COLOGO
A Graph Language

Advisor: Stephen A. Edwards

Shen Wang (sw2613@columbia.edu)

Lixing Dong (ld2505@columbia.edu)

Siyuan Lu (sl3352@columbia.edu)

Chao Song (cs2994@columbia.edu)

Zhou Ma (zm2167@columbia.edu)

Description:

Our COLOGO language is an effective programming language for drawing 2D graphics. The COLOGO language is designed in spirit of low threshold, which enables easy entry by novices and yet meet the needs of high-powered users. We can use COLOGO for education as it contains basic computer concepts appropriate for beginners. We can also draw interesting pictures and design complicated logos with COLOGO so that the language could be widely used for entertainment or commercial area.

Features:

Euclidean: COLOGO operates in a Euclidean space using relative measures and angles, without an origin, unlike coordinate-addressed systems such as Cartisian geometry.

Functional: In our COLOGO language, users can create their own functions to perform a specific task. This helps programmers to decompose the complex program to simple steps. Also, this feature allows users to reuse the code across different programs.

Recursive: Recursion is supported in our COLOGO language. This allows users to simplify their code by dividing a problem to subproblems of the same type.

Iridescent: COLOGO support drawing lines of different colors and line width, making your drawing experience more colorful.

Objectives:

The main goal of our programming language is to provide an easy way to draw 2D graphics. These graphics, and hence our language, can be used for representing mathematical formulas, teaching geometric concepts, simple arithmetical operation and simulation of robots routing. Also, COLOGO is an appropriate language for teaching basic programming language concepts. Basic data types will be supported in COLOGO, such as integers, floats, and strings. Some simple data structures like list will also be implemented in it.

Sample Code:

```
function void triangleSpiral( int intEdgeToDraw, double dblEdgeLength )
{
        FD dblEdgeLength;
        RT 2 / 3 * Pi;
        if ( intEdgeToDraw > 0 )
        {
                triangleSpiral( intEdgeToDraw - 1, dblEdgeLength * 0.9 );
        }
}

function void main()
{
    int intStarCount = 20;
    double dblLength;
    double dblAngle;
    for ( int intStarIndex = 0; intStarIndex < intStarCount; intStarIndex ++ )
    {
            dblLength = rand() * 10;
            dblAngle = rand() * Pi;
            RT dblAngle;
            PU;                         // pen up
            FD dblLength;
            PD;                         // pen down
            triangleSpiral( 10, dblLength );
    }
}
```

Syntax:

In the sample code, the following syntaxes are involved:

Draw command (draw keywords followed by 0 or more parameters):

```
FD dblEdgeLength;       // Draw line as moving forward
RT dblAngle;            // Turn right by dblAngle
PU;                     // Pen up, move without drawing
PF;                     // Pen up/down flip
```

Variable declaration (variable class followed by variable name):
double dblLength;

Assignment:
*dblLength = rand() * 10;*

Variable definition (variable type followed by variable name and assignment):
int intStarIndex = 0;

Function definition (keyword *function* followed by return type, function name, parameter list and function body which is surrounded by braces):

```
function void triangleSpiral( int intEdgeToDraw, double dblEdgeLength )
{
        ...
}
```

Function call (function name followed by parameter list):
*triangleSpiral(intEdgeToDraw - 1, dblEdgeLength * 0.9);*

Comments (start with //):
// pen up

Flow control (iterational and conditional flow control (for, while, if) structured similarly to C program language):

```
for ( int intStarIndex = 0; intStarIndex < intStarCount; intStarIndex ++ )
{
        ...
}

if ( intEdgeToDraw > 0 )
{
        triangleSpiral( intEdgeToDraw - 1, dblEdgeLength * 0.9 );
}
```