233b11b6bfd43
Author: 江桐 <koukiri@dyn-129-236-238-220.dyn.columbia.edu>
Date:   Sun Dec 20 23:00:04 2015 -0500

    report

commit 5f7cc71644deafa2ad3e15fe7ad82bdd16f5bc10
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 19:29:26 2015 -0500

    getEndpoints added

commit 47903b10fef2a7062bce21797ca02b2be78d3eac
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 19:27:29 2015 -0500

    not found error fixed
```

```
commit db3d59acb40951bec56d1abf1009312dea1dce1d
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 19:19:42 2015 -0500

    hardcode removed & syslib updated

commit 4d5786565aa59e0099d3678159cf8b889cee8623
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 19:07:48 2015 -0500

    hardcode removed & syslib updated

commit fe8dcf5df7b01ad9656c0f713daf8815bd4758b0
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 18:35:27 2015 -0500

    func tab fixed, sysfunc updated, semantic check updated

commit 2ab7d2a8b2c6a6fa9befc3a97873bd509d8ef3ff
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 16:53:25 2015 -0500

    char eq

commit 4ed4cb4c36c95042a99e84d8fdee56b53aa4e87a
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 16:52:45 2015 -0500

    char eq

commit 020981f7e099c9da15718c3e43ecc680bd60526f
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 16:38:52 2015 -0500

    delete sympy

commit f7151c6d900551ffc67a2e630052804113954a06
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 16:37:20 2015 -0500
```

```
    tab fixed

commit 910598c6992001af2d57a8b42261aecf50c37481
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 15:46:34 2015 -0500

    and or<->eq neq

commit eebb8adbc3539d290b6b3bafb4a6ea8a5d75c824
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 15:45:25 2015 -0500

    python ast added

commit 09f0e5fd0f65da7f57e853b3da246473a4f8adea
Author: chaozc <zichen.chao@gmail.com>
Date:   Sun Dec 20 13:20:32 2015 -0500

    import sysgeo

commit 198101f441966b042b7fb36c15472b5508df19c7
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sun Dec 20 02:32:17 2015 -0500

    sysgeo - fix bugs in line.rotatebyarc

commit b7100804d457382f5535663407cab1856c7db21e
Merge: 60219f1 57b425b
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 22:41:15 2015 -0500

    Merge branch 'master' of github.com:chaozc/Geo

commit 60219f1a16c9a4b7d1a07d3408a455c2e14f394a
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 22:41:01 2015 -0500

    for i check
```

```
commit 57b425b66945fc25d6dbbc6711528f213957a2da
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sat Dec 19 22:32:42 2015 -0500

    renew sysgeo -> Done without debug

commit 6faae02b88dff8f200f9cb013e24089ddb675d53
Merge: 295ab14 c4ef44d
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 22:02:27 2015 -0500

    Merge branch 'master' of github.com:chaozc/Geo

commit 295ab14228f4a6fe9cffe2106dd02f1d049e88d5
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 22:02:11 2015 -0500

    list check

commit c4ef44d3e5de5e11d88d598ae4d40ee86f09746b
Author: Yuechen-Zhao <zhaoyuechen@dyn-160-39-133-196.dyn.columbia.edu>
Date:   Sat Dec 19 21:52:38 2015 -0500

    gtk graphical

    hehehe

commit a459f319cf1d2b28c75cab3818f66500964d552b
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sat Dec 19 20:23:44 2015 -0500

    Renew sysgeo.py

commit e96f70031940729b20da6feae370fafb1777dc1c
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 20:22:13 2015 -0500

    list

commit 21ee233ce63a8d8d0743241e73cdf470f30994d9
```

```
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 19:09:07 2015 -0500

    float, bool fixed

commit 21ba6952ee1946792b73d0a89c4fb95aa3bc6b1f
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 18:46:39 2015 -0500

    operator type check

commit 16d0e96c49c48299be5f5443f6ec758bc77e64e6
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 16:45:24 2015 -0500

    check function type

commit a99f2028c2070a7c0861530e76ee00070bb329ac
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 16:07:17 2015 -0500

    check function type

commit 0b86400f0e680c3c1cea6877641b1a39bbacdd1a
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 15:55:06 2015 -0500

    check function type

commit b3aa7a3b8e9b0690a4cf7db2d073375769a30b3c
Merge: 9fa758b 5f9f491
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 15:46:41 2015 -0500

    Merge branch 'master' of github.com:chaozc/Geo

commit 9fa758ba405d52a1a57cc3de9018606140d6828b
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 15:46:17 2015 -0500
```

```
    check function type

commit 5f9f491f5bde1ee017729087f43065fa356dec82
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sat Dec 19 15:08:33 2015 -0500

    New update to sysgeo

commit 20016a043e03e48d884e90bda5fb36763512eefa
Merge: 17b0b0f b6a23ab
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 11:45:06 2015 -0500

    merge

commit 17b0b0fcd01f33cb19f637e149e5c6442991fdd3
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Dec 19 11:43:15 2015 -0500

    check vars

commit b6a23aba7149dd7bf4639fd5ce944249e6b06c16
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sat Dec 19 00:14:10 2015 -0500

    geo geotype realisation - unfinished

    12/18/2015

commit 748bacd67efe60fa41510f5908d402ed6b800d75
Author: chaozc <zichen.chao@gmail.com>
Date:   Fri Dec 18 16:24:17 2015 -0500

    bool

commit 5f7bd3b2194733e94b5f06e4522de50861bbc056
Author: chaozc <zichen.chao@gmail.com>
Date:   Fri Dec 18 16:23:29 2015 -0500

    bool
```

```
commit c09eedc8ffc8190ec4f56d7198644eb04f5b6470
Author: chaozc <zichen.chao@gmail.com>
Date:   Fri Dec 18 15:10:08 2015 -0500

    dot circle line

commit 038a36f53fd2624bbb73a753d13be4cf318ffae0
Author: chaozc <zichen.chao@gmail.com>
Date:   Fri Dec 18 11:12:23 2015 -0500

    chaozc

commit 616e282e41167e19e96a402e6f24305ee0b98c93
Author: chaozc <zichen.chao@gmail.com>
Date:   Fri Dec 18 10:52:34 2015 -0500

    printT

commit fc61b503abcccd7deb7e198235aa89fd5a8a87f5
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sun Dec 6 22:05:02 2015 -0500

    Update: compile .g using geo.sh 12/06/2015

commit 997b54cd3ed9667e71a8097f5c76b84dc4f3a61a
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sun Nov 15 00:22:59 2015 -0500

    Update src. Current version 1.1

commit c72f7b3cead595a042df77a3de05e92dea28a58c
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Thu Nov 12 09:32:10 2015 -0500

    Geo to python reference in compile.ml 11/12/2015

commit 7eee2e471b3e63bd67ec2e8d0a004eefd2cb15a5
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Wed Nov 11 11:22:58 2015 -0500
```

```
    Log of syntax and semantic changes

commit 1d11eacb5c4e0f6501564a1ea23bb144d28b625e
Author: chaozc <zichen.chao@gmail.com>
Date:   Tue Nov 10 22:29:33 2015 -0500

    README

commit 9c69df285eb0b59729d6904b29cd3d75ef6c4517
Author: chaozc <zichen.chao@gmail.com>
Date:   Tue Nov 10 21:31:43 2015 -0500

    Simple helloworld

commit 026ff619fd00b3cc2ffb977fb1d4ed3db04733e2
Author: chaozc <zichen.chao@gmail.com>
Date:   Tue Nov 10 00:19:17 2015 -0500

    parser ast v1

commit 1da0285f1c1c521f7d9d04dc73b9d5c7bb13a9f7
Merge: 8d90fe9 006d8cb
Author: chaozc <zichen.chao@gmail.com>
Date:   Mon Nov 9 22:47:56 2015 -0500

    Merge branch 'master' of github.com:chaozc/Geo

commit 8d90fe9f8697a7c225d46868ddc8612416756adc
Author: chaozc <zichen.chao@gmail.com>
Date:   Mon Nov 9 22:46:39 2015 -0500

    parser

commit 006d8cb4401b2fb7fc20271c1ba7b03731993f45
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Mon Nov 9 18:59:15 2015 -0500

    Update std.glib, current version v0.1 11/09/2015
```

```
commit b880e5103ae801b94c7956c5952b1386dcaaf4c0
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Mon Nov 9 11:56:35 2015 -0500

    Remove token '@' and add two CFGs 11/09/2015

    REMOVE
    | '@' { AT }
    ADD
    | "@end" {ENDOFPROGRAM}
    | '@'(['a' - 'z' 'A' - 'Z']+ as pre) {PRESET(pre)}

commit 1047bb793c3ff11b869deb8fa7a67bbe1a093fb6
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sat Nov 7 22:34:55 2015 -0500

    Increase token '@' 11/07/2015

commit 46801b1f7eaf5cd8c4ea254e5375fcb1758061c2
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sat Nov 7 22:28:55 2015 -0500

    Update scanner.mll 11/07/2015

    <int> INTEGER -> <int> INTEGERLIT
    .etc

commit a26968cc05ed7e72abe846a688b03595f556c471
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sat Nov 7 20:19:53 2015 -0500

    Upload scanner 11/07/2015

    Lichen & Qi, please note the comment on the top of scanner.mll.

commit 33c8c0eca445c9ba2e6ae00797af1922148bda90
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Nov 7 18:44:28 2015 -0500

    src added
```

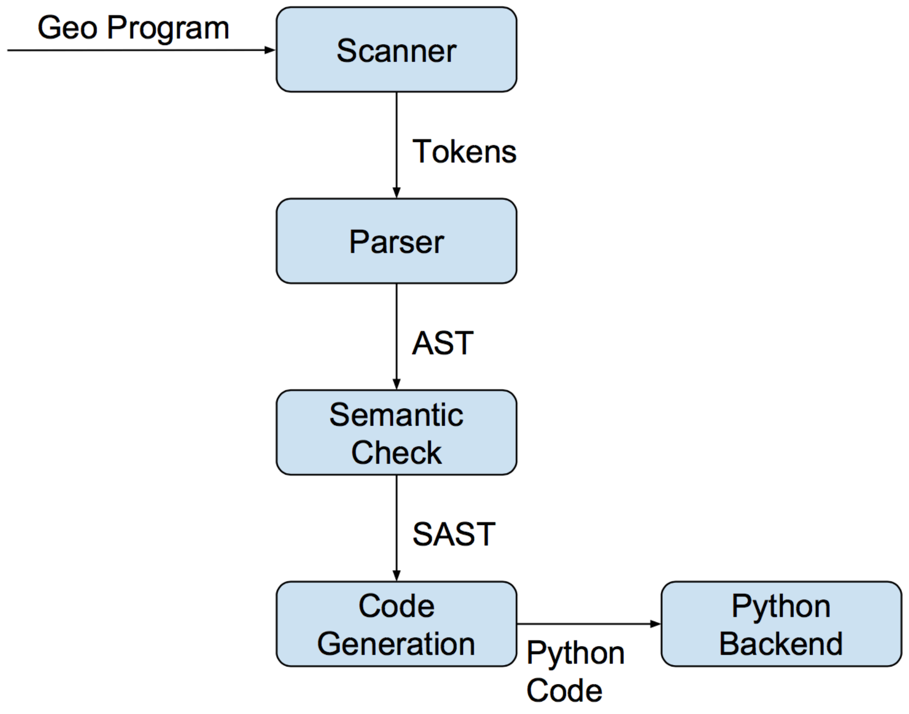
```
commit be5dcfa1b37e5b7b216f2165ca78f823f49974d5
Author: chaozc <zichen.chao@gmail.com>
Date:   Sat Nov 7 16:45:35 2015 -0500

    microc sample

commit 8c4af5c1de0d91e8fda4820c66b5839e47f279fa
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sat Nov 7 16:34:24 2015 -0500

    Trivial action 11/07/2015

    Delete file "hello"

commit f193478d19f23915946a28e4af1d6cff27446915
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Sat Nov 7 16:30:31 2015 -0500

    Update LRM v1.1 11/07/2015

    In v1.1, Geo
    1. increases keyword "void"
    2. modifies (non-return-value) function definition.

commit 9ca3d66442cc095af8749e04e0e3ff72527e7ad1
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Fri Nov 6 17:54:54 2015 -0500

    Update 11/06/2015

commit e7d29017141856abe1c77e6bcb10d6d1c84e2394
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Fri Nov 6 17:54:31 2015 -0500

    Language Demo v0.0

commit e2387b6e0942128963eeb25d946eb5137700cfad
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Fri Nov 6 17:46:01 2015 -0500
```

```
    Geo std library 11/06/2015

commit 5988bcff6f7ca9f32b4c7b48e7c4fff8f55ee7c1
Author: Ziyi Luo <zl2471@columbia.edu>
Date:   Fri Nov 6 17:41:33 2015 -0500

    Reference doc renew 11/06/2015

commit a571b1602f67df940075b7391362caca420b2327
Author: 子仪_ <samluo@dyn-209-2-216-195.dyn.columbia.edu>
Date:   Mon Sep 28 15:21:09 2015 -0400

    Mod by Ziyi 09/28

commit e6999a33e9697715c975eba1f27a39cd057e3845
Author: Yuechen-Zhao <zhaoyuechen@dyn-160-39-178-121.dyn.columbia.edu>
Date:   Fri Sep 25 16:47:51 2015 -0400

    Proposal Draft

    A total draft… Please edit in google docs if its more convenient!

commit f77dc63f8b5cf16fc0865e20bcd7909215c4070b
Author: Yuechen-Zhao <zhaoyuechen@dyn-160-39-178-121.dyn.columbia.edu>
Date:   Fri Sep 25 16:44:41 2015 -0400

    Hi

commit 12e4127e5c6d5aad6cb60056ed6c38e3b4a82846
Author: Qi Wang <wang.qi@columbia.edu>
Date:   Wed Sep 23 23:44:20 2015 -0400

    Create hello

commit 727c15962dfcc8e7fab64f20adbbc36cf68cc396
Author: 子仪_ <samluo@ziyi-deMacBook-Pro.local>
Date:   Wed Sep 23 23:40:24 2015 -0400

    Ziyi adds a map of Columbia.
```

```
commit eaace90218857c2ed4730022310452a6d4d1b0f3
Author: chaozc <zichen.chao@gmail.com>
Date:   Wed Sep 23 23:02:49 2015 -0400

    first geo file

commit 38602f1aca8ffc03d95250344ca91b80d1f8af3c
Author: chaozc <zichen.chao@gmail.com>
Date:   Wed Sep 23 22:50:33 2015 -0400

    readme

commit 4cb6d4fd7116a521c96cae76adf103561bb2f566
Author: chaozc <zichen.chao@gmail.com>
Date:   Wed Sep 23 15:38:09 2015 -0400

    Initial commit
```

Chapter 5 - Architectural Design

5.1 Block Diagram



5.2 Components

5.2.1 Scanner

The scanner was implemented with ocamellex, with scanner.mll as the corresponding file. The scanner breaks down formatted input into tokens and translates individual tokens to some basic data types like int and char, discarding unnecessary information for compilation like whitespaces and comments. Also, it does basic syntax checking, any illegal symbols like # or $ are discovered and rejected.

5.2.2 Parser and AST

We use ocamlyacc to implement the parser and ast, with parser.mly and ast.ml as the corresponding files. The parser parses the token stream produced by the scanner to an abstract syntax tree, which represents the abstract syntactic structure of the source code. The parsing process does further check of syntax following the specifications in ast file.

5.2.3 Semantic Check and SAST

The semantic check was implemented using OCaml, and the associated file is compile_sc_py.ml, which takes in an abstract syntax tree and produces a semantically checked tree of python. A standard python AST, pyast.ml, is used for comparing. Errors including inconsistent types in expressions, reference to undefined functions or variables are discovered during this process. We use two maps named vars and funcs to record variables and functions in the scope and raise undefined exceptions if there is no match in maps.

5.2.4 Code Generator

The generator was implemented using OCaml and the associated file is compile_to_pycode.ml. The generator takes in the sast produced before and generates python code from it. Most of the code it generates is hard coded by translating python.

Chapter 6 - Test Plan

6.1 Geo Test Code

6.1.1 test-assignments.g

```
@panel panel1
a=3;
b=3;
c=123e-1;
d="Ocaml sucks!";
e='@';
f=3.4*2==4.;
printT(a);
printT(b);
printT(c);
printT(d);
printT(e);
printT(f);
@end
```

6.1.2 test-circle.g

```
@panel panel1
dot1=[3,5];
circle1=circle([3,5],5);
circle2=circle([0,0],5);
printT(circle1);
printT(circle2);
printT(circle2.intersect(circle1));
printT(circle2.intersect(line([0,2],[1,2])));
printT(circle2.intersect(line([0,2],[1,2],-1,1)));
printT(circle2.intersect(line([0,-2],[0,2])));
printT(circle2.intersect(line([0,-2],[0,2],0,0)));
printT(circle1.getRadius());
printT(circle2.getCenter());
circle1.setCenter([2,2]);
printT(circle1);
circle2.setRadius(3);
printT(circle2);
printT(circle2.getPointbyarc(PI/4));
@end
```

6.1.3 test-comparison.g

```
@panel panel1
printT(5>3);
printT(5==5 & 3<2);
a=4.3;
b=8.6/2.0;
c=a==b;
printT(!c);
printT(a==b);
printT(false | true);
printT(!false & true);
@end
```

6.1.4 test-dot.g

```
@panel panel1
dot1=[3,5];
dot2=[5,7];
printT(dot1.distance(dot2));
printT(dot1.getX());
printT(dot1.getY());
@end
```

6.1.5 test-fit.g

```
@panel fib
function fib(x:int):int:
  if (x == 1):
    return(1);
  else:
    if (x == 2):
          return(1);
    else:
          return(fib(x-1)+fib(x-2));
    end
  end
end
i = 1;
while (i < 10):
  print(fib(i));
  i = i+1;
end
@end
```

6.1.6 test-for.g

```
@panel panel1
for i in {1,"zcCHAO",2,"zyLUO",3,"qWANG",4,"ycZHAO",true,';'}:
printT(i);
end
@end
```

6.1.7 test-function.g

```
@panel panel1


function test1(str:string):bool:
if(str=="PLT"):
return true;
else:
//end
return false;
end
end

function test2():void:
if(test1("PLT")):
printT("is PLT");
else:
printT("is not PLT");
end
end
test2();
@end

/*

function test1(str:string):bool:
if(str=="PLT"):
return true;
//else:
end
return false;
//end
end

function test2():void:
if(test1("PLT")):
printT("is PLT");
//else:
end
printT("is not PLT");
//end
end
test2();

*/
```

6.1.8 test-gcd.g

```
@panel gcd
function gcd(a:int, b:int):int:
  if (a<b):
    return (gcd(b,a));
  else:
    if (a == b):
          return (a);
    else:
          return(gcd(a-b, b));
    end
  end
end
print(gcd(70,28));
print(gcd(147,21));
@end
```

6.1.9 test-if.g

```
@panel panel1
if(5==3):
printT("5==3");
else:
if(5>3):
printT("5>3");
else:
printT("5<3");
end
end
@end
```

6.1.10 test-list.g

```
@panel panel1
d={1,2,3,4,5,6};
e={"str",45,'c',3.4};
printT(d);
printT(d#[2]);
printT(e);
printT(e#[0]);
@end
```

6.1.11 test-line.g

```
@panel panel1
```

```
dot1=[3,5];
line1=line(4,5);
line2=line([1,2],[2,3]);
line3=line(4,5,-5,5);
line4=line([0,-2],[0,2],0,0);
printT(line1);
printT(line2);
printT(line3);
printT(line4);
printT(line2.intersect(line4));
printT(line3.distance(dot1));
printT(line4.length());
printT(line2.pointAway([2,3],5));
printT(line4.pointAway([0,0],1));
printT(line3.getEndpoints());
printT(line3.getMidpoint());
printT(line1.isParallel(line2));
printT(line1.getX(5));
printT(line4.getX(1));
printT(line2.getY(0));
line2.setRunstep('a',0.5);
line3.setRunstep('b',1.0);
line3.setRunstep('x',1.0);
// r1=runset(8,0.1);
// r1.addPara(line2,'a');
// r1.addPara(line3,'b');
// r1.addPara(line3,'x');
// run r1:
// printT(line2);
// printT(line3);
// printT(line4);
// line4.rotateonPoint([0,0],PI/4);
// end
@end
```

6.1.12 test-operation.g

```
@panel panel1
//int+int
printT(1+1);
//float+float
printT(.1+1.0);
//int+float
printT(1+1.0);
//float-int
printT(.1-5);
//int
printT(1+2*3-4/2);
//float
printT(1.+2.*3-4./3);
//int
printT(1-5*2^3+4*5/2*7%3);
```

```
//float mix with int
printT(1.+2.*3-4/3);
//string+string
printT("Hello "+"World");
@end
```

6.1.13 polygon.g

```
@panel panel1
li={[0,0],[5,0],[5,3],[0,3]};
l2={[0,0],[2,2],[0,4],[-2,2]};
p1=polygon(li);
p2=polygon(l2);
print(p1);
print(p1.getPoints());
print(p1.getArea());
print(p1.getAngle());
print(p1.getParimeter());
print(p1.getCentroid());
print(p1.getSides());
print(p1.intersect([5,6]));
print(p1.intersect(line([0,0],[2,2])));
print(p1.intersect(circle([0,0],3)));
print(p1.intersect(p2));
@end

/*
r1=runset(5,0.1);
circle1.addRunstep('x',1);
circle1.addRunstep('y',-1);
circle2.addRunstep('r',1);
printT(circle1.getRunstep());
r1.addPara(circle1,'x');
r1.addPara(circle1,'y');
r1.addPara(circle2,'r');
run r1:
printT(circle1);
printT(circle2);
end
*/
```

6.1.14 test-print.g

```
@panel panel1
printT(123456);
printT(123.456);
printT(5e-5);
printT('a');
printT('%');
printT(true);
printT(false);
printT("Hello world!");
@end
```

6.1.15 test-qsort.g

```
@panel qsort
function qsort(a:list, l:int, r:int):list:
  i = l;
  j = r;
  mid = (l+r)/2;

  while (i <= j):
    while (i <= j & a#[i] < a#[mid]):
          i = i+1;
    end
    while (i <= j & a#[j]> a#[mid]):
          j = j-1;
    end

    if (i <= j):
          k = a#[i];
          a#[i] = a#[j];
          a#[j] = k;
          i = i+1;
          j = j-1;
    end

  end


  if (l < j):
    a = qsort(a, l, j);
  end
  if (i < r):
    a = qsort(a, i, r);
  end
  return(a);
end
b = {3,7,8,32,1,4,7,9,2,5};
b = qsort(b, 0, len(b)-1);
print(b);
@end
```

6.1.16 test-while.g

```
@panel panel1
count=0;
while(count<11):
printT(count);
count=count+2;
```

```
end
@end
```

6.2 Test Suite

Here is the automatic test suite script.

```
#!/bin/bash
########################################
#               geo.sh
#
#   Geo compiler shell script
#           Version 0.01
#            12/06/2015
########################################

GEO='./geo'
opfile=''
copfile=''
keep=0
Usage() {
  echo "************************************************"
  echo -e "\t\tGeo tests shell\t\t"
    echo "Usage: geo.sh [options] [.g files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    echo "************************************************"
    exit 1
}

Compile(){
  eval "$GEO<$1>$2"
}

Execute(){
  opname=`echo $1 | sed 's/.py/.out/'`
  eval "python $1 > $opname"
}

Precat(){

  grep '@[^eE]' $1 >&2
  if [ -e std.glib ]; then
    cat std.glib >&2
  else
    echo "$0 warning: std.glib does not exist."
  fi
  cat $1 | sed 's/^@.*$//' >&2
  echo >&2
  echo @end >&2
}
```

```
while getopts kh c; do
    case $c in
   k) # Keep intermediate files
       keep=1
       ;;
   h) # Help
       Usage
       ;;
    esac
done

shift `expr $OPTIND - 1`

if [ $# -ge 1 ]; then
   opfile=$1'pre'
   rm -f $opfile
   Precat $1 2>> $opfile
   if [ $# -eq 1 ]; then
     copfile=`echo $opfile | sed 's/.gpre/.py/'`
   else
     copfile=$2
   fi
   Compile $opfile $copfile
   if [ $keep -eq 0 ]; then
     rm -f $opfile
   fi
   Execute $copfile
Fi
```

Chapter 7 - Lessons Learned

7.1 Qi Wang

I have learned how to do a team project. When a project is extremely large and complex, like this one, it's impossible to finish all the works individually. So communications are really important, we need to keep communicating with others about our own sub-tasks, and we should be able to combine different parts together. This is not easy, since many details are involved. And plan is very important, the first thing before start is to make a plan and follow it strictly. Even if we have a perfect plan, we still need to keep communicate with our teammates during the group project. Sometimes things can be different from what we have imaged. For example, the semantic check is far more difficult than we thought, so as soon as I found out that it might take us much more time, I told my teammates and we discussed on how to change the schedule.

7.2 Zichen Chao

The most important thing I have learned from this project is that we need to keep the whole picture in mind. There are many technologies, tools, theories and other details involved in making a compiler. In each part of the project, we have to apply several of these in our codes or designs. When we are doing the coding or design, these sub-tasks seem to be independent with each other. For example, the parsing tree of the code and the semantic check of the code seem to be two very different tasks. It is possible that we may not see any connection between these two things at the first sight. However, when we were implementing the semantic check, we found that several modifications on the parsing tree had to be made to help with the semantic check.

7.3 Yuechen Zhao

If I can use only one word to describe the key to success in this project, I will say communication. This is a large project and we have 4 members in our team. When working together, we have to go through many discussions because different people have different opinions. We need to make an agreement on one plan before we can start to divide the work, and it often takes us a lot of time to discuss the advantages and disadvantages of ideas. But the discussions do help us to save time from trying things that not work for us. During the discussion, we should be able to explain our opinions clearly and listen carefully to others. All these make our communications efficient.

7.4 Ziyi Luo

Before this course project, I have never realized that how important testing is in a project. After finishing the most part of our project, we decided to try several test cases. I thought our compiler was able to handle all the test cases, or we could make some little modifications to correct it.

However, when the test was done, we discovered so many problems and we were not able to resolve them by a few lines of codes. We had to adjusts the foundation part of our complier and rewrite several parts. We realized that we should spend more time on testing before finishing the compiler, since it’s much harder to resolve all the bugs in the end. So we designed many test cases, each one testing a single function of our compiler, to resolve problems one by one.

Chapter 8 – Appendix

8.1 Scanner

```
(*
scanner.mll - updated 11:25AM 11/09/2015
NOTE FROM ZIYI TO ZICHEN & QI 11/09/2015:
REMOVE '@' AS A TOKEN
ADD
| "@end" {ENDOFPROGRAM}
| '@'(['a' - 'z' 'A' - 'Z']+ as pre) {PRESET(pre)}
TO RECOGNIZE A PRESET
THEREFORE WE NEED TO ADD
%token <string> PRESET
%token ENDOFPROGRAM
in parser.mly
IN PARSER, THE PROCESSING SUGGESTION IS:
PRESET ID {Preset($1,$2)}
*)

{ open Parser } (* Get the token types *)
rule token = parse
(* Whitespace *)
[' ' '\t' '\r' '\n'] { token lexbuf }

(* Comments *)
| "/*"    { comment lexbuf }  | "//"    { comment2 lexbuf}

(* Basic tokens *)
| '(' { LPAREN } | ')' { RPAREN } (* Parentheses *)
| '{' { LBRACE } | '}' { RBRACE } (* Braces *)
| '[' { LSQUAR } | ']' { RSQUAR } (* Square brackets *)
| ';' { SEMI }   | ':' { COLON }
| '.' {GET}             | ',' { COMMA }
| '=' { ASSIGN } (*'@' { AT }*)
| '#' { DOLL}

(* Arithmetic operators *)
| '+' { PLUS }   | '-' { MINUS }
| '*' { TIMES }  | '/' { DIVIDE }
| '%' { PERCENT } | '^' {EXP}

(* Logic operators *)
| "==" { EQ }
| "!=" { NEQ }   | '<' { LT }
| "<=" { LEQ }   | '>' { GT }
| ">=" { GEQ }   | '!' {NOT}
| '&' {AND}             | '|' {OR}

(* Keywords (totally 23) *)
| "else" { ELSE }           | "end" { END }
| "for" { FOR }
| "function" {FUNCTION}      | "if" { IF }
| "import" { IMPORT } | "in" { IN }
| "polygon" { POLYGON }            | "break" { BREAK }
| "return" { RETURN }
| "run" {RUN}
| "line" {LINE}
| "while" { WHILE }     | "circle" {CIRCLE}
| "runset" { RUNSET }

| ("int"|"float"|"void"|"bool"|"char"|"string"|"list") as tp {TYPE(tp)} (*
Type *)

| "print" {PRINT}
| "printT" {PRINTT}
| eof { EOF } (* End of file *)

(* Integers *)
| ['0' - '9']+ as lxm { INTEGERLIT(int_of_string lxm) }

(* Float *)
| ['0'-'9']*'.'['0'-'9']+ ('e'['+' '-']?['0'-'9']+)? as flo
{FLOATLIT(float_of_string flo)}
| ['0'-'9']+'.'['0'-'9']* ('e'['+' '-']?['0'-'9']+)? as flo
{FLOATLIT(float_of_string flo)}
| ['0'-'9']+ ('e'['+' '-']?['0'-'9']+) as flo {FLOATLIT(float_of_string flo)}

(* Char *)
| '''(['\000' - '\038' '\040' - '\127'] as chr)''' {CHARLIT(chr)}

(* Bool *)
| ("true"|"false") as bl {BOOLLIT(bool_of_string bl)}

(* String *)
| '"'(['\000' - '\033' '\035' - '\127']* as str)'"' {STRINGLIT(str)}

(* ID *)
| ['a' - 'z' 'A' - 'Z']['a' - 'z' 'A' - 'Z' '0' - '9' '_']* as lxm
{ ID(lxm) }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }

(* Preset *)
| "@end" {ENDOFPROGRAM}
| '@'(['a' - 'z' 'A' - 'Z']+ as pre) {PRESET(pre)}

and comment = parse
  "*/" { token lexbuf } (* Endofcomment*)
| _ { comment lexbuf } (* Eat everything else *)

and comment2 = parse
  '\n' {token lexbuf}
| _ {comment2 lexbuf}
```

8.2 Parser

8.2.1 parser.mly

```
%{ open Ast %}
%token LPAREN RPAREN LBRACE RBRACE LSQUAR RSQUAR SEMI COLON GET COMMA ASSIGN
DOLL
%token PLUS MINUS TIMES DIVIDE PERCENT EXP
%token EQ NEQ LT LEQ GT GEQ NOT   AND OR
%token BREAK ELSE END FOR FUNCTION RETURN RUN WHILE IF IN LINE CIRCLE RUNSET
POLYGON
%token <string> TYPE
%token PRINT PRINTT
%token ENDOFPROGRAM
%token EOF

%token <int> INTEGERLIT
%token <float> FLOATLIT
%token <char> CHARLIT
%token <bool> BOOLLIT
%token <string> STRINGLIT
%token <string> ID
%token <string> PRESET

%right ASSIGN
%left AND OR
%left EQ NEQ
%right NOT
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE PERCENT
%left EXP
%left GET

%start program
%type <Ast.program> program
%%

program:
  preset bodies ENDOFPROGRAM EOF { $2 }

preset:
  PRESET ID { Preset($1, $2) }

bodies:
    /* nothing */ { [], [] }
  | bodies decl { ($2 :: fst $1), snd $1 }
  | bodies stmt { fst $1, ($2 :: snd $1) }

decl:
  fdecl { $1 }

fdecl:
  FUNCTION ID LPAREN paras_opt RPAREN COLON TYPE COLON stmt_list END
  {{
    tp = $7;
    fname = $2;
    paras = $4;
    body = List.rev $9

  }}

paras_opt:
      /* nothing */ { [] }
  | paras_list   { List.rev $1 }

paras_list:
      ID COLON TYPE                { [($1, $3)] }
  | paras_list COMMA ID COLON TYPE { ($3, $5) :: $1 }

stmt_list:
      /* nothing */  { [] }
  | stmt_list stmt { $2 :: $1 }

stmt:
      expr SEMI { Expr($1) }
    | RETURN expr SEMI { Return($2) }
    | IF LPAREN expr RPAREN COLON stmt_list END
      { If($3, $6, []) }
    | IF LPAREN expr RPAREN COLON stmt_list ELSE COLON stmt_list END
      { If($3, $6, $9) }
    | WHILE LPAREN expr RPAREN COLON stmt_list END { While($3, $6) }
    | PRINT LPAREN expr RPAREN SEMI { Print($3) }
    | PRINTT LPAREN expr RPAREN SEMI { PrintT($3) }
    | FOR expr IN expr COLON stmt_list END { For($2, $4, $6) }
    | RUN expr COLON stmt_list END { Run($2, $4) }
    | ID ASSIGN expr SEMI  { Assign($1, $3, Noexpr) }
    | ID DOLL LSQUAR expr RSQUAR ASSIGN expr SEMI { Assign($1, $7, $4) }
    | BREAK SEMI{ Break }

expr_opt:
    /* nothing */ { Noexpr }
    | expr { $1 }

digit:
      INTEGERLIT       { Int($1) }
    | FLOATLIT         { Float($1) }

expr:
      INTEGERLIT       { Int($1) }
  | FLOATLIT         { Float($1) }
  | CHARLIT          { Char($1) }
  | BOOLLIT          { Bool($1) }
  | STRINGLIT        { String($1) }
  | ID               { Id($1) }
    | MINUS expr       { Minus($2)}
  | expr PLUS   expr { Binop($1, Add,   $3) }
    | expr MINUS  expr { Binop($1, Sub,   $3) }
  | expr TIMES  expr { Binop($1, Mult,  $3) }
  | expr DIVIDE expr { Binop($1, Div,   $3) }
  | expr EQ     expr { Binop($1, Equal, $3) }
  | expr NEQ    expr { Binop($1, Neq,   $3) }
  | expr LT     expr { Binop($1, Less,  $3) }
  | expr LEQ    expr { Binop($1, Leq,   $3) }
  | expr GT     expr { Binop($1, Greater,  $3) }
  | expr GEQ    expr { Binop($1, Geq,   $3) }
    | expr AND    expr { Binop($1, And,   $3) }
    | expr OR     expr { Binop($1, Or,   $3) }
    | expr PERCENT expr { Binop($1, Mod,   $3)}
    | expr EXP    expr { Binop($1, Exp,   $3) }
    | NOT expr         { Not($2) }
  | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
  | LPAREN expr RPAREN { $2 }
    | LSQUAR expr COMMA expr RSQUAR { Dot($2, $4) }
    | expr GET expr { Get_Call($1, $3) }
    | LINE LPAREN actuals_opt RPAREN { Line($3) }
    | POLYGON LPAREN actuals_opt RPAREN { Polygon($3) }
    | CIRCLE LPAREN actuals_opt RPAREN { Circle($3) }
    | RUNSET LPAREN actuals_opt RPAREN { Runset($3) }
    | LBRACE actuals_opt RBRACE { List($2) }
    | ID DOLL LSQUAR expr RSQUAR { ListEle($1, $4) }



actuals_opt:
    /* nothing */ { [] }
  | actuals_list  { List.rev $1 }

actuals_list:
    expr                    { [$1] }
  | actuals_list COMMA expr { $3 :: $1 }
```

8.2.2 parser.mli

```
type token =
  | LPAREN
  | RPAREN
  | LBRACE
  | RBRACE
  | LSQUAR
  | RSQUAR
  | SEMI
  | COLON
  | GET
  | COMMA
  | ASSIGN
  | DOLL
  | PLUS
  | MINUS
```

```
    | TIMES
    | DIVIDE
    | PERCENT
    | EXP
    | EQ
    | NEQ
    | LT
    | LEQ
    | GT
    | GEQ
    | NOT
    | AND
    | OR
    | BREAK
    | CONST
    | ELSE
    | END
    | FOR
    | FUNCTION
    | IMPORT
    | MODEL
    | RETURN
    | RUN
    | SUBMODEL
    | WHILE
    | IF
    | IN
    | LINE
    | CIRCLE
    | RUNSET
    | POLYGON
    | TYPE of (string)
    | PRINT
    | PRINTT
    | ENDOFPROGRAM
    | EOF
    | INTEGERLIT of (int)
    | FLOATLIT of (float)
    | CHARLIT of (char)
    | BOOLLIT of (bool)
    | STRINGLIT of (string)
    | ID of (string)
    | PRESET of (string)

  val program :
    (Lexing.lexbuf  -> token) -> Lexing.lexbuf -> Ast.program
```

8.3 AST

```
  type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater | Geq
  | And | Or | Mod | Exp
```

```
type preset =
   Preset of string * string


type expr =
    Noexpr
  | Int of int
  | Float of float
  | Bool of  bool
  | String of string
  | Char of char
  | Id of string
  | Binop of expr * op * expr
  | Call of string * expr list
  | Dot of expr * expr
  | Get_Call of expr * expr
  | Line of expr list
  | Circle of expr list
  | List of expr list
  | Polygon of expr list
  | Not of expr
  | Minus of expr
  | ListEle of string * expr
  | Runset of expr list


type stmt =
  | Expr of expr
  | Return of expr
  | If of expr * stmt list * stmt list
  | For of expr * expr * stmt list
  | While of expr * stmt list
  | Print of expr
  | PrintT of expr
  | Run of expr * stmt list
  | Assign of string * expr * expr
  | Break


type fdecl = {
    fname : string;
    paras : (string * string) list;
    body : stmt list;
    tp : string;

  }

type program = fdecl list * stmt list
```

8.4 Semantic Check

```
open Ast
open Pyast

module StringMap = Map.Make(String)

type envType = {
    vars : string StringMap.t;
    funcs : string StringMap.t;
    get_call : string;
    func_opt: string list StringMap.t;
};;

let env = ref {
    vars = StringMap.empty;
    funcs = StringMap.empty;
    get_call = "";
    func_opt = StringMap.empty;
};;

(*translate geoAst to pyAst*)

env := {vars = StringMap.empty; funcs = StringMap.add ":len" "int"
env.contents.funcs; get_call = ""; func_opt = StringMap.add ":len" ["list"]
env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "list:append" "void"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "list:append"
["any"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "list:pop" "list_ele"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "list:pop" []
env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add ":sin" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add ":sin" ["float"]
env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add ":cos" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add ":cos" ["float"]
env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add ":tan" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add ":tan" ["float"]
env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add ":cot" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add ":cot" ["float"]
env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "dot:getX" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "dot:getX" []
env.contents.func_opt};;
```

```
env := {vars = StringMap.empty; funcs = StringMap.add "dot:getY" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "dot:getY" []
env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "dot:distance" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "dot:distance"
["dot"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:distance" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "line:distance"
["dot"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:getPara" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "line:getPara"
["char"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:setPara" "void"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "line:setPara"
["char";"float"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:getRunstep"
"float" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"line:getRunstep" ["char"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:setRunstep"
"void" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"line:setRunstep" ["char";"float"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:getY" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "line:getY"
["float"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:getX" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "line:getX"
["float"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:contains" "bool"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "line:contains"
["dot"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:getMidpoint"
"dot" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"line:getMidpoint" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:getEndpoints"
"list" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"line:getEndpoints" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:length" "float"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "line:length" []
env.contents.func_opt};;
```

```
env := {vars = StringMap.empty; funcs = StringMap.add "line:pointAway" "dot"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "line:pointAway"
["dot";"float"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:isParallel"
"bool" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"line:isParallel" ["line"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:intersect" "list"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "line:intersect"
["shape"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "circle:getCenter"
"dot" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"circle:getCenter" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "circle:getRadius"
"float" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"circle:getRadius" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "circle:setCenter"
"void" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"circle:setCenter" ["dot"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "circle:setRadius"
"void" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"circle:setRadius" ["float"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "circle:getRunstep"
"float" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"circle:getRunstep" ["char"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "circle:setRunstep"
"void" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"circle:setRunstep" ["char";"float"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "circle:getPointbyarc"
"dot" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"circle:getPointbyarc" ["float"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "circle:intersect"
"list" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"circle:intersect" ["shape"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "runset:addPara" "bool"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "runset:addPara"
["shape";"char"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "runset:renew" "void"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "runset:renew" []
env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "runset:enableRun"
"void" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"runset:enableRun" [] env.contents.func_opt};;
```

```
env := {vars = StringMap.empty; funcs = StringMap.add "runset:disableRun"
"void" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"runset:disableRun" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "runset:removePara"
"bool" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"runset:removePara" ["shape";"char"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "line:rotateonPoint"
"void" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"line:rotateonPoint" ["dot";"float"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "polygon:setRunstep"
"void" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"polygon:setRunstep" ["char";"float"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "polygon:getRunstep"
"float" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"polygon:getRunstep" ["char"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "polygon:getPoints"
"list" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"polygon:getPoints" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "polygon:getArea"
"float" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"polygon:getArea" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "polygon:getAngle"
"list" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"polygon:getAngle" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "polygon:getParimeter"
"float" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"polygon:getParimeter" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "polygon:getCentroid"
"dot" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"polygon:getCentroid" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "polygon:getSides"
"list" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"polygon:getSides" [] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "polygon:intersect"
"list" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"polygon:intersect" ["shape"] env.contents.func_opt};;

env := {vars = StringMap.empty; funcs = StringMap.add "runset:mark" "void"
env.contents.funcs; get_call = ""; func_opt = StringMap.add "runset:mark"
["dot"] env.contents.func_opt};;
```

```
env := {vars = StringMap.empty; funcs = StringMap.add "runset:getRuncount"
"int" env.contents.funcs; get_call = ""; func_opt = StringMap.add
"runset:getRuncount" [] env.contents.func_opt};;

let translate (declarations, statements) =
    let rec py_of_expr = function
        Int(l) -> (PyInt(l), "int")
      | Float(l) -> (PyFloat(l), "float")
      | String(s) -> (PyString(s), "string")
      | Char(l) -> (PyChar(l), "char")
      | Bool(l) -> (PyBool(l), "bool")
      | Id(s) ->  (PyId(s), try StringMap.find s env.contents.vars with
Not_found -> raise(Failure("Undeclared Variable " ^ env.contents.get_call ^
":"^s)))

      | Binop(e1, o, e2) ->
              let result1 = py_of_expr e1 and result2 = py_of_expr e2 in
              let digit_op_match op a b =
              if
(((a="string"||a="int"||a="float")&&a=b)||a="list_ele"||b="list_ele") then
(true, a) else if ((a="float" && b="int")||(a="int" && b="float")) then (true,
"float") else raise(Failure("Undefined Operation: " ^ a ^ op ^ b)) in
              let eq_op_match op a b =
              if
(((a=b&&(a="string"||a="int"||a="float"||a="bool"||a="char"))&&a=b)||a="list_
ele"||b="list_ele") then (true, "bool") else raise(Failure("Undefined
Operation: " ^ a ^ op ^ b)) in
              let bool_op_match op a b =
              if (((a=b && a="bool")&&a=b)||a="list_ele"||b="list_ele") then
(true, "bool") else raise(Failure("Undefined Operation: " ^ a ^ op ^ b)) in
          let get_type_bi o =
          (match o with
        Add -> let ck = digit_op_match "+" (snd result1) (snd result2) in
(snd ck, "+")
      | Sub -> let ck = digit_op_match "-" (snd result1) (snd result2) in
(snd ck, "-")
      | Mult -> let ck = digit_op_match "*" (snd result1) (snd result2) in
(snd ck, "*")
      | Div -> let ck = digit_op_match "/" (snd result1) (snd result2) in
(snd ck, "/")
      | Mod -> let ck = digit_op_match "%" (snd result1) (snd result2) in
(snd ck, "%")
      | Exp -> let ck = digit_op_match "^" (snd result1) (snd result2) in
(snd ck, "**")
        | Equal -> let ck = eq_op_match "==" (snd result1) (snd result2) in
(snd ck, "==")
        | Neq -> let ck = eq_op_match "!=" (snd result1) (snd result2) in
(snd ck, "!=")
        | Less -> let ck = eq_op_match "<" (snd result1) (snd result2) in
(snd ck, "<")
        | Leq -> let ck = eq_op_match "<=" (snd result1) (snd result2) in
(snd ck, "<=")
        | Greater -> let ck = eq_op_match ">" (snd result1) (snd result2) in
(snd ck, ">")
```

```
        | Geq -> let ck = eq_op_match ">=" (snd result1) (snd result2) in
(snd ck, ">=")
        | And -> let ck = bool_op_match "&" (snd result1) (snd result2) in
(snd ck, "and")
        | Or -> let ck = bool_op_match "|" (snd result1) (snd result2) in
(snd ck, "or")
        ) in

          let result_op = get_type_bi o in
          (PyBinop((fst result1), snd result_op, (fst result2)), fst
result_op)
      | Minus(e) ->

            let result = py_of_expr e in

            let ck_minus x = if (x="int"||x="float") then x else
raise(Failure("Undefined Operation: -" ^ x)) in

            let tp = ck_minus (snd result) in

            (PyMinus(fst result), tp)

      | Not(e) -> let result = py_of_expr e in

      let ck_not x = if (x="bool") then x else raise(Failure("Undefined
Operation: !(" ^ x^")")) in

      (ck_not (snd result); (PyNot(fst result), snd result))



      | Noexpr -> (PyNoexpr, "none")

      | Dot(x, y) -> let result1 = py_of_expr x and result2 = py_of_expr y in

                   (PyCall("dot", [fst result1;fst result2]), "dot")

      | Get_Call(x, y) -> let result1 = py_of_expr x

                                    in (env := {vars = env.contents.vars;
funcs = env.contents.funcs; get_call = snd result1; func_opt =
env.contents.func_opt};

                                          let result2 =  py_of_expr y in

                                                (PyGet_Call(fst result1,fst
result2), snd result2))

      | Line(x) -> (PyCall("line", (List.map fst (List.map py_of_expr x))),
"line")

      | Circle(x) ->  (PyCall("circle", (List.map fst (List.map py_of_expr
x))), "circle")
```

```
      | Polygon(x) ->  (PyCall("polygon", (List.map fst (List.map py_of_expr
x))), "polygon")

      | Runset(x) ->  (PyCall("runset", (List.map fst (List.map py_of_expr
x))), "runset")

      | List(x) -> (PyList(List.map fst (List.map py_of_expr x)), "list")

      | ListEle(x1, x2) ->

                  let ln_ = try (x1, StringMap.find x1 env.contents.vars)
with Not_found -> raise(Failure("Undeclared List " ^ x1)) and  id_ =
py_of_expr x2 in

                        let ck_list ln id = if ((snd ln = "list")&&((snd id)
= "int" || (snd id)="list_ele")) then (PyListEle((fst ln),(fst id)),
"list_ele") else raise(Failure("List Error: " ^ (snd ln) ^ "[" ^ (snd id) ^
"]")) in

                        let ans = ck_list ln_ id_ in ans

      | Call(f, el) ->

          let gtc = env.contents.get_call in

          let result_el = List.map py_of_expr el in

          let returnType =  try StringMap.find (gtc ^ ":" ^ f)
env.contents.funcs with Not_found -> raise(Failure("Undeclared Fuction " ^
gtc ^ ":" ^ f))

      in  let func_opt_types = StringMap.find (gtc ^ ":" ^ f)
env.contents.func_opt

    in    let opt_match a b = if (a=b) then true else if (b="int" &&
a="float") then true else if (a="any"||b="list_ele") then true else if
(a="shape"&&(b="dot"||b="line"||b="circle"||b="polygon")) then true else
false

in        let opts_match a b = if (List.for_all2 opt_match a b) then true
else raise(Failure("Fuction Parameter Not Match\n" ^ "Required " ^ gtc ^ ":"
^ f ^ "(" ^ (String.concat "," func_opt_types) ^ ")\n" ^ "Get " ^gtc ^ ":" ^
f ^ "(" ^ (String.concat "," (List.map snd result_el)) ^ ")"))

in let mat = try opts_match func_opt_types (List.map snd result_el) with
Invalid_argument "List.for_all2" -> raise(Failure("Fuction Parameter Not
Match\n" ^ "Required " ^ gtc ^ ":" ^ f ^ "(" ^ (String.concat ","
func_opt_types) ^ ")\n" ^ "Get " ^gtc ^ ":" ^ f ^ "(" ^ (String.concat ","
(List.map snd result_el)) ^ ")")) in

          (PyCall(f,(List.map fst result_el)),returnType)
    in let addTab s = "\t" ^ s

in let ck_bool tp rb = if ((snd rb)="bool") then true else raise(Failure(tp ^
" Statement Need Bool Expression"))

    in let rec py_of_stmt = function

      Expr(e) -> (env := {vars = env.contents.vars; funcs =
env.contents.funcs; get_call = ""; func_opt = env.contents.func_opt};
PyExpr(fst (py_of_expr e)))

      | Assign(v, e, id) ->

      let result = py_of_expr e in

      (env := {vars = env.contents.vars; funcs = env.contents.funcs; get_call
= ""; func_opt = env.contents.func_opt};

       if (id=Noexpr)

       then  (env := {vars = StringMap.add v (snd result) env.contents.vars;
funcs = env.contents.funcs; get_call = env.contents.get_call; func_opt =
env.contents.func_opt}; PyAssign(v, fst result, PyNoexpr))

       else PyAssign(v, fst result, fst (py_of_expr id)))

      | PrintT(expr) -> (env := {vars = env.contents.vars; funcs =
env.contents.funcs; get_call = ""; func_opt = env.contents.func_opt};
PyPrintT(fst (py_of_expr expr)))

      | Print(expr) -> (env := {vars = env.contents.vars; funcs =
env.contents.funcs; get_call = ""; func_opt = env.contents.func_opt};
PyPrint(fst (py_of_expr expr)))

      | While(e, s) -> let rb = py_of_expr e in (ck_bool "While" rb;env :=
{vars = env.contents.vars; funcs = env.contents.funcs; get_call = "";
func_opt = env.contents.func_opt}; PyWhile(fst rb, List.map py_of_stmt
(List.rev s)))

      | For(e1, e2, s) ->

      let r1_ck_fori = function

            Id(s) -> (env := {vars = StringMap.add s "list_ele"
env.contents.vars; funcs = env.contents.funcs; get_call = ""; func_opt =
env.contents.func_opt}; py_of_expr e1)

      | _ -> raise(Failure("Geo Syntax Error Forloop"))

    in

      let r1 = r1_ck_fori e1 and r2 = py_of_expr e2 in
      let r2_ck_fori r2 = if ((snd r2)="list") then true else
raise(Failure("Geo Syntax Error Forloop")) in

      (r2_ck_fori r2;env := {vars = env.contents.vars; funcs =
env.contents.funcs; get_call = ""; func_opt = env.contents.func_opt};

            PyFor(fst r1, fst r2, (List.map py_of_stmt (List.rev s))))

      | Return(e) -> (env := {vars = env.contents.vars; funcs =
env.contents.funcs; get_call = ""; func_opt = env.contents.func_opt};
PyReturn(fst (py_of_expr e)))

      | If(e1, s1, s2) ->

      let rb = py_of_expr e1 in

      (ck_bool "If" rb; env := {vars = env.contents.vars; funcs =
env.contents.funcs; get_call = ""; func_opt = env.contents.func_opt};

                  PyIf(fst rb, (List.map py_of_stmt (List.rev s1)), (List.map
py_of_stmt (List.rev s2))))

      | Run(e, s) ->

      let re = py_of_expr e in

      let ck_id = function

            Id(i) -> if ((snd re)="runset") then i else raise(Failure("Run
must be followed by a variable Id with type 'runset' , not "^ (snd re)))

            | _ -> raise(Failure("Run must be followed by a variable with
type 'runset' "))

      in (env := {vars = env.contents.vars; funcs = env.contents.funcs;
get_call = ""; func_opt = env.contents.func_opt}; PyRun(PyId(ck_id e),
List.map py_of_stmt (List.rev s)))

    | Break -> (env := {vars = env.contents.vars; funcs = env.contents.funcs;
get_call = ""; func_opt = env.contents.func_opt}; PyBreak)



    in let rec translate_stmts = function

        [] -> []

      | hd::tl -> let result1 = py_of_stmt hd and result2 = (translate_stmts
tl) in (result1::result2)

    in let para_var paras_tp = env := {vars = StringMap.add  (fst paras_tp)
(snd paras_tp) env.contents.vars; funcs = env.contents.funcs; get_call =
env.contents.get_call; func_opt = env.contents.func_opt}
```

```
in let para_type fn paras_types = env := {vars = env.contents.vars; funcs =
env.contents.funcs; get_call = env.contents.get_call; func_opt =
StringMap.add fn paras_types env.contents.func_opt}

    in let py_of_func fdecl =  (env := {vars = env.contents.vars; funcs =
StringMap.add (":" ^ fdecl.fname) fdecl.tp env.contents.funcs; get_call =
env.contents.get_call; func_opt = env.contents.func_opt};

                                     List.map para_var fdecl.paras;

                                      para_type (":" ^ fdecl.fname) (List.map
snd fdecl.paras);

                                      let ans = {pyfname = fdecl.fname;
pyparas = List.map fst fdecl.paras; pybody = translate_stmts fdecl.body} in

                                      (env := {vars = StringMap.empty; funcs
= env.contents.funcs; get_call = ""; func_opt = env.contents.func_opt}; ans))




    in let rec translate_funcs = function

        [] -> []

      | hd::tl -> (py_of_func hd)::(translate_funcs tl)

    in let dofunc = ((List.rev (translate_funcs (declarations)))) and dostms
= (((translate_stmts (List.rev statements))))

    in (dofunc,dostms)
```

8.5 SAST

```
  type pyExpr =
      PyNoexpr
    | PyInt of int
    | PyFloat of float
    | PyBool of  bool
    | PyString of string
    | PyChar of char
    | PyId of string
    | PyBinop of pyExpr * string * pyExpr
    | PyCall of string * pyExpr list
    (*| PyDot of pyExpr * pyExpr*)
    | PyGet_Call of pyExpr * pyExpr
    (*| PyLine of pyExpr list
    | PyCircle of pyExpr list*)
    | PyList of pyExpr list
    | PyNot of pyExpr
    | PyMinus of pyExpr
    | PyListEle of string * pyExpr

  type pyStmt =
      PyExpr of pyExpr
    | PyReturn of pyExpr
    | PyIf of pyExpr * pyStmt list * pyStmt list
    | PyFor of pyExpr * pyExpr * pyStmt list
    | PyWhile of pyExpr * pyStmt list
    | PyPrint of pyExpr
    | PyPrintT of pyExpr
    | PyAssign of string * pyExpr * pyExpr
    | PyRun of  pyExpr * pyStmt list
    | PyBreak

  type pyFuncDecl = {
      pyfname : string;
      pyparas : string list;
      pybody : pyStmt list;
  }

  type pyProgram = pyFuncDecl list * pyStmt list
```

8.6 Code Generation

```
open Pyast
let prestring = ["from Tkinter import *\n";
                 "from sysgeo import *\n" ]

let finalstring = [""]


let translate (declarations, statements) =
  let rec string_of_expr = function
      PyInt(l) -> string_of_int l
    | PyFloat(l) -> string_of_float l
    | PyString(s) -> "\"" ^ s ^ "\""
    | PyChar(l) -> "\'" ^ (String.make 1 l) ^ "\'"
    | PyBool(l) -> let tran_bool x = if (x=true) then "True" else "False"
in (tran_bool l)
    | PyId(s) ->  s
    | PyBinop(e1, o, e2) ->
            let result1 = string_of_expr e1 and result2 = string_of_expr e2
in
        "(" ^ result1 ^ " "^ o ^ " "^result2 ^ ")"
    | PyMinus(e) ->
          let result = string_of_expr e in "-" ^ result
    | PyNot(e) -> let result = string_of_expr e in "not(" ^ result ^ ")"

    | PyNoexpr -> ""
    | PyGet_Call(x, y) -> let result1 = string_of_expr x and result2 =
string_of_expr y
                                  in result1 ^ "." ^ result2
    | PyList(x) -> "[" ^  String.concat ", " (List.map string_of_expr x) ^
"]"
    | PyListEle(x1, x2) ->
              let id = string_of_expr x2 in x1 ^ "[" ^ id ^ "]"
    | PyCall(f, el) ->
        let result_el = List.map string_of_expr el in
        f ^ "(" ^ String.concat ", " result_el ^ ")"


  in let addTab s = "\t" ^ s
  in let rec string_of_stmt = function
    PyExpr(e) -> (string_of_expr e)  :: []
    | PyPrintT(expr) -> ("print " ^ (string_of_expr expr)) :: []
    | PyPrint(expr) -> ("print " ^ (string_of_expr expr)) :: []
    | PyWhile(e, s) -> let rb = string_of_expr e in ("while (" ^ rb ^
"):"):: (List.map addTab (List.concat ((List.map string_of_stmt s))))
    | PyFor(e1, e2, s) ->
    let r1 = string_of_expr e1 and r2 = string_of_expr e2 in
          ("for " ^ r1 ^ " in " ^ r2 ^ ":")
          :: (List.map addTab (List.concat ((List.map string_of_stmt s))))
    | PyReturn(e) -> ("return " ^ (string_of_expr e)) :: []
    | PyAssign(v, e, id) ->
    let result = string_of_expr e in
    if (id=PyNoexpr)
          then ((v) ^ " = " ^ result) :: []
          else ((v) ^ "[" ^ (string_of_expr id) ^ "] = " ^ result) :: []
    | PyBreak -> ["break"]
    | PyRun(e, s) -> let str=string_of_expr e in (("def runfun__(" ^ str ^
"):") ::  (List.map addTab (List.concat ((List.map string_of_stmt s))))) @
["drawmain(" ^ str ^ ")"]
    | PyIf(e1, s1, s2) ->
    let rb = string_of_expr e1 in
              match s2 with
              [] -> ("if " ^ rb ^ ":") ::
                    (List.map addTab (List.concat ((List.map
string_of_stmt s1))))
              |_ -> (("if " ^ rb ^ ":") ::
                    (List.map addTab (List.concat ((List.map
string_of_stmt s1))))) @
                    ("else:" :: (List.map addTab (List.concat ((List.map
string_of_stmt s2)))))



  in let rec translate_stmts = function
      [] -> []
    | hd::tl -> let result1 = (string_of_stmt hd) and result2 =
(translate_stmts tl) in (result1 @ result2)
  in let string_of_func fdecl =  "def " ^ fdecl.pyfname ^ "(" ^
(String.concat ", " fdecl.pyparas) ^ "):\n\t" ^ (String.concat "\n\t"
(translate_stmts fdecl.pybody))
```

```
    in let rec translate_funcs = function
        [] -> []
      | hd::tl -> (string_of_func hd)::(translate_funcs tl)
    in let dofunc = (String.concat "\n" ( (translate_funcs (declarations))))
  and dostms = (String.concat "\n" ((translate_stmts (statements))))
    in let tra = dofunc ^ "\n" ^ dostms ^ "\n"
    in (String.concat "" prestring) ^ tra ^ (String.concat "" finalstring)
```

8.7 Tests

8.7.1 test-assignments.g

```
@panel panel1
a=3;
b=3;
c=123e-1;
d="Ocaml sucks!";
e='@';
f=3.4*2==4.;
printT(a);
printT(b);
printT(c);
printT(d);
printT(e);
printT(f);
@end
```

8.7.2 test-circle.g

```
@panel panel1
dot1=[3,5];
circle1=circle([3,5],5);
circle2=circle([0,0],5);
printT(circle1);
printT(circle2);
printT(circle2.intersect(circle1));
printT(circle2.intersect(line([0,2],[1,2])));
printT(circle2.intersect(line([0,2],[1,2],-1,1)));
printT(circle2.intersect(line([0,-2],[0,2])));
printT(circle2.intersect(line([0,-2],[0,2],0,0)));
printT(circle1.getRadius());
printT(circle2.getCenter());
circle1.setCenter([2,2]);
printT(circle1);
circle2.setRadius(3);
printT(circle2);
printT(circle2.getPointbyarc(PI/4));
@end
```

8.7.3 test-comparison.g

```
@panel panel1
printT(5>3);
printT(5==5 & 3<2);
a=4.3;
b=8.6/2.0;
c=a==b;
printT(!c);
printT(a==b);
printT(false | true);
printT(!false & true);
@end
```

8.7.4 test-dot.g

```
@panel panel1
dot1=[3,5];
dot2=[5,7];
printT(dot1.distance(dot2));
printT(dot1.getX());
printT(dot1.getY());
@end
```

8.7.5 test-fib.g

```
@panel fib
function fib(x:int):int:
      if (x == 1):
            return(1);
      else:
            if (x == 2):
                  return(1);
            else:
                  return(fib(x-1)+fib(x-2));
            end
      end
end
i = 1;
while (i < 10):
      print(fib(i));
      i = i+1;
end
@end
```

8.7.6 test-for.g

```
@panel panel1
for i in {1,"zcCHAO",2,"zyLUO",3,"qWANG",4,"ycZHAO",true,';'}:
printT(i);
end
@end
```

8.7.7 test-function.g

```
@PANEL PANEL1
FUNCTION TEST1(STR:STRING):BOOL:
IF(STR=="PLT"):
RETURN TRUE;
ELSE:
RETURN FALSE;
END
END

FUNCTION TEST2():VOID:
IF(TEST1("PLT")):
PRINTT("IS PLT");
ELSE:
PRINTT("IS NOT PLT");
END
END
TEST2();
@END
```

8.7.8 test-gcd.g

```
@PANEL GCD
FUNCTION GCD(A:INT, B:INT):INT:
      IF (A<B):
            RETURN (GCD(B,A));
      ELSE:
            IF (A == B):
                  RETURN (A);
            ELSE:
                  RETURN(GCD(A-B, B));
            END
      END
END
PRINT(GCD(70,28));
PRINT(GCD(147,21));
@END
```

8.7.9 test-if.g

```
@PANEL PANEL1
IF(5==3):
PRINTT("5==3");
ELSE:
IF(5>3):
PRINTT("5>3");
ELSE:
PRINTT("5<3");
END
END
@END
```

8.7.10 test-line.g

```
@PANEL PANEL1
```

```
dot1=[3,5];
line1=line(4,5);
line2=line([1,2],[2,3]);
line3=line(4,5,-5,5);
line4=line([0,-2],[0,2],0,0);
printT(line1);
printT(line2);
printT(line3);
printT(line4);
printT(line2.intersect(line4));
printT(line3.distance(dot1));
printT(line4.length());
printT(line2.pointAway([2,3],5));
printT(line4.pointAway([0,0],1));
printT(line3.getEndpoints());
printT(line3.getMidpoint());
printT(line1.isParallel(line2));
printT(line1.getX(5));
printT(line4.getX(1));
printT(line2.getY(0));
line2.setRunstep('a',0.5);
line3.setRunstep('b',1.0);
line3.setRunstep('x',1.0);
@end
```

8.7.11 test-list.g

```
@panel panel1
d={1,2,3,4,5,6};
e={"str",45,'c',3.4};
printT(d);
printT(d#[2]);
printT(e);
printT(e#[0]);
@end
```

8.7.12 test-operations.g

```
@panel panel1
//int+int
printT(1+1);
//float+float
printT(.1+1.0);
//int+float
printT(1+1.0);
//float-int
printT(.1-5);
//int
printT(1+2*3-4/2);
//float
printT(1.+2.*3-4./3);
//int
printT(1-5*2^3+4*5/2*7%3);
//float mix with int
printT(1.+2.*3-4/3);
//string+string
```

```
printT("Hello "+"World");
@end
```

8.7.13 test-polygon.g

```
@panel panel1
lI={[0,0],[5,0],[5,3],[0,3]};
l2={[0,0],[2,2],[0,4],[-2,2]};
p1=polygon(lI);
p2=polygon(l2);
print(p1);
print(p1.getPoints());
print(p1.getArea());
print(p1.getAngle());
print(p1.getParimeter());
print(p1.getCentroid());
print(p1.getSides());
print(p1.intersect([5,6]));
print(p1.intersect(line([0,0],[2,2])));
print(p1.intersect(circle([0,0],3)));
print(p1.intersect(p2));
@end
```

8.7.14 test-print.g

```
@panel panel1
printT(123456);
printT(123.456);
printT(5e-5);
printT('a');
printT('%');
printT(true);
printT(false);
printT("Hello world!");
@end
```

8.7.15 test-qsort.g

```
@panel qsort
function qsort(a:list, l:int, r:int):list:
      i = l;
      j = r;
      mid = (l+r)/2;

      while (i <= j):
            while (i <= j & a#[i] < a#[mid]):
                  i = i+1;
            end
            while (i <= j & a#[j]> a#[mid]):
                  j = j-1;
            end

            if (i <= j):
                  k = a#[i];
                  a#[i] = a#[j];
                  a#[j] = k;
```

```
                I = I+1;
                J = J-1;
            END

        END


        IF (L < J):
            A = QSORT(A, L, J);
        END
        IF (I < R):
            A = QSORT(A, I, R);
        END
        RETURN(A);
END
B = {3,7,8,32,1,4,7,9,2,5};
B = QSORT(B, 0, LEN(B)-1);
PRINT(B);
@END
```

8.7.16 test-while.g

```
@PANEL PANEL1
COUNT=0;
WHILE(COUNT<11):
PRINTT(COUNT);
COUNT=COUNT+2;
END
@END
```

8.7.17 error-semantics1.g

```
a=foo(3);
```

8.7.18 error-semantics2.g

```
function foo(i:int):int:
  return 1;
end;

foo(0.1);
```

8.7.19 error-syntax1.g

```
function foo(i:int):
```

8.7.20 error-syntax2.g

```
end
```

8.7.21 error-syntax3.g

```
a=3
```

8.7.22 error-syntax4.g

```
if():
```

8.7.23 error-syntax5.g

```
if()

else
```