Sketchpad Graphics Language

Yan Peng, Yichen Liu, Zhongyu Wang
Overview

- Compass-and-straightedge construction
- Dependence Relationship
- Subfunction
- Symbol Table
- Statically scoped, Byval and Byref.
Simple, but Strong!
Geometry Statements

- Geometry Statements are usually not about a certain graph, but about a set of graphs in some certain constraint.
- E.g. the three perpendicular bisector of the three edges of a triangle meet at one point.
- Our language gives an easy way to check different instances in a certain constraint. :Move some part of the graph, the other parts will reshape.
Types and Operators

- integer, float, string, bool
- Point, Line, Circle.
- Basic arithmetic and logic operators.
- Basic library functions
Basic Syntax:

- A combination of C++ and VB
Structure of the program
Some Features

- sub functions
- Anywhere variable declaration
- Statically scoped
- By value and by reference.
Code generation

- Symbol Table: Managing Scopes
- Sketch Pad: managing dependency relationship
- Function translation: put them ahead, name them according to their appearance order.
- Code generator need to keep track of the environment:
Code generation

```c
int function1(int n){
    int _return_value; Symbol_Table_Node* parent=_CurSymbolTable;
    Symbol_Table_Node* _CurSymbolTable=new Symbol_Table_Node(parent);
    _CurSymbolTable->add_var(n,"n");
    _CurSymbolTable->add_function((func_pointer)&function1,"fib");

    _return_value=((int(*)(int,int))_CurSymbolTable->get_function("fib"))(*((int*)_CurSymbolTable->get_var("n")),1.5);
    goto endline;
    endline:
    delete _CurSymbolTable;
    return _return_value;
}
```
Demo: snowflake fractal
integer snowflake(integer depth)
integer n(Point A, Point B, Point Center, integer depth)
    if depth==0 then
        library.LineST(A, B, 0);
        return 1;
    end
    Point C=onethird(A, B, 1);
    Point D=onethird(A, B, 2);
    Line 1=library.LineST(A, C, 0);
    1.setvisible(false);
    Line 2=library.LineST(D, B, 0);
    2.setvisible(false);
    Circle c1=library.DrawCircle(C, 11);
    c1.setvisible(false);
    Circle c2=library.DrawCircle(D, 12);
    c2.setvisible(false);
    Point K=library.intersect(c1, c2, true);
    Point F=library.intersect(c1, c2, false);
    if Dist(Center, K)>Dist(Center, F) then
        F.setvisible(false);
    else
        F.setvisible(true);
        E=F;
    end
    Point O1=library.PointXY((C.getX())+D.getX()+E.getX())/3.0, (C.getY())+D.getY()+E.getY())/3.0);
    01.setvisible(false);
    iter(A, C, Center, depth-1);
    iter(C, E, O1, depth-1);
    iter(B, D, O1, depth-1);
    iter(D, B, Center, depth-1);
    return 1;
end
Point A=library.PointXY(200.0, 200.0);
Point E=library.PointXY(600.0, 200.0);
Point D=library.PointXY(400.200+200*1.73205080800);
Point B=library.PointXY(400.200+200*1.73205080800/3.0);
0.setvisible(false);
iter(A, B, O, depth);
iter(E, C, O, depth);
iter(C, A, O, depth);
return 1;
end
snowflake(4);
library.point();
Lessons Learned

- Tests are never enough
- Keep a good structure so that it can be easily modified when you have new ideas.
- Communication is important in group work
Thanks!