COMS 4115
Programming Languages and Translators
Curve Final Report

Kun An
John Chan
David Mauskop
Wisdom Omuya
Zitong Wang

December 19, 2012

Contents

1 Introduction

Curve is a simple but powerful programming language that produces two–dimensional graphics and animations. It is motivated by the observation that many, if not all, of the basic shapes that are often included in graphics languages generalize toBézier curves. We aimed to keep our language as small as possible, without sacrificing functionality or convenience to the programmer. In line with this aim, we reduced the language to a minimal set of built-in types and operators that can still be combined to produce concise and expressive code. We imagine our language being used in tandem with a rich standard library that would provide additional convenience to the programmer. In this sense, we have followed the model of the C programming language. That is, keep the language small, and let programmers write libraries that tailor the language to their specific needs. Curve supports global and local variables with statically scoping. It allows for user-defined functions, which are call by value. Curve] is strongly typed.

2 Language Tutorial

Curve makes it easy to create create geometric shapes. Before we begin the tutorial, you should know the two broad classes of objects supported in Curve.

2.1 Basic Types

There are three basic types defined by the Curve language. Type identifiers always begin with an upper-case letter followed by a sequence of one or more legal identifier characters. The built-in types include:

- `Point`: A pair of `int`s representing Cartesian coordinates.
- `Curve`: A list of four `Point`s defining a Bézier curve. The first and last points are the anchor points, the second and third points are the control points.
- `Layer`: A list of up to ten `Curve`s, which can be manipulated and drawn as a unit.

2.2 Atom Types

The only legal atom type allowed in Curve is an `int`: An `int` object may be used to describe `Point`s or `Curve`s. See section 3.5 for a discussion of operations possible for `int`s.

2.3 Basic syntax

```
// Declaration must precede assignment
int i;
```

```
Point p;
Curve c1;
Curve c2;
Layer l;

i = 0;
// These both assign Point p to the origin
p = (i, i);
p = (0, 0);
// Assign a curve
c1 = (0, 0)(100, 0)(200, 100)(200, 200);
// Use method from standard library
c2 = rectangleP(p, 10, 100);
// Group curves in a layer
l = [c1, c2];
```

2.4 Program structure

Programs written in Curve must define a main method with the following declaration:

```
int main()
```

The main method can call methods from the standard library (see section 4), or other user-defined methods, which may be recursive. In addition to local variables defined within the scope of a specific method, the user can define global variables outside the scope of any method. These variables can be accessed by any of the program's methods.

2.5 Examples

2.5.1 Koch snowflake - Professor Edwards 6 hour challenge

We completed Professor Edwards' challenge to implement the Koch snowflake with Curve in the six hours between the end of our presentation and the submission deadline. Prof. Edwards was right. This was easy to implement elegantly in Curve.

```
1  Point pointAlongLine(Curve line, int mult, int div)
2  {
3      int x;
4      int y;
5      Point p1;
6      Point p2;
7
8      p1 = line.getPoint(0);
9      p2 = line.getPoint(3);
10
11     x = p1.getX() + ((p2.getX() - p1.getX())*mult)/div;
12     y = p1.getY() + ((p2.getY() - p1.getY())*mult)/div;
13
```

```
14      return (x, y);
15  }
16
17  Point topPoint(Point a, Point b)
18  {
19      int delX;
20      int delY;
21      int x;
22      int y;
23
24      delX = b.getX() - a.getX();
25      delY = b.getY() - a.getY();
26
27      x = a.getX() + delX/2 - delY*1732/2000;
28      y = a.getY() + delY/2 + delX*1732/2000;
29
30      return (x, y);
31  }
32
33  int snowflake(Point a, Point e, int n)
34  {
35      Point b;
36      Point c;
37      Point d;
38      Curve line;
39
40      line = lineP(a, e);
41
42      if (n == 1) {
43          draw([line]);
44          return 1;
45      }
46
47      b = pointAlongLine(line, 1, 3);
48      d = pointAlongLine(line, 2, 3);
49      c = topPoint(b, d);
50
51      snowflake(a, b, n-1);
52      snowflake(b, c, n-1);
53      snowflake(c, d, n-1);
54      snowflake(d, e, n-1);
55  }
56
57  int main()
58  {
59      int n;
60      Point p0;
61      Point q0;
62      Point r0;
63
64      p0 = (350, 260);
65      q0 = (650, 780);
66      r0 = (950, 260);
67
68      for (n = 1; n < 9; n++) {
69          snowflake(p0, q0, n);
70          snowflake(q0, r0, n);
71          snowflake(r0, p0, n);
72          pause(3000);
73          clear();
74      }
75  }
```

2.5.2 Draw a Circle

The example program below uses the `circle` method to draw a circle with radius `r` at coordinates passed in `Point c`. Each Curve within the returned Layer contains a quadrant's arc:

```
Layer circle(int r, Point c) {
  int x;
  int y;
  Curve tr;
  Curve br;
  Curve bl;
  Curve tl;
  Layer cir;
  int ctrl;

  x = c.getX();
  y = c.getY();

  ctrl = 552*r/1000;
  tr = (x, r+y)(ctrl+x, r+y)(r+x, ctrl+y)(r+x, y);
  br = (x, -r+y)(ctrl+x, -r+y)(r+x, -ctrl+y)(r+x, y);
  bl = (x, -r+y)(-ctrl+x, -r+y)(-r+x, -ctrl+y)(-r+x, y);
  tl = (x, r+y)(-ctrl+x, r+y)(-r+x, ctrl+y)(-r+x, y);
  cir = [tr, br, bl, tl];
  return cir;
}
```

2.5.3 Friendly Screensaver

The example above created a circle; now, let's make it move! The example below uses the `circle` method in the above example to create a simple screensaver animation.

```
int main()
{
  int i;
  int x;
  int y;
  int r;
  int xinc;
  int yinc;

  xinc = 1;
  yinc = 1;

  while(1) {
    x = x + xinc;
    y = y + yinc;
    if (x == 590)
      xinc = -1;
    if (x == 10)
      xinc = 1;
    if (y == 440)
      yinc = -1;
    if (y == 10)
      yinc = 1;
    r = 10;
    draw(circle(r, (x, y)));
    draw(circle(r + 5, (x + xinc * 5, y + yinc * 5)));
```

```
27      pause(5);
28      clear();
29    }
30  }
```

2.5.4 Bow and Arrow

The `Curve` type is the core type of our language. All kinds of shapes can be thought of as compositions of curves. Under the hood, curve is implemented using beizer curve. By changing the critical and control points of the curve over time, we can obtain very smooth animation effects. The following example demostrate the movements of bow, string, and arrow.

```
1   int main()
2   {
3     int i;
4
5     int bcx;
6     int bcy;
7     Curve bow1;
8     Curve bow2;
9     Curve str;
10    Curve arr;
11    Curve hed;
12    Layer bsah;
13    int ela;
14    int inc;
15    int elac;
16
17    bcx = 200;
18    bcy = 400;
19    ela = 0;
20    inc = 1;
21    elac = 0;
22
23    while (1) {
24      i++;
25      ela = ela + inc;
26      if (ela == 25) {
27        inc = -1;
28        elac = elac + 1;
29      }
30      if (ela == 0) {
31        inc = 1;
32        elac = elac + 1;
33      }
34      if (elac > 1)
35        inc = 0;
36      bow1 = (bcx - ela * 2, bcy + 100)(bcx + 20, bcy + 20)(bcx + 20, ↩
            bcy - 20)(bcx - ela * 2, bcy -100);
37      bow2 = (bcx - ela * 2, bcy + 100)(bcx + 15, bcy + 20)(bcx + 15, ↩
            bcy - 20)(bcx - ela * 2, bcy -100);
38      str = (bcx - ela * 2, bcy + 100)(bcx - ela * 6, bcy)(bcx - ela * ↩
            6, bcy)(bcx - ela * 2, bcy -100);
39      arr = (bcx - ela * 5, bcy)(bcx - ela, bcy)(bcx - ela, bcy)(bcx - ↩
            ela * 5 + 150, bcy);
40      hed = (bcx - ela * 5 + 150 - 10, bcy + 5)(bcx - ela * 5 + 150, bcy↩
            )
41              (bcx - ela * 5 + 150, bcy)(bcx - ela * 5 + 150 - 10, bcy - ↩
                5);
42      if (elac > 1) {
```

```
      arr = (bcx + i - 50, bcy)(bcx + i, bcy)(bcx + i, bcy)(bcx + i + 100, bcy);
      hed = (bcx + i + 100 - 10, bcy + 5)(bcx + i + 100, bcy)
            (bcx + i + 100, bcy)(bcx + i + 100 - 10, bcy - 5);
    }
    bsah = [bow1, bow2, str, arr, hed];
    draw(bsah);
    if (inc == 1)
      pause(50);
    else
      pause(3);
    clear();
    if (i == 1000) {
      i = 0;
      ela = 0;
      elac = 0;
      inc = 1;
    }
  }
}
```

2.5.5 Tree Creation and Traversal

By using recursion and animation, we can draw a tree recursively and showcase how each step is done. After we are done with tree creation, we do a in-order traversal.

```
int exp(int x, int n) {
  int i;
  int acc;

  acc = 1;

  for (i = 0; i < n; i++) {
    acc = acc * x;
  }
  return acc;
}

int drawTree(int x, int y, int n) {
  Layer l;
  Curve left;
  Curve right;
  Layer cirL;
  Layer cirR;

  if (n == 0) {
    return 1;
  }

  drawTree(x - exp(2, n), y - 50, n - 1);
  drawTree(x + exp(2, n), y - 50, n - 1);

  cirL = circle(5, (x - exp(2, n), y - 50));
  draw(cirL);
  cirR = circle(5, (x + exp(2, n), y - 50));
  draw(cirR);
  left = lineP((x, y), (x - exp(2, n), y - 50));
  right = lineP((x, y), (x + exp(2, n), y - 50));
  draw([left, right]);
  pause(100);
}

```

```
37 int inOrder(int x, int y, int n) {
38   Layer mark;
39
40   if (n == 0) {
41     mark = circle(8, (x, y));
42     draw(mark);
43     pause(200);
44     return 1;
45   }
46
47   inOrder(x - exp(2, n), y - 50, n - 1);
48
49   mark = circle(8, (x, y));
50   draw(mark);
51   pause(200);
52
53   inOrder(x + exp(2, n), y - 50, n - 1);
54 }
55
56 int main()
57 {
58   drawTree(700, 700, 8);
59
60   inOrder(700, 700, 8);
61
62   while(1) {
63
64   }
65 }
```

2.5.6 Windmill

Curve supports rotation, scaling and many other transformations. User can also easily define their own transformations using our language. The following example demonstrates how to create a windmill and make it rotate. Note that in the loop we apply the rotateL function to the layer, which is the windmill. By applying rotateL, we make the windmill rotate 3 degrees per frame.

```
1  int main()
2  {
3    int i;
4    Curve c1;
5    Curve c2;
6    Curve c3;
7    Curve c4;
8    Curve cc1;
9    Curve cc2;
10   Curve cc3;
11   Curve cc4;
12   Layer l;
13   int cx;
14   int cy;
15
16   cx = 700;
17   cy = 500;
18
19   c1 = (cx - 500, cy)(cx - 500, cy - 200)(cx - 200, cx - 300)(cx, cy);
20   c2 = rotateC(c1, cx, cy, 1, 0, 1);
21   c3 = rotateC(c2, cx, cy, 1, 0, 1);
22   c4 = rotateC(c3, cx, cy, 1, 0, 1);
23   cc1 = (cx - 500, cy)(cx, cy)(cx, cy)(cx, cy);
24   cc2 = rotateC(cc1, cx, cy, 1, 0, 1);
```

```
  cc3 = rotateC(cc2, cx, cy, 1, 0, 1);
  cc4 = rotateC(cc3, cx, cy, 1, 0, 1);
  l = [c1, c2, c3, c4, cc1, cc2, cc3, cc4];

  while(1) {
    i++;
    l = rotateL(l, cx, cy, -348995, 9993908, 10000000);
    draw(l);
    pause(30);
    clear();
  }
}
```

2.5.7 Clock

With rotation and animation, you can do even more. Let's draw a working clock!

```
Layer rotateL6(Layer l, int cx, int cy) {
  return rotateL(l, cx, cy, 1045285, 9945219, 10000000);
}

int main()
{
  int i;
  int j;
  int k;
  int c;
  Curve s;
  Layer sl;
  Layer sbs;
  Layer cir;
  Layer cirbs;
  Curve m;
  Layer ml;
  Layer mbs;
  Curve h;
  Layer hl;
  Layer hbs;
  Curve c;
  Layer cl;
  Layer clbs;
  int cx;
  int cy;

  cx = 700;
  cy = 500;

  s = (cx, cy + 310)(cx, cy)(cx, cy)(cx, cy);
  sl = [s];
  sbs = sl;
  cir = circle(20, (cx, cy + 260));
  cirbs = cir;

  m = (cx, cy + 240)(cx, cy)(cx, cy)(cx, cy);
  ml = [m];
  mbs = ml;

  h = (cx, cy + 180)(cx, cy)(cx, cy)(cx, cy);
  hl = [h];
  hbs = hl;

  c = (cx, cy + 360)(cx, cy + 380)(cx, cy + 380)(cx, cy + 380);
  cl = [c];
```

```
47    clbs = cl;
48
49    while(1) {
50      for (c = 0; c < 12; c++) {
51        draw(cl);
52        cl = rotateL(cl, cx, cy, 50000000000, 86602540378, 100000000000)↩
              ;
53      }
54      cl = clbs;
55      draw(sl);
56      draw(cir);
57      draw(ml);
58      draw(hl);
59      i++;
60      sl = rotateL6(sl, cx, cy);
61      cir = rotateL6(cir, cx, cy);
62      if (i == 60) {
63        sl = sbs;
64        cir = cirbs;
65        i = 0;
66        j++;
67        ml = rotateL6(ml, cx, cy);
68        if (j == 12) {
69          hl = rotateL6(hl, cx, cy);
70        }
71        if (j == 24) {
72          hl = rotateL6(hl, cx, cy);
73        }
74        if (j == 36) {
75          hl = rotateL6(hl, cx, cy);
76        }
77        if (j == 48) {
78          hl = rotateL6(hl, cx, cy);
79        }
80        if (j == 60) {
81          ml = mbs;
82          j = 0;
83          k++;
84          hl = rotateL6(hl, cx, cy);
85          if (k == 12) {
86            hl = hbs;
87            k = 0;
88          }
89        }
90      }
91      pause(998);
92      clear();
93    }
94  }
```

2.5.8 Pattern Generation

We introduced random numbers to our language. This makes curve even fun to play with. By drawing a shape with random parameters and doing all kinds of transformations, you can easily create interesting patterns in seconds. This example creates spiral-like patterns all over the screen.

```
1   int main()
2   {
3     int i;
4     Layer l;
5     int x;
6     int y;
```

```
7     int s;
8     int c;
9     int d;
10
11    while(1) {
12      s = 10 + random(5);
13      c = 100;
14      x = random(1400);
15      y = random(800);
16      d = random(2);
17      if (d == 1) {
18        s = -s;
19      }
20      l = rectangleP((x, y), random(100), random(100));
21      for (i = 0; i < 100 + random(500); i++) {
22        l = rotateL(l, x, y, s, c, 100);
23        draw(l);
24        pause(2);
25      }
26      pause(10);
27    }
28  }
```

3 Language Reference Manual

3.1 Introduction

Curve is a simple vector graphics language specifically targeted for graphic animations. It allows for creation and rendering of graphical objects, both static and moving and is based on the simple concept of Bézier curves.

The goal of the Curve syntax is to make conceptually simple graphics and manipulations using simple, yet powerful, language constructs. The type system in Curve is described in subsection 3.4. The basic type, Curve, while simple, is very expressive. e.g. a group of curves may be used to represent a layer, a group of layers used to form an image – successive rendering of layers thus creating animations. This language reference manual describes the syntax of Curve.

3.2 Program Definition

A Curve program consists of a sequence of zero or more expressions.

3.3 Lexical Conventions

3.3.1 Comments

In-line comments are preceded by `//`, while block comments are delimited by `/*` and `*/`. Nesting will not be allowed.

3.3.2 Identifiers

Identifiers are comprised of uppercase letters, lowercase letters, digits and underscore (`_`). The first character cannot be a digit, nor can it be an underscore (`_`).

3.3.3 Keywords

The following keywords are reserved:

```
int
Layer
Curve
Point
if
else
for
while
main
random
print
draw
pause
clear
getX
getY
setX
setY
setPoint
getPoint
setCurve
getCurve
getSize
```

3.3.4 Constants

`int`s are represented by a combination of digits and nothing else. Only decimal (as opposed to hexidecimal, or octal) representations are allowed. Example:

```
//Declaration and assignment of an int (must be separate)
int m;
m = 99;
```

3.3.5 Operators

Operator	Definition
+	Addition
-	Subtraction
*	Multiplication
/	Division
=	Assignment

3.3.6 Punctuators

The following symbols are used to organize code and to specify different organizations of objects:

Symbol	Definition
;	Marks the end of a statement
[]	Marks the beginning and end of a group of Curves - i.e. a Layer
{}	Marks the beginning and end of a group of statements
,	Separates different dimension values of a Point
	Separates different Curves of a Layer
.	Used for dot notation access - e.g. accessing Points of a curve
()	Used for grouping parameters in function call
	Used for specifying Points

3.4 Object Types

An Curve object is a manipulatable region of storage. There are three broad classes of objects supported in Curve.

3.4.1 Basic Types

There are three basic types defined by the Curve language. Type identifiers always begin with an upper-case letter followed by a sequence of one or more legal identifier characters. The built-in types include:

- `Point`: A pair of `int`s representing Cartesian coordinates
- `Curve`: A list of `Point`s where the first `Point` is the start point of the `Curve`, the last is the end point, while the two intermediate `Point`s are controls for the Bézier curve.
- `Layer`: A list of `Curve`s

3.4.2 Atom Types

The only legal atom type accepted in Curve is an `int`. See subsection 3.5 for a discussion of operations possible for `int`s.

3.4.3 Object Scope

All variable declaration must come *before* (and separate from) variable assignment. Curve distinguishes between two kinds of variables with respect to scope.

Local Variables Variables that are declared inside a function or block are local variables. They can be used only by statements that are inside that function or block of code. Local variables are not known to functions outside their own. The following is the example using local variables:

```
int addition ()
{
  // Local variable declaration:
  int a;
  int c;

  // actual initialization
  a = 10;
  c = 12;

  return a + b;
}
```

Global Variables Global variables are defined outside of all the functions, usually on top of the program. The global variables will hold their value throughout the lifetime of your program.

A global variable can be accessed by any function. That is, a global variable is available for use throughout your entire program after its declaration. Following is the example using global and local variables:

```
int b;

int scale ()
{
  // Local variable declaration:
  int a;

  // actual initialization
  a = 10;

  return a * b;
}
```

All identifiers in Curve are either global or local. A program can have same name for local and global variables but value of local variable inside a function will take preference. For example:

```
int b;
b = 2;

int scale ()
{
  // Local variable declaration:
  int b;

  // actual initialization
  b = 10;
  return b;
}
print(scale());
```

When the above code is compiled and executed, it produces following result:

```
10
```

3.5 Expressions and Operations

In this subsection we describe the built-in operators for Curve and define what constitutes an expression in our language. Operators are listed in order of precedence. All operators which associate, do so left to right.

1. Primary expressions
   (a) identifier
   See section 3.3.2.
   (b) constant
   See section 3.3.4.
   (c) (expr)
   (d) (expr, expr)
   Define a point, expressions must be ints
   (e) (expr, expr)(expr, expr)(expr, expr)(expr, expr)
   Define a curve, expressions must be ints
   (f) [list of up to 10 expressions]
   Define a Layer, expressions must be curves
   (g) identifier.system-function(list of expressions, comma separated)
   See section 3.8 for the list of system functions
   (h) identifier(list of 0 or more expressions, comma separated)
   Call a function with optional arguments.
2. Multiplicative operators
   (a) expression * expression
   Multiplication is valid between two ints
   (b) expression / expression
   Division is defined identically to multiplication, except of course that division by zero is not allowed.
3. Additive operators
   (a) expression + expression
   Addition is valid between two ints
   (b) expression - expression
   Subtraction is defined identically to addition.
   (c) expression++
   Increment integer by one
4. Relational operators
   (a) expression < expression
   Less than.
   (b) expression > expression
   Greater than.
   (c) expression <= expression
   Less than or equal to.
   (d) expression >= expression
   Greater than or equal to.

5. Equality operators

   (a) expression == expression
   Equal to.

   (b) expression ! = expression
   Not equal to.

   The relational and equality operators are valid for comparison between two ints, the result is an int, 1 for true or 0 for false. The standard library functions true() and false() can also be used to avoid confusion.

6. Assignment operators

   (a) primary-expression = expression
   The value of the expression replaces the value of the object that the primary-expression refers to. Types must match.

To summarize, the basic arithmetic and boolean operations are defined on integers. The Point, Curve, and Layer types rely on get and set methods, which are described in section 3.8.

3.6 Declarations

3.6.1 Function Declaration

Our language supports user-defined functions. Functions are declared and implemented at the same time, which means before using an user-defined function user must first implement it. The return type of a function needs to be declared. A function can have any number of parameters, whose types must also be declared. The parameters are passed by value. Here is an example:

```
Point dist (int x1, int y1, int x2, int y2) {
  return ((x1-x2), (y1-y2));
}
```

Each program written in Curve must have a main method defined with return type `int` and no parameters. This method is where the program begins execution.

3.6.2 Variable Declaration

Users need to declare the type of variable. When a local variable is defined, it is not initialized by the system, you must initialize it yourself. Global variables are initialized automatically by the system when you define them as follows:

```
int:
    int a;
Point:
```

```
    Point p;
Curve:
    Curve c;
Layer:
    Layer l;
```

Variable declaration is typically followed by variable assignment:

```
Curve:
    Curve c;
    c = (0,0)(1,1)(3,1)(4,0);
Layer:
    Layer l;
    l = [c1, c2]; // c1, c2 are curves
```

3.7 Statements

All statements in Curve end in a semi-colon.

3.7.1 Expression statement

```
expression;
```

An expression statement is typically an assignment or a function call.

3.7.2 Conditional statement

```
if (expression) { } else { }
```

The only condition statement supported by Curve is an if statement followed by an optional else.

3.7.3 While statement

```
while (expression) { }
```

The while loop behaves as is typical

3.7.4 For statement

```
for (expression; expression; expression) { }
```

The for loop behaves as is typical

3.7.5 Return statement

```
return 0;
```

Each method must have a return statement whose type matches the method's declared return type.

3.8 System Functions

3.8.1 Print Function

The print function is used to display text on standard output.

```
print(9);
```

The print function only accepts a int. It could however be used in conjunction with other system functions to print coordinates of Curves, Points and Layers.

3.8.2 Draw Function

The draw function is used to draw a layer to the graphics window:

```
draw(l);
```

The draw function only accepts a Layer (which could consist of a single curve). In the example above, it renders all curves – within the Layer l – passed to the draw function.

3.8.3 Pause Function

The pause function is used to simulate a delay in the current execution of the program:

```
pause(5000);
```

The pause function accepts an int that represents the number of milliseconds for which you want to suspend the execution of the program.

3.8.4 Clear Function

The clear function is used to remove all renderings currently on the graphics window.

```
clear();
```

The clear function accepts no arguments.

3.8.5 getX Function

The getX function is used to get the x-coordinate of a Point.

```
p.getX();
```

The getX accepts no arguments. In the example above, it returns the x-coordinate of the Point, p.

3.8.6 getY Function

The getY function is used to get the y-coordinate of a Point.

```
p.getY();
```

The getY accepts no arguments. In the example above, it returns the y-coordinate of the Point, p.

3.8.7 setX Function

The setX function is used to set the x-coordinate of a named Point.

```
p.setX(10);
```

The statement above changes the x-coordinate of a Point, p, to 10.

3.8.8 setY Function

The setY function is used to set the y-coordinate of a named Point.

```
p.setY(12);
```

The statement above changes the y-coordinate of a Point, p, to 12.

3.8.9 getPoint Function

The getPoint function is used to get a Point from a named Curve.

```
c.getPoint(1);
```

The getPoint built-in accepts exactly one argument – the index of the Point to be retried[1]. The example above returns the first control point of the supplied Curve, c.

3.8.10 setPoint Function

The setPoint function is used to set a Point in a named Curve.

```
c.setPoint(3,p);
```

The setPoint built-in accepts exactly two arguments – the first being the index of the point to be modified, the second, the target point. The example above sets the end point of the supplied Curve, c, to Point p.

[1] Note that this is zero-indexed

3.8.11 getCurve Function

The getCurve function is used to get a Curve from a Layer.

```
l.getCurve(1);
```

The getCurve built-in accepts one argument – the index of the Curve to be retried. The example above returns the second Curve in the Layer, l.

3.8.12 setCurve Function

The setCurve function is used to set a Curve in a Layer.

```
 l.setCurve(0,a);
```

The setCurve built-in accepts exactly two arguments – the first being the position of the Layer to mutate, and the second an identifier/expression for the new Curve. The example above changes the first Curve in the Layer, l to a named Curve a.

3.8.13 getSize Function

The getSize function is used to retrieve the number of Curves in a Layer.

```
l.getSize();
```

The getSize accepts no arguments and is called on the Layer object. It returns the number of named Curves within the Layer. The example above returns the number of Curves in the Layer l.

4 Standard Library

We wrote a small standard library to demonstrate how our language can be tailored to the specific needs of the programmer. This removes any of the inconvenience caused by our language's limited set of built-in types and operators. The standard library is written entirely in Curve and simply appended to the end of new source files. If the programmer, whether intentionally or accidentally, re-defines one of the methods in the standard library, this newly defined method is used. Our basic standard library has the methods described below. However, we can imagine many other ways to expand this library.

- `int printp(Point p)`
  Prints x and y coordinates of point p
- `int printc(Curve c)`
  Prints points making up curve c
- `int printl(Layer l)`
  Prints curves making up layer l

- `int true()`
  Returns 1, the integer value corresponding to "true"
- `int false()`
  Returns 0, the integer value corresponding to "false"
- `int curveSize()`
  Returns 4, the number of Points in a Curve
- `int maxLayerSize()`
  Returns 10, the maximum number of curves allowed in a layer
- `int nullI()`
  Returns -1, the integer value we use to signify null
- `int nullP()`
  Returns a point equal to (-1,-1) to signify null
- `int nullC()`
  Returns a curve with four null points, to signify null
- `int nullL()`
  Returns a layer with ten null curves, to signify null
- `int equalsP(Point p, Point q)`
  Returns true() if p equals q, false() otherwise
- `int equalsC(Curve c, Curve d)`
  Returns true() if c equals d, false() otherwise
- `int equalsL(Layer l, Layer m)`
  Returns true() if l equals m, false() otherwise
- `Curve translateC(Curve c, int x, int y)`
  Returns a new curve that equals curve c translated according to vector (x, y)
- `Layer translateL(Layer l, int x, int y)`
  Returns a new layer that equals layer l with each of its curves translated by (x, y)
- `Curve transformC(Curve cv, int a, int b, int c, int d)`
  Returns a new curve that equals curve c transformed according to the specified 2x2 matrix
- `Layer transformL(Layer l, int a, int b, int c, int d)`
  Returns a new layer that equals layer l with each of its curves transformed according to the 2x2 matrix specified
- `Curve lineXY(int x1, int y1, int x2, int y2)`
  Returns a curve that's a line going from (x1, y1) to (x2, y2)

- `Curve lineP(Point p, Point q)`
  Returns a curve that's a line going from Point p to Point q
- `Layer triangleP(Point p, Point q, Point r)`
  Returns a Layer that's a triangle with vertices p, q, and r
- `Layer triangleXY(int x1, int y1, int x2, int y2, int x3, int y3)`
  Returns a Layer that's a triangle with vertices (x1, y1), (x2, y2), and (x3, y3)
- `Layer rectangleXY(int x, int y, int height, int width)`
  Returns a Layer that's a rectangle with lower-left vertex at (x, y) and the specified height and width
- `Layer rectangleP(Point p, int height, int width)`
  Returns a Layer that's a rectangle with lower-left vertex at point p and the specified height and width
- `int isLine(Curve c)`
  Return true() if c is a straight line, false otherwise
- `int isRectangle(Layer r)`
  Returns true() if the specified Layer is a rectangle, false() otherwise
- `Point getRectangleBase(Layer r)`
  Returns the lower-left vertex of the specified layer if it is a rectangle, null otherwise
- `int getRectangleWidth(Layer r)`
  Returns the width of the specified layer if it is a rectangle, null otherwise
- `int getRectangleHeight(Layer r)`
  Returns the heigh of the specified layer if it is a rectangle, null otherwise
- `Layer fillRectangleL(Layer r, int lightness)`
  If the specified layer is in fact a rectangle, draws the rectangle with the specified fill density, returns r
- `Layer fillRectangleP(Point p, int height, int width, int lightness)`
  Creates a rectangle with rectangleP(p, height width) then calls fillRectangleL with the specified fill density, returns this new rectangle
- `Layer squareXY(int x, int y, int size)`
  Creates a square with lower-left vertex at (x, y) and side length equal to size
- `Layer squareP(Point p, int size)`
  Creates a square with lower-left vertex at point p and side length equal to size

- `Layer quadP(Point p1, Point p2, Point p3, Point p4)`
  Returns a layer that's a quadrilateral with the specified vertices
- `Layer circle(int r, Point c)`
  Returns a layer that's a circle with center c and radius r
- `Curve scaleC(Curve c, int x, int y, int n, int d)`
  Returns a new curve that equals c scaled by n/d with reference to point (x,y)
- `Layer scaleL(Layer l, int x, int y, int n, int d)`
  Returns a new layer that equals l with each of its curves scaled by n/d with reference to point (x,y)
- `Curve rotateC(Curve c, int x, int y, int s, int c, int d)`
  Returns a new curve that equals c rotated according to parameters s, c, and d. The x and y values specify the pivot point. The s and c values can be used as the sin and cos of the degree of rotation, respectively. Due to the fact that we don't support floats, the d is a scaling factor, and the s and c values will be divided by this.
- `Curve rotateL(Layer l, int x, int y, int s, int c, int d)`
  Returns a new layer that equals l with each of its curves rotated according to parameters s, c, and d
- `exp(int x, int n)`
  Returns x raised to the power of n

5 Project Plan

As soon as we had decided on what we wanted our language to do, we began extended discussions about how we wanted to define our Abstract Syntax Tree – as it forms the core from which other parts of our compiler interact. Once this was completed, we began development in earnest. We had a fast iterative development cycle where small incremental changes were made, tested and then checked in. Version Control was very helpful as team members could simultaneously commit code, find and fix bugs. Due to our large team size, we decided to divide and conquer. The teams we set up are described in 5.3.

5.1 Project Processes

5.1.1 Planning

As soon as we formed our group, we began meeting each Monday after lecture to discuss the project, assess our progress, and assign responsibilities for tasks that needed to be completed before the next meeting. This enabled us to make small but substantial progress in planning and implementing our language.

5.1.2 Specification

After we submitted our LRM, as we began the implementation process, we quickly discovered that we had underspecified our language – we needed to be more precise about what the types were and how they all worked together; consequently, we came up with a simple and succinct specification for the type system which allowed us to have a much cleaner AST interface.

5.1.3 Development

We utilized a fast iterative approach to development where whenever a feature was implemented, a test had to be written for it and it immediately underwent regression tests to ensure the change broke nothing. This was invaluable in catching bugs and making final verification easy. The scanner was easily written but we spent a substantial amount of time (weeks) deciding what the interface for the AST should be — it was time well spent. With this in hand, we were able to proceed in implementing various parts of our language in tandem.

5.1.4 Testing

The testing process was managed collaboratively but primarily by Dave with tests written as new features were added by team members. The testing process is described in greater detail in section 7 of this document.

5.2 Style Guide

In writing code, we conformed to the following coding convention styles:

1. Maximum line length is 100 characters
2. Code blocks are indented with tabs
3. Camel casing for function names
4. Generally only one statement per line

5.3 Team Responsibilities

With the AST interface agreed upon, we divided our team into two groups to increase efficiency and not have too many cooks working on one soup. Our teams were:

- Frontend Team (Dave, Kun, Zitong): Responsible for scanner, parser, semantic check
- Backend Team (John, Wisdom): Responsible for bytecode generation and execution

Once the core features each team was responsible for had been completed, we came together as a whole to perform more holistic verification and validation.

5.4 Project Timeline

The first few weeks were spend deciding what interface we wanted to expose via the Abstract Syntax Tree. While we spent a great deal of time discussing and refining our initial design, the effort was not wasted. In the end, we all had a strong grasp of what we wanted to accomplish and how we wanted to do it. We arrived at v1.0 of our AST around mid-November – by this time, the scanner was already completed.

5.5 Project Log

```
b505330 - Zitong Wang, 2 weeks ago : Add semantic check.
4f153ee - Kun An, 2 weeks ago : Changed function return type to curvet.
d0a5e88 - Kun An, 2 weeks ago : Used type curvet instead of string to denote variable type.
6894b06 - Kun An, 3 weeks ago : Return type is added to function declaration; added the other two types to formal.
4b373d6 - Kun An, 4 weeks ago : Added :: operation. I did not modify the global variable code yet. Local variables are sufficient for testing purposes.
10ac440 - Kun An, 4 weeks ago : Defined how curve's id is stored in a layer.
14fc4c9 - Kun An, 4 weeks ago : Added t field in var_decl to explicitly declare the type. Will no longer use the length of int list to determine the type.
f0d1196 - Kun An, 4 weeks ago : Layer is added  to the language. Will modify draw to deal with drawing a layer of curves.
bd3ed89 - Kun An, 4 weeks ago : Added the matrix expr to allow transformation by applying a matrix.
5235459 - Kun An, 4 weeks ago : Added transformation support >> (_, _, _, _).
c9f52e0 - Kun An, 4 weeks ago : Added >> support, which can be followed by (_, _) or  (_, _, _, _). For curve or point, we can do operations such as draw(c >> (2,1) >> (-1, -1)). The shift operation implementation is done. The translation will be added later on.
7fcc9fd - Kun An, 4 weeks ago : Converted tab to spaces.
1fbf686 - Kun An, 4 weeks ago : Dot operation is now available. May add more operations later on. Curve type is also added. Note that there is no type checking and out of bound checking in the interpreter right now. Should add those later on.
03004ed - Kun An, 4 weeks ago : Now function call returns int list. Run "./microc -i < test-arith1.mc" to see how it works.
03764be - Kun An, 4 weeks ago : Point works! TODO: change return exception type; add type checking; modify global variable code. Notice: Now the type of variable is inferred by the length of the value field, which is a int list; the vtype field is therefore deleted.
361280a - Kun An, 4 weeks ago : Added Point expr; Removed PointAssign as it is no long useful; Thinking of use the length of int list to implicitly indicate the type of the variable.
d2b18c2 - Kun An, 4 weeks ago : "let rec eval env = function" now returns int list * env; Need to change the return exception to type int list * env.
473ff4e - Kun An, 4 weeks ago : Added point assign expr; Omit global assignment for now; Need to let eval return (int list, env) rather than (int , env).
b225c12 - Kun An, 4 weeks ago : Compilable version that contains type in parameter list.
85e39e6 - Kun An, 4 weeks ago : Removed complied files.
```

```
2de22dd - Kun An, 4 weeks ago : Added microc files.
5d5071e - Kun An, 4 weeks ago : added readme
```

5.5.1 Master

```
d50ba2b - Zitong Wang, Wed Dec 19 15:53:16 : add slide for interpreter
1829b2f - Zitong Wang, Wed Dec 19 15:01:29 : correct typo
7d88710 - David Mauskop, Wed Dec 19 02:02:37 : Final report updates and typo correction
868c253 - David B Mauskop, Wed Dec 19 00:03:38 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
1fadd03 - David B Mauskop, Wed Dec 19 00:02:32 : Hanoi without grayscale
bd66ecb - Wisdom Omuya, Tue Dec 18 23:10:35 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project presentation update
8a3312b - Wisdom Omuya, Tue Dec 18 23:10:27 : small updated to presentation
64e2843 - David Mauskop, Tue Dec 18 22:58:24 : Std lib updates
eee9325 - David Mauskop, Tue Dec 18 21:56:12 : Verify function paramter types
43ee8e5 - John Chan, Tue Dec 18 20:09:00 : merged
b0d52fd - John Chan, Tue Dec 18 20:07:33 : changed presentation for backend
b518d66 - David Mauskop, Tue Dec 18 19:34:37 : Overview section, bezier example, basic syntax
1949a01 - John Chan, Tue Dec 18 19:33:17 : added more to the rotate function
9df9b97 - David Mauskop, Tue Dec 18 19:30:40 : Overview section, bezier example, basic syntax
c33e77e - Kun An, Tue Dec 18 18:47:07 : minor changes
6b5f0b9 - David Mauskop, Tue Dec 18 16:47:45 : Lessons learned
a477e99 - John Chan, Tue Dec 18 07:56:16 : added to backend
736bb70 - Wisdom Omuya, Mon Dec 17 16:34:41 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project merge
1f4549f - Wisdom Omuya, Mon Dec 17 16:34:34 : added ninjas
0cc67e0 - Kun An, Mon Dec 17 16:27:36 : tree.png
e511b85 - Kun An, Mon Dec 17 16:23:38 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
e8e4121 - Kun An, Mon Dec 17 16:23:24 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
518cb4d - Wisdom Omuya, Mon Dec 17 16:23:19 : syntax
f1def30 - Kun An, Mon Dec 17 16:23:18 : add tree.png
75722fd - Wisdom Omuya, Mon Dec 17 16:18:58 : small changes
309365a - David Mauskop, Mon Dec 17 16:14:39 : Std lib section added
26eb954 - Zitong Wang, Mon Dec 17 16:12:17 : Merge branches 'master' and 'master' of https://bitbucket.org/John Chan/plt-project
4311ffd - Zitong Wang, Mon Dec 17 16:10:56 : frontend part in slides
94557d5 - Wisdom Omuya, Mon Dec 17 16:10:48 : clean up presentation
9a84664 - Wisdom Omuya, Mon Dec 17 16:08:32 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project added lessons learned
33b8739 - Wisdom Omuya, Mon Dec 17 16:08:23 : lessons learned
a64404a - Zitong Wang, Mon Dec 17 16:07:30 : frontend images
fb4c613 - Zitong Wang, Mon Dec 17 16:02:59 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
6c23be2 - Zitong Wang, Mon Dec 17 16:02:51 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
f1d3ecc - Wisdom Omuya, Mon Dec 17 15:54:31 : updated lessons learned
0899ca8 - Zitong Wang, Mon Dec 17 15:53:32 : frontend in slides
59c449e - Wisdom Omuya, Mon Dec 17 15:44:47 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project added subsection to test plan
344fa36 - Wisdom Omuya, Mon Dec 17 15:44:43 : added subsection
e85f2e5 - Kun An, Mon Dec 17 15:43:46 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
```

```
f7a798d - Kun An, Mon Dec 17 15:43:19 : Added example code to presentation and added my lessons to final report.
eb719f0 - John Chan, Mon Dec 17 20:41:38 2012 +0000 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
cbf24fa - John Chan, Mon Dec 17 20:41:29 2012 +0000 : edited final.tex
8c15f04 - Wisdom Omuya, Mon Dec 17 15:40:03 : moved frontend
559ac78 - Wisdom Omuya, Mon Dec 17 15:39:44 : added lessons learned
67e3c61 - Wisdom Omuya, Mon Dec 17 15:36:48 : removed headers
98d26dd - Wisdom Omuya, Mon Dec 17 15:30:00 : oh goodness!
fbea68e - Wisdom Omuya, Mon Dec 17 15:18:50 : added log entries
cfa84a8 - Wisdom Omuya, Mon Dec 17 15:17:56 : added log entries
2225f0a - John Chan, Mon Dec 17 20:14:53 2012 +0000 : conflict
c208917 - John Chan, Mon Dec 17 20:10:34 2012 +0000 : changed testsuite
b0fb43d - Wisdom Omuya, Mon Dec 17 15:09:00 : added log
af6d0c0 - Wisdom Omuya, Mon Dec 17 15:01:30 : fixed conflict
5e8914e - Wisdom Omuya, Mon Dec 17 14:59:04 : updates to final report
c66a45f - John Chan, Mon Dec 17 19:57:52 2012 +0000 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
b59d126 - John Chan, Mon Dec 17 19:57:47 2012 +0000 : changed testsuite
e0e0385 - David Mauskop, Mon Dec 17 14:54:45 : intro updated
c22bfb0 - John Chan, Mon Dec 17 19:49:11 2012 +0000 : resolved conflicts
4dfcd50 - John Chan, Mon Dec 17 19:46:26 2012 +0000 : added to section 6
1870cff - Wisdom Omuya, Mon Dec 17 14:45:10 : style
2153255 - Wisdom Omuya, Mon Dec 17 14:43:09 : cleaned language
f3e511e - Wisdom Omuya, Mon Dec 17 14:41:57 : revert
7c49fb4 - Wisdom Omuya, Mon Dec 17 14:37:25 : added style guide
2f753d6 - Wisdom Omuya, Mon Dec 17 14:36:32 : added style guide
313b6fb - David Mauskop, Mon Dec 17 14:33:44 : Code table added
f440e5c - Zitong Wang, Mon Dec 17 14:30:58 : lesson learned of Zitong
c5cdba7 - Zitong Wang, Mon Dec 17 14:30:06 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
7430039 - Zitong Wang, Mon Dec 17 14:29:58 : lesson learned of Zitong
0289a18 - Wisdom Omuya, Mon Dec 17 14:25:51 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project changes to project plan
0ba248f - Wisdom Omuya, Mon Dec 17 14:25:39 : changes to project plan
2621af9 - David Mauskop, Mon Dec 17 14:13:44 : Intro added
e57ca00 - Kun An, Mon Dec 17 13:59:46 : Updated the tree example.
58e41fe - John Chan, Mon Dec 17 09:53:10 : removed some files, edited gitignore
1d0bba6 - John Chan, Mon Dec 17 09:35:58 : added backend section to presentation
dfb9893 - John Chan, Sun Dec 16 23:36:49 : added some notes for presentation
1925976 - John Chan, Sun Dec 16 22:03:57 : added close_graph so it doesnt freeze my machine
96acb00 - David Mauskop, Sun Dec 16 19:02:20 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
20760e2 - David Mauskop, Sun Dec 16 19:02:05 : Added emptyLayer to stdlib
926664e - John Chan, Sun Dec 16 17:51:50 : pdf
4ab1c3a - John Chan, Sun Dec 16 17:51:27 : merge conflict fixed
6f6f9f1 - John Chan, Sun Dec 16 17:50:16 : fixed & problem and also patched my section from previous commit
a7b177f - Kun An, Sun Dec 16 17:43:31 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
f3fb186 - Kun An, Sun Dec 16 17:43:26 : Added return statement. Fixed a bug in clock.cv.
6ef2fdb - Zitong Wang, Sun Dec 16 17:41:55 : fixed an error in final.tex
ba22dcc - Zitong Wang, Sun Dec 16 17:32:17 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
b1bb5b4 - Kun An, Sun Dec 16 17:31:35 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
```

```
0f5013f - Zitong Wang, Sun Dec 16 17:31:34 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
8de1b9c - Kun An, Sun Dec 16 17:31:29 : Added return statement to bezier.cv
7abdd12 - Zitong Wang, Sun Dec 16 17:31:27 : semantic port in final report
c52678d - John Chan, Sun Dec 16 17:25:43 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
eb97ed0 - John Chan, Sun Dec 16 17:25:40 : cleaned up appendix
41ee38f - David Mauskop, Sun Dec 16 17:19:06 : Animated curve logo
6570550 - John Chan, Sun Dec 16 17:16:45 : fixed tests to conform with following of strict requirement of return statements
e271311 - John Chan, Sun Dec 16 17:09:49 : handled merge
544e2b5 - John Chan, Sun Dec 16 16:59:18 : final sections
a460e68 - Zitong Wang, Sun Dec 16 16:57:25 : delete addtolayer from token of parser
e8e63ff - Kun An, Sun Dec 16 16:42:09 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
a6af069 - Kun An, Sun Dec 16 16:40:43 : Modified the example section in LRM.
83695b8 - Zitong Wang, Sun Dec 16 16:36:01 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
81c1084 - Zitong Wang, Sun Dec 16 16:35:40 : LRM change for fund and var declarition
fe058bb - Wisdom Omuya, Sun Dec 16 16:34:15 : several changes
91ba025 - Wisdom Omuya, Sun Dec 16 16:26:02 : updated references
6677e9a - Wisdom Omuya, Sun Dec 16 16:14:53 : fixing merge conflict
d0c7710 - Wisdom Omuya, Sun Dec 16 16:13:18 : updated lrm code listing format
c8573fb - Wisdom Omuya, Sun Dec 16 16:12:20 : updated section headers
aab96de - Wisdom Omuya, Sun Dec 16 16:08:05 : updated LRM as well
a6d6230 - Wisdom Omuya, Sun Dec 16 16:06:15 : fixed max nesting
d80c6b8 - Wisdom Omuya, Sun Dec 16 16:03:06 : updated final report
b66844c - Zitong Wang, Sun Dec 16 15:59:25 : add check for return statement
0e56dbf - Kun An, Sun Dec 16 15:39:23 : Added the example section and part of front end section.
57ffd4a - Kun An, Sun Dec 16 12:07:29 : Added example: tree creation and traversal.
1aa9dbf - John Chan, Sun Dec 16 09:14:50 : bytecode final.tex
4cd366b - John Chan, Sat Dec 15 17:34:09 : flow added to final
2a8d2e7 - John Chan, Sat Dec 15 17:26:00 : added flowchart
e1aed05 - John Chan, Sat Dec 15 16:00:30 : overview of architectural design
19d150d - John Chan, Sat Dec 15 11:05:26 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
acb61ed - John Chan, Sat Dec 15 11:05:19 : edited LRM section
3ab5706 - David B Mauskop, Sat Dec 15 10:37:31 : Stdlib updates
0ccdfa1 - David B Mauskop, Sat Dec 15 10:31:02 : Hanoi now prettier
25922f4 - Wisdom Omuya, Sat Dec 15 00:24:35 : added rough project plan
e789d13 - Wisdom Omuya, Sat Dec 15 00:17:15 : added a bit of timeline
cbb1691 - Kun An, Fri Dec 14 22:52:33 : Added the clock example and two more patterns.
4d76256 - John Chan, Fri Dec 14 21:24:12 : found and replaced all getLSize with getSize
3139831 - John Chan, Fri Dec 14 20:54:47 : changed the getX and getY to be in dot op form in some of the tests, all pass
5fb3bd3 - Wisdom Omuya, Fri Dec 14 17:50:12 : added dave :)
6345e06 - Wisdom Omuya, Fri Dec 14 17:48:54 : restructured final report
442b262 - Wisdom Omuya, Fri Dec 14 17:45:26 : Merge branch 'master' of bitbucket.org:John Chan/plt-project make changes
80e6673 - Wisdom Omuya, Fri Dec 14 17:45:07 : changes to final report
0305a0f - Zitong Wang, Fri Dec 14 17:37:22 : remove AddToLayer
2a17ebc - Wisdom Omuya, Fri Dec 14 17:29:04 : override built-ins
85d9f50 - John Chan, Fri Dec 14 15:12:37 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
```

```
4ff2f16 - John Chan, Fri Dec 14 15:12:33 : fixed default curve action to be compile, the index was incorrect (curve.ml)
d5cef46 - Kun An, Fri Dec 14 14:58:15 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
efa904b - Kun An, Fri Dec 14 14:58:02 : Moved examples
76ec316 - John Chan, Fri Dec 14 14:55:27 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
972448a - Kun An, Fri Dec 14 14:48:06 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
b1c8bd2 - Kun An, Fri Dec 14 14:48:01 : Added rotate and scale to stl. Added two more examples.
691596b - David B Mauskop, Fri Dec 14 14:39:36 : Stdlib updates and hanoi example added
c80c58c - John Chan, Fri Dec 14 13:40:49 : changed the debug flag in execute
d6698c3 - John Chan, Fri Dec 14 09:18:10 : removed Trans from code, this is not needed
a602b9f - Wisdom Omuya, Thu Dec 13 21:33:31 : not exactly but anyways
495afe1 - Wisdom Omuya, Thu Dec 13 21:27:20 : clean before commit
7d377e6 - Wisdom Omuya, Thu Dec 13 21:26:13 : cleaned things up a bit
f847f4b - Wisdom Omuya, Thu Dec 13 21:22:16 : cleaned files
9ea1cfa - Wisdom Omuya, Thu Dec 13 21:21:59 : updated .gitignore
4b6d98e - Wisdom Omuya, Thu Dec 13 21:20:42 : moved graphics example
e54a6c3 - Wisdom Omuya, Thu Dec 13 21:06:14 : Merge branch 'master' of bitbucket.org:John Chan/plt-project updated example
ab93953 - Wisdom Omuya, Thu Dec 13 21:06:09 : updated example
522fe43 - Kun An, Thu Dec 13 20:54:42 : reverted the example back
1061fd4 - Kun An, Thu Dec 13 20:51:47 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
464141e - Kun An, Thu Dec 13 20:49:26 : i++ and i-- is working
e2303d3 - Zitong Wang, Thu Dec 13 14:11:45 : Add support for recursive function in semantic.ml
7196ec4 - John Chan, Thu Dec 13 12:16:17 : removed run.sh
64e867b - John Chan, Thu Dec 13 12:12:49 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
485a70a - John Chan, Thu Dec 13 12:12:45 : added old tests from microc, removed tests folder (but test is still intact)
42d8cda - Wisdom Omuya, Thu Dec 13 11:35:01 : removed random test
29f4478 - Wisdom Omuya, Thu Dec 13 11:33:26 : fixed random function
39e0dd5 - John Chan, Thu Dec 13 11:20:30 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
904feb0 - John Chan, Thu Dec 13 11:19:52 : transform works, similar to translate
be7ebd0 - Zitong Wang, Thu Dec 13 11:18:21 : add check for random()
de4dc7a - John Chan, Thu Dec 13 10:58:18 : added testb, which bypasses all checking, fordebugging purposes
efe5b23 - John Chan, Thu Dec 13 10:54:24 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
fa11e9c - John Chan, Thu Dec 13 10:53:56 : refactored curve.ml, added a bypass function to bypass semantic checker
f4da9e7 - Zitong Wang, Thu Dec 13 10:50:45 : fix bug of semantic.ml
b85d705 - John Chan, Thu Dec 13 10:28:07 : fixed freezing index out of bounds issue
9c28907 - John Chan, Thu Dec 13 06:38:40 : fixed nth, random is the only test to fail
bb09859 - Wisdom Omuya, Thu Dec 13 02:43:16 : caps check
20243ac - Wisdom Omuya, Thu Dec 13 02:41:13 : removed extraneous getY
09393bd - Wisdom Omuya, Thu Dec 13 02:31:57 : updated lrm
4f6b449 - Zitong Wang, Thu Dec 13 01:49:15 : Change system fund part of LRM
57c605b - Zitong Wang, Thu Dec 13 01:31:11 : add setPoint back
bf122a8 - John Chan, Thu Dec 13 00:33:50 : modified test code to use the dot format
540fa43 - Zitong Wang, Wed Dec 12 23:00:18 : change in semantic.ml
e2e21fe - Wisdom Omuya, Wed Dec 12 20:10:35 : updated lrm
70c0bc6 - Wisdom Omuya, Wed Dec 12 20:06:06 : updated example
a236dc9 - Wisdom Omuya, Wed Dec 12 20:02:56 : updated example
dd7a586 - Wisdom Omuya, Wed Dec 12 19:59:27 : updated example
```

```
571a0da - Wisdom Omuya, Wed Dec 12 19:07:55 : minor changes
67a2512 - Wisdom Omuya, Wed Dec 12 18:58:18 : incorporating semantic...
71ed973 - Zitong Wang, Wed Dec 12 18:52:08 : Merge commit '997aba6a9da90f14f365e8d8bb9d340eeb052a9e'
2b86e98 - Zitong Wang, Wed Dec 12 18:50:57 : Update of semantic
997aba6 - Wisdom Omuya, Wed Dec 12 18:42:23 : all tests now pass
8ce7b65 - Wisdom Omuya, Wed Dec 12 18:41:35 : removed unused tokens
784407c - Wisdom Omuya, Wed Dec 12 18:38:02 : updated stdlib
4ac7a8d - Kun An, Wed Dec 12 18:36:31 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
4a7b193 - Kun An, Wed Dec 12 18:36:25 : Added circle to STL
98e169c - Wisdom Omuya, Wed Dec 12 18:35:56 : updated makefile
cd63aec - Wisdom Omuya, Wed Dec 12 18:33:27 : changed stlib to stdlib, updated curve.ml
235efa5 - Wisdom Omuya, Wed Dec 12 18:30:02 : incorporated stdlib into prog
dee2c24 - Wisdom Omuya, Wed Dec 12 17:48:35 : removed main from stdlib
0510ee8 - Wisdom Omuya, Wed Dec 12 17:45:01 : removed unnecessary bytecode instrs.
053bb7b - Wisdom Omuya, Wed Dec 12 17:29:03 : Updated stack space Merge branch 'master' of bitbucket.org:John Chan/plt-project
f9cab0e - Wisdom Omuya, Wed Dec 12 17:28:51 : increased stack space
f5db2ab - Kun An, Wed Dec 12 17:23:45 : Added to example programs.
dcbc6e3 - David B Mauskop, Wed Dec 12 16:56:27 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
9ae4cf3 - David B Mauskop, Wed Dec 12 16:53:58 : Added to standard library
acc3072 - Wisdom Omuya, Wed Dec 12 16:43:24 : added 'expected' random outcome
3dff651 - Wisdom Omuya, Wed Dec 12 16:38:41 : added random builtin
b589388 - Wisdom Omuya, Wed Dec 12 16:24:13 : updated lrm
739d74b - John Chan, Wed Dec 12 10:49:11 : translate works
90ed8d5 - John Chan, Wed Dec 12 09:52:03 : added print layer to stl
5ae0023 - John Chan, Wed Dec 12 09:50:54 : set curve works
2896ff0 - John Chan, Tue Dec 11 18:15:58 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
971b5a7 - John Chan, Tue Dec 11 17:59:09 : lsize
5fb33c7 - Wisdom Omuya, Tue Dec 11 16:44:50 : minor edits
4593697 - John Chan, Tue Dec 11 16:43:47 : added dot notation as well
376ffbb - John Chan, Tue Dec 11 16:42:58 : added layer size function
284bd79 - Wisdom Omuya, Tue Dec 11 16:40:14 : updated lrm
1cdf6e4 - Wisdom Omuya, Tue Dec 11 16:39:51 : removed boolean; cleaned report
be19444 - Wisdom Omuya, Tue Dec 11 16:30:36 : some progress on the final report
bef4185 - Wisdom Omuya, Tue Dec 11 16:10:50 : Merge branch 'master' of bitbucket.org:John Chan/plt-project full screen on execute
66d46cc - Wisdom Omuya, Tue Dec 11 16:10:42 : changed execute to open full screen
e0bd5a3 - Wisdom Omuya, Tue Dec 11 16:09:03 : updated final report
50a3eca - John Chan, Tue Dec 11 15:56:17 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
4cfe4a1 - John Chan, Tue Dec 11 15:56:10 : added stl
196ce2b - Wisdom Omuya, Tue Dec 11 15:42:03 : incorporated lrm into final report
9871a77 - Wisdom Omuya, Tue Dec 11 15:38:37 : removed string and boolean constants
29b986f - Wisdom Omuya, Tue Dec 11 15:30:09 : lrm changes Merge branch 'master' of bitbucket.org:John Chan/plt-project
14d2311 - Wisdom Omuya, Tue Dec 11 15:30:02 : further cleaned up lrm
7cf23f8 - Wisdom Omuya, Tue Dec 11 15:25:15 : cleaned up example listings
d25d8ea - John Chan, Tue Dec 11 15:08:11 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
9bf70d2 - John Chan, Tue Dec 11 15:08:06 : notes
c061481 - Wisdom Omuya, Tue Dec 11 15:07:12 : added layer builtins
1eb147b - Wisdom Omuya, Tue Dec 11 15:01:52 : added curve builtins
```

```
97e9118 - Wisdom Omuya, Tue Dec 11 14:58:36 : added point builtins
270ee3a - Wisdom Omuya, Tue Dec 11 14:56:31 : added clear builtin
928e3cc - John Chan, Tue Dec 11 14:56:05 : getcurve works
318ec39 - Wisdom Omuya, Tue Dec 11 14:55:48 : added print builtin
f8e69c1 - Wisdom Omuya, Tue Dec 11 14:54:19 : added pause builtin
14d6dac - Wisdom Omuya, Tue Dec 11 14:52:12 : removed math section
3bef391 - Wisdom Omuya, Tue Dec 11 14:51:58 : updated draw description
7413f35 - Wisdom Omuya, Tue Dec 11 14:49:50 : edited variable ↩
    declaration
2af58ca - Wisdom Omuya, Tue Dec 11 14:48:13 : removed doubles from lrm
fa0304e - Wisdom Omuya, Tue Dec 11 14:42:04 : removed doubles from lrm
c264e3c - Wisdom Omuya, Tue Dec 11 14:40:01 : updating lrm
38ab2c3 - Wisdom Omuya, Tue Dec 11 14:34:22 : working on language ↩
    tutorial
b682a40 - Wisdom Omuya, Tue Dec 11 14:29:44 : added introduction
0e44198 - Wisdom Omuya, Tue Dec 11 14:27:08 : added introduction
f228e01 - John Chan, Tue Dec 11 14:08:22 : started get curve and added↩
     rta
135e989 - John Chan, Tue Dec 11 11:47:33 : added tests for new ↩
    bytecode instruction rta
0b5fd91 - John Chan, Tue Dec 11 11:37:19 : fixed bug with overlapping ↩
    stack values
68ca0dd - John Chan, Tue Dec 11 11:30:05 : found bug with overlapping ↩
    stack values
69b9771 - John Chan, Tue Dec 11 10:29:07 : added rta
7847373 - Wisdom Omuya, Mon Dec 10 19:31:49 : created presentation ↩
    skeleton
b4e25be - Wisdom Omuya, Sat Dec 8 18:14:10 : updated test
ea99bb9 - John Chan, Sat Dec 8 17:41:35 : added return curve test
d0e3c70 - John Chan, Fri Dec 7 22:11:04 : notes
f256d1d - John Chan, Fri Dec 7 20:42:26 : renamed get to getPoint, etc
89a7372 - John Chan, Fri Dec 7 16:30:16 : notes
b9c1237 - John Chan, Fri Dec 7 16:16:21 : refactor
79441e3 - John Chan, Fri Dec 7 15:03:12 : notes
75fcd96 - Wisdom Omuya, Wed Dec 5 21:55:39 : edited scoping section
ffa9c97 - Wisdom Omuya, Wed Dec 5 21:49:45 : changed Integer to int
bfc198b - Wisdom Omuya, Wed Dec 5 21:49:25 : updated some sections
22a8f0f - Wisdom Omuya, Wed Dec 5 21:38:40 : added initial final ↩
    report doc
5276bdf - John Chan, Tue Dec 4 18:09:57 : dot operator for set works
f1e9574 - Ubuntu, Tue Dec 4 19:41:07 2012 +0000 : notes
f3e0a0c - Ubuntu, Tue Dec 4 19:11:57 2012 +0000 : set works
b267b21 - Ubuntu, Tue Dec 4 16:41:37 2012 +0000 : middle of set ↩
    function
9b2ad9b - John Chan, Mon Dec 3 21:54:09 : refactoreed
27ededе - Wisdom Omuya, Mon Dec 3 18:47:44 : added clear
da351b2 - John Chan, Mon Dec 3 10:54:16 : notes
bcc5125 - Wisdom Omuya, Mon Dec 3 00:16:00 : updated curve.ml for ↩
    semantic
ab9c460 - Wisdom Omuya, Mon Dec 3 00:15:01 : updated makefile
2ab25cc - Wisdom Omuya, Mon Dec 3 00:10:31 : added semantic.ml
70b2827 - John Chan, Sat Dec 1 17:58:58 : refactored
59ba8d1 - John Chan, Sat Dec 1 16:09:38 : added multi expression ↩
    testing
937654e - John Chan, Sat Dec 1 16:08:54 : fixed bug on stack when ↩
    multi drops are required
005a8aa - John Chan, Sat Dec 1 11:29:31 : call
a3bf9e2 - John Chan, Sat Dec 1 11:19:33 : fixed dropsize area
ec8d35a - John Chan, Sat Dec 1 10:59:10 : fixed sizeoftype
9756476 - John Chan, Sat Dec 1 00:49:40 : tested dot op on get
33838d8 - John Chan, Sat Dec 1 00:40:41 : better implementation of Ind
ec2c8f1 - John Chan, Sat Dec 1 00:37:13 : get for point works, but ↩
    ugly implementation of Ind
1a78f61 - John Chan, Fri Nov 30 18:55:17 : notes
dcfd249 - John Chan, Fri Nov 30 15:15:13 : notes
d00b93a - John Chan, Fri Nov 30 13:05:37 : notes
8bc7bf6 - John Chan, Thu Nov 29 17:00:05 : setx sety work
78f5fd7 - John Chan, Thu Nov 29 16:54:01 : setx works
```

```
81d5d33 - John Chan, Thu Nov 29 13:30:31 : added dotop tests
afc389a - John Chan, Thu Nov 29 13:25:53 : improved index counting for built_in_functions
b49d966 - John Chan, Thu Nov 29 13:17:26 : dot op works for getx() gety()
2912e06 - Wisdom Omuya, Wed Nov 28 19:49:54 : revert to saner animation
6b88faa - Wisdom Omuya, Wed Nov 28 19:49:19 : increased stack space
c12d552 - Wisdom Omuya, Wed Nov 28 19:49:09 : cleaned up pause
e6678bf - Wisdom Omuya, Wed Nov 28 18:52:08 : less awkward layer
65a8279 - Wisdom Omuya, Wed Nov 28 18:43:43 : layers working!
0dcd4a3 - Wisdom Omuya, Wed Nov 28 16:53:03 : revert for tests
8563777 - Wisdom Omuya, Wed Nov 28 16:34:14 : updated curve drawing mechanism
38deccf - John Chan, Wed Nov 28 15:06:54 : lists load correctly
3336174 - Wisdom Omuya, Wed Nov 28 14:01:09 : passing graphics test
3b91265 - Wisdom Omuya, Wed Nov 28 13:57:21 : updated interpreter
79cb5a5 - Wisdom Omuya, Wed Nov 28 13:55:19 : fixed ogr bug
3b23a7a - Wisdom Omuya, Wed Nov 28 13:46:37 : reverted to working compile.ml
4624f06 - Wisdom Omuya, Wed Nov 28 13:43:23 : moved curve tests
dfe10d3 - Wisdom Omuya, Wed Nov 28 13:21:54 : added pause to compile
d96ed26 - Wisdom Omuya, Wed Nov 28 13:21:20 : updated test file
83b942b - Wisdom Omuya, Wed Nov 28 13:18:03 : updated compile to open graph
6f2ab4b - Wisdom Omuya, Wed Nov 28 13:14:20 : added draw function
5eb370d - Wisdom Omuya, Wed Nov 28 13:13:34 : added top-level .gitignore
2bc4ab1 - Wisdom Omuya, Wed Nov 28 13:12:46 : added test-curv1.cv
7388dab - Wisdom Omuya, Wed Nov 28 12:59:41 : updated makefile/test-layer1.out
9d15987 - Wisdom Omuya, Wed Nov 28 12:56:40 : updated makefile
61c5d15 - Wisdom Omuya, Wed Nov 28 12:56:23 : add, jsr(-2), ogr, cgr
fa2fdaf - John Chan, Wed Nov 28 12:50:01 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
c35422c - John Chan, Wed Nov 28 12:49:42 : added layer test, changed execute
a821923 - Wisdom Omuya, Wed Nov 28 12:47:37 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
2a37231 - Wisdom Omuya, Wed Nov 28 12:47:29 : removed microc.ml
20db217 - John Chan, Wed Nov 28 12:16:47 : added solocall test
534bb61 - John Chan, Wed Nov 28 12:16:11 : fixed single call to fn stack inconsistency
90d4de9 - John Chan, Wed Nov 28 11:42:16 : enumerated types, added a return type, fixed tests to include return types
1ae3d3f - John Chan, Wed Nov 28 11:06:34 : changed testall script, test is now the correct folder
34e9520 - John Chan, Tue Nov 27 20:38:37 : fixed bug
09dc0fe - John Chan, Tue Nov 27 15:08:57 : merged
9a606ab - John Chan, Tue Nov 27 15:03:58 : moved some files
6e1c580 - John Chan, Tue Nov 27 14:53:23 : removed drpe
5a8337e - John Chan, Tue Nov 27 14:31:33 : added test case
93a6d54 - John Chan, Tue Nov 27 14:16:27 : return point works
0faaf93 - John Chan, Tue Nov 27 00:17:37 : sizeofexpr
97475d8 - John Chan, Mon Nov 26 18:59:07 : added run
01a113e - John Chan, Mon Nov 26 16:30:45 : bug
c6fec72 - John Chan, Mon Nov 26 15:04:04 : getx gety work
b23efd2 - John Chan, Mon Nov 26 14:09:32 : added gety
8d029be - John Chan, Mon Nov 26 13:46:15 : getx works
5dba127 - John Chan, Sun Nov 25 17:11:47 : debugged stack op
7cb2e28 - John Chan, Sun Nov 25 08:51:59 : fixed size of id
053b4c6 - John Chan, Sat Nov 24 11:04:59 : need to fix printing 2vals
21a849b - John Chan, Sat Nov 24 09:50:26 : checkpoint
dfa79e9 - John Chan, Sat Nov 24 09:27:19 : checkpoint
fc0e952 - John Chan, Sat Nov 24 09:07:59 : finished drp in stmt
a0ef768 - John Chan, Sat Nov 24 08:58:41 : fixed rts
11a3221 - John Chan, Fri Nov 23 17:31:40 : ed
35ab3d2 - John Chan, Fri Nov 23 09:18:34 : edit
```

```
d6901a4 - John Chan, Thu Nov 22 23:53:05 : edit
d884ba7 - John Chan, Thu Nov 22 23:52:31 : ed
b3ae68e - Ubuntu, Thu Nov 22 23:12:50 2012 +0000 : ed
883dbb4 - John Chan, Thu Nov 22 00:25:02 : notes
7a89546 - John Chan, Wed Nov 21 20:53:05 : notes
c998a12 - John Chan, Wed Nov 21 20:46:38 : made mapping for vdecls
95af749 - John Chan, Wed Nov 21 09:45:11 : edited notes
9c5ee9f - John Chan, Wed Nov 21 09:37:33 : finished expr in compile
07d9f56 - John Chan, Wed Nov 21 09:09:30 : added enum_var
f53dadc - John Chan, Tue Nov 20 23:37:01 : reverted back
e5d4fc7 - John Chan, Tue Nov 20 13:20:20 : added notesw
0f625df - John Chan, Tue Nov 20 11:41:35 : patched files
5b5cf80 - John Chan, Tue Nov 20 11:41:18 : copied master to this
a8bcabe - John Chan, Tue Nov 20 11:25:08 : notes
d412c49 - John Chan, Mon Nov 19 00:50:46 : removed cexpr
3d35b04 - John Chan, Sat Nov 17 17:14:51 : ed
3fa97b3 - John Chan, Sat Nov 17 15:46:36 : bytecode
c96a4e7 - John Chan, Sat Nov 17 11:16:36 : notes
60c3a30 - John Chan, Sat Nov 17 08:41:48 : lrm
8dc85d4 - John Chan, Fri Nov 16 22:09:57 : note
19b7324 - John Chan, Fri Nov 16 20:27:34 : notes
a5205ac - John Chan, Fri Nov 16 11:21:47 : cleaner interpret
b06757c - John Chan, Fri Nov 16 10:52:36 : updated lrm on jc branch
4d760fa - John Chan, Fri Nov 16 01:53:37 : fix two cexpr errors (todo)
5f17e2f - John Chan, Fri Nov 16 00:51:47 : added some more to parser, ↩
    and scanner
d53387f - John Chan, Thu Nov 15 14:06:48 : displays point
91ac389 - John Chan, Wed Nov 14 23:47:28 : kind of works for ↩
    displaying through interpreter
5d93a3b - John Chan, Wed Nov 14 20:49:32 : added grtest example
cfc1bd5 - John Chan, Wed Nov 14 16:06:56 : notes
3759227 - John Chan, Wed Nov 14 15:25:14 : slight edits to jc
75a470d - John Chan, Wed Nov 14 15:15:02 : edit
14c48c8 - John Chan, Wed Nov 14 14:12:14 : added notes
b9993e3 - John Chan, Fri Nov 9 17:53:18 : Point added
f60426d - John Chan, Wed Nov 7 11:26:47 : changed formatting of ↩
    scanner.mll; fixed test-func2.mc - fun is a keyword for our ↩
    language, and they use this to call their function.  This threw ↩
    the test off.  It works now.
ca8f9b1 - John Chan, Mon Nov 5 14:29:49 : slight edit to scanner.
653f9a6 - John Chan, Mon Nov 5 14:23:59 : revered back to the old ↩
    testall script.
6a81f9c - John Chan, Mon Nov 5 14:17:42 : Added necessary files.  Did ↩
    not work before because of missing files.  .out line removed from ↩
    gitignore
1eba528 - Wisdom Omuya, Fri Nov 2 14:53:34 2012 -0400 : fixed command ↩
    not found error
85275f2 - Wisdom Omuya, Fri Nov 2 00:00:00 2012 -0400 : fixed scanner/↩
    testall so tests still pass
a0a7ba2 - Wisdom Omuya, Thu Nov 1 23:40:11 2012 -0400 : updated ↩
    scanner/parser
ca93690 - Wisdom Omuya, Thu Nov 1 17:18:45 2012 -0400 : reverted to ↩
    working condition
aaf9253 - Wisdom Omuya, Thu Nov 1 17:12:26 2012 -0400 : Merge branch '↩
    master' of https://bitbucket.org/John Chan/plt-project
b7a2d5d - Wisdom Omuya, Thu Nov 1 17:12:01 2012 -0400 : separated ↩
    interface
7319799 - John Chan, Thu Nov 1 16:44:48 2012 -0400 : removed compiled ↩
    code
73a7c23 - Wisdom Omuya, Thu Nov 1 16:43:28 2012 -0400 : updated .↩
    gitignore
a541ef6 - Wisdom Omuya, Thu Nov 1 16:43:03 2012 -0400 : updated .↩
    gitignore
77e214d - Wisdom Omuya, Thu Nov 1 16:40:42 2012 -0400 : added micro-c ↩
    code
24dc3f1 - Wisdom Omuya, Sat Oct 27 13:15:39 2012 -0400 : updated ↩
    object types section
```

```
b626bbd - Wisdom Omuya, Sat Oct 27 12:48:58 2012 -0400 : claimed back our swag
73fc83d - Wisdom Omuya, Sat Oct 27 12:48:25 2012 -0400 : updated object types section
c7bb21e - ka2401, Sat Oct 27 11:38:08 2012 -0400 : modified examples in LRM
d6e769e - Zitong Wang, Sat Oct 27 01:08:29 2012 -0400 : Declarations and Sys functions section updated
ae90441 - Zitong Wang, Sat Oct 27 01:04:15 2012 -0400 : Merge branch 'master' of https://bitbucket.org/John_Chan/plt-project
3306422 - Zitong Wang, Sat Oct 27 00:56:25 2012 -0400 : Declarations and System Functions section of LRM updated
2446c9c - John Chan, Sat Oct 27 00:27:45 2012 -0400 : changed section 3
f12db21 - David Mauskop, Fri Oct 26 21:45:09 2012 -0400 : Expressions and Operators section of LRM updated
b3c399a - Wisdom Omuya, Fri Oct 26 16:06:01 2012 -0400 : changed boolean case
a2ab46b - Wisdom Omuya, Fri Oct 26 16:05:12 2012 -0400 : relinquished our swag
734c39d - Wisdom Omuya, Fri Oct 26 15:57:35 2012 -0400 : updated scanner
4688f2b - Wisdom Omuya, Fri Oct 26 15:56:43 2012 -0400 : updated scanner
5c41831 - John Chan, Fri Oct 26 15:40:04 2012 -0400 : removed 2 from footer
54e77fe - Wisdom Omuya, Wed Oct 24 16:13:03 2012 -0400 : updated scanner
5a129d2 - ka2401, Wed Oct 24 15:04:49 2012 -0400 : Added the example section.
6b39ed5 - Wisdom Omuya, Wed Oct 24 10:46:17 2012 -0400 : updated .gitignore
2485721 - Wisdom Omuya, Wed Oct 24 10:45:06 2012 -0400 : initial version for scanner
a4aa25e - John Chan, Wed Oct 24 00:27:35 2012 -0400 : added <<, dot notation, print keyword exception
72126c3 - John Chan, Tue Oct 23 22:30:29 2012 -0400 : removed toc and added it to the gitignore
95d5e2a - Wisdom Omuya, Mon Oct 22 19:39:03 2012 -0400 : removed 'start' as required
245679e - Wisdom Omuya, Mon Oct 22 17:44:19 2012 -0400 : compiled pdf
3d4cb0b - David Mauskop, Mon Oct 22 16:41:24 2012 -0400 : Added operator precedence and associativity
cffdde3 - Wisdom Omuya, Sat Oct 20 18:44:09 2012 -0400 : added zitong's section
bba6561 - Wisdom Omuya, Thu Oct 18 20:31:53 2012 -0400 : removed .toc file
9a1bc9b - Wisdom Omuya, Thu Oct 18 20:30:55 2012 -0400 : .gitignore now working
7056e46 - Wisdom Omuya, Thu Oct 18 20:26:56 2012 -0400 : updated .gitignore
6989cfd - John Chan, Thu Oct 18 20:23:22 2012 -0400 : removed the toc file
3355f8d - Wisdom Omuya, Thu Oct 18 20:21:51 2012 -0400 : fixed typos
878427c - Wisdom Omuya, Thu Oct 18 20:20:28 2012 -0400 : added cover page
74d1082 - Wisdom Omuya, Thu Oct 18 20:12:55 2012 -0400 : added header/footer
9a218e7 - Wisdom Omuya, Thu Oct 18 20:04:33 2012 -0400 : updated program definition
e64ae73 - Wisdom Omuya, Thu Oct 18 19:52:22 2012 -0400 : added statements
80a3303 - Wisdom Omuya, Thu Oct 18 19:47:00 2012 -0400 : added scoping section
e4dfbf7 - Wisdom Omuya, Thu Oct 18 19:29:43 2012 -0400 : added introduction/program def.
5d2a6a4 - John Chan, Thu Oct 18 13:44:04 2012 -0400 : almost finished with my part (3)
```

```
498d7ab - John Chan, Wed Oct 17 18:21:17 2012 -0400 : edited my part
4831976 - John Chan, Wed Oct 17 17:35:21 2012 -0400 : added table of contents
f99999d - Wisdom Omuya, Wed Oct 17 16:51:13 2012 -0400 : edits for print/booleans
bb7e5e9 - Wisdom Omuya, Mon Oct 15 18:26:10 2012 -0400 : updated lrm
5e3c42a - Wisdom Omuya, Mon Oct 15 17:31:21 2012 -0400 : added lrm
41d8fac - Wisdom Omuya, Mon Oct 15 13:39:00 2012 -0400 : removed little mac fritter
d3ab4bb - Wisdom Omuya, Mon Oct 15 13:29:25 2012 -0400 : whitespace
f09c22d - Wisdom Omuya, Mon Oct 15 13:27:33 2012 -0400 : changed layer syntax
994d8f6 - Wisdom Omuya, Mon Oct 15 13:27:09 2012 -0400 : changed control point syntax
46baa09 - Wisdom Omuya, Mon Oct 15 13:26:33 2012 -0400 : fixed typo
8b3a39f - Wisdom Omuya, Sun Oct 14 15:50:37 2012 -0400 : included conditionals
7cc9dcf - Wisdom Omuya, Sun Oct 14 15:40:53 2012 -0400 : updated for loop
4a5459b - Wisdom Omuya, Sun Oct 14 15:39:08 2012 -0400 : render all objects in namespace
89c091b - John Chan, Thu Oct 11 18:28:36 2012 -0400 : Merge branch 'master' of bitbucket.org:John Chan/plt-project
2ec6f2c - John Chan, Thu Oct 11 18:27:20 2012 -0400 : added more information, and some examples
eef4c8f - John Chan, Thu Oct 11 18:24:38 2012 -0400 : added more information, and some examples
fd98798 - John Chan, Thu Oct 11 00:06:48 2012 -0400 : added some more to curve2
db80c6d - John Chan, Tue Oct 9 13:21:06 2012 -0400 : minor changes, added layer and graphic
df99bb6 - John Chan, Tue Oct 9 13:11:26 2012 -0400 : started to make changes to 2nd iteration of proposal
b5ef945 - John Chan, Wed Oct 3 13:52:29 2012 -0400 : added Feedback for PLT Group Curve.pdf
75f61ab - Wisdom Omuya, Wed Sep 26 21:53:26 2012 -0400 : changed file name
2903b84 - Wisdom Omuya, Wed Sep 26 18:16:13 2012 -0400 : updated doc
1ae793d - Wisdom Omuya, Wed Sep 26 16:22:11 2012 -0400 : reduced # pages
4f23a8a - David Mauskop, Wed Sep 26 16:16:29 2012 -0400 : Added built-in types and operators
fdc2db9 - John Chan, Tue Sep 25 23:58:29 2012 -0400 : added some more control statements
f550bc9 - John Chan, Tue Sep 25 18:40:06 2012 -0400 : fixed margins
c9d43b1 - Wisdom Omuya, Tue Sep 25 15:14:13 2012 -0400 : cleaned motivation, features, example
e024452 - John Chan, Mon Sep 24 22:23:03 2012 -0400 : added some more to Key features
ccc0461 - Wisdom Omuya, Mon Sep 24 19:52:59 2012 -0400 : added section owners
1405913 - John Chan, Sat Sep 22 17:55:26 2012 -0400 : added more code examples.
1ff9bb5 - John Chan, Sat Sep 22 00:06:10 2012 -0400 : changed gitignore
095a94a - John Chan, Sat Sep 22 00:05:40 2012 -0400 : slight modifications - we should decide how the language should look like and what it can do
e5785ea - Wisdom Omuya, Thu Sep 20 12:12:59 2012 -0400 : added this time
9ed39d3 - John Chan, Thu Sep 20 12:00:21 2012 -0400 : added contacts back in
f47dee2 - Wisdom Omuya, Thu Sep 20 11:34:06 2012 -0400 : rough initial proposal draft
2691e71 - Wisdom Omuya, Thu Sep 6 17:01:15 2012 -0400 : added .gitignore
d61e69d - John Chan, Thu Sep 6 16:38:03 2012 -0400 : added contact information
```

5.5.2 Frontend

- Sep. & Oct.
  Series of meetings, discussing language features, scope, AST
- Nov. 15 Completed AST design; Scanner complete
- Nov. 24 Parser completed
- Dec. 1 Basic working compiler working
- Dec. 14 Builtins standard library completed
- Dec. 16 Code freeze, examples written
- Dec. 18 Documentation completed

5.6 Development Environment

The project was developed primarily on Mac OS X using Ocaml, and the following major components:

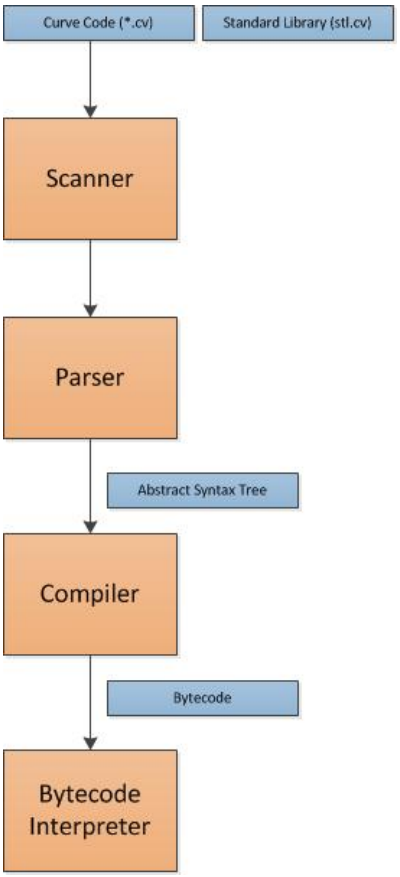
- OCaml 4.00.1 : Primary Development Language
- OCamllex : OCaml variant of lex utility for lexical analysis
- OCamlyacc : OCaml variant of yacc utility for parsing
- Git : Version control for source code
- Makefile and Regression test scripts : Make and bash scripts

6 Architectural Design

This section deals with the details of our conceptual design for Curve.

6.1 Overview

The Curve compiler has two intermediate represenations, the abstract syntax tree and the bytecode. After tokenizing with the Lexer, our Parser creates the abstract syntax tree, stripping away all unnecessary token information, such as parenthesis, commas, etc. The compiler takes in the abstract syntax tree generated by the parser and creates bytecode. Finally, the bytecode is executed by the bytecode interpreter. The bytecode is stack oriented, and so almost all of the instructions operate on the stack. The flow chart below depicts the steps that Curve code goes through, from its raw form to the tangible actions performed by the bytecode interpreter.



6.2 Frontend

6.2.1 Scanner

The scanner is used to tokenize the input file into lexemes. At this stage, we strip the comments and the whitespace off the code. Furthermore, any illegal characters or Curve reserved keywords are caught and failure is reported. The scanner is built using the ocamllex utility.

6.2.2 Parser

The parser generates the abstract syntax tree (AST) from the stream of tokens that it receives from the scanner. See the Appendix for a listing of the grammar that the parser accepts. The parser is built using the ocamlyacc utility.

6.2.3 Abstract Syntax Tree

Our AST is succinct yet enriched. We defined the value of variables as a list of integers. This definition makes all the types consistent and easy to implement. At the same time, the integer list is expressive as it can be used to represent integer, point, curve, and layer since they are just lists of different length. We also added the type field to the variable definition so that the semantic analysis can directly check the validity of variables by inspecting the type field.

6.2.4 Interpreter

The interpreter is not part of our final deliverable. However, it served as a very useful testing tool when implementing the scanner, parser, and AST. This is because the interpreter was much easier to implement and modify than the compiler. The language features were added one at a time. Each time a new front end feature was created, we used the interpreter to test whether the front end behaved as expected.

6.2.5 Semantic Analysis

The main framework of the semantic checker comes from the interpreter. Instead of evaluating the value of a statement, the semantic check evaluates the type of a statement. The checker is able to detect following errors:
(1) All kinds of type mismatch including variable assignment, LHS & RHS of an assignment statement or binary operation, parameter of user-defined function, built-in function, standard library function, etc.
(2) Number of parameters mismatched with the definition of the function.
(3) Return type mismatches with the definition of the function's return type.
(4) Undeclared variables or functions.
(5) Lack of return statement for user-defined function.

6.3 Backend

6.3.1 Bytecode

Due to the fact that there is only a limited number of useful instructions that operate on the stack, the set of bytecode instructions is relatively small. The following chart describes the different instructions that make up bytecode created by the compiler.

Instruction	Usage
`Lit of int`	Pushes a single integer value on to the top of the stack.
`Drp`	Pops the stack.
`Bin of Ast.op`	Performs the operation specified by Ast.op on the top two elements of the stack. The result replaces the first of the two elements on top of the stack.
`Lod of int`	Fetches the global variable specified by the int.
`Str of int`	Stores the global variable specified by the int.
`Lfp of int`	Loads a value indicated by the int, relative to the frame pointer.
`Sfp of int`	Puts the value from the top of the stack to a location indicated by the int, relative to the frame pointer.
`Jsr of int`	Moves the program counter to the location specified by the int. Used for jumping into a subroutine.
`Ent of int`	Prepares the stack for a new activation record by allocating appropriate number of bytes for local variables.
`Rta`	Prepares for a return by storing new program counter and new frame pointer. This is necessary because these values may be overwritten if the return object is big enough.
`Rts of int`	Returns from a subroutine. The new program counter and frame pointer values are loaded, and the stack is popped until the top of the return object is reached.
`Beq of int`	Branch if equal.
`Bne of int`	Branch not equal.
`Bra of int`	Branch.
`Ind of int`	Indirect load. The int specifies where the pointer is, and the value that this pointer points to indicates another location where the value at this location should be pushed on to the stack. The int is relative to the frame pointer, whereas the pointer that it points to is relative to itself.
`Ins of int`	Similar to `Ind`, but the value on the top of the stack is stored at the location instead.
`Hlt`	Halt.
`Ogr`	Open Graph.

6.3.2 Compiler

The main goal of the compiler is to take an abstract syntax tree as input and to convert it into a form that is easily portable. As the compiler traverses the abstract syntax tree, records of function size, types, and variable names are kept so that they can be referred to in later portions of the code. This record keeping is very important in maintaining stack consistency. Done incorrectly, the program will suffer from memory leaks or will have bugs due to the over

dropping of values from the stack. Some of the challenges in implementation include dealing with indirect values (pointers), maintaining stack consistency and writing bytecode for the built in procedures.

6.3.3 Execute

The bytecode interpreter takes in bytecode and creates actions from the bytecode. This is where the graphs are initialized, and where the stack and global variables are maintained. The goal of having an intermediate representation is so that all of the high level details of the langauge are stripped away, leaving only simple and universal operations. The simplicity of our bytecode propogates through to this interpreter, and so moving to another platform should be a very trivial task.

7 Test Plan

7.1 Sources with Target Language Programs

The following are two simple programs in curve, and its respective bytecode:

7.1.1 source – print.cv

```
1  int main()
2  {
3     print(1);
4
5  }
```

7.1.2 target – print.cv

```
1  0 global variables
2  0 Ogr
3  1 Jsr 113
4  2 Hlt
5  3 Ent 0
6  4 Lfp -3
7  5 Rta
8  6 Rts 2
9  7 Ent 0
10 8 Lfp -2
11 9 Rta
12 10 Sfp -3
13 11 Rts 2
14 12 Ent 0
15 13 Ind 1
16 14 Ind 0
17 15 Rta
18 16 Sfp -10
19 17 Drp
20 18 Sfp -11
21 19 Rts 9
```

```
20 Ent 0
21 Lfp -3
22 Lfp -2
23 Ins 2
24 Drp
25 Ins 3
26 Rta
27 Rts 5
28 Ent 0
29 Lfp -3
30 Lfp -2
31 Sub
32 Lit 8
33 Mul
34 Lit 4
35 Add
36 Lfp 1
37 Lfp 1
38 Lfp 1
39 Lfp 1
40 Lfp 1
41 Lfp 1
42 Lfp 1
43 Ind -3
44 Ind -4
45 Ind -5
46 Ind -6
47 Ind -7
48 Ind -8
49 Ind -9
50 Ind -10
51 Rta
52 Sfp -76
53 Drp
54 Sfp -77
55 Drp
56 Sfp -78
57 Drp
58 Sfp -79
59 Drp
60 Sfp -80
61 Drp
62 Sfp -81
63 Drp
64 Sfp -82
65 Drp
66 Sfp -83
67 Rts 75
68 Ent 0
69 Lfp -2
70 Rta
71 Sfp -82
72 Rts 81
73 Ent 0
74 Lfp -11
75 Lfp -10
76 Sub
77 Lit 8
78 Mul
79 Lit 12
80 Add
81 Lfp 1
82 Lfp 1
83 Lfp 1
84 Lfp 1
85 Lfp 1
86 Lfp 1
87 Lfp 1
```

```
88 Lfp -9
89 Lfp -8
90 Lfp -7
91 Lfp -6
92 Lfp -5
93 Lfp -4
94 Lfp -3
95 Lfp -2
96 Ins -10
97 Drp
98 Ins -9
99 Drp
100 Ins -8
101 Drp
102 Ins -7
103 Drp
104 Ins -6
105 Drp
106 Ins -5
107 Drp
108 Ins -4
109 Drp
110 Ins -3
111 Rta
112 Rts 10
113 Ent 1
114 Lit 1
115 Jsr -1
116 Drp
117 Rta
118 Lit 0
119 Rts 0
```

7.1.3 source – points.cv

```
int main()
{
   Point a;
   a = (1,2);
   print(a.getX());
   print(a.getY());
}
```

7.1.4 target – points.cv

```
0 global variables
0 Ogr
1 Jsr 113
2 Hlt
3 Ent 0
4 Lfp -3
5 Rta
6 Rts 2
7 Ent 0
8 Lfp -2
9 Rta
10 Sfp -3
11 Rts 2
12 Ent 0
```

```
13 Ind 1
14 Ind 0
15 Rta
16 Sfp -10
17 Drp
18 Sfp -11
19 Rts 9
20 Ent 0
21 Lfp -3
22 Lfp -2
23 Ins 2
24 Drp
25 Ins 3
26 Rta
27 Rts 5
28 Ent 0
29 Lfp -3
30 Lfp -2
31 Sub
32 Lit 8
33 Mul
34 Lit 4
35 Add
36 Lfp 1
37 Lfp 1
38 Lfp 1
39 Lfp 1
40 Lfp 1
41 Lfp 1
42 Lfp 1
43 Ind -3
44 Ind -4
45 Ind -5
46 Ind -6
47 Ind -7
48 Ind -8
49 Ind -9
50 Ind -10
51 Rta
52 Sfp -76
53 Drp
54 Sfp -77
55 Drp
56 Sfp -78
57 Drp
58 Sfp -79
59 Drp
60 Sfp -80
61 Drp
62 Sfp -81
63 Drp
64 Sfp -82
65 Drp
66 Sfp -83
67 Rts 75
68 Ent 0
69 Lfp -2
70 Rta
71 Sfp -82
72 Rts 81
73 Ent 0
74 Lfp -11
75 Lfp -10
76 Sub
77 Lit 8
78 Mul
79 Lit 12
80 Add
```

```
81 Lfp 1
82 Lfp 1
83 Lfp 1
84 Lfp 1
85 Lfp 1
86 Lfp 1
87 Lfp 1
88 Lfp -9
89 Lfp -8
90 Lfp -7
91 Lfp -6
92 Lfp -5
93 Lfp -4
94 Lfp -3
95 Lfp -2
96 Ins -10
97 Drp
98 Ins -9
99 Drp
100 Ins -8
101 Drp
102 Ins -7
103 Drp
104 Ins -6
105 Drp
106 Ins -5
107 Drp
108 Ins -4
109 Drp
110 Ins -3
111 Rta
112 Rts 10
113 Ent 2
114 Lit 1
115 Lit 2
116 Sfp 2
117 Drp
118 Sfp 1
119 Drp
120 Lfp 1
121 Lfp 2
122 Jsr 3
123 Jsr -1
124 Drp
125 Lfp 1
126 Lfp 2
127 Jsr 7
128 Jsr -1
129 Drp
130 Rta
131 Lit 0
132 Rts 0
```

Even though the built in functions were mostly unused, they are still in the bytecode. This explains why it is lengthy.

7.2 Test Suites

For each feature described in in the LRM, we provided a test. As we utilized a quick iterative development strategy, every new feature added was only accepted after passing all regression tests. The comprehensive listing of our tests

is listed below, and can also be found in the test directory of the source file. The reference files are named the same, but with the "out" extension rather than the "cv" extension. Overall, there were about 900 lines of test code. Using the Makefile, these tests are run automatically after entering 'make test'. The outputs are saved and then compared against golden references. If they match exactly, the temporary output files are deleted. If they fail, they are kept so that the errors can be looked at.

```
test-arith1.cv     test-lsize.cv
test-arith2.cv     test-multiexp.cv
test-dotop.cv      test-ops1.cv
test-fib.cv        test-plusplus.cv
test-for1.cv       test-returnCurve.cv
test-func1.cv      test-rta1.cv
test-func2.cv      test-rta2.cv
test-func3.cv      test-rta4.cv
test-gcd.cv        test-rta.cv
test-getCurve.cv   test-set2.cv
test-get.cv        test-setCurve2.cv
test-global1.cv    test-setCurve.cv
test-hello.cv      test-set.cv
test-if1.cv        test-setx.cv
test-if2.cv        test-setxy.cv
test-if3.cv        test-solocall.cv
test-if4.cv        test-transform.cv
test-jc2.cv        test-translate.cv
test-jc3.cv        test-var1.cv
test-jc.cv         test-while1.cv
test-layer1.cv
```

Testing the compiler requires looking at the stack as a program is being run. In the execute.ml file, there is a flag that can be set to allow for debugging output. The output consists of the instruction, program counters and the stack. Here is an example of the debugging output:

```
Ogr fp: 0 sp: 0 pc: 0      | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Jsr 12276 fp: 0 sp: 0 pc: 1      | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0 0
Ent 2 fp: 0 sp: 1 pc: 12276      2 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0 0
Lit 1 fp: 1 sp: 4 pc: 12277      2 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0 0
Lit 2 fp: 1 sp: 5 pc: 12278      2 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0 0
Sfp 2 fp: 1 sp: 6 pc: 12279      2 0 0 0 1 2 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0 0
Drp fp: 1 sp: 6 pc: 12280      2 0 0 2 1 2 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0
Sfp 1 fp: 1 sp: 5 pc: 12281      2 0 0 2 1 | 2 0 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0 0
Drp fp: 1 sp: 5 pc: 12282      2 0 1 2 1 | 2 0 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0
Lfp 1 fp: 1 sp: 4 pc: 12283      2 0 1 2 | 1 2 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0 0
Lfp 2 fp: 1 sp: 5 pc: 12284      2 0 1 2 1 | 2 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0 0
Jsr 3 fp: 1 sp: 6 pc: 12285      2 0 1 2 1 2 | 0 0 0 0 0 0 0 0 0 0 0 0 ↩
    0 0
```

```
13 Ent 0 fp: 1 sp: 7 pc: 3      2 0 1 2 1 2 12286 | 0 0 0 0 0 0 0 0 0 0 0 ↩
       0 0
14 Lfp -3 fp: 7 sp: 8 pc: 4      2 0 1 2 1 2 12286 1 | 0 0 0 0 0 0 0 0 0 0↩
        0 0
15 Rta  fp: 7 sp: 9 pc: 5      2 0 1 2 1 2 12286 1 1 | 0 0 0 0 0 0 0 0 0 0↩
        0
16 Rts 2 fp: 7 sp: 9 pc: 6      2 0 1 2 1 2 12286 1 1 | 0 0 0 0 0 0 0 0 0 ↩
       0 0
17 Jsr -1 fp: 1 sp: 5 pc: 12286      2 0 1 2 1 | 2 12286 1 1 0 0 0 0 0 0 0↩
        0 0 0 0
18 1
19 Drp fp: 1 sp: 5 pc: 12287      2 0 1 2 1 | 2 12286 1 1 0 0 0 0 0 0 0 0 ↩
       0 0 0
20 Lfp 1 fp: 1 sp: 4 pc: 12288      2 0 1 2 | 1 2 12286 1 1 0 0 0 0 0 0 0 ↩
       0 0 0 0
21 Lfp 2 fp: 1 sp: 5 pc: 12289      2 0 1 2 1 | 2 12286 1 1 0 0 0 0 0 0 0 ↩
       0 0 0 0
22 Jsr 7 fp: 1 sp: 6 pc: 12290      2 0 1 2 1 2 | 12286 1 1 0 0 0 0 0 0 0 ↩
       0 0 0 0
23 Ent 0 fp: 1 sp: 7 pc: 7      2 0 1 2 1 2 12291 | 1 1 0 0 0 0 0 0 0 0 0 ↩
       0 0
24 Lfp -2 fp: 7 sp: 8 pc: 8      2 0 1 2 1 2 12291 1 | 1 0 0 0 0 0 0 0 0 0↩
        0 0
25 Rta  fp: 7 sp: 9 pc: 9      2 0 1 2 1 2 12291 1 2 | 0 0 0 0 0 0 0 0 0 0↩
        0
26 Sfp -3 fp: 7 sp: 9 pc: 10      2 0 1 2 1 2 12291 1 2 | 0 0 0 0 0 0 0 0 ↩
       0 0 0
27 Rts 2 fp: 7 sp: 9 pc: 11      2 0 1 2 2 2 12291 1 2 | 0 0 0 0 0 0 0 0 0↩
        0 0
28 Jsr -1 fp: 1 sp: 5 pc: 12291      2 0 1 2 2 | 2 12291 1 2 0 0 0 0 0 0 0↩
        0 0 0 0
29 2
30 Drp fp: 1 sp: 5 pc: 12292      2 0 1 2 2 | 2 12291 1 2 0 0 0 0 0 0 0 0 ↩
       0 0 0
31 Rta  fp: 1 sp: 4 pc: 12293      2 0 1 2 | 2 2 12291 1 2 0 0 0 0 0 0 0 0↩
        0 0 0
32 Lit 0 fp: 1 sp: 4 pc: 12294      2 0 1 2 | 2 2 12291 1 2 0 0 0 0 0 0 0 ↩
       0 0 0 0
33 Rts 0 fp: 1 sp: 5 pc: 12295      2 0 1 2 0 | 2 12291 1 2 0 0 0 0 0 0 0 ↩
       0 0 0 0
34 Hlt fp: 0 sp: 1 pc: 2      2 | 0 1 2 0 2 12291 1 2 0 0 0 0 0 0 0 0 0 ↩
       0
```

8 Lessons Learned

As many a wise team from years past have suggested – start early! We did and the whole experience was very painless.

8.1 Kun An

Time management is always important for team projects. Especially for a team of five, sometimes it is difficult to find a time for everyone to meet. Therefore, meetings should always be efficient and effective. It should be efficient in that everyone should have sufficient preparations so that everyone can get involved. The meeting should also be effective in that every team member would have a clear idea as to future tasks. Also, previous errors should be addressed at the meeting. It is great that we met every Monday after class, and discussed for about one to two hours. Every meeting has achieved the expected results.

Though the interpreter is not part of the final deliverables of our language, I found it quite useful to implement an interpreter. Each time a new feature is added to the front-end code, the interpreter was used to test if the scanner, parser, and AST work as expected. The front-end features were added one at a time, which made the bugs easy to track down and fix.

The beauty of a language, whether it is a natural language or programming language, is that one can use a handful basic building blocks to form a great variety of meaningful expressions and statements. With this philosophy in mind, We made curve as the core type of our language because basically any shape can be thought of a curve or composition of curves.

In conclusion, I had great fun working on this project and learned a lot.

8.2 John Chan

Being in a big group like ours poses several challenges. For example, meeting times were tough to set at first, as we had different and conflicting schedules. We did not require the whole group to be present so this made it a lot easier. Also, good minutes were taken during the meetings so that there was a record of what was talked about during the meeting.

Splitting the tasks in a fair and reasonable way was also very important. It would be highly inefficient if we had coded together, because most members would probably be idle during that time. We were successful in defining good boundaries for the work that each member did, and so we were very efficient in completing our tasks. Dependencies were removed almost immediately, as we agreed on an abstract syntax tree from the start. For testing, an interpreter was written by the Frontend group, so that a lot of the bytecode dependencies were removed.

Finally, using GIT proved very helpful. All of the members of our team knew how to use this version control system, so a lot of the disasters where members were working on different versions of code were avoided. We also used Bitbucket's issue tracking system, so that there was a central place where all the issues were kept. Assignments were made so that there was a responsible owner of these issues, and the discussions were had using this tool.

Overall this has been a great experience, and most problems were solved quickly and without a hitch.

8.3 David Mauskop

In terms of group dynamics, I think it was crucial that we had a strong team leader in Wisdom, who enforced regular meetings throughout the semester. An-

other crucial point was the way we handled disagreements over the design of our language. Everyone's opinion was valued, and though we were not afraid to disagree with each other, this always happened in a respectful manner.

In terms of implementation, a willingness to deviate from our initial plan was important. Inevitably, unforeseen challenges arise, and if we had been too rigid in sticking with the plan we set out in our initial proposal and LRM, the whole process would have been more of a struggle. The incremental approach we took was also very successful. Rather than developing several components in parallel and trying to fit them together at the end, we developed the whole system in incremental stages, so that we always had a recent working version. Finally, I think it was valuable to write an interpreter, even though this was not a part of the project specifications. This allowed us to test the front end (scanner, parser, ast, semantic check) before the back end (byte code, compiler) was completed.

8.4 Wisdom Omuya

Things aren't always as bad as they seem; starting on time, properly defining roles, and setting reasonable milestones made the project both enjoyable and educative. I'd advise future teams to clearly delineate roles and responsibilities, use version control with built-in issue tracking, and have everyone be involved – at least in some capacity – in **all** stages of the project.

8.5 Zitong Wang

I think the most important lesson I learnt and the reason why our project works quite successful is that always having something workable. Building a prototype and then adding things one by one turns out to be a very good method to finish a project like this. The prototype we used is microc. Before linking with the compiler and executer, we used the interpreter to test our frontend part: the scanner, the parser and the ast. So we are always confident that what we write are workable. And it motivates us to make the frontend more consistent and powerful.

Team leader is a crucial role in this project. I have to say that Wisdom really did great in that role. Every week we have new challenges to deal with and there is always some progress in our project. Of course, teamwork is the key to the success of our project. And dividing us into the frontend and backend group turns out to be useful and efficient, because small group meetings are more flexible and frequent.

Having a version control system is always a necessity for a project like this. GIT helps us a lot and makes our life much easier.

9 Appendix

This appendix contains the code listing for Curve.

File name	Lines of Code
scanner.mll	62
ast.ml	43
parser.mly	147
bytecode.ml	62
compile.ml	366
execute.ml	101
semantic.ml	385
curve.ml	57
interpret.ml	222
stl.cv	564
Makefile	66

9.1 scanner.mll

```
{
    open Parser
    exception LexError of string
}

let DIGIT = ['0'-'9']+
let ALPHABET      = ['a'-'z' 'A'-'Z']
let ALPHA_NUMERIC   = DIGIT | ALPHABET
let IDENTIFIER = ALPHABET (ALPHA_NUMERIC | '_')*
let WHITESPACE = [' ' '\t' '\r' '\n']


rule token = parse
    WHITESPACE { token lexbuf }
    | "/*"    { block_comment lexbuf }
    | "//"  { line_comment lexbuf }

    | '('       { LPAREN }
    | ')'       { RPAREN }
    | '['       { LBRACK }
    | ']'       { RBRACK }
    | '{'       { LBRACE }
    | '}'       { RBRACE }
    | ';'       { SEMI }
    | ','       { COMMA }
    | '+'       { PLUS }
    | '-'       { MINUS }
    | '*'       { TIMES }
    | '/'       { DIVIDE }
    | '='       { ASSIGN }
    | "=="      { EQ }
    | "!="      { NEQ }
    | '<'       { LT }
    | "<="      { LEQ }
    | ">"       { GT }
    | ">="      { GEQ }
    | "if"      { IF }
    | "else"    { ELSE }
```

```
39      | "for"     { FOR }
40      | "while"   { WHILE }
41      | "while"   { WHILE }
42      | "return"  { RETURN }
43      | "int"     { INT }
44      | "Point"   { POINT }
45      | "Curve"   { CURVE }
46      | "Layer"   { LAYER }
47      | "."       { DOT }
48      | "++"      { PLUSONE }
49      | "--"      { MINUSONE }
50      | DIGIT as lxm { LITERAL(int_of_string lxm) }
51      | IDENTIFIER as lxm { ID(lxm) }
52      | eof { EOF }
53      | _ as char { raise (Failure("illegal character " ^ Char.escaped ↩
            char)) }
54
55  and block_comment = parse
56    | "*/"    { token lexbuf }
57    | eof     { raise (LexError("unterminated block_comment!")) }
58    | _       { block_comment lexbuf }
59
60  and line_comment = parse
61    | ['\n' '\r'] { token lexbuf }
62    | _           { line_comment lexbuf }
```

9.2 ast.ml

```
1   type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater ↩
        | Geq
2
3   type curvet =
4      | Literalt
5      | Curvet
6      | Pointt
7      | Layert
8
9   type expr =
10    | Literal of int
11    | Dotop of string * string * expr list
12    | Curve of expr * expr * expr * expr * expr * expr * expr * expr
13    | Point of expr * expr
14    | Layer of string list
15    | Id of string
16    | Binop of expr * op * expr
17    | Assign of string * expr
18    | Call of string * expr list
19    | Noexpr
20
21  type stmt =
22    | Block of stmt list
23    | Expr of expr
24    | Return of expr
25    | If of expr * stmt * stmt
26    | For of expr * expr * expr * stmt
27    | While of expr * stmt
28
29  type var_decl = {
30     t : curvet;
31     name : string;
32     value : int list;
33  }
34
```

```
35 type func_decl = {
36    return : curvet;
37    fname : string;
38    formals : var_decl list;
39    locals : var_decl list;
40    body : stmt list;
41 }
42
43 type program = var_decl list * func_decl list
```

9.3 parser.mly

```
1  %{ open Ast %}
2
3  %token SEMI LPAREN RPAREN LBRACK RBRACK LBRACE RBRACE COMMA
4  %token PLUS MINUS TIMES DIVIDE ASSIGN
5  %token EQ NEQ LT LEQ GT GEQ
6  %token RETURN IF ELSE FOR WHILE
7  %token INT POINT CURVE DOT LAYER
8  %token PLUSONE MINUSONE
9  %token <int> LITERAL
10 %token <string> ID
11 %token EOF
12
13 %nonassoc NOELSE
14 %nonassoc ELSE
15 %right ASSIGN
16 %left EQ NEQ
17 %left LT GT LEQ GEQ
18 %left PLUS MINUS
19 %left TIMES DIVIDE
20
21 %start program
22 %type <Ast.program> program
23
24 %%
25
26 program:
27    /* nothing */ { [], [] }
28  | program vdecl { ($2 :: fst $1), snd $1 }
29  | program fdecl { fst $1, ($2 :: snd $1) }
30
31 fdecl:
32    | INT ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list ↩
        RBRACE
33      { {
34         return = Literalt;
35         fname = $2;
36         formals = $4;
37         locals = List.rev $7;
38         body = List.rev $8 } }
39    | POINT ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list ↩
        RBRACE
40      { {
41         return = Pointt;
42         fname = $2;
43         formals = $4;
44         locals = List.rev $7;
45         body = List.rev $8 } }
46    | CURVE ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list ↩
        RBRACE
47      { {
48         return = Curvet;
```

```
49          fname = $2;
50          formals = $4;
51          locals = List.rev $7;
52          body = List.rev $8 } }
53    | LAYER ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list ↩
        RBRACE
54      { {
55          return = Layert;
56          fname = $2;
57          formals = $4;
58          locals = List.rev $7;
59          body = List.rev $8 } }
60
61  formals_opt:
62      /* nothing */ { [] }
63    | formal_list   { List.rev $1 }
64
65  formal_list:
66      formal                  { [$1] }
67    | formal_list COMMA formal { $3 :: $1 }
68
69  formal:
70    | INT ID { { t = Literalt; name = $2; value = [0] } }
71    | POINT ID { { t = Pointt; name = $2; value = [0; 0] } }
72    | CURVE ID { { t = Curvet; name = $2; value = [0; 0; 0; 0; 0; 0; 0; ↩
        0] } }
73    | LAYER ID { { t = Layert; name = $2; value = [] } }
74
75
76  vdecl_list:
77      /* nothing */    { [] }
78    | vdecl_list vdecl { $2 :: $1 }
79
80  vdecl:
81    | INT ID SEMI { { t = Literalt; name = $2; value = [0] } }
82    | POINT ID SEMI { { t = Pointt; name = $2; value = [0; 0] } }
83    | CURVE ID SEMI { { t = Curvet; name = $2; value = [0; 0; 0; 0; 0; ↩
        0; 0; 0] } }
84    | LAYER ID SEMI { { t = Layert; name = $2; value = [] } }
85
86  stmt_list:
87      /* nothing */  { [] }
88    | stmt_list stmt { $2 :: $1 }
89
90  stmt:
91      expr SEMI { Expr($1) }
92    | RETURN expr SEMI { Return($2) }
93    | LBRACE stmt_list RBRACE { Block(List.rev $2) }
94    | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([])) }
95    | IF LPAREN expr RPAREN stmt ELSE stmt    { If($3, $5, $7) }
96    | FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt
97       { For($3, $5, $7, $9) }
98    | WHILE LPAREN expr RPAREN stmt { While($3, $5) }
99
100 expr_opt:
101     /* nothing */ { Noexpr }
102   | expr          { $1 }
103
104 expr:
105   | LITERAL          { Literal($1) }
106   | ID DOT ID LPAREN actuals_opt RPAREN
107       { Dotop($1, $3, $5)}
108   | LPAREN expr COMMA expr RPAREN
109     LPAREN expr COMMA expr RPAREN
110     LPAREN expr COMMA expr RPAREN
111     LPAREN expr COMMA expr RPAREN
112       { Curve($2, $4, $7, $9, $12, $14, $17, $19)}
113   | LPAREN expr COMMA expr RPAREN { Point($2, $4)}
```

```
114   | LBRACK cvs_opt RBRACK { Layer($2) }
115   | ID                  { Id($1) }
116   | expr PLUS   expr { Binop($1, Add,   $3) }
117   | expr MINUS  expr { Binop($1, Sub,   $3) }
118   | expr TIMES  expr { Binop($1, Mult,  $3) }
119   | expr DIVIDE expr { Binop($1, Div,   $3) }
120   | MINUS  expr      { Binop(Literal(0), Sub,   $2) }
121   | expr EQ     expr { Binop($1, Equal, $3) }
122   | expr NEQ    expr { Binop($1, Neq,   $3) }
123   | expr LT     expr { Binop($1, Less,  $3) }
124   | expr LEQ    expr { Binop($1, Leq,   $3) }
125   | expr GT     expr { Binop($1, Greater, $3) }
126   | expr GEQ    expr { Binop($1, Geq,   $3) }
127   | ID ASSIGN expr   { Assign($1, $3) }
128   | ID PLUSONE       { Assign($1, Binop(Id($1), Add, Literal(1))) }
129   | ID MINUSONE      { Assign($1, Binop(Id($1), Sub, Literal(1))) }
130   | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
131   | LPAREN expr RPAREN { $2 }
132
133 actuals_opt:
134     /* nothing */ { [] }
135   | actuals_list  { List.rev $1 }
136
137 actuals_list:
138     expr                    { [$1] }
139   | actuals_list COMMA expr { $3 :: $1 }
140
141 cvs_opt:
142     /* nothing */ { [] }
143   | cvs_list  { List.rev $1 }
144
145 cvs_list:
146     ID                    { [$1] }
147   | cvs_list COMMA ID { $3 :: $1 }
```

9.4 bytecode.ml

```
1  type bstmt =
2    | Lit of int     (* Push a literal *)
3    | Drp            (* Discard a value *)
4    | Bin of Ast.op (* Perform arithmetic on top of stack *)
5    | Lod of int     (* Fetch global variable *)
6    | Str of int     (* Store global variable *)
7    | Lfp of int     (* Load frame pointer relative *)
8    | Sfp of int     (* Store frame pointer relative *)
9    | Jsr of int     (* Call function by absolute address *)
10   | Ent of int     (* Push FP, FP -> SP, SP += i *)
11   | Rta            (* sets new pc and fp *)
12   | Rts of int     (* Restore FP, SP, consume formals, push result *)
13   | Beq of int     (* Branch relative if top-of-stack is zero *)
14   | Bne of int     (* Branch relative if top-of-stack is non-zero *)
15   | Bra of int     (* Branch relative *)
16   | Ind of int     (* Loads value relative to the top of the stack *)
17   | Ins of int     (* Stores value relative to the top of the stack *)
18   | Hlt            (* Terminate *)
19
20   (* curve related instructions *)
21   | Ogr            (* Open graph *)
22
23
24 type prog = {
25    num_globals : int;   (* Number of global variables *)
26    text : bstmt array; (* Code for all the functions *)
```

```
27 }
28
29 let string_of_stmt = function
30     | Ogr -> "Ogr"
31     | Lit(i) -> "Lit " ^ string_of_int i
32     | Drp -> "Drp"
33     | Bin(Ast.Add) -> "Add"
34     | Bin(Ast.Sub) -> "Sub"
35     | Bin(Ast.Mult) -> "Mul"
36     | Bin(Ast.Div) -> "Div"
37     | Bin(Ast.Equal) -> "Eql"
38     | Bin(Ast.Neq) -> "Neq"
39     | Bin(Ast.Less) -> "Lt"
40     | Bin(Ast.Leq) -> "Leq"
41     | Bin(Ast.Greater) -> "Gt"
42     | Bin(Ast.Geq) -> "Geq"
43     | Lod(i) -> "Lod " ^ string_of_int i
44     | Str(i) -> "Str " ^ string_of_int i
45     | Lfp(i) -> "Lfp " ^ string_of_int i
46     | Sfp(i) -> "Sfp " ^ string_of_int i
47     | Jsr(i) -> "Jsr " ^ string_of_int i
48     | Ent(i) -> "Ent " ^ string_of_int i
49     | Rts(i) -> "Rts " ^ string_of_int i
50     | Rta    -> "Rta "
51     | Bne(i) -> "Bne " ^ string_of_int i
52     | Beq(i) -> "Beq " ^ string_of_int i
53     | Bra(i) -> "Bra " ^ string_of_int i
54     | Ind(i) -> "Ind " ^ string_of_int i
55     | Ins(i) -> "Ins " ^ string_of_int i
56     | Hlt    -> "Hlt"
57
58 let string_of_prog p =
59   string_of_int p.num_globals ^ " global variables\n" ^
60   let funca = Array.mapi
61       (fun i s -> string_of_int i ^ " " ^ string_of_stmt s) p.text
62   in String.concat "\n" (Array.to_list funca)
```

9.5 compile.ml

```
1  open Ast
2  open Bytecode
3
4  module StringMap = Map.Make(String)
5
6  (* Symbol table: Information about all the names in scope *)
7  type env = {
8     function_index : int StringMap.t; (* Index for each function *)
9     global_index   : int StringMap.t; (* "Address" for global variables↩
       *)
10    local_index    : int StringMap.t; (* FP offset for args, locals *)
11 }
12
13 (* val enum : int -> 'a list -> (int * 'a) list *)
14 let rec enum stride n = function
15     | [] -> []
16     | hd::tl -> (n, hd) :: enum stride (n+stride) tl
17
18 let sizeoftype = function
19     | Literalt -> 1
20     | Pointt -> 2
21     | Curvet -> 8
22     | Layert -> 81
23
```

```
24 let sizereq acc l =
25       List.fold_left (fun ac el -> ac + (sizeoftype el.t))
26          acc l
27
28 let rec multiLfp index i n =
29    if i = n then
30       []
31    else
32       [Lfp (index + i)] @ (multiLfp index (i+1) n)
33
34 let rec multiLod index i n =
35    if i = n then
36       []
37    else
38       [Lod (index + i)] @ (multiLod index (i+1) n)
39
40 let rec multiSfp index i n =
41    if i = n then
42       []
43    else
44       [Drp] @ [Sfp (index + i)] @ (multiSfp index (i+1) n)
45
46 let rec multiStr index i n =
47    if i = n then
48       []
49    else
50       [Drp] @ [Str (index + i)] @ (multiStr index (i+1) n)
51
52 (* enum for variables, dir is direction *)
53 let rec enum_var dir n = function
54    | [] -> []
55    | hd:: tl -> let sz =
56       sizeoftype hd.t in
57       if dir < 0 then
58          (n-sz+1, hd.name) :: enum_var dir (n + dir*sz) tl
59       else
60          (n, hd.name) :: enum_var dir (n + dir*sz) tl
61
62
63 (* val string_map_pairs StringMap 'a -> (int * 'a) list -> StringMap '↩
       a *)
64 let string_map_pairs map pairs =
65    List.fold_left (fun m (i, n) -> StringMap.add n i m) map pairs
66
67 let vdeclmap map vdlist =
68    List.fold_left (fun m v -> StringMap.add v.name v m) map vdlist
69
70 let fdeclmap map fdlist =
71    List.fold_left (fun m f -> StringMap.add f.fname f.return m) map ↩
       fdlist
72
73 (** Translate a program in AST form into a bytecode program.  Throw an
74     exception if something is wrong, e.g., a reference to an unknown
75     variable or function *)
76 let translate (globals, functions) =
77
78    (* Allocate "addresses" for each global variable *)
79    let global_indexes = string_map_pairs StringMap.empty
80       (enum_var 1 0 globals) in
81
82    let funcdmap = fdeclmap StringMap.empty functions in
83    let funcdmap = StringMap.add "print" (Literalt) funcdmap in
84    let funcdmap = StringMap.add "draw" (Layert) funcdmap in
85    let funcdmap = StringMap.add "pause" (Literalt) funcdmap in
86    let funcdmap = StringMap.add "clear" (Literalt) funcdmap in
87    let funcdmap = StringMap.add "random" (Literalt) funcdmap in
88    let funcdmap = StringMap.add "getX" (Literalt) funcdmap in
89    let funcdmap = StringMap.add "getY" (Literalt) funcdmap in
```

```
90    let funcdmap = StringMap.add "setX" (Literalt) funcdmap in
91    let funcdmap = StringMap.add "setY" (Literalt) funcdmap in
92    let funcdmap = StringMap.add "getPoint" (Pointt) funcdmap in
93    let funcdmap = StringMap.add "setPoint" (Literalt) funcdmap in
94    let funcdmap = StringMap.add "getCurve" (Curvet) funcdmap in
95    let funcdmap = StringMap.add "getSize" (Literalt) funcdmap in
96    let funcdmap = StringMap.add "setCurveh" (Curvet) funcdmap in
97    let funcdmap = StringMap.add "setCurve" (Literalt) funcdmap in
98    let globvdmap = vdeclmap StringMap.empty globals in
99
100   (* Assign indexes to function names; built-in "print" is special *)
101   let graphics_functions = StringMap.add "print" (-1) StringMap.empty↩
        in
102   let graphics_functions = StringMap.add "draw" (-2) ↩
        graphics_functions in
103   let graphics_functions = StringMap.add "pause" (-3) ↩
        graphics_functions in
104   let graphics_functions = StringMap.add "clear" (-4) ↩
        graphics_functions in
105   let graphics_functions = StringMap.add "random" (-5) ↩
        graphics_functions in
106   let built_in_functions = StringMap.add "getX" (1) ↩
        graphics_functions in
107   let built_in_functions = StringMap.add "getY" (2) ↩
        built_in_functions in
108   let built_in_functions = StringMap.add "getPoint" (3) ↩
        built_in_functions in
109   let built_in_functions = StringMap.add "setPoint" (4) ↩
        built_in_functions in
110   let built_in_functions = StringMap.add "getCurve" (5) ↩
        built_in_functions in
111   let built_in_functions = StringMap.add "getSize" (6) ↩
        built_in_functions in
112   let built_in_functions = StringMap.add "setCurveh" (7) ↩
        built_in_functions in
113   let function_indexes = string_map_pairs built_in_functions
114      (enum 1 ((StringMap.cardinal built_in_functions) -
115        (StringMap.cardinal graphics_functions) + 1)
116        (List.map (fun f -> f.fname) functions)) in
117
118   (* Translate a function in AST form into a list of bytecode ↩
        statements *)
119   let translate env fdecl =
120   (* Bookkeeping: FP offsets for locals and arguments *)
121   let sz_formals = sizereq 0 fdecl.formals
122   and sz_locals = sizereq 0 fdecl.locals
123   and sz_ret = sizeoftype fdecl.return
124   and local_offsets = enum_var 1 1 fdecl.locals
125   and locvdmap = vdeclmap StringMap.empty (fdecl.locals@fdecl.formals↩
        )
126   and formal_offsets = enum_var (-1) (-2) fdecl.formals in
127   let rec sizeofexpr = function
128      | Literal i -> 1
129      | Dotop (id, fn, e) -> sizeofexpr (Call(fn,[Id(id)]@e))
130      | Curve (a,b,c,d,e,f,g,h) ->  8
131      | Point (a,b) -> 2
132      | Layer lst -> 81
133      | Id s->
134        (try
135              sizeoftype (StringMap.find s locvdmap).t
136        with Not_found -> try
137              sizeoftype (StringMap.find s globvdmap).t
138        with Not_found -> raise (Failure ("Undeclared variable " ^ s)↩
              ))
139      | Binop (e1,op,e2) -> 1
140      | Assign(s,e) -> sizeofexpr(Id(s))
141      | Call(fname, arg) ->
142        (let rett =
```

```
143             (try
144                (StringMap.find fname funcdmap)
145             with Not_found ->
146                raise (Failure ("undefined function " ^ fname)))
147          in sizeoftype rett)
148       | Noexpr -> 1 in
149    let env = { env with local_index = string_map_pairs
150       StringMap.empty (local_offsets @ formal_offsets) } in
151
152    let rec expr = function
153       | Literal i -> [Lit i]
154       | Point (a,b) -> expr a @ expr b
155       | Curve (a,b,c,d,e,f,g,h) ->
156          expr a @
157          expr b @
158          expr c @
159          expr d @
160          expr e @
161          expr f @
162          expr g @
163          expr h
164       | Dotop (id, fn, e) -> expr (Call(fn,[Id(id)]@e))
165       | Layer lst ->
166          (let rec blanks i n =
167             if i = n then
168                []
169             else
170                [Lit (0)] @ (blanks (i+1) n) in
171             blanks 0 (8*(10-(List.length lst))))
172          @ List.concat (List.map (fun el -> expr (Id(el))) lst)
173          @ [Lit (List.length lst)]
174       | Id s ->
175          (try
176             let start = (StringMap.find s env.local_index)
177             and sz = sizeoftype (StringMap.find s locvdmap).t in
178             multiLfp start 0 sz (* @ [Lit sz] *)
179          with Not_found -> try
180             let startg = (StringMap.find s env.global_index)
181             and sz = sizeoftype (StringMap.find s globvdmap).t in
182             multiLod startg 0 sz (* @ [Lit sz] *)
183          with Not_found -> raise (Failure ("Undeclared variable " ^ s)↩
               ))
184       | Binop (e1, op, e2) -> expr e1 @ expr e2 @ [Bin op]
185       | Assign (s, e) ->
186             expr e @
187             (try
188                let start = (StringMap.find s env.local_index) in
189                   let sz = sizeoftype (StringMap.find s locvdmap).t in
190                List.rev (List.tl (multiSfp start 0 sz))
191             with Not_found -> try
192                let startg = (StringMap.find s env.global_index) in
193                   let sz = sizeoftype (StringMap.find s globvdmap).t ↩
                       in
194                List.rev (List.tl (multiStr startg 0 sz))
195             with Not_found -> raise (Failure ("undeclared variable " ^↩
                  s)))
196       | Call (fname, actuals) -> (match fname with
197          | "setX" -> expr (Assign(
198             (match (List.hd actuals) with
199                | Id (s)-> s | _ -> ""),
200             Point(
201                List.hd (List.tl actuals), Call("getY", [(List.hd ↩
                     actuals)])
202                )))
203          | "setY" -> expr (Assign(
204             (match (List.hd actuals) with
205                | Id (s)-> s | _ -> ""),
206             Point(
```

```
207                 Call("getX",[(List.hd actuals)]),List.hd (List.tl ↩
                        actuals)
208                 )))
209           | "getPoint" ->
210             (* a b c d e f g h 8 8 a b *)
211             let indindx =
212                (Binop(Binop(Literal(4),Sub,(List.hd (List.tl actuals))↩
                       ),
213                   Mult,Literal(2))) in
214             (try
215                expr (List.hd actuals) @
216                expr indindx @ expr indindx @
217                [Jsr (StringMap.find fname env.function_index) ]
218                with Not_found ->
219                   raise (Failure ("undefined function " ^ fname))
220             )
221           | "setPoint" ->
222             (* a b c d e f g h 8 8 n1 n2 *)
223             let s = match (List.hd actuals) with
224                | Id (s)-> s | _ -> "" in
225             let indindx =
226                (Binop(Binop(Literal(4),Sub,(List.hd (List.tl actuals))↩
                       ),
227                   Mult,Literal(2))) in
228             (try
229                expr (List.hd actuals) @
230                expr indindx @ expr indindx @
231                expr
232                (List.hd (List.rev actuals))
233                @
234                [Jsr (StringMap.find fname env.function_index) ]
235                with Not_found ->
236                   raise (Failure ("undefined function " ^ fname)))
237                @
238                (try
239                   let start = (StringMap.find s env.local_index) in
240                      let sz = sizeoftype (StringMap.find s locvdmap).t↩
                             in
241                   List.rev (List.tl (multiSfp start 0 sz))
242                with Not_found -> try
243                   let startg = (StringMap.find s env.global_index) in
244                      let sz = sizeoftype (StringMap.find s globvdmap).↩
                             t in
245                   List.rev (List.tl (multiStr startg 0 sz))
246                with Not_found -> raise (Failure ("undeclared variable ↩
                       " ^ s))
247                )
248           | "getCurve" ->
249             (try
250                expr (List.hd actuals) @
251                expr (List.hd (List.tl actuals)) @
252                [Jsr (StringMap.find fname env.function_index) ]
253                with Not_found ->
254                   raise (Failure ("undefined function " ^ fname))
255             )
256           | "getSize" ->
257             (try
258                expr (List.hd actuals) @
259                [Jsr (StringMap.find fname env.function_index) ]
260                with Not_found ->
261                   raise (Failure ("undefined function " ^ fname))
262             )
263           | "setCurveh" ->
264             (try
265                expr (List.hd actuals) @
266                expr (List.hd (List.tl actuals)) @
267                expr (List.hd (List.rev actuals)) @
268                [Jsr (StringMap.find fname env.function_index) ]
```

```
269                 with Not_found ->
270                    raise (Failure ("undefined function " ^ fname))
271             )
272
273           | "setCurve" ->
274              expr (Assign(
275              (match (List.hd actuals) with
276                 | Id (s)-> s | _ -> ""),
277              Call("setCurveh", actuals)))
278           | _ ->
279              (try
280                 (List.concat (List.map expr (List.rev actuals))) @
281                 [Jsr (StringMap.find fname env.function_index) ]
282                 with Not_found ->
283                    raise (Failure ("undefined function " ^ fname))))
284       | Noexpr -> []
285
286    in let rec stmt = function
287       | Block sl     -> List.concat (List.map stmt sl)
288       | Expr e       -> (
289          let rec drpsize a = match a with
290             | Assign(s,e) -> 1
291             | _-> sizeofexpr a in
292          let rec drpe i ex =
293             if i = drpsize ex then
294                []
295             else
296                [Drp] @ drpe (i+1) ex in
297          expr e @ drpe 0 e)
298       | Return e     ->
299          (let rec loadretval i exp sargs =
300             let sz = sizeofexpr exp in
301             if i = sz then
302                []
303             else
304                [Sfp (-sargs + sz -2 -i )] @ [Drp] @
305                loadretval (i+1) exp sargs
306          in
307          expr e @ [Rta] @ (loadretval 0 e sz_formals) @
308          [Rts (sz_formals - (sizeofexpr e) + 1)])
309       | If (p, t, f) -> let t' = stmt t and f' = stmt f in
310          expr p @ [Beq(2 + List.length t')] @
311          t' @ [Bra(1 + List.length f')] @ f'
312       | For (e1, e2, e3, b) ->
313          stmt (Block([Expr(e1); While(e2, Block([b; Expr(e3)]))]))
314       | While (e, b) ->
315          let b' = stmt b and e' = expr e in
316             [Bra (1+ List.length b')] @ b' @ e' @
317             [Bne (-(List.length b' + List.length e'))]
318
319    in
320       [Ent (max sz_locals sz_ret)] @(* Entry: allocate space for ↩
               locals *)
321       stmt (Block fdecl.body) @  (* Body *)
322       [Rta; Lit 0; Rts sz_formals]   (* Default = return 0 *)
323
324    in let env = { function_index = function_indexes;
325       global_index = global_indexes;
326       local_index = StringMap.empty } in
327
328    (* Code executed to start the program: Jsr main; halt *)
329    let entry_function = try
330       [Ogr; Jsr (StringMap.find "main" function_indexes); Hlt]
331       with Not_found -> raise (Failure ("no \"main\" function"))
332    in
333
334    (* Compile the functions *)
335    let func_bodies = entry_function ::
```

```
336         [[Ent 0;Lfp (-3);Rta;Rts 2]] @
337         [[Ent 0;Lfp (-2);Rta;Sfp (-3);Rts 2]] @
338         [[Ent 0;Ind 1;Ind 0;Rta; Sfp (-10);Drp;Sfp (-11);Rts 9]] @
339         [[Ent 0;Lfp(-3);Lfp(-2);Ins 2;Drp;Ins 3;Rta;Rts 5]] @
340         [[Ent 0;Lfp(-3);Lfp(-2);Bin(Sub);Lit 8;Bin(Mult);Lit 4;Bin(Add);
341            Lfp (1);Lfp (1);Lfp (1);Lfp (1);Lfp (1);Lfp (1);Lfp (1);
342            Ind (-3);Ind (-4);Ind (-5);Ind (-6);Ind (-7);Ind (-8);
343            Ind (-9);Ind (-10);Rta;Sfp(-76);Drp;Sfp(-77);Drp;
344            Sfp(-78);Drp;Sfp(-79);Drp;Sfp(-80);Drp;Sfp(-81);Drp;Sfp(-82);
345            Drp;Sfp(-83);Rts (75);]] @
346         [[Ent 0;Lfp(-2);Rta;Sfp(-82);Rts(81)]] @
347         [[Ent 0;Lfp(-11);Lfp(-10);Bin(Sub);Lit 8;Bin(Mult);Lit 12;Bin(↩
              Add);
348            Lfp (1);Lfp (1);Lfp (1);Lfp (1);Lfp (1);Lfp (1);Lfp (1);
349            Lfp (-9);Lfp(-8);Lfp(-7);Lfp(-6);Lfp(-5);Lfp(-4);Lfp(-3);Lfp↩
                  (-2);
350            Ins(-10);Drp;Ins(-9);Drp;Ins(-8);Drp;
351            Ins(-7);Drp;Ins(-6);Drp;Ins(-5);Drp;Ins(-4);Drp;
352            Ins(-3);Rta;Rts (10);]] @
353         (List.map (translate env) functions) in
354
355     (* Calculate function entry points by adding their lengths *)
356     let (fun_offset_list, _) = List.fold_left
357        (fun (l,i) f -> (i :: l, (i + List.length f))) ([],0) ↩
              func_bodies in
358     let func_offset = Array.of_list (List.rev fun_offset_list) in
359
360     { num_globals = sizereq 0 globals;
361           (* Concatenate the compiled functions and replace the ↩
                 function
362            indexes in Jsr statements with PC values *)
363       text = Array.of_list (List.map (function
364           | Jsr i when i > 0 -> Jsr func_offset.(i)
365           | _ as s -> s) (List.concat func_bodies))
366     }
```

9.6 execute.ml

```
1  open Ast
2  open Bytecode
3  open Thread
4  open Random
5
6
7  (* Stack layout just after "Ent":
8
9                <-- SP
10    Local n
11    ...
12    Local 0
13    Saved FP   <-- FP
14    Saved PC
15    Arg 0
16    ...
17    Arg n *)
18
19 let rec printstack i sp = function
20    | [] -> []
21    | hd :: tl ->
22       if i = sp then
23          ["|"] @ [string_of_int hd] @ printstack (i+1) sp tl
24       else
25          [string_of_int hd ] @ printstack (i+1) sp tl
```

```
26
27  let execute_prog prog =
28    let debug = false in
29    let stack = Array.make (1024*1024) 0
30    and globals = Array.make prog.num_globals 0
31    and new_fp = ref 0 and new_pc = ref 0 in
32
33    let sleep n  = Thread.join(Thread.create(Thread.delay)(float_of_int ↩
          n /. 1000.0)) in
34
35    let random n = Random.self_init(); Random.int n in
36
37    let drawcurve m =
38        Graphics.moveto  (stack.(m)) (stack.(m+1));
39        Graphics.curveto ((stack.(m+2)),stack.(m+3)) ((stack.(m+4)),↩
              stack.(m+5)) ((stack.(m+6)),stack.(m+7)) in
40
41    let rec exec fp sp pc =
42        if debug then
43           print_endline( (Bytecode.string_of_stmt prog.text.(pc)) ^
44              " fp: " ^ string_of_int fp ^
45              " sp: " ^ string_of_int sp ^
46              " pc: " ^ string_of_int pc ^ "      " ^
47              String.concat " " (printstack 0 sp
48                 (Array.to_list (Array.sub stack 0 300))))
49        else ();
50    match prog.text.(pc) with
51    | Ogr -> Graphics.open_graph " 1440x900"; exec fp (sp) (pc+1)
52    | Lit i  -> stack.(sp) <- i ; exec fp (sp+1) (pc+1)
53    | Drp    -> exec fp (sp-1) (pc+1)
54    | Bin op -> let op1 = stack.(sp-2) and op2 = stack.(sp-1) in
55      stack.(sp-2) <- (let boolean i = if i then 1 else 0 in
56      match op with
57       | Add     -> op1 + op2
58      | Sub     -> op1 - op2
59      | Mult    -> op1 * op2
60      | Div     -> op1 / op2
61      | Equal   -> boolean (op1 =  op2)
62      | Neq     -> boolean (op1 != op2)
63      | Less    -> boolean (op1 <  op2)
64      | Leq     -> boolean (op1 <= op2)
65      | Greater -> boolean (op1 >  op2)
66      | Geq     -> boolean (op1 >= op2)) ;
67      exec fp (sp-1) (pc+1)
68    | Lod i   -> stack.(sp)   <- globals.(i)  ; exec fp (sp+1) (pc+1)
69    | Str i   -> globals.(i)  <- stack.(sp-1) ; exec fp sp     (pc+1)
70    | Lfp i   -> stack.(sp)   <- stack.(fp+i) ; exec fp (sp+1) (pc+1)
71    | Sfp i   -> stack.(fp+i) <- stack.(sp-1) ; exec fp sp     (pc+1)
72    | Ind i   -> stack.(sp) <- stack.(fp-stack.(fp-i-2)-i-2) ;
73                 exec fp (sp+1) (pc+1)
74    | Ins i   -> stack.(fp-stack.(fp-i-2)-i-2) <- stack.(sp-1) ;
75                 exec fp sp (pc+1)
76    | Jsr(-1) -> print_endline (string_of_int stack.(sp-1));
77                 exec fp sp (pc+1)
78    | Jsr(-2) -> for j=1 to stack.(sp-1) do
79                  ignore(drawcurve (sp - 1 - (j * 8)));
80                 done;
81                 exec fp sp (pc+1)
82    | Jsr(-3) -> sleep stack.(sp-1);
83                 exec fp (sp) (pc+1)
84    | Jsr(-4) -> Graphics.clear_graph ();
85                 exec fp (sp+1) (pc+1)
86    | Jsr(-5) -> stack.(sp-1) <- random stack.(sp-1);
87                 exec fp (sp) (pc+1)
88    | Jsr i   -> stack.(sp)   <- pc + 1        ; exec fp (sp+1) i
89    | Ent i   -> stack.(sp)   <- fp            ; exec sp (sp+i+1) (pc+1)
90    | Rta     -> ignore(new_fp := stack.(fp));
91                 ignore(new_pc := stack.(fp-1));
```

```
92                exec fp (sp) (pc+1)
93    | Rts i    -> (* let new_fp = stack.(fp) and new_pc = stack.(fp-1) in *)
94                (*stack.(fp-i-1) <- stack.(sp-1); *)
95                exec !new_fp (fp-i) !new_pc
96    | Beq i    -> exec fp (sp-1) (pc + if stack.(sp-1) =  0 then i else 1)
97    | Bne i    -> exec fp (sp-1) (pc + if stack.(sp-1) != 0 then i else 1)
98    | Bra i    -> exec fp sp (pc+i)
99    | Hlt      -> Graphics.close_graph ()
100
101   in exec 0 0 0
```

9.7 semantic.ml

```
1   open Ast
2
3   module NameMap = Map.Make(struct
4     type t = string
5     let compare x y = Pervasives.compare x y
6   end)
7
8   exception ReturnException of curvet * var_decl NameMap.t
9
10
11  (* Main entry point: check a program *)
12
13  let rec check_layer cvs env count =
14    let locals, globals = env in
15    match cvs with
16    | [] -> true
17    | hd :: tl ->
18        if count < 11 then
19          if NameMap.mem hd locals && (NameMap.find hd locals).t =
                Curvet then
20            check_layer tl env (count+1)
21          else if NameMap.mem hd globals && (NameMap.find hd globals).t
                = Curvet then
22            check_layer tl env (count+1)
23          else false
24        else
25          raise (Failure ("Too many curves in a layer!"))
26
27  let check_semantic (vars, funcs) =
28    (* Put function declarations in a symbol table *)
29    let func_decls = List.fold_left
30        (fun funcs fdecl -> NameMap.add fdecl.fname fdecl funcs)
31        NameMap.empty funcs
32    in
33
34    (* Invoke a function and return an updated global symbol table *)
35    let rec call fdecl actuals globals checked =
36
37      (* Evaluate an expression and return (value, updated environment)
            *)
38      let rec eval env = function
39          Literal(i) -> Literalt, env;
40        | Dotop(var, op, e) ->
41      let (locals, globals) = env in
42      if op = "getX" || op = "getY" then
43        if NameMap.mem var locals then
44          if (NameMap.find var locals).t = Pointt then
```

```
45            if List.length e = 0 then
46              Literalt, env
47            else
48              raise (Failure ("Invalid parameter for "^op))
49          else
50            raise (Failure (var^" is not a Point"))
51        else if NameMap.mem var globals then
52          if (NameMap.find var globals).t = Pointt then
53            if List.length e = 0 then
54              Literalt, env
55            else
56              raise (Failure ("Invalid parameter for "^op))
57          else
58            raise (Failure (var^" is not a Point"))
59        else
60          raise (Failure ("Undeclared variable "^var))
61      else if op = "setX" || op = "setY" then
62        if NameMap.mem var locals then
63          if (NameMap.find var locals).t = Pointt then
64            if List.length e = 1 then
65              let l, (locals, globals) = eval env (List.hd e) in
66              if l = Literalt then
67                Literalt, env
68              else
69                raise (Failure ("The parameter should be integer for "^↩
                      op))
70            else
71              raise (Failure ("Invalid parameter for "^op))
72          else
73            raise (Failure (var^" is not a Point"))
74        else if NameMap.mem var globals then
75          if (NameMap.find var globals).t = Pointt then
76            if List.length e = 1 then
77              let l, (locals, globals) = eval env (List.hd e) in
78              if l = Literalt then
79                Literalt, env
80              else
81                raise (Failure ("The parameter should be integer for "^↩
                      op))
82            else
83              raise (Failure ("Invalid parameter for "^op))
84          else
85            raise (Failure (var^" is not a Point"))
86        else
87          raise (Failure ("Undeclared variable "^var))
88      else if op = "getPoint" then
89        if NameMap.mem var locals then
90          if (NameMap.find var locals).t = Curvet then
91            if List.length e = 1 then
92              let l, (locals, globals) = eval env (List.hd e) in
93              if l = Literalt then
94                Pointt, env
95              else
96                raise (Failure ("Parameter should be 0-3 for "^op))
97            else
98              raise (Failure ("Invalid parameter for "^op))
99          else
100           raise (Failure (var^" is not a Curve"))
101       else if NameMap.mem var globals then
102         if (NameMap.find var globals).t = Curvet then
103           if List.length e = 1 then
104             let l, (locals, globals) = eval env (List.hd e) in
105             if l = Literalt then
106               Pointt, env
107             else
108               raise (Failure ("Parameter should be 0-3 for "^op))
109           else
110             raise (Failure ("Invalid parameter for "^op))
```

```
111           else
112             raise (Failure (var^" is not a Curve"))
113         else
114           raise (Failure ("Undeclared variable "^var))
115       else if op = "setPoint" then
116         if NameMap.mem var locals then
117           if (NameMap.find var locals).t = Curvet then
118             if List.length e = 2 then
119               let l1, (locals, globals) = eval env (List.hd e) in
120               let l2, (locals, globals) = eval env (List.nth e 1) in
121               if l1 = Literalt then
122                 if l2 = Pointt then
123                   Literalt, env
124                 else
125                   raise (Failure ("The 2nd parameter should be a Point ↩
                          for "^op))
126               else
127                 raise (Failure ("The 1st parameter should be 0-3 for "^↩
                        op))
128             else
129               raise (Failure ("Invalid parameter for "^op))
130           else
131             raise (Failure (var^" is not a Curve"))
132         else if NameMap.mem var globals then
133           if (NameMap.find var globals).t = Curvet then
134             if List.length e = 2 then
135               let l1, (locals, globals) = eval env (List.hd e) in
136               let l2, (locals, globals) = eval env (List.nth e 1) in
137               if l1 = Literalt then
138                 if l2 = Pointt then
139                   Literalt, env
140                 else
141                   raise (Failure ("The 2nd parameter should be a Point ↩
                          for "^op))
142               else
143                 raise (Failure ("The 1st parameter should be 0-3 for "^↩
                        op))
144             else
145               raise (Failure ("Invalid parameter for "^op))
146           else
147             raise (Failure (var^" is not a Curve"))
148         else
149           raise (Failure ("Undeclared variable "^var))
150       else if op = "getCurve" then
151         if NameMap.mem var locals then
152           if (NameMap.find var locals).t = Layert then
153             if List.length e = 1 then
154               let l, (locals, globals) = eval env (List.hd e) in
155               if l = Literalt then
156                 Curvet, env
157               else
158                 raise (Failure ("Parameter should be integer for "^op))
159             else
160               raise (Failure ("Invalid parameter for "^op))
161           else
162             raise (Failure (var^" is not a Layer"))
163         else if NameMap.mem var globals then
164           if (NameMap.find var globals).t = Layert then
165             if List.length e = 1 then
166               let l, (locals, globals) = eval env (List.hd e) in
167               if l = Literalt then
168                 Curvet, env
169               else
170                 raise (Failure ("Parameter should be integer for "^op))
171             else
172               raise (Failure ("Invalid parameter for "^op))
173           else
174             raise (Failure (var^" is not a Layer"))
```

```
175       else
176         raise (Failure ("Undeclared variable "^var))
177     else if op = "setCurve" then
178       if NameMap.mem var locals then
179         if (NameMap.find var locals).t = Layert then
180           if List.length e = 2 then
181             let l1, (locals, globals) = eval env (List.hd e) in
182             let l2, (locals, globals) = eval env (List.nth e 1) in
183             if l1 = Literalt then
184               if l2 = Curvet then
185                 Curvet, env
186               else
187                 raise (Failure ("The 2nd parameter should be a Curve ↩
                        for "^op))
188             else
189               raise (Failure ("The 1st parameter should be an integer ↩
                      for "^op))
190           else
191             raise (Failure ("Invalid parameter for "^op))
192         else
193           raise (Failure (var^" is not a Layer"))
194       else if NameMap.mem var globals then
195         if (NameMap.find var globals).t = Layert then
196           if List.length e = 2 then
197             let l1, (locals, globals) = eval env (List.hd e) in
198             let l2, (locals, globals) = eval env (List.nth e 1) in
199             if l1 = Literalt then
200               if l2 = Curvet then
201                 Curvet, env
202               else
203                 raise (Failure ("The 2nd parameter should be a Curve ↩
                        for "^op))
204             else
205               raise (Failure ("The 1st parameter should be an integer ↩
                      for "^op))
206           else
207             raise (Failure ("Invalid parameter for "^op))
208         else
209           raise (Failure (var^" is not a Layer"))
210       else
211         raise (Failure ("Undeclared variable "^var))
212     else if op = "getSize" then
213       if NameMap.mem var locals then
214         if (NameMap.find var locals).t = Layert then
215           if List.length e = 0 then
216             Literalt, env
217           else
218             raise (Failure ("Invalid parameter for "^op))
219         else
220           raise (Failure (var^" is not a Layer"))
221       else if NameMap.mem var globals then
222         if (NameMap.find var globals).t = Layert then
223           if List.length e = 0 then
224             Literalt, env
225           else
226             raise (Failure ("Invalid parameter for "^op))
227         else
228           raise (Failure (var^" is not a Layer"))
229       else
230         raise (Failure ("Undeclared variable "^var))
231     else raise (Failure ("Undeclared dot operation " ^ op))
232     | Curve(e1, e2, e3, e4, e5, e6, e7, e8) ->
233     let l1, env = eval env e1 in
234       let l2, env = eval env e2 in
235         let l3, env = eval env e3 in
236           let l4, env = eval env e4 in
237             let l5, env = eval env e5 in
238               let l6, env = eval env e6 in
```

```
239                 let l7, env = eval env e7 in
240                   let l8, env = eval env e8 in
241                     if List.map (fun(x) -> if x = Literalt then 1 else↩
                            0) [l1;l2;l3;l4;l5;l6;l7;l8] = ↩
                           [1;1;1;1;1;1;1;1] then
242                       Curvet, env
243                     else raise (Failure ("Invalid curve definition."))
244       | Point(e1, e2) ->
245     let l1, env = eval env e1 in
246       let l2, env = eval env e2 in
247         if l1 = Literalt && l2 = Literalt then
248           Pointt, env
249         else raise (Failure ("Invalid point definition"))
250       | Layer(cvs) ->
251     if check_layer cvs env 0 then
252       Layert, env
253     else raise (Failure ("Invalid layer definition"))
254       | Noexpr -> Literalt, env (* must be non-zero for the for loop ↩
            predicate *)
255       | Id(var) ->
256     let locals, globals = env in
257     if NameMap.mem var locals then
258       (NameMap.find var locals).t, env
259     else if NameMap.mem var globals then
260       (NameMap.find var globals).t, env
261     else raise (Failure ("Undeclared identifier " ^ var))
262       | Binop(e1, op, e2) ->
263     let l1, env = eval env e1 in
264       let l2, env = eval env e2 in
265         if l1 = Literalt && l1 = l2 then
266           Literalt, env
267         else raise (Failure ("Parameter should be integer for binary ↩
              operation"))
268       | Assign(var, e) ->
269     let l, (locals, globals) = eval env e in
270     if NameMap.mem var locals then
271       if (NameMap.find var locals).t = l then
272         l, (locals, globals)
273       else raise (Failure ("Type mismatch for "^var))
274     else if NameMap.mem var globals then
275       if (NameMap.find var globals).t = l then
276         l, (locals, globals)
277       else raise (Failure ("Type mismatch for "^var))
278     else raise (Failure ("Undeclared identifier " ^ var))
279       | Call("draw", e) ->
280     if List.length e = 1 then
281     let l, env = eval env (List.hd e) in
282       if l = Layert then
283         (Literalt, env)
284       else raise (Failure ("The parameter of draw() function should be↩
             Layer"))
285     else raise (Failure ("Invalid parameter for draw()"))
286       | Call("print", e) ->
287     if List.length e = 1 then
288     let l, env = eval env (List.hd e) in
289       if l = Literalt then
290         (Literalt, env)
291       else raise (Failure ("The parameter of print() function should ↩
            be integer"))
292     else raise (Failure ("Invalid parameter for print()"))
293       | Call("pause", e) ->
294     if List.length e = 1 then
295     let l, env = eval env (List.hd e) in
296       if l = Literalt then
297         (Literalt, env)
298       else raise (Failure ("The parameter of pause() function should ↩
            be integer"))
299     else raise (Failure ("Invalid parameter for pause()"))
```

```
300       | Call("random", e) ->
301     if List.length e = 1 then
302     let l, env = eval env (List.hd e) in
303       if l = Literalt then
304         (Literalt, env)
305       else raise (Failure ("The parameter of random() function should ↩
              be integer"))
306     else raise (Failure ("Invalid parameter for random()"))
307       | Call("clear", e) ->
308     if List.length e = 0 then
309         (Literalt, env)
310     else raise (Failure ("Invalid parameter for clear()"))
311       | Call(f, actuals) ->
312     let fdecl =
313       try NameMap.find f func_decls;
314       with Not_found -> raise (Failure ("Undefined function " ^ f))
315     in
316     let actuals, env = List.fold_left
317         (fun (actuals, env) actual ->
318     let v, env = eval env actual in v :: actuals, env)
319           ([], env) (List.rev actuals)
320     in
321     let (locals, globals) = env in
322     try
323       if NameMap.mem f checked then
324         (NameMap.find f func_decls).return, (locals, globals)
325       else
326         let globals = call fdecl actuals globals (NameMap.add f 1 ↩
              checked)
327         in fdecl.return, (locals, globals)
328     with ReturnException(v, globals) ->
329       if v = fdecl.return then
330         v, (locals, globals)
331       else
332         raise (Failure ("Return type mismatch for function "^f))
333     in
334
335     (* Execute a statement and return an updated environment *)
336     let rec exec env = function
337   Block(stmts) -> List.fold_left exec env stmts
338       | Expr(e) -> let _, env = eval env e in env
339       | If(e, s1, s2) ->
340     let l, env = eval env e in
341       if l = Literalt then
342         List.fold_left exec env [s1;s2]
343       else raise (Failure ("Invalid conditional statement."))
344       | While(e, s) ->
345     let l, env = eval env e in
346       if l = Literalt then
347         exec env s
348       else raise (Failure ("Invalid conditional statement."))
349       | For(e1, e2, e3, s) ->
350     let _, env = eval env e1 in
351       let l, env = eval env e2 in
352         if l = Literalt then
353           let _, env = eval (exec env s) e3 in
354           exec env s
355         else raise (Failure ("Invalid conditional statement."))
356       | Return(e) ->
357     let v, (locals, globals) = eval env e in
358     raise (ReturnException(v, globals))
359     in
360
361     (* Enter the function: bind actual values to formal arguments *)
362     let locals =
363       try List.fold_left2
364     (fun locals formal actual -> if formal.t = actual then
```

```
365                                       NameMap.add formal.name { t = ↩
                                              formal.t; name = formal.name; ↩
                                              value = [0] } locals
366                                     else raise (Failure ("Type mismatch ↩
                                            for parameter " ^ formal.name ^ "↩
                                             in function " ^ fdecl.fname)))
367       NameMap.empty fdecl.formals actuals
368         with Invalid_argument(_) ->
369     raise (Failure ("wrong number of arguments passed to " ^ fdecl.fname↩
          ))
370       in
371       let locals = List.fold_left
372     (fun locals local -> NameMap.add local.name { t = local.t; name = ↩
          local.name; value = local.value } locals) locals fdecl.locals
373       in
374       (* Execute each statement in sequence, return updated global ↩
            symbol table *)
375       let env = List.fold_left exec (locals, globals) fdecl.body
376       in
377       if fdecl.fname = "main" then snd env
378       else raise (Failure ("No return statement for function "^fdecl.↩
            fname))
379
380     (* Run a program: initialize global variables to 0, find and run "↩
          main" *)
381     in let globals = List.fold_left
382         (fun globals vdecl -> NameMap.add vdecl.name { t = vdecl.t; name↩
              = vdecl.name; value = vdecl.value } globals) NameMap.empty ↩
             vars
383     in try
384       ignore (call (NameMap.find "main" func_decls) [] globals NameMap.↩
            empty)
385     with Not_found -> raise (Failure ("did not find the main() function"↩
          ))
```

9.8 curve.ml

```
1  open Parsing
2  open Lexing
3
4  type action = Bytecode | Compile | Semantic | Bypass | NoStl
5
6  let read_file filename =
7    let lines = ref [] in
8    let chan = open_in filename in
9    try
10     while true; do
11       lines := input_line chan :: !lines
12     done; []
13   with End_of_file ->
14     close_in chan;
15     List.rev !lines;;
16
17 let check prog = Semantic.check_semantic prog in
18    let arglength = Array.length Sys.argv in
19    let action = if arglength > 2 then
20    List.assoc Sys.argv.(1) [
21                   ("-b", Bytecode);
22                   ("-c", Compile);
23                   ("-y", Bypass);
24                   ("-n", NoStl);
25                ("-s", Semantic)]
26   else Compile in
```

```
27      let lexbuf =
28      match action with
29          | NoStl ->
30           Lexing.from_string
31           (String.concat "\n"
32              (read_file Sys.argv.(arglength-1) ))
33          | _ ->
34           Lexing.from_string
35           (String.concat "\n"
36              (read_file Sys.argv.(arglength-1) @ read_file "stdlib/stl.↩
                    cv")) in
37      let program = try Parser.program Scanner.token lexbuf
38       with Parse_error -> raise (Failure("Syntax error in program at "
39         ^ string_of_int((Parsing.symbol_start_pos()).pos_cnum))) in
40      match action with
41       | NoStl ->
42          check program;
43          let listing =
44             Bytecode.string_of_prog (Compile.translate program)
45          in print_endline listing
46       | Bypass ->
47          Execute.execute_prog (Compile.translate program)
48       | Bytecode ->
49          check program;
50          let listing =
51             Bytecode.string_of_prog (Compile.translate program)
52          in print_endline listing
53       | Compile ->
54          check program;
55          Execute.execute_prog (Compile.translate program)
56       | Semantic ->
57          check program
```

9.9 interpret.ml

```
1  open Ast
2
3  module NameMap = Map.Make(struct
4    type t = string
5    let compare x y = Pervasives.compare x y
6  end)
7
8  exception ReturnException of int list * int NameMap.t
9
10 let rec transform mat00 mat01 mat10 mat11 = function
11   | [] -> []
12   | [x] -> [x]
13   | x :: y :: t ->
14         (x * mat00 + y * mat01)
15         :: (x * mat10 + y * mat11)
16         :: transform mat00 mat01 mat10 mat11 t
17
18 let explode s =
19   let rec exp i l =
20     if i < 0 then l else exp (i - 1) (s.[i] :: l) in
21   exp (String.length s - 1) []
22
23 let implode l =
24   let res = String.create (List.length l) in
25   let rec imp i = function
26   | [] -> res
27   | c :: l -> res.[i] <- c; imp (i + 1) l in
28   imp 0 l
```

```
29
30 let string_to_int_list s = List.map Char.code (explode s)@[-1]
31
32 let int_list_to_string l =
33   let ascii_to_char = function
34   | -1 -> ','
35   | x -> Char.chr x
36   in let rec char_to_string res temp = function
37   | [] -> res
38   | ',' :: tl -> char_to_string ((implode (List.rev temp))::res) [] tl
39   | x :: tl -> char_to_string res (x::temp) tl
40   in List.rev (char_to_string [] [] (List.map ascii_to_char l))
41
42 (* Main entry point: run a program *)
43
44 let run (vars, funcs) =
45   (* Put function declarations in a symbol table *)
46   let func_decls = List.fold_left
47       (fun funcs fdecl -> NameMap.add fdecl.fname fdecl funcs)
48       NameMap.empty funcs
49   in
50
51   (* Invoke a function and return an updated global symbol table *)
52   let rec call fdecl actuals globals =
53
54     (* Evaluate an expression and return (value, updated environment) ↩
         *)
55     let rec eval env = function
56         Literal(i) -> [i], env
57       | Dotop(var, op, [e]) ->
58     let l, (locals, globals) = eval env e in
59       let v = List.hd l in
60         if op = "nth" then
61           [List.nth (NameMap.find var locals).value v], env
62         else if op = "getX" then
63           let index = v * 2 in
64             [List.nth (NameMap.find var locals).value index], env
65         else if op = "getY" then
66           let index = v * 2 + 1 in
67           [List.nth (NameMap.find var locals).value index], env
68         else raise (Failure ("undeclared dot operation " ^ op))
69       | Curve(e1, e2, e3, e4, e5, e6, e7, e8) ->
70     let l1, env = eval env e1 in
71       let l2, env = eval env e2 in
72         let l3, env = eval env e3 in
73           let l4, env = eval env e4 in
74             let l5, env = eval env e5 in
75               let l6, env = eval env e6 in
76                 let l7, env = eval env e7 in
77                   let l8, env = eval env e8 in
78     let v1 = List.hd l1 in
79       let v2 = List.hd l2 in
80         let v3 = List.hd l3 in
81           let v4 = List.hd l4 in
82             let v5 = List.hd l5 in
83               let v6 = List.hd l6 in
84                 let v7 = List.hd l7 in
85                   let v8 = List.hd l8 in
86     [v1; v2; v3; v4; v5; v6; v7; v8], env
87
88       | Point(e1, e2) ->
89     let l1, env = eval env e1 in
90       let l2, env = eval env e2 in
91         let v1 = List.hd l1 in
92           let v2 = List.hd l2 in
93           [v1; v2], env
94       | Layer(cvs) ->
```

```
95      List.fold_left (fun result s -> result@string_to_int_list s) [] ↩
            cvs, env
96        | AddToLayer(cv, ly) ->
97      let locals, globals = env in
98        (NameMap.find ly locals).value@string_to_int_list cv, env
99        | Noexpr -> [1], env (* must be non-zero for the for loop ↩
              predicate *)
100       | Id(var) ->
101     let locals, globals = env in
102     if NameMap.mem var locals then
103       (NameMap.find var locals).value, env
104     else if NameMap.mem var globals then
105       [(NameMap.find var globals)], env
106     else raise (Failure ("undeclared identifier " ^ var))
107       | Binop(e1, op, e2) ->
108     let l1, env = eval env e1 in
109       let l2, env = eval env e2 in
110         let v1 = List.hd l1 in
111           let v2 = List.hd l2 in
112     let boolean i = if i then 1 else 0 in
113     (match op with
114       Add -> [v1 + v2]
115     | Sub -> [v1 - v2]
116     | Mult -> [v1 * v2]
117     | Div -> [v1 / v2]
118     | Equal -> [boolean (v1 = v2)]
119     | Neq -> [boolean (v1 != v2)]
120     | Less -> [boolean (v1 < v2)]
121     | Leq -> [boolean (v1 <= v2)]
122     | Greater -> [boolean (v1 > v2)]
123     | Geq -> [boolean (v1 >= v2)]), env
124     | Assign(var, e) ->
125     let l, (locals, globals) = eval env e in
126     if NameMap.mem var locals then
127       l, (NameMap.add var { t = (NameMap.find var locals).t; name = ↩
              var; value = l } locals, globals)
128     else if NameMap.mem var globals then
129       l, (locals, NameMap.add var (List.hd l) globals)
130     else raise (Failure ("undeclared identifier " ^ var))
131       | Call("print", [e]) ->
132     let l, env = eval env e in
133       let v = List.hd l in
134     print_endline (string_of_int v);
135     [0], env
136       | Call("draw", [e]) ->
137     let l, env = eval env e in
138       print_endline "draw starts";
139       ignore (List.map (fun x -> print_endline (string_of_int x)) l);
140       print_endline "draw ends";
141     [0], env
142       | Call("pause", [e]) ->
143     let l, env = eval env e in
144       print_endline "pause starts";
145       ignore (List.map (fun x -> print_endline (string_of_int x)) l);
146       print_endline "pause ends";
147     [0], env
148       | Call("drawLayer", [e]) ->
149     let l, env = eval env e in
150       let cvs = int_list_to_string l in
151         print_endline "draw layer starts";
152         ignore (List.map (fun x -> print_endline x) cvs);
153         print_endline "draw layer ends";
154     [0], env
155       | Call(f, actuals) ->
156     let fdecl =
157       try NameMap.find f func_decls
158       with Not_found -> raise (Failure ("undefined function " ^ f))
159     in
```

```
160     let actuals, env = List.fold_left
161         (fun (actuals, env) actual ->
162       let v, env = eval env actual in v :: actuals, env)
163           ([], env) (List.rev actuals)
164     in
165     let (locals, globals) = env in
166     try
167       let globals = call fdecl actuals globals
168       in [0], (locals, globals)
169     with ReturnException(v, globals) -> v, (locals, globals)
170     in
171
172     (* Execute a statement and return an updated environment *)
173     let rec exec env = function
174   Block(stmts) -> List.fold_left exec env stmts
175       | Expr(e) -> let _, env = eval env e in env
176       | If(e, s1, s2) ->
177     let l, env = eval env e in
178       let v = List.hd l in
179     exec env (if v != 0 then s1 else s2)
180       | While(e, s) ->
181     let rec loop env =
182       let l, env = eval env e in
183         let v = List.hd l in
184       if v != 0 then loop (exec env s) else env
185     in loop env
186       | For(e1, e2, e3, s) ->
187     let _, env = eval env e1 in
188     let rec loop env =
189       let l, env = eval env e2 in
190         let v = List.hd l in
191       if v != 0 then
192         let _, env = eval (exec env s) e3 in
193         loop env
194       else
195         env
196     in loop env
197       | Return(e) ->
198     let v, (locals, globals) = eval env e in
199     raise (ReturnException(v, globals))
200     in
201
202     (* Enter the function: bind actual values to formal arguments *)
203     let locals =
204       try List.fold_left2
205     (fun locals formal actual -> NameMap.add formal.name { t = formal.↩
            t; name = formal.name; value = actual } locals)
206     NameMap.empty fdecl.formals actuals
207       with Invalid_argument(_) ->
208   raise (Failure ("wrong number of arguments passed to " ^ fdecl.fname↩
        ))
209     in
210     (* Initialize local variables to 0 *)
211     let locals = List.fold_left
212   (fun locals local -> NameMap.add local.name { t = local.t; name = ↩
        local.name; value = local.value } locals) locals fdecl.locals
213     in
214     (* Execute each statement in sequence, return updated global ↩
          symbol table *)
215     snd (List.fold_left exec (locals, globals) fdecl.body)
216
217   (* Run a program: initialize global variables to 0, find and run "↩
        main" *)
218   in let globals = List.fold_left
219       (fun globals vdecl -> NameMap.add vdecl 0 globals) NameMap.empty↩
            (List.map (fun x -> x.name) vars)
220   in try
221     call (NameMap.find "main" func_decls) [] globals
```

```
222   with Not_found -> raise (Failure ("did not find the main() function"↩
          ))
```

9.10 stl.cv

```
1   int printp(Point p)
2   {
3      print(p.getX());
4      print(p.getY());
5      return 0;
6   }
7
8   int printc(Curve c)
9   {
10     int i;
11     for(i = 0; i < 4; i = i+1)
12     {
13        printp(c.getPoint(i));
14     }
15     return 0;
16  }
17
18  int printl(Layer l)
19  {
20     int sz;
21     int i;
22     Curve tmp;
23     sz = l.getSize();
24     for (i=0 ; i < sz; i = i+1)
25     {
26        tmp = l.getCurve(i);
27        printc(tmp);
28     }
29     return 0;
30  }
31
32  int true()
33  {
34      return 1;
35  }
36
37  int false()
38  {
39      return 0;
40  }
41
42  int curveSize()
43  {
44      return 4;
45  }
46
47  int maxLayerSize()
48  {
49      return 10;
50  }
51
52  int nullI()
53  {
54      return -1;
55  }
56
57  Point nullP()
58  {
```

```
59      return (-1, -1);
60  }
61
62  Curve nullC()
63  {
64      return (-1, -1)(-1, -1)(-1, -1)(-1, -1);
65  }
66
67  Layer nullL()
68  {
69      Curve c;
70      Layer nullLayer;
71
72      c = nullC();
73      nullLayer = [c,c,c,c,c,c,c,c,c,c];
74
75      return nullLayer;
76  }
77
78  int equalsP(Point p, Point q)
79  {
80      if (p.getX() != q.getX()) {
81          return false();
82      }
83      if (p.getY() != q.getY()) {
84          return false();
85      }
86      return true();
87  }
88
89  int equalsC(Curve c, Curve d)
90  {
91      int i;
92      for (i = 0; i < curveSize(); i++) {
93          if (equalsP(c.getPoint(i), d.getPoint(i)) == false()) {
94              return false();
95          }
96      }
97      return true();
98  }
99
100 int equalsL(Layer l, Layer m)
101 {
102     int i;
103     int lsize;
104     int msize;
105
106     lsize = l.getSize();
107     msize = m.getSize();
108
109     if (lsize != msize) {
110         return false();
111     }
112     for (i = 0; i < lsize; i++) {
113         if (equalsC(l.getCurve(i), m.getCurve(i)) == false()) {
114             return false();
115         }
116     }
117     return true();
118 }
119
120 Curve translateC(Curve c, int x, int y)
121 {
122    int i;
123    Point a;
124    for(i = 0; i < 4; i = i+1)
125    {
126       a = c.getPoint(i);
```

```
127         a.setX(a.getX() + x);
128         a.setY(a.getY() + y);
129         c.setPoint(i,a);
130      }
131      return c;
132   }
133
134   Layer translateL(Layer l, int x, int y)
135   {
136      int sz;
137      int i;
138      Curve tmp;
139      sz = l.getSize();
140      for (i = 0; i< sz; i=i+1)
141      {
142         tmp = l.getCurve(i);
143         tmp = translateC(tmp,x,y);
144         l.setCurve(i,tmp);
145      }
146      return l;
147   }
148
149   Curve transformC(Curve cv, int a, int b, int c, int d)
150   {
151      int i;
152      Point p;
153      Point m;
154      for(i = 0; i < 4; i = i+1)
155      {
156         p = cv.getPoint(i);
157         m.setX(p.getX()*a + p.getY()*c);
158         m.setY(p.getX()*b + p.getY()*d);
159         cv.setPoint(i,m);
160      }
161      return cv;
162   }
163
164   Layer transformL(Layer l, int a, int b, int c, int d)
165   {
166      int sz;
167      int i;
168      Curve tmp;
169      sz = l.getSize();
170      for (i = 0; i< sz; i=i+1)
171      {
172         tmp = l.getCurve(i);
173         tmp = transformC(tmp,a,b,c,d);
174         l.setCurve(i,tmp);
175      }
176      return l;
177   }
178
179   Curve lineXY(int x1, int y1, int x2, int y2)
180   {
181       Curve l;
182       l  = (x1, y1)(x1,y1)(x1,y1)(x2,y2);
183       return l;
184   }
185
186   Curve lineP(Point p, Point q)
187   {
188       Curve l;
189       l = lineXY(p.getX(), p.getY(), q.getX(), q.getY());
190       return l;
191   }
192
193   Layer triangleP(Point p, Point q, Point r)
194   {
```

```
195     Layer tri;
196     Curve s1;
197     Curve s2;
198     Curve s3;
199     s1 = lineP(p, q);
200     s2 = lineP(q, r);
201     s3 = lineP(r, p);
202     tri = [s1, s2, s3];
203     return tri;
204 }
205
206 Layer triangleXY(int x1, int y1, int x2, int y2, int x3, int y3) {
207     Layer tri;
208     Point p;
209     Point q;
210     Point r;
211     p = (x1, y1);
212     q = (x2, y2);
213     r = (x3, y3);
214     tri = triangleP(p, q, r);
215     return tri;
216 }
217
218
219 Layer rectangleXY(int x, int y, int height, int width)
220 {
221     Curve s1;
222     Curve s2;
223     Curve s3;
224     Curve s4;
225     Layer rect;
226
227     s1 = lineXY(x, y, x+width, y);
228     s2 = lineXY(x+width, y, x+width, y+height);
229     s3 = lineXY(x+width, y+height, x, y+height);
230     s4 = lineXY(x, y+height, x, y);
231     rect = [s1, s2, s3, s4];
232
233     return rect;
234 }
235
236 Layer rectangleP(Point p, int height, int width)
237 {
238     Layer r;
239     int x;
240     int y;
241     x = p.getX();
242     y = p.getY();
243     r = rectangleXY(x, y, height, width);
244     return r;
245 }
246
247 int isLine(Curve c)
248 {
249     Point end1;
250     Point middle1;
251     Point middle2;
252     Point end2;
253
254
255     end1 = c.getPoint(0);
256     middle1 = c.getPoint(1);
257     middle2 = c.getPoint(2);
258     end2 = c.getPoint(3);
259
260     if (equalsP(end1, end2)) {
261         return false();
262     }
```

```
263     if (equalsP(end1, middle1) == false()) {
264         if (equalsP(end2, middle1) == false()) {
265             return false();
266         }
267     }
268     if (equalsP(end1, middle2) == false()) {
269         if (equalsP(end2, middle2) == false()) {
270             return false();
271         }
272     }
273     return true();
274 }
275
276 int isRectangle(Layer r)
277 {
278     Point p1;
279     Point p2;
280     Curve c1;
281     Curve c2;
282     Curve c3;
283     Curve c4;
284     int i;
285
286     if (r.getSize() != 4)
287         return 0;
288
289     for (i = 0; i < 4; i++) {
290         if (isLine(r.getCurve(i)) == false()) {
291             return false();
292         }
293     }
294
295     c1 = r.getCurve(0);
296     c2 = r.getCurve(1);
297     c3 = r.getCurve(2);
298     c4 = r.getCurve(3);
299
300     p1 = c1.getPoint(3);
301     p2 = c2.getPoint(0);
302     if (equalsP(p1, p2) == false()) {
303         return false();
304     }
305
306     p1 = c2.getPoint(3);
307     p2 = c3.getPoint(0);
308     if (equalsP(p1, p2) == false()) {
309         return false();
310     }
311
312     p1 = c3.getPoint(3);
313     p2 = c4.getPoint(0);
314     if (equalsP(p1, p2) == false()) {
315         return false();
316     }
317
318     p1 = c4.getPoint(3);
319     p2 = c1.getPoint(0);
320     if (equalsP(p1, p2) == false()) {
321         return false();
322     }
323
324     return true();
325 }
326
327 Point getRectangleBase(Layer r)
328 {
329     Curve c;
330
```

```
331     if (isRectangle(r) == false()) {
332         return nullP();
333     }
334     c = r.getCurve(0);
335     return c.getPoint(0);
336 }
337 
338 int getRectangleWidth(Layer r)
339 {
340     Curve c;
341     Point p1;
342     Point p2;
343 
344     if (isRectangle(r) == false()) {
345         return nullI();
346     }
347     c = r.getCurve(0);
348     p1 = c.getPoint(0);
349     p2 = c.getPoint(3);
350     return p2.getX() - p1.getX();
351 }
352 
353 int getRectangleHeight(Layer r)
354 {
355     Curve c;
356     Point p1;
357     Point p2;
358 
359     if (isRectangle(r) == false()) {
360         return nullI();
361     }
362     c = r.getCurve(1);
363     p1 = c.getPoint(0);
364     p2 = c.getPoint(3);
365     return p2.getY() - p1.getY();
366 }
367 
368 Layer fillRectangleL(Layer r, int lightness)
369 {
370     int i;
371     Curve line;
372     Layer lineLayer;
373     Layer rectangle;
374     Point left;
375     Point right;
376     Point top;
377     Point bottom;
378 
379     Point p;
380     int height;
381     int width;
382 
383     if (isRectangle(r) == false()) {
384         return nullL();
385     }
386 
387     p = getRectangleBase(r);
388     height = getRectangleHeight(r);
389     width = getRectangleWidth(r);
390 
391     for (i = 0; i < height; i = i + lightness) {
392         left = (p.getX(), p.getY() + i);
393         right = (p.getX() + width, p.getY() + i);
394         line = lineP(left, right);
395         lineLayer = [line];
396         draw(lineLayer);
397     }
398 
```

```
399     for (i = 0; i < width; i = i + lightness) {
400         top = (p.getX()+i, p.getY());
401         bottom = (p.getX()+i, p.getY()+height);
402         line = lineP(top, bottom);
403         lineLayer = [line];
404         draw(lineLayer);
405     }
406
407     rectangle = rectangleP(p, height, width);
408     draw(rectangle);
409     return rectangle;
410 }
411
412 Layer fillRectangleP(Point p, int height, int width, int lightness)
413 {
414     int i;
415     Curve line;
416     Layer lineLayer;
417     Layer rectangle;
418     Point left;
419     Point right;
420     Point top;
421     Point bottom;
422
423     for (i = 0; i < height; i = i + lightness) {
424         left = (p.getX(), p.getY() + i);
425         right = (p.getX() + width, p.getY() + i);
426         line = lineP(left, right);
427         lineLayer = [line];
428         draw(lineLayer);
429     }
430
431     for (i = 0; i < width; i = i + lightness) {
432         top = (p.getX()+i, p.getY());
433         bottom = (p.getX()+i, p.getY()+height);
434         line = lineP(top, bottom);
435         lineLayer = [line];
436         draw(lineLayer);
437     }
438
439     rectangle = rectangleP(p, height, width);
440     return rectangle;
441 }
442
443 Layer squareXY(int x, int y, int size)
444 {
445     Layer sq;
446     sq = rectangleXY(x, y, size, size);
447     return sq;
448 }
449
450 Layer squareP(Point p, int size)
451 {
452     Layer sq;
453     int x;
454     int y;
455     sq = squareXY(x, y, size);
456     return sq;
457 }
458
459 Layer quadP(Point p1, Point p2, Point p3, Point p4)
460 {
461     Layer quad;
462     Curve s1;
463     Curve s2;
464     Curve s3;
465     Curve s4;
466
```

```
467     s1 = lineP(p1, p2);
468     s2 = lineP(p2, p3);
469     s3 = lineP(p3, p4);
470     s4 = lineP(p4, p1);
471     quad = [s1, s2, s3, s4];
472 
473     return quad;
474 }
475 
476 Layer circle(int r, Point c) {
477     int x;
478     int y;
479     Curve tr;
480     Curve br;
481     Curve bl;
482     Curve tl;
483     Layer cir;
484     int ctrl;
485 
486     x = c.getX();
487     y = c.getY();
488 
489     ctrl = 552*r/1000;
490     tr = (x, r+y)(ctrl+x, r+y)(r+x, ctrl+y)(r+x, y);
491     br = (x, -r+y)(ctrl+x, -r+y)(r+x, -ctrl+y)(r+x, y);
492     bl = (x, -r+y)(-ctrl+x, -r+y)(-r+x, -ctrl+y)(-r+x, y);
493     tl = (x, r+y)(-ctrl+x, r+y)(-r+x, ctrl+y)(-r+x, y);
494     cir = [tr, br, bl, tl];
495     return cir;
496 }
497 
498 Curve scaleC(Curve cv, int cx, int cy, int n, int d) {
499      int i;
500    Point p;
501    Point m;
502 
503    for(i = 0; i < 4; i = i+1)
504    {
505       p = cv.getPoint(i);
506       m.setX((p.getX()-cx)*n/d + cx);
507       m.setY((p.getY()-cy)*n/d + cy);
508       cv.setPoint(i,m);
509    }
510    return cv;
511 }
512 
513 Layer scaleL(Layer l, int cx, int cy, int n, int d) {
514     int sz;
515     int i;
516     Curve tmp;
517     sz = l.getSize();
518     for (i = 0; i< sz; i=i+1)
519     {
520       tmp = l.getCurve(i);
521       tmp = scaleC(tmp,cx,cy,n,d);
522       l.setCurve(i,tmp);
523     }
524     return l;
525 }
526 
527 Curve rotateC(Curve cv, int cx, int cy, int s, int c, int d) {
528    int i;
529    Point p;
530    Point m;
531 
532    for(i = 0; i < 4; i = i+1)
533    {
534       p = cv.getPoint(i);
```

```
535         m.setX((((p.getX()-cx)*c + (p.getY()-cy)*s) + cx*d)/d);
536         m.setY((((p.getX()-cx)*(-s) + (p.getY()-cy)*c) + cy*d)/d);
537         cv.setPoint(i,m);
538       }
539       return cv;
540   }
541
542   Layer rotateL(Layer l, int cx, int cy, int s, int c, int d) {
543       int sz;
544       int i;
545       Curve tmp;
546       sz = l.getSize();
547       for (i = 0; i< sz; i=i+1)
548       {
549         tmp = l.getCurve(i);
550         tmp = rotateC(tmp,cx,cy,s,c,d);
551         l.setCurve(i,tmp);
552       }
553       return l;
554   }
555
556   int exp(int x, int n) {
557     int i;
558     int acc;
559     acc = 1;
560     for (i = 0; i < n; i++) {
561       acc = acc * x;
562     }
563     return acc;
564   }
```

9.11 Makefile

```
1   OBJS = ast.cmo parser.cmo scanner.cmo compile.cmo semantic.cmo ↩
        bytecode.cmo execute.cmo curve.cmo
2
3   LIBS=$(WITHGRAPHICS) $(WITHUNIX) $(WITHTHREADS)
4
5   CONF=-I +threads
6
7   WITHGRAPHICS =graphics.cma -cclib -lgraphics -cclib -L/usr/X11R6/lib -↩
        cclib -lX11
8
9   WITHUNIX =unix.cma -cclib -lunix
10
11  WITHSTR =str.cma -cclib -lstr
12
13  WITHNUMS =nums.cma -cclib -lnums
14
15  WITHTHREADS =threads.cma -cclib -lthreads
16
17  WITHDBM =dbm.cma -cclib -lmldbm -cclib -lndbm
18
19  curve : $(OBJS)
20      ocamlc $(CONF) -o curve $(LIBS) $(OBJS)
21
22  .PHONY : test
23  test : curve testall.sh
24      ./testall.sh
25
26  .PHONY : testb
27  testb : curve testallb.sh
28      ./testallb.sh
```

```
29
30 scanner.ml : scanner.mll
31     ocamllex scanner.mll
32
33 parser.ml parser.mli : parser.mly
34     ocamlyacc parser.mly
35
36 %.cmo : %.ml
37     ocamlc $(CONF) -c $<
38
39 %.cmi : %.mli
40     ocamlc -c $<
41
42 .PHONY : clean
43 clean :
44     rm -f curve parser.ml parser.mli scanner.ml testall.log \
45     *.cmo *.cmi *.out *.diff
46
47 # Generated by ocamldep *.ml *.mli
48 ast.cmo:
49 ast.cmx:
50 bytecode.cmo: ast.cmo
51 bytecode.cmx: ast.cmx
52 compile.cmo: bytecode.cmo ast.cmo
53 compile.cmx: bytecode.cmx ast.cmx
54 execute.cmo: bytecode.cmo ast.cmo
55 execute.cmx: bytecode.cmx ast.cmx
56 semantic.cmo:
57 semantic.cmx:
58 curve.cmo: scanner.cmo parser.cmi execute.cmo \
59     bytecode.cmo ast.cmo semantic.cmo
60 curve.cmx: scanner.cmx parser.cmx execute.cmx \
61     bytecode.cmx ast.cmx semantic.cmx
62 parser.cmo: ast.cmo parser.cmi
63 parser.cmx: ast.cmx parser.cmi
64 scanner.cmo: parser.cmi
65 scanner.cmx: parser.cmx
66 parser.cmi: ast.cmo
```