Progress Report of COMP3021 Project

EasyDraw: A Programming Language for Drawing

The Hong Kong Polytechnic University

LI Jinlin:18081569d
LIU Yanting: 18082826d
ZHANG Caiqi: 18085481d

Contents

Contribution statement

Contribution

Liu Yanting: Compiler design and language design. (Task 2, Task 3, Task 4, Task7)
LI Jinlin : Language Design and Test. (Task 3, Task 5, Task 6 and Task 7)
Zhang Caqi : Language Design and Test. (Task 2, Task 5, Task 6 and Task 7)

Section of report

Liu Yanting : Implementation (Majority), language Design.
Zhang Caiqi : Introduction, Problem analysis, Reuqirement analysis
LI Jinlin : Implementation (Minority), Testing Cases, and Conclusion

In general, we divided the workload fairly and efficiently. We worked as a team and finished the project with the whole team effort.

Introduction

Drawing pictures is one of the most commonly used functions on computers in our daily life. There are many programming languages that can be used to draw pictures. For example,in C++ and Java we can use the drawing functions by calling library functions. Meanwhile, there are also some programming languages that are designed especially for drawing, such as Logo.

However, these programming languages all have their own strength and drawbacks. In this project, we developed our own programming language: EasyDraw, which aims to combine the strengths of the current drawing programming languages. The EasyDraw has three of the most significant advantages: it is easy to use, straightforward and fully functional.

In this progress report, we will first analyze the advantages and disadvantages of current programming language for drawing and summarize the requirements of our own language. After that, we will illustrate the design of data types supported by our language, the syntax and the semantics.

Problem analysis

In this section, we will compare existing programming languages on drawing, such as Turtle in Python, Swing in Java, TikZ in LateX and Logo. We will also give an easy demo to each language to let readers have a direct impression. Then we will compare these four programming languages horizontally by using language evaluation criteria (e.g., readability, writability, reliability). Finally, we will define the requirements of our project from functional and non-functional perspectives.

Turtle in Python:

Turtle is a pre-installed Python library that enables users to create pictures and shapes by providing them with a virtual canvas. The onscreen pen that people use for drawing is called the turtle and this is what gives the library its name. A simple coding demo to draw a square is as follows:

```
import turtle
def draw_square():
    brad = turtle.Turtle()
    brad.forward(100)  # forward takes a number which is the distance to move
    brad.right(90)  # turn right
    brad.forward(100)
    brad.right(90)
    brad.forward(100)
    brad.right(90)
    brad.forward(100)
    brad.right(90)
```

Swing in Java:

Swing in java is part of Java foundation class which is lightweight and platform independent. It is used for creating window based applications. It includes components like buttons, scroll bar, text field etc. Putting together all these components makes a graphical user interface. A simple coding demo to draw a square is as follows:

```
import javax.swing.*;
import java.awt.*;
public class Test extends JPanel {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                JFrame frame = new JFrame();
                frame.add(new Test());
                frame.setVisible(true);
                frame.pack();
            }
        });
    }
    public Dimension getPreferrdSize() {
        return new Dimension(200, 200);
    }
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawRect(10, 10, 150, 150);
    }
}
```

TikZ in LaTeX:

TikZ is a LaTeX package that allows people to create high quality diagrams—and often quite complex ones too. A simple coding demo to draw a square is as follows:

```
# This will draw a square.
\begin{tikzpicture}
\draw (0,0) -- (4,0) -- (4,4) -- (0,4) -- (0,0);
\end{tikzpicture}
```

Logo:

The LOGO programming language is designed to help kids learn programming hands on. Instead of memorizing theory or using complicated programming structures, LOGO users learn programming basics with simple words and directions. An object, usually a turtle, might be directed to move forward 20 steps. A simple coding demo to draw a square is as follows:

```
FD 100 RT 90
FD 100 RT 90
FD 100 RT 90
FD 100 RT 90
```

Comparison table:

According to our analysis and the feedback from the users collected from the internet, we conclude the following table. The marks range from 1 to 5, from the lowest to the highest. We can notice that all of them have very satisfied reliability but the readability and writability need to be improved. The EasyDraw is thus designed to combine their advantages and improve their disadvantages.

	readability	writability	reliability
Turtle	3	3	5
Swing	2	2	5
LaTeX	3	3	5
Logo	2	1	5

Requirements of EasyDraw:

Non-functional requirements:

- It is easy to use. It is better to be a hands on tool. People only need a little practice to get familiar with it.
- It should be straightforward. The instructions should be easy to understand and in easy words. Users can memorize them easily.
- It should be fully functional. It can fulfill almost all the needs for the basic requirements of people.

Functional requirements:

- Be able to set canvas size, background color.
- Be able to draw pictures at designated locations.
- Be able to draw any polygons.
- Be able to **quickly** draw basic graphics:
    - Straight line,
    - Rectangle,
    - Square,
    - Circle,
    - Oval…
- Be able to fill in the shapes with different colors
- Be able to put many drawings inside a single canva
- Be able to allow users to customize, such as
    - Line color,
    - Thickness,
    - Graph's length and width...

Why to use EasyDraw?

There are two most significant reasons that you should choose the EasyDraw.

- **It is easy to use and very straightforward.** The keywords and the grammar are easy to remember. Most of the keywords are in natural language. It is a hands on tool.
- **The EasyDraw has well designed encapsulation.** It takes less lines of codes to draw the graph than the existing programming languages.

Here we give two examples here. More examples are in the "Testing" section.

Compared with Swing

If we want to draw a line in swing, we must first calculate (x, y) then use the Draw line. But in our language, we only need to determine the direction then use forward.

Swing Code:

```
x2 = pen.X - Math.abs((Math.cos(Math.PI*(pen.Direction/180.0))) *
distance);
y2 = pen.Y - Math.abs((Math.sin(Math.PI*(pen.Direction/180.0))) *
distance);
g2.drawLine(pen.X, pen.Y, (int)x2, (int)y2);
```

Our Code:

```
Set Pen Direction Degree
Forward Distance
```

Compared with Turtle

In addition to Turtle, we implement Stamp and Brush. For example, we want to draw an Rectangle.

Turtle Code:

```
import turtle
turtle.title('Rectangle')
turtle.begin_fill()
turtle.fillcolor('#069')
for x in range(4):
    turtle.forward(100 + x*40)
    turtle.right(90)
turtle.end_fill()
turtle.hideturtle()
turtle.done()
```

Our code:

```
Start Brush Color R G B
Draw Rectangle X Y Length Width
```

Clearly, we only have two lines of code, which is much easier..

Language design

Language Overview

The new language is the tool for simple draw. It has four Global Static Data: Canvas, Pen, Stamp and Brush. Canvas is the base for a program, programmers must first establish it in order to begin the program. Once the Canvas is set, the programmer can do the operation for Pen and Brush, like the library turtle of python.

A programmer can

1. set the direction of Pen and let it draw a simple line,
2. set the features of base (Canvas)
3. or he can set the color of brush to set an area to be the color of brush. The
4. draw a frame on the canvas, like a rectangle frame, by using stamp

Data types and semantic

We only have two kinds of data types, which are objects and integers.

- The objects only have three distinct values : Canvas, Pen and Brush. Canvas is used to contain the picture, pen is used to draw the line and Brush is used to draw or erase a particular size of area.
- The integers are used to determine many attributes in the language. It will represent the pixels when setting the size or thickness, and moving in a direction as well. When

the program tries to set a direction or rotate, it will set the angle based on degrees (from 0 to 360, if the number is above 360, the effect will be the same as in 0 to 360 since this function is periodic). When we are going to set the color, the integers are set for 8-bit color.

EasyDraw does not need very complex design of scope and value assignment, and the scope is relatively simple compared to the existing language, such as java and C. The design of semantics is succinct in order to spare the resource for other parts. The scope is basically the global scope. As to the type checking, it is also very straightforward according to the types given above. We will discuss more in the following sections.

Syntax of EasyDraw in EBNF

```
<digit excluding zero> = "1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
digit = "0"| <digit excluding zero>
< int > = "0" | <digit excluding zero> , { <digit> }
+ = ' ', {' '}
< opi > = "Start"| "Stop" | "Set"
< opm > = "Forward" | "Spin" | "Draw" | "Erase" | "Rotate" | "Frame"
< shape > = "Rectangle" | "Circle" | "Oval" | "Dot"
< direction > = "Clockwise" | "Anticlockwise"
< obj >  = "Canvas" | "Pen" | "Brush" | "Stamp"
< Features >  = "Color" | "Thickness" | "Direction"
< expr > = < int > [+ < expr >]
< iniExpr > =  < Features > +  <expr>
< oprExpr > =  <shape> | <direction>
< iniCmd > = < opi > + < obj > + [ <iniExpr> ]
< oprCmd > = < opm > + [ <oprExpr> ] + expr
< cmd > = <iniCmd> | <oprCmd>
```

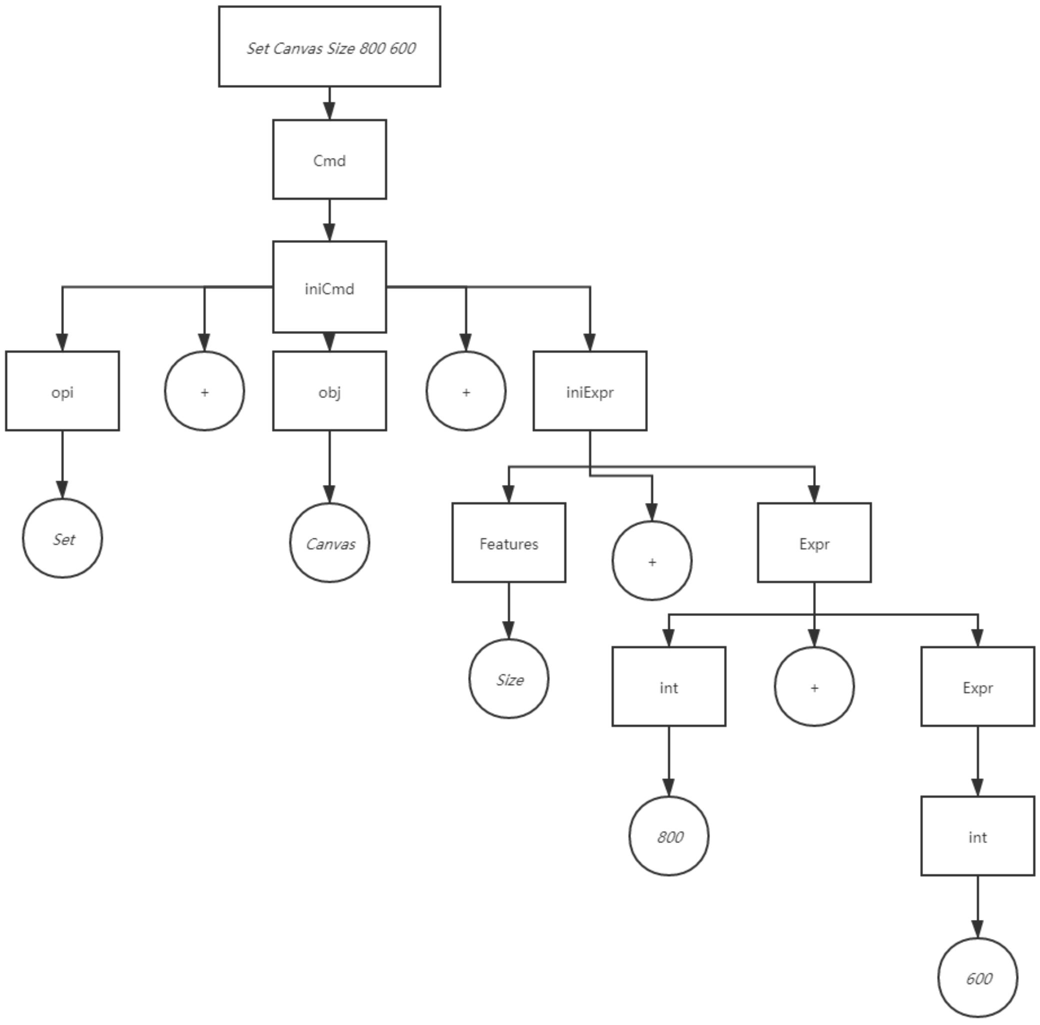
Note

1. + means space with length larger than 1
2. Command has two kinds, initialization operation and moving operation. For example, "Start Canvas" is the first kinds of commands and "Forward 100" is the second kinds of commands
3. <opi> is the operation for initialization operation and <opm> is for moving operation
4. <iniExpr> and <oprExpr> are two different expression for two kinds of command relatively
5. For convenience, <int> will be any natural number in the illustration
6. In the EBNF formula, "[]" means optional and "{}" means repetition, just like slides.

Examples

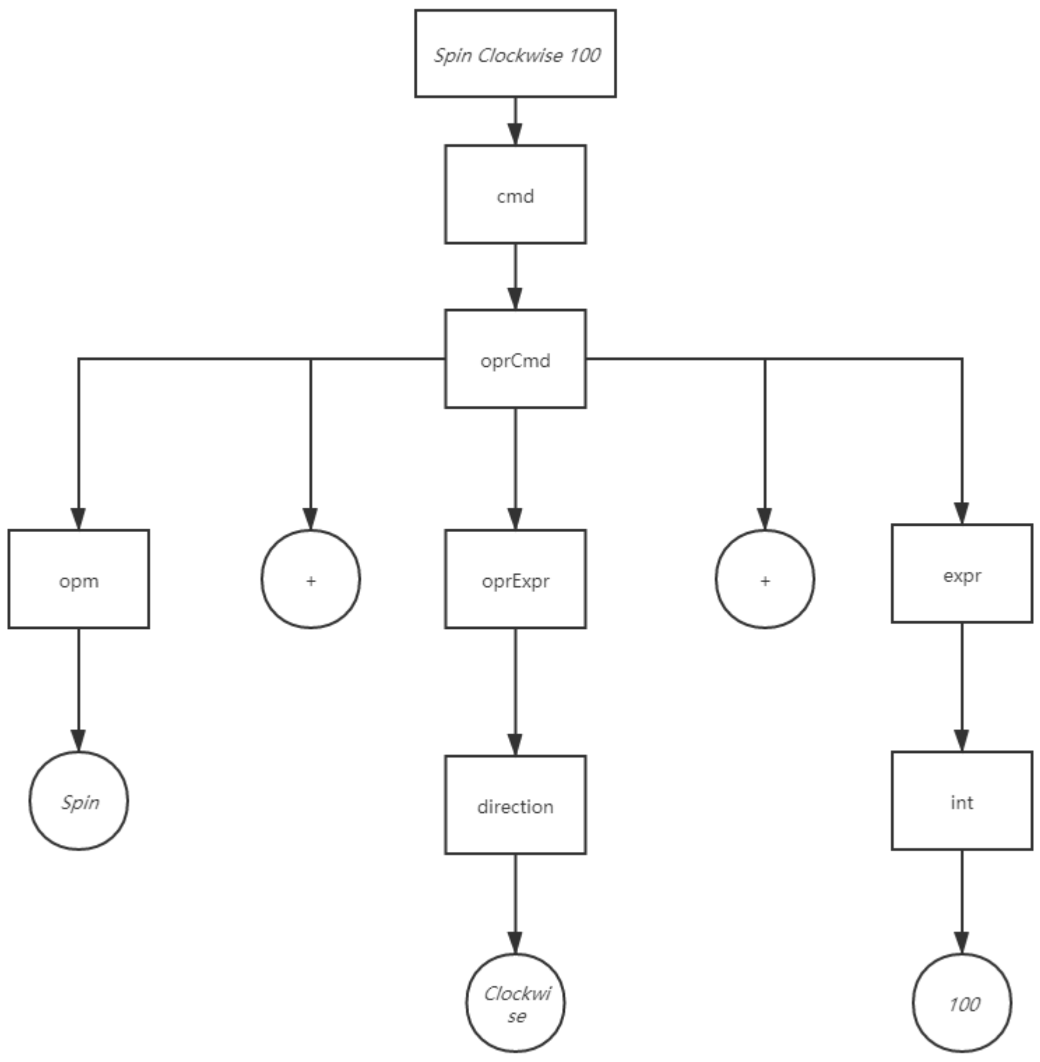
Here we will offer several examples

Example 1 : Set Canvas Size 800 600



Notice + is for space and we make it easier to represent natural numbers in illustration. The detailed way to represent numbers can be seen at EBNF.

Example 2 : Spin Clockwise 100



Grammar

We differ our data type to obj (Canvas, Pen, and Brush) and integers. In the following rules, Obj is for Canvas, Pen, and Brush and Int is for integers. Below is command rules

1. Start/Stop Obj
2. Start Obj Color red: Int yellow: Int blue: Int
3. Start Canvas Size length: Int width: Int
4. Start Pen Thickness thickness: Int
5. Start Pen Direction degree: Int
6. Set Pen Thickness thickness: Int
7. Set Canvas Size length: Int length: Int
8. Set Obj Color red: Int yellow: Int blue: Int
9. Set Pen Direction degree: Int

10. Forward/Backward distance: Int
11. Spin Clockwise/Anticlockwise degree: Int
12. Rotate x: Int y: Int degree: Int
13. Draw/Frame Rectangular x: Int y: Int long: Int short: Int
14. Draw/Frame Circle x: Int y: Int r: Int
15. Draw/Frame Oval x: Int y: Int long: Int short: Int
16. Draw Dot x: Int y: Int
17. Erase Rectangular x: Int y: Int long: Int short: Int
18. Erase Circle x: Int y: Int r: Int
19. Erase Oval x: Int y: Int long: Int short: Int
20. Erase Dot x: Int y: Int

Attribute Grammar

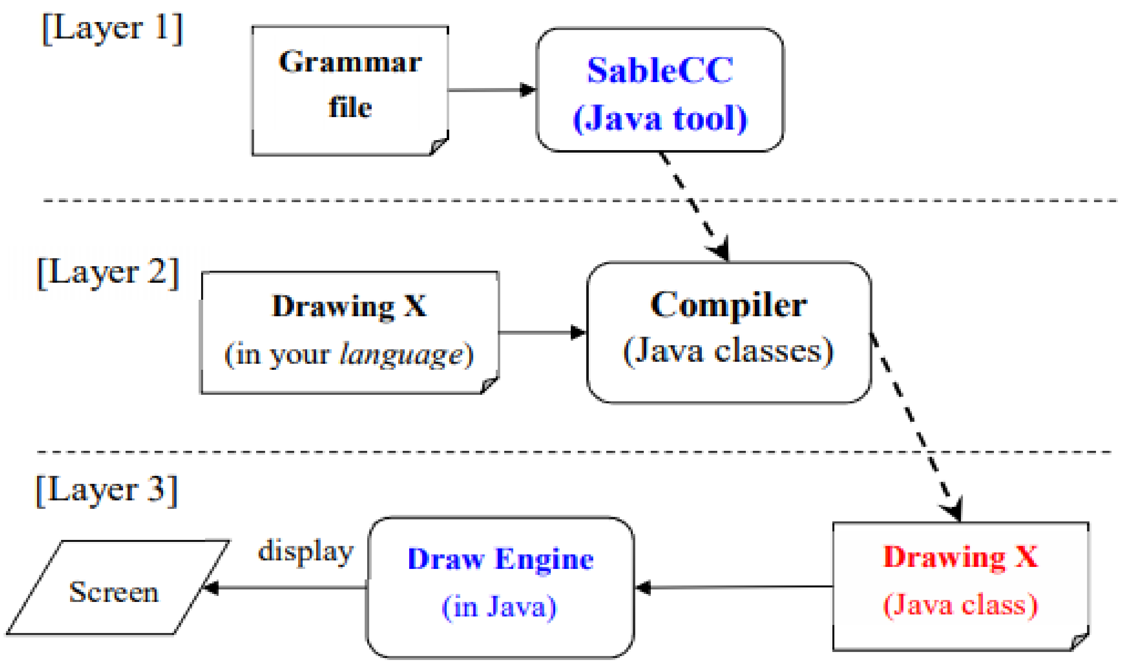
expr = int [expr.value = int.value]

Scope rules

Every object is a Global Static Data. Every integer is a constant value.

Implementation

Overview



We follow the implementation flow given in the project description with little changes to suit our language. We firstly write our grammar file, then use SableCC to get the basic semantic of our grammar. Then we Modify compiler classes to implement semantic analysis and code generation. We rewrite the Compiler and we implement another class called “**Interpreter”** to handle output from the Compiler and transform it to the final picture output (A png image). During this process, the Interpreter class is “Drawing X” and the Draw Engine we used in Java is mainly **Swing components**. (Notice that our output screen is a png picture to save memory.)

Firstly, we have made a grammar file and then use sablecc java tools to generate grammar class. Here is our Grammar text.

```
Package Paintlang;

Helpers
tab = 9;
cr = 13;
lf   = 10;
digit = ['0'..'9'];

Tokens
number = ('0' | [digit - '0'] digit*);
```

```
object = ('Canvas' | 'Pen' | 'Brush' | 'Stamp');
opi = ('Start' | 'Stop' | 'Set');
opm = ('Forward' | 'Backward' | 'Spin' | 'Rotate' | 'Draw' | 'Erase' | 'Frame');
shape = ('Rectangle' | 'Circle' | 'Oval' | 'Dot');
features = ('Color' | 'Thickness' | 'Position' | 'Direction' | 'Size');
direction = ('Clockwise' | 'Anticlockwise');
blank = (' ' | 9 | 13 | 10)+;

Ignored Tokens
blank;

Productions

program =
        cmd+;

cmd =
        {inicmd} opi object |
        {multiinicmd} opi object features expr |
        {oprcmd} opm expr |
        {oprcmdshape} opm shape expr |
        {oprcmddirection} opm direction expr;

expr =
        {number} number |
        {multinumber} expr number;
```

We use Tokens to determine the keyword we need. Furthermore, in productions, we use cmd to represent the normal command and expr to determine normal expression, that is composed with numbers.

Compiler

Firstly, we define several classes to abstract the data and function.

Pen

The Pen has several attributes `(Thickness, XY coordinate, RGB color and direction)` and `"activated"` to determine whether it is used

Brush and Stamp

The class Brush and Stamp have the same main attribute `(RGB color)`, besides, it does not need direction since it only gives an area with color.

Function Paint (Interpreter)

Then the compiler will interpret all the words in the input grammar.

```
static void paint(String par){
        String[] Parser = Par.split(" ");
        Pen pen = new Pen();
        Brush brush = new Brush();
        Brush stamp = new Brush();
        BufferedImage img = new BufferedImage(1000, 618,
BufferedImage.TYPE_INT_ARGB);
        Graphics2D g2 = img.createGraphics();
        int i = 0;

        while(i < Parse.length){
            switch(Parse[i]){
                case "Set": // implementation
                    break;
                case "Start": // implementation
                    break;
                case "Stop": // implementation
                    break;
                case "Forward": // implementation
                    break;
                case "Backward": // implementation
                    break;
                case "Rotate": // implementation
                    break;
                case "Draw": // implementation
                    break;
                case "Erase": // implementation
                    break;
                case "Frame": // implementation
                    break;
                default: // implementation
            }
        }

        File f = new File(".\\Output.png");
        try {
            ImageIO.write(img, "png", f);
        } catch (Exception e) {
            e.printStackTrace();
        }
        g2.dispose();
    }
```

The Program will first read the result of the parser, and initialize the Pen, Brush and Stamp object. Then corresponding to the operation of the program, like “Set” or “Forward”, then use the Engine to draw them.

Testing Cases

Basic functions

Canvas setting

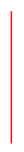
```
Start Canvas Size 1000 600
Set Canvas Color 255 255 255
Set Canvas Size 1800 1600
Start Pen Position 500 500
Set Pen Thickness 4
Set Pen Color 255 0 0
Stop Pen
Stop Canvas
```

We use this function as our basic setting. It will set the size of background to 1000*600, color to white.
Then the program reset the size to 1800 1600.
Then the program set the Pen Position to 500 500, with thickness 4 and red color.
Now since we have not done any operation, we will have a clean picture without any figure.

Pen

Now we consider command Forward X. It will lead the pen in X distance.

```
Forward 400
```



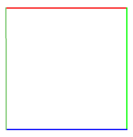
Spin Clockwise / Anticlockwise X

Spin C/A X means spin the Pen in X degree by clockwise direction or anticlockwise direction.

```
Set Pen Color 255 0 0
Forward 400
Set Pen Color 0 255 0
Spin Anticlockwise 90
Forward 400
Set Pen Color 0 0 255
```

```
Spin Anticlockwise 90
Forward 400
Spin Anticlockwise 90
Set Pen Color 123 200 111
Forward 400
```

This graph illustrate a rectangular drawn by Pen

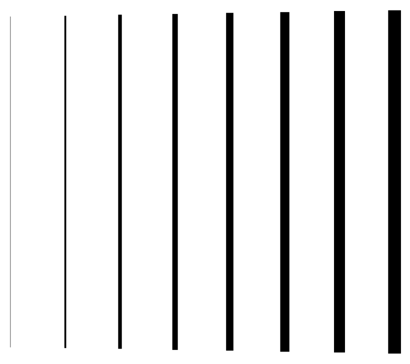


Thickness of Pen

In this demo, we draw different sizes of Pen.

```
Start Canvas Size 1000 600
Set Canvas Color 255 255 0
Set Canvas Size 1800 1800
Start Pen Position 300 300
Set Pen Color 0 0 0
Set Pen Thickness 1
Set Pen Direction 90
Forward 900
Set Pen Position 450 300
Set Pen Thickness 5
Forward 900
Set Pen Position 600 300
Set Pen Thickness 10
Forward 900
Set Pen Position 750 300
Set Pen Thickness 15
Forward 900
Set Pen Position 900 300
Set Pen Thickness 20
Forward 900
Set Pen Position 1050 300
Set Pen Thickness 25
```

```
Forward 900
Set Pen Position 1200 300
Set Pen Thickness 30
Forward 900
Set Pen Position 1350 300
Set Pen Thickness 35
Forward 900
Stop Canvas
Stop Stamp
```



Draw with Brush

Draw

Figure : [Rectangle, Circle, Oval]

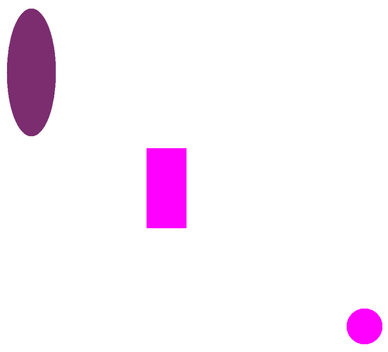
Expr : [(X, Y, Length, Width), (X, Y, Radius), (X, Y, Long Radius , Short Radius)]

This Command will draw a figure with the corresponding expression.

Example

```
Start Brush
Set Brush Color 255 0 255
Draw Rectangle 800 800 100 200
Draw Circle 1300 1200 90
Set Brush Color 123 45 111
Draw Oval 450 450 123 320
```

It will have the following output



Earse

Earse

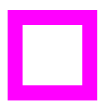
Figure : [Rectangle, Circle, Oval]
Expr : [(X, Y, Length, Width), (X, Y, Radius), (X, Y, Long Radius , Short Radius)]

This function will erase a figure on the output picture.

Example

```
Start Brush
Set Brush Color 255 0 255
Draw Rectangle 800 800 300 300
Erase Rectangle 850 850 200 200
```

The Output is



We first draw an Rectangular, then erase a hole inside it.

Stamp and Frame

Frame

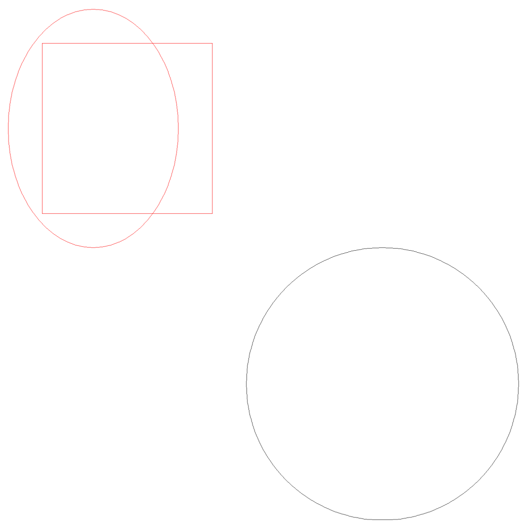
Figure : [Rectangle, Circle, Oval]

Expr : [(X, Y, Length, Width), (X, Y, Radius), (X, Y, Long Radius , Short Radius)]

The Frame command is very similar to Draw and Erase. But it gives a frame of a figure, rather a whole area.

```
Start Stamp Color 255 0 0
Frame Oval 100 100 500 700
Frame Rectangle 200 200 500 500
Set Stamp Color 0 0 0
Frame Circle 800 800 800
```

The Output is



Here we frame a Rectangle, a circle and an Oval.

Now we introduce some application of EsayDraw

Functions in x-y coordinates

Below is code for circle and Absolute function in X,y Coordinate.

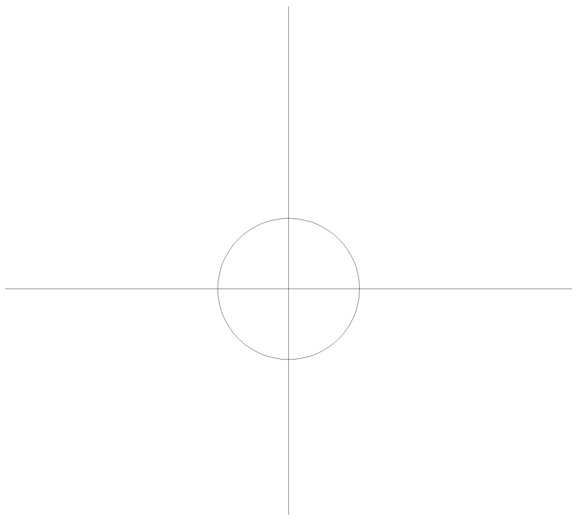
```
Start Canvas Size 1000 600
Set Canvas Color 0 0 0
Set Canvas Size 1800 1800
```

```
Start Stamp Color 0 0 0
Start Pen Position 900 0
Set Pen Direction 90
Forward 1800
Set Pen Position 0 900
Set Pen Direction 0
Forward 1800
Frame Circle 675 675 450
Stop Pen
Stop Canvas
```

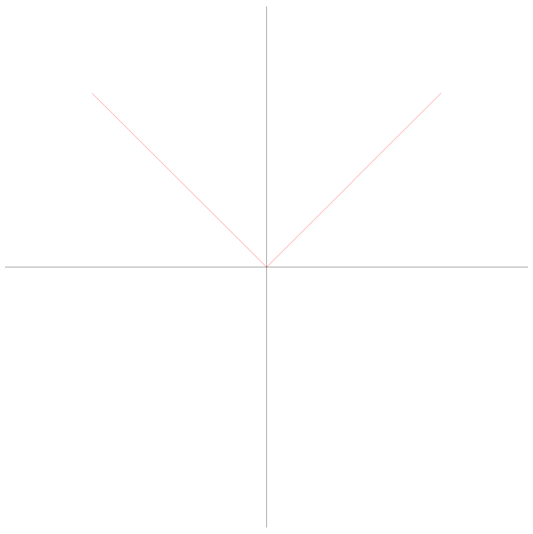
And

```
Start Canvas Size 1000 600
Set Canvas Color 0 0 0
Set Canvas Size 1800 1800

Start Stamp Color 0 0 0
Start Pen Position 900 0
Set Pen Direction 90
Forward 1800
Set Pen Position 0 900
Set Pen Direction 0
Forward 1800


Set Pen Position 300 300
Set Pen Color 255 0 0
Set Pen Direction 45
Forward 849
Spin Clockwise 90
Forward 849


Stop Pen
Stop Canvas
```

The result is



and



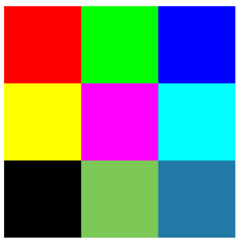
Flags

We use our code to color an area with different color

```
Start Canvas Size 1000 600
Set Canvas Color 255 255 0
Set Canvas Size 1800 1800

Start Brush
Set Brush Color 255 0 0
Draw Rectangle 300 300 300 300
Set Brush Color 0 255 0
Draw Rectangle 600 300 300 300
Set Brush Color 0 0 255
Draw Rectangle 900 300 300 300
```

```
Set Brush Color 255 255 0
Draw Rectangle 300 600 300 300
Set Brush Color 255 0 255
Draw Rectangle 600 600 300 300
Set Brush Color 0 255 255
Draw Rectangle 900 600 300 300

Set Brush Color 0 0 0
Draw Rectangle 300 900 300 300
Set Brush Color 123 200 86
Draw Rectangle 600 900 300 300
Set Brush Color 33 122 166
Draw Rectangle 900 900 300 300

Stop Canvas
Stop Stamp
```

Result is

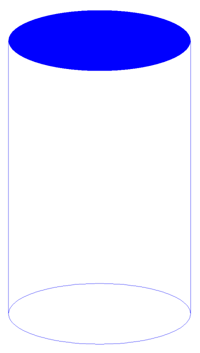


Cylinder

We use our language to draw a cylinder, combine two ovals and two lines.

```
Start Canvas Size 1000 600
Set Canvas Color 255 255 0
Set Canvas Size 1800 1800
Start Stamp Color 0 0 255
Start Pen Color 0 0 255
Start Brush Color 0 0 255
Draw Oval 500 300 600 200
```

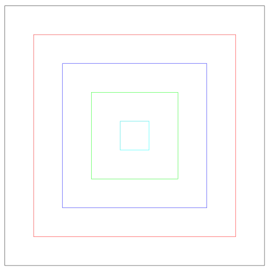
```
Frame Oval 500 1200 600 200
Set Pen Position 500 400
Set Pen Direction 90
Forward 900
Set Pen Position 1100 400
Forward 900
Stop Canvas
Stop Stamp
```



Nested similar shapes

Chinese character "Hui" (回) is combined by different size of rectangle with same core. We use our language to implement a more complex "Hui".

```
Start Canvas Size 1000 600
Set Canvas Color 255 255 0
Set Canvas Size 1800 1800
Start Stamp Color 0 0 0
Frame Rectangle 450 450 900 900
Set Stamp Color 255 0 0
Frame Rectangle 550 550 700 700
Set Stamp Color 0 0 255
Frame Rectangle 650 650 500 500
Set Stamp Color 0 255 0
Frame Rectangle 750 750 300 300
Set Stamp Color 0 255 255
Frame Rectangle 850 850 100 100
Stop Canvas
Stop Stamp
```

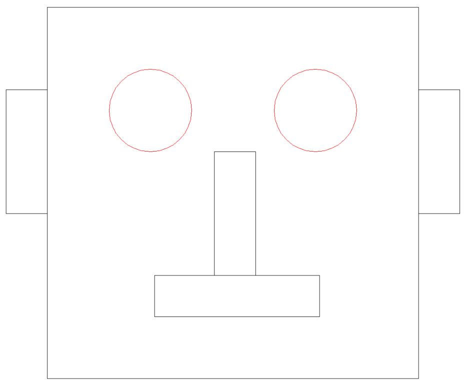


Robot

We use our language to paint a simple robot.

```
Start Canvas Size 1000 600
Set Canvas Color 0 0 0
Set Canvas Size 1800 1800
Start Stamp
Frame Rectangle 450 450 900 900
Set Stamp Color 255 0 0
Frame Circle 600 600 200
Frame Circle 1000 600 200
Set Stamp Color 0 0 0
Frame Rectangle 855 800 100 300
Frame Rectangle 710 1100 400 100
Set Stamp Color  0 0 0
Frame Rectangle 350 650 100 300
Frame Rectangle 1350 650 100 300
Stop Canvas
Stop Stamp
```

Result is

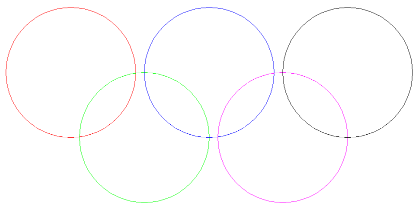


Olympic symbol

A famous figure about Sport.

```
Start Canvas Size 1000 600
Set Canvas Color 255 255 0
Set Canvas Size 1800 1800
Start Stamp Color 255 0 0
Frame Circle 300 700 300
Set Stamp Color 0 255 0
Frame Circle 470 850 300
Set Stamp Color 0 0 255
Frame Circle 620 700 300
Set Stamp Color 255 0 255
Frame Circle 790 850 300
Set Stamp Color 0 0 0
Frame Circle 940 700 300
Stop Canvas
Stop Stamp
```

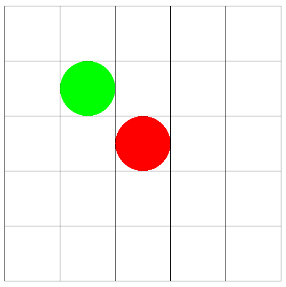
Result is



Tic Tac Toe

We use our language to simulate tic tac toe. We use different kinds of color to represent the different sides of the player.

```
Start Canvas Size 1000 800
Set Canvas Color 255 255 255
Start Stamp Color 0 0 0
Frame Rectangle 100 100 500 100
Frame Rectangle 100 300 500 100
Frame Rectangle 100 500 500 100
Frame Rectangle 100 100 100 500
Frame Rectangle 300 100 100 500
Frame Rectangle 500 100 100 500
Start Brush
Set Brush Color 0 255 0
Draw Circle 200 200 100
Set Brush Color 255 0 0
Draw Circle 300 300 100
Stop Brush
Stop Stamp
Stop Canvas
```



Conclusion

We develop a new language that can draw simple graphs by computer. It is developed by java language, and uses the engine of java swing. Combining existing graph language in the market, EasyDraw shows the advantages of easy usage, and high encapsulation. Based on the features of EasyDraw, it reveals user-friendliness of itself. From the abundant test cases, we can feel the strength of the EasyDraw and its briefness. In conclusion, for neophytes in programming, they can easily get familiar with it and find fun in using language to draw. In the future we will try to implement more built in functions and try to continuously improve the EasyDraw according to the market trend. We believe that the EasyDraw will be one of their top choices for the users when choosing drawing programming languages.