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 3

2 Language Tutorial 3
  2.1 Basic Types ........................................... 3
  2.2 Atom Types ........................................... 3
  2.3 Basic syntax .......................................... 3
  2.4 Program structure ................................. 4
  2.5 Examples ............................................. 4
    2.5.1 Koch snowflake - Prof. Edwards 6 hour challenge ... 4
    2.5.2 Draw a Circle ..................................... 6
    2.5.3 Friendly Screensaver ............................. 6
    2.5.4 Bow and Arrow .................................... 7
    2.5.5 Tree Creation and Traversal ..................... 8
    2.5.6 Windmill .......................................... 9
    2.5.7 Clock ............................................ 10
    2.5.8 Pattern Generation ............................. 11

3 Language Reference Manual 12
  3.1 Introduction ......................................... 12
  3.2 Program Definition .................................. 12
  3.3 Lexical Conventions .................................. 12
    3.3.1 Comments ........................................ 12
    3.3.2 Identifiers ....................................... 12
    3.3.3 Keywords ......................................... 13
    3.3.4 Constants ........................................ 13
    3.3.5 Operators ........................................ 13
    3.3.6 Punctuators ...................................... 13
  3.4 Object Types ......................................... 14
    3.4.1 Basic Types ...................................... 14
    3.4.2 Atom Types ...................................... 14
    3.4.3 Object Scope ..................................... 14
  3.5 Expressions and Operations ......................... 15
  3.6 Declarations ......................................... 17
    3.6.1 Function Declaration ............................ 17
    3.6.2 Variable Declaration ............................ 17
  3.7 Statements ............................................ 18
    3.7.1 Expression statement ........................... 18
    3.7.2 Conditional statement ........................... 18
    3.7.3 While statement ................................ 18
    3.7.4 For statement .................................. 18
    3.7.5 Return statement ................................ 18
  3.8 System Functions ..................................... 19
    3.8.1 Print Function .................................. 19
    3.8.2 Draw Function .................................. 19
8 Lessons Learned

8.1 Kun An .......................... 47
8.2 John Chan ........................ 48
8.3 David Mauskop ......................... 48
8.4 Wisdom Omuya ........................ 49
8.5 Zitong Wang ........................ 49

9 Appendix

9.1 scanner.mll ........................ 50
9.2 ast.ml ............................. 51
9.3 parser.mly .......................... 52
9.4 bytecode.ml ........................ 54
9.5 compile.ml .......................... 55
9.6 execute.ml .......................... 61
9.7 semantic.ml ........................ 63
9.8 curve.ml ............................ 69
9.9 interpret.ml ........................ 70
9.10 stl.cv .............................. 74
9.11 Makefile ........................... 82
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 to Bézier 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 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 ints representing Cartesian coordinates.
- **Curve**: A list of four Points 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 Curves, 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 Points or Curves. See section 3.5 for a discussion of operations possible for ints.

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:

```java
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.

```java
Point pointAlongLine(Curve line, int mult, int div)
{
    int x;
    int y;
    Point p1;
    Point p2;
    p1 = line.getPoint(0);
    p2 = line.getPoint(3);
    x = p1.getX() + ((p2.getX() - p1.getX())*mult)/div;
    y = p1.getY() + ((p2.getY() - p1.getY())*mult)/div;
}
```
return (x, y);
}

Point topPoint(Point a, Point b)
{
    int delX;
    int delY;
    int x;
    int y;
    delX = b.getX() - a.getX();
    delY = b.getY() - a.getY();
    x = a.getX() + delX/2 - delY*1732/2000;
    y = a.getY() + delY/2 + delX*1732/2000;
    return (x, y);
}

int snowflake(Point a, Point e, int n)
{
    Point b;
    Point c;
    Point d;
    Curve line;
    line = lineP(a, e);
    if (n == 1) {
        draw([line]);
        return 1;
    }
    b = pointAlongLine(line, 1, 3);
    d = pointAlongLine(line, 2, 3);
    c = topPoint(b, d);
    snowflake(a, b, n-1);
    snowflake(b, c, n-1);
    snowflake(c, d, n-1);
    snowflake(d, e, n-1);
}

int main()
{
    int n;
    Point p0;
    Point q0;
    Point r0;
    p0 = (350, 260);
    q0 = (650, 780);
    r0 = (950, 260);
    for (n = 1; n < 9; n++) {
        snowflake(p0, q0, n);
        snowflake(q0, r0, n);
        snowflake(r0, p0, n);
        pause(3000);
    }
    clear();
}
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:

```java
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.

```java
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 == 500)
      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)));
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 bezier curve. By changing the critical and control points of the curve over time, we can obtain very smooth animation effects. The following example demonstrate the movements of bow, string, and arrow.

```c
int main()
{
    int i;
    int bcx;
    int bcy;
    Curve bow1;
    Curve bow2;
    Curve str;
    Curve arr;
    Curve hed;
    Layer bsah;
    int ela;
    int inc;
    int elac;
    bcx = 200;
    bcy = 400;
    ela = 0;
    inc = 1;
    elac = 0;
    while (1) {
        i++;
        ela = ela + inc;
        if (ela == 25) {
            inc = -1;
            elac = elac + 1;
        }
        if (ela == 0) {
            inc = 1;
            elac = elac + 1;
        }
        if (elac > 1)
            inc = 0;
        bow1 = (bcx - ela * 2, bcy + 100)(bcx + 20, bcy + 20)(bcx + 20, ←
            bcy - 20)(bcx - ela * 2, bcy -100);
        bow2 = (bcx - ela * 2, bcy + 100)(bcx + 15, bcy + 20)(bcx + 15, ←
            bcy - 20)(bcx - ela * 2, bcy -100);
        str = (bcx - ela * 2, bcy + 100)(bcx - ela * 6, bcy)(bcx - ela * ←
            6, bcy)(bcx - ela * 2, bcy -100);
        arr = (bcx - ela * 5, bcy)(bcx - ela, bcy)(bcx - ela, bcy)(bcx - ←
            ela * 5 + 150, bcy);
        hed = (bcx - ela * 5 + 150 - 10, bcy + 5)(bcx - ela * 5 + 150, bcy←
            )
            (bcx - ela * 5 + 150 - 10, bcy + 5)(bcx - ela * 5 + 150, bcy←
            5);
        if (elac > 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.

```c
int exp(int x, int n) {
    int i;
    int acc = 1;
    for (i = 0; i < n; i++) {
        acc = acc * x;
    }
    return acc;
}

int drawTree(int x, int y, int n) {
    Layer 1;
    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);
}
```
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.

```c
int main() {
    int i;
    Curve c1;
    Curve c2;
    Curve c3;
    Curve c4;
    Curve cc1;
    Curve cc2;
    Curve cc3;
    Curve cc4;
    Layer l;
    int cx;
    int cy;
    cx = 700;
    cy = 500;
    c1 = (cx - 500, cy)(cx - 500, cy - 200)(cx - 200, cx - 300)(cx, cy);
    c2 = rotateC(c1, cx, cy, 1, 0, 1);
    c3 = rotateC(c2, cx, cy, 1, 0, 1);
    c4 = rotateC(c3, cx, cy, 1, 0, 1);
    cc1 = (cx - 500, cy)(cx, cy)(cx, cy)(cx, cy);
    cc2 = rotateC(cc1, cx, cy, 1, 0, 1);
    return 1;
}
```
cc3 = rotateC(cc2, cx, cy, 1, 0, 1);  
cp = rotateC(cc3, cx, cy, 1, 0, 1);  
l = [c1, c2, c3, c4, cc1, cc2, cc3, cc4];  

while (1) {
    i + 1;  
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 rotateL8(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];
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.

```cpp
int main()
{
    int i;
    Layer l;
    int x;
    int y;
    clbs = cl;

    while (1) {
        for (c = 0; c < 12; c++) {
            draw(cl);
            cl = rotateL(cl, cx, cy, 50000000000, 86602540378, 100000000000);
        }
        cl = clbs;
        draw(sl);
        draw(cir);
        draw(ml);
        draw(hl);
        i++;
        sl = rotateL6(sl, cx, cy);
        cir = rotateL6(cir, cx, cy);
        if (i == 60) {
            sl = sbs;
            cir = cirbs;
            i = 0;
            j++;
            ml = rotateL6(ml, cx, cy);
            if (j == 12) {
                hl = rotateL6(hl, cx, cy);
            }
            if (j == 24) {
                hl = rotateL6(hl, cx, cy);
            }
            if (j == 36) {
                hl = rotateL6(hl, cx, cy);
            }
            if (j == 48) {
                hl = rotateL6(hl, cx, cy);
            }
            if (j == 60) {
                ml = mbs;
                j = 0;
                k++;
                hl = rotateL6(hl, cx, cy);
                if (k == 12) {
                    hl = hbs;
                    k = 0;
                }
            }
        }
        pause(998);
        clear();
    }
}
```
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`
- `setCurve`
- `getSize`

3.3.4 Constants

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

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

```c
m = 99;
```

3.3.5 Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td>=</td>
<td>Assignment</td>
</tr>
</tbody>
</table>

3.3.6 Punctuators

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>Marks the end of a statement</td>
</tr>
<tr>
<td>[]</td>
<td>Marks the beginning and end of a group of Curves - i.e. a Layer</td>
</tr>
<tr>
<td>{}</td>
<td>Marks the beginning and end of a group of statements</td>
</tr>
<tr>
<td>,</td>
<td>Separates different dimension values of a Point</td>
</tr>
<tr>
<td>.</td>
<td>Used for dot notation access - e.g. accessing Points of a curve</td>
</tr>
<tr>
<td>()</td>
<td>Used for grouping parameters in function call</td>
</tr>
<tr>
<td></td>
<td>Used for specifying Points</td>
</tr>
</tbody>
</table>
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 ints representing Cartesian coordinates
- **Curve**: A list of Points where the first Point is the start point of the Curve, the last is the end point, while the two intermediate Points are controls for the Bézier curve.
- **Layer**: A list of Curves

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 ints.

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:

```c
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:

```c
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:

```c
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:

```cpp
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:

```cpp
int:
   int a;
Point:
```
Point p;
Curve:

Curve c;
Layer:
Layer 1;

Variable declaration is typically followed by variable assignment:

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

3.7 Statements
All statements in Curve end in a semi-colon.

3.7.1 Expression statement
equation;

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

3.7.2 Conditional statement
if (equation) { } else { }

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

3.7.3 While statement
while (equation) { }

The while loop behaves as is typical

3.7.4 For statement
for (equation; equation; equation) { }

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.

\texttt{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:

\texttt{draw(1);}

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 1 – 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:

\texttt{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.

\texttt{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.

\texttt{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.

```java
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.

```java
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.

```java
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.

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

The `getPoint` built-in accepts exactly one argument – the index of the Point to be retrieved\(^1\). 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.

```java
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 retrieved. 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 height 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 spent 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

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b505330</td>
<td>Zitong Wang, 2 weeks ago: Add semantic check.</td>
</tr>
<tr>
<td>4f153ee</td>
<td>Kun An, 2 weeks ago: Changed function return type to curvet.</td>
</tr>
<tr>
<td>d0a5e88</td>
<td>Kun An, 2 weeks ago: Used type curvet instead of string to denote variable type.</td>
</tr>
<tr>
<td>6894b08</td>
<td>Kun An, 3 weeks ago: Return type is added to function declaration; added the other two types to formal.</td>
</tr>
<tr>
<td>4b37d04</td>
<td>Kun An, 4 weeks ago: Added :: operation. I did not modify the global variable code yet. Local variables are sufficient for testing purposes.</td>
</tr>
<tr>
<td>1oa440</td>
<td>Kun An, 4 weeks ago: Defined how curve’s id is stored in a layer.</td>
</tr>
<tr>
<td>14fc4c9</td>
<td>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.</td>
</tr>
<tr>
<td>f0d1196</td>
<td>Kun An, 4 weeks ago: Layer is added to the language. Will modify draw to deal with drawing a layer of curves.</td>
</tr>
<tr>
<td>bd3ed99</td>
<td>Kun An, 4 weeks ago: Added the matrix expr to allow transformation by applying a matrix.</td>
</tr>
<tr>
<td>5235459</td>
<td>Kun An, 4 weeks ago: Added transformation support &gt;&gt; (_, _, _, _). For curve or point, we can do operations such as draw(c &gt;&gt; (2.1 &gt;&gt; (-1. -1))). The shift transformation implementation is done. The translation will be added later on.</td>
</tr>
<tr>
<td>c9f52e0</td>
<td>Kun An, 4 weeks ago: Added &gt;&gt; support, which can be followed by (_, <em>) or (</em>, _, _, _). For curve or point, we can do operations for form (draw(c &gt;&gt; (2.1 &gt;&gt; (-1. -1))). The shift transformation implementation is done. The translation will be added later on.</td>
</tr>
<tr>
<td>7fccc9fd</td>
<td>Kun An, 4 weeks ago: Converted tab to spaces.</td>
</tr>
<tr>
<td>1ff6886</td>
<td>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.</td>
</tr>
<tr>
<td>03004ed</td>
<td>Kun An, 4 weeks ago: Now function call returns int list. Run ./microc -i &lt; test-arith1.mc to see how it works.</td>
</tr>
<tr>
<td>03764be</td>
<td>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 an int list; the vtype field is therefore deleted.</td>
</tr>
<tr>
<td>361290a</td>
<td>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.</td>
</tr>
<tr>
<td>d2b18c2</td>
<td>Kun An, 4 weeks ago: &quot;let rec eval env = function&quot; now returns int list + env; Need to change the return exception to type int list + env.</td>
</tr>
<tr>
<td>473ff4e</td>
<td>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).</td>
</tr>
<tr>
<td>b25c12</td>
<td>Kun An, 4 weeks ago: Compilable version that contains type in parameter list.</td>
</tr>
<tr>
<td>85e39e6</td>
<td>Kun An, 4 weeks ago: Removed compiled files.</td>
</tr>
</tbody>
</table>
5.5.1 Master

d50ba2b - Zitong Wang, Wed Dec 19 15:53:16: add slide for interpreter and type correction
1829b24 - Zitong Wang, Wed Dec 19 15:01:29: correct typo
7f88710 - David Mauskop, Wed Dec 19 02:02:37: Final report updates to grayscale
868c293 - 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
bd66c8c - Wisdom Omuya, Tue Dec 18 23:10:35: Merge branch 'master' of https://bitbucket.org/John Chan/plt-project presentation update
8a33107 - Wisdom Omuya, Tue Dec 18 23:10:27: small updated to presentation
64e28d2 - David Mauskop, Tue Dec 18 22:58:24: Std lib updates
9ee9325 - David Mauskop, Tue Dec 18 21:56:12: Verify function and pararent types
43ee8e6 - John Chan, Tue Dec 18 20:09:00: merged
b0d52fd - John Chan, Tue Dec 18 20:07:33: changed presentation for backend
b518368 - 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
c33e77a - Kun An, Tue Dec 18 18:47:07: minor changes
6b5f0b9 - David Mauskop, Tue Dec 18 16:47:45: Lessons learned
a777e99 - John Chan, Tue Dec 18 07:56:16: added to backend
736b70 - 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
0c9e7f2 - Kun An, Mon Dec 17 16:27:36: tree.png
518cb44 - Wisdom Omuya, Mon Dec 17 16:23:19: syntax
f1def3d - Kun An, Mon Dec 17 16:23:18: add tree.png
75723f4 - Wisdom Omuya, Mon Dec 17 16:18:58: small changes
309355a - 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
4311f4c - Zitong Wang, Mon Dec 17 16:10:56: frontend part in slides
9a55f16 - 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 16:02:31: Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
1f3d4ec - Wisdom Omuya, Mon Dec 17 15:54:31: updated lessons learned
0899cb8 - Zitong Wang, Mon Dec 17 15:53:32: frontend in slides
59c449a - Wisdom Omuya, Mon Dec 17 15:44:47: frontend branch 'master' of https://bitbucket.org/John Chan/plt-project added subsection to test plan
334fa36 - Wisdom Omuya, Mon Dec 17 15:44:43: added subsection
85f2eb5 - Kun An, Mon Dec 17 15:43:46: Merge branch 'master' of https://bitbucket.org/John Chan/plt-project

Page 29 of 83
f7a798d – Kun An, Mon Dec 17 15:43:19 : Added example code to presentation and added my lessons to final report.
41 eb719f0 – John Chan, Mon Dec 17 20:41:38 2012 +0000 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
42 cbf24fa – John Chan, Mon Dec 17 20:41:29 2012 +0000 : edited final.tex
83 8c15f04 – Wisdom Onuya, Mon Dec 17 15:40:03 : moved frontend
55 559ac78 – Wisdom Onuya, Mon Dec 17 15:39:44 : added lessons learned
67 673c61 – Wisdom Onuya, Mon Dec 17 15:36:48 : removed headers
98 98d266d – Wisdom Onuya, Mon Dec 17 15:30:00 : oh goodness!
62 fbea68e – Wisdom Onuya, Mon Dec 17 15:18:50 : added log entries
cfa84a8 – Wisdom Onuya, Mon Dec 17 15:17:56 : added log entries
22 2225f0a – John Chan, Mon Dec 17 20:14:53 2012 +0000 : conflict
c9 208917e – John Chan, Mon Dec 17 20:10:34 2012 +0000 : changed →
test suite
50 b0fb43d – Wisdom Onuya, Mon Dec 17 15:09:00 : added log
51 af6d0c0 – Wisdom Onuya, Mon Dec 17 15:01:30 : fixed conflict
53 5e8914f – Wisdom Onuya, Mon Dec 17 14:59:04 : updates to final report
c6 6a45f – John Chan, Mon Dec 17 19:57:52 2012 +0000 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
55 b59d126 – John Chan, Mon Dec 17 19:57:47 2012 +0000 : changed →
test suite
e0e0386 – David Hauskop, Mon Dec 17 14:54:45 : intro updated
c22f5b0 – John Chan, Mon Dec 17 19:49:11 2012 +0000 : resolved → conflicts
4d 4ffcd50 – John Chan, Mon Dec 17 19:46:26 2012 +0000 : added to section ←
test suite
1870cc – Wisdom Onuya, Mon Dec 17 14:45:10 : style
21 2153255 – Wisdom Onuya, Mon Dec 17 14:43:09 : cleaned language
61 f3e511e – Wisdom Onuya, Mon Dec 17 14:41:57 : revert
7c 7c49fb4 – Wisdom Onuya, Mon Dec 17 14:37:25 : added style guide
27 27f53d6 – Wisdom Onuya, Mon Dec 17 14:36:32 : added style guide
31 313b6fb – David Hauskop, Mon Dec 17 14:33:44 : Code table added
f4 540e6c – Zitong Wang, Mon Dec 17 14:30:58 : lesson learned of Zitong
56 c6cb7a7 – Zitong Wang, Mon Dec 17 14:30:06 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
74 7430039 – Zitong Wang, Mon Dec 17 14:29:58 : lesson learned of Zitong
02 0289a18 – Wisdom Onuya, Mon Dec 17 14:25:51 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
changes to project plan
9b 9ba248f – Wisdom Onuya, Mon Dec 17 14:25:39 : changes to project plan
20 2021af9 – David Hauskop, Mon Dec 17 14:13:44 : Intro added
e8 87c3a0 – Kun An, Mon Dec 17 13:59:46 : Updated the tree example.
e6 6e4df1e – John Chan, Mon Dec 17 09:53:10 : removed some files, edited .gitignore
1d 1dbba6 – 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
19 1925976 – John Chan, Sun Dec 16 22:03:57 : added close_graph so it doesn't freeze my machine
96 96ab90 – David Hauskop, Sun Dec 16 19:02:20 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
20 20760e2 – David Hauskop, Sun Dec 16 19:02:05 : Added emptyLayer to stdlib
92 926664e – John Chan, Sun Dec 16 17:51:50 : pdf
4b 4abc3a – John Chan, Sun Dec 16 17:51:27 : merge conflict fixed
6f 6f6bf9f – John Chan, Sun Dec 16 17:50:16 : fixed & problem and also patched my section from previous commit
6e 6ef2fbd – Zitong Wang, Sun Dec 16 17:41:55 : fixed an error in final.tex
b1 1b5b54 – Kun An, Sun Dec 16 17:31:35 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
7abdd12 – Zitong Wang, Sun Dec 16 17:31:27 : semantic port in final report
eb97ed0 – John Chan, Sun Dec 16 17:25:40 : cleaned up appendix
41ee3f1 – David Hauskopf, 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
544e2b1 – John Chan, Sun Dec 16 16:59:18 : final sections
a46e6f8 – Zitong Wang, Sun Dec 16 16:57:25 : delete addtolayer from token of parser
a6af6eb – Kun An, Sun Dec 16 16:40:43 : Modified the example section in LRM.
81c108e – Zitong Wang, Sun Dec 16 16:35:40 : LRM change for fund and var declaration
fe058b8 – Wisdom Omuya, Sun Dec 16 16:34:15 : several changes
91ba025 – Wisdom Omuya, Sun Dec 16 16:26:02 : updated references
6077e9a – 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
a8b96e4 – Wisdom Omuya, Sun Dec 16 16:08:05 : updated LRM as well
a6d6230 – Wisdom Omuya, Sun Dec 16 16:06:15 : fixed max nesting
d80c6bb – Wisdom Omuya, Sun Dec 16 16:03:06 : updated final report
0e56db1 – Kun An, Sun Dec 16 15:39:23 : Added the example section and part of front end section.
57ff44a – Kun An, Sun Dec 16 12:07:29 : Added example: tree creation and traversal.
1a9a8d7 – John Chan, Sun Dec 16 09:14:50 : bytecode final.tex
4cd3d6b – John Chan, Sat Dec 15 17:34:09 : flow added to final
2x8d2e7 – John Chan, Sat Dec 15 17:26:00 : added flowchart
e1aed05 – John Chan, Sat Dec 15 16:00:30 : overview of architectural design
19d1d5a – John Chan, Sat Dec 15 11:05:26 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
ac6b1ed – John Chan, Sat Dec 15 11:05:19 : edited LRM section
3ab5706 – David B Hauskopf, Sat Dec 15 10:37:31 : Stdlib updates
0ccdf41 – David B Hauskopf, 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
5ff3bdf – Wisdom Omuya, Fri Dec 14 17:50:12 : added dave :}
6445e06 – Wisdom Omuya, Fri Dec 14 17:48:54 : restructured final report
442b262 – Wisdom Omuya, Fri Dec 14 17:45:26 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project make changes
80e6673 – Wisdom Omuya, Fri Dec 14 17:45:07 : changes to final report
0305a02 – Zitong Wang, Fri Dec 14 17:37:22 : remove AddToLayer
2a17ebc – Wisdom Omuya, Fri Dec 14 17:29:04 : override built-ins
85df5f0 – John Chan, Fri Dec 14 15:12:37 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
14f2f616 - John Chan, Fri Dec 14 15:12:33 : fixed default curve action → to be compilable, the index was incorrect (curve.ml)
1d5c4f68 - Kun An, Fri Dec 14 14:58:15 : Merge branch 'master' of https://bitbucket.org/John Chan/plt-project
1efa04b - Kun An, Fri Dec 13 14:58:02 : Moved examples
1b1c8bd2 - Kun An, Fri Dec 14 14:48:01 : Added rotate and scale to stl. →
1691596c - David Bauskog, Fri Dec 14 14:39:36 : Stdlib updates and → hanoi example added
1c80cc8c - John Chan, Fri Dec 13 14:40:49 : changed the debug flag in ← execute
1d6698c3 - John Chan, Fri Dec 14 09:18:10 : removed Trans from code, ← this is not needed
1a062b9f - Wisdom Omuya, Thu Dec 13 21:33:31 : not exactly but anyways
1405aef1 - Wisdom Omuya, Thu Dec 13 21:27:20 : clean before commit
17d377e6 - Wisdom Omuya, Thu Dec 13 21:26:13 : cleaned things up a bit
1f847f4b - Wisdom Omuya, Thu Dec 13 21:22:16 : cleaned files
19ea1cfa - Wisdom Omuya, Thu Dec 13 21:21:59 : updated gitignore
14b6d96e - Wisdom Omuya, Thu Dec 13 21:20:42 : moved graphics example
1e5aad63 - Wisdom Omuya, Thu Dec 13 21:06:14 : Merge branch 'master' of← bitbucket.org:John Chan/plt-project updated example
1aba9953 - Wisdom Omuya, Thu Dec 13 21:06:09 : updated example
1522fe43 - Kun An, Thu Dec 13 20:54:42 : reverted the example back
14b6414e - Kun An, Thu Dec 13 20:49:26 : i++ and i— is working
1e033d3a - Zitong Wang, Thu Dec 13 14:11:45 : Add support for recursive function in semantic.ml
7196e4c4 - 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
1485a70a - John Chan, Thu Dec 13 12:12:45 : added old tests from microc← removed tests folder (but test is still intact)
42d8f4a9 - Wisdom Omuya, Thu Dec 13 11:55:01 : removed random test
29f447ab - Wisdom Omuya, Thu Dec 13 11:33:26 : fixed random function
39e0dd59 - John Chan, Thu Dec 13 11:20:30 : Merge branch 'master' of← bitbucket.org:John Chan/plt-project
904f6eb8 - John Chan, Thu Dec 13 11:19:52 : transform works, similar to← translate
1be7e9db - Zitong Wang, Thu Dec 13 11:18:21 : add check for random()
1de4dc7a - John Chan, Thu Dec 13 10:58:18 : added testb, which bypasses← all checking, for debugging purposes←
1efe5bb3 - John Chan, Thu Dec 13 10:54:24 : Merge branch 'master' of← bitbucket.org:John Chan/plt-project
1fa11e89c - John Chan, Thu Dec 13 10:53:56 : refactored curve.ml, added← a bypass function to bypass semantic checker
1f4da9e7 - Zitong Wang, Thu Dec 13 10:50:45 : fix bug of semantic.ml
1b85d705 - John Chan, Thu Dec 13 10:28:07 : fixed freezing index out of← bounds issue
19c28907 - John Chan, Thu Dec 13 09:38:40 : fixed nth, random is the← only test to fail
1bb09859 - Wisdom Omuya, Thu Dec 13 02:43:16 : caps check
120243ac - Wisdom Omuya, Thu Dec 13 02:41:13 : removed extraneous getY
109303bd - Wisdom Omuya, Thu Dec 13 02:31:57 : updated lrm
14f6b449 - Zitong Wang, Thu Dec 13 01:49:15 : Change system fund part ← of LRM
157c605b - Zitong Wang, Thu Dec 13 01:31:11 : add setPoint back
1bf122a8 - John Chan, Thu Dec 13 00:35:50 : modified test code to use ← the dot format
1540fa43 - Zitong Wang, Wed Dec 12 23:00:18 : change in semantic.ml
1e2e21fe - Wisdom Omuya, Wed Dec 12 20:10:35 : updated lrm
170fc0cb - Wisdom Omuya, Wed Dec 12 20:06:06 : updated example
1a236d9c - Wisdom Omuya, Wed Dec 12 20:02:56 : updated example
1dd7a588 - Wisdom Omuya, Wed Dec 12 19:59:27 : updated example
571a04a - Wisdom Omuya, Wed Dec 12 19:07:55 : minor changes
67a2512 - Wisdom Omuya, Wed Dec 12 18:58:18 : incorporating semantic
71ed073 - Zitong Wang, Wed Dec 12 18:52:08 : Merge commit '997←→
aba6a9d90f1f4365e8d8bb9d3400eb052a9e'
997a9a6 - 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
47b1934 - Kun An, Wed Dec 12 18:36:25 : Added circle to STL
98e169c - Wisdom Omuya, Wed Dec 12 18:35:56 : updated makefile
cd63ae5 - Wisdom Omuya, Wed Dec 12 18:33:27 : changed stlib to stdlib,←
235ef56 - 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
05100eb - 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
f9cabc0 - Wisdom Omuya, Wed Dec 12 17:28:51 : increased stack space
f5db2ab - Kun An, Wed Dec 12 17:23:45 : Added to example programs.
dcbe6e3 - David B Hauskop, Wed Dec 12 16:56:27 : Merge branch 'master'←
of https://bitbucket.org/John Chan/plt-project
9ae4cf3 - David B Hauskop, Wed Dec 12 16:53:58 : Added to standard←
library
acc3072 - Wisdom Omuya, Wed Dec 12 16:43:24 : added 'expected' random←
outcome
34ff661 - Wisdom Omuya, Wed Dec 12 16:38:41 : added random builtin
73974f8 - John Chan, Wed Dec 12 10:49:11 : translate works
90ed8d5 - John Chan, Wed Dec 12 09:52:03 : added print layer to stl
5a00223 - John Chan, Wed Dec 12 09:50:54 : set curve works
2996ff0 - John Chan, Tue Dec 11 18:15:58 : Merge branch 'master' of←
bitbucket.org:John Chan/plt-project
971b5af - John Chan, Tue Dec 11 17:59:09 : lsize
5fb33c7 - Wisdom Omuya, Tue Dec 11 16:44:50 : minor edits
4096697 - John Chan, Tue Dec 11 16:43:47 : added dot notation as well
370f5bb - John Chan, Tue Dec 11 16:42:58 : added layer size function
284bd79 - Wisdom Omuya, Tue Dec 11 16:40:14 : updated lrms
1cdef6a - 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 lrms 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 : lrms changes Merge branch←
'master' of bitbucket.org:John Chan/plt-project
14d2311 - Wisdom Omuya, Tue Dec 11 15:30:02 : further cleaned up lrms
7cf236b - 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
Curve Final Report December 19, 2012

97e9118 - Wisdom Omuya, Tue Dec 11 14:58:36 : added point builtins
dcfd249 - John Chan, Fri Nov 30 18:55:17 : notes
2bf58ca - Wisdom Omuya, Tue Dec 11 14:48:13 : removed doubles from lrm
fa0304a - 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
b682ae0 - 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
0bsfd91 - 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
setx works

b4e25be - Wisdom Omuya, Sat Dec 8 18:14:10 : updated test
ea99b99 - 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
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
b267d21 - Ubuntu, Tue Dec 4 16:41:37 2012 +0000 : middle of set
function

9b2ad9b - John Chan, Mon Dec 3 21:54:09 : refactorereed
27edeed - Wisdom Omuya, Mon Dec 3 18:47:44 : added clear
6baf3eb2 - John Chan, Mon Dec 3 10:54:16 : notes
bccc5125 - 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
00f8a8a - John Chan, Sat Dec 1 11:29:31 : call
3bf9e32 - John Chan, Sat Dec 1 11:19:33 : fixed dropsize area
ec8d35a - John Chan, Sat Dec 1 10:59:10 : fixed sizeof type
9756476 - John Chan, Sat Dec 1 00:49:40 : tested dot op on get
33838db - John Chan, Sat Dec 1 00:40:41 : better implementation of Ind ugly implementation of Ind
ec2bf41 - John Chan, Sat Dec 1 00:37:13 : get for point works, but u
la78f61 - John Chan, Fri Nov 30 18:55:17 : notes
dcfdf49 - John Chan, Fri Nov 30 15:15:13 : notes
d00b39a - John Chan, Fri Nov 30 18:05:37 : notes
8bc7bf6 - John Chan, Thu Nov 29 17:00:05 : setx sety work
78f5fd7 - John Chan, Thu Nov 29 16:54:01 : setx works
Curve Final Report

December 19, 2012

Page 35 of 83

John Chan, Thu Nov 29 13:30:31: added dotop tests
John Chan, Thu Nov 29 13:25:53: improved index counting
for built_in_functions
John Chan, Thu Nov 29 13:17:26: dot op works for getx() ← gety()
John Chan, Thu Nov 29 13:17:32: updated curve drawing ←
mechanism
Wisdom Omuya, Wed Nov 28 19:49:54: revert to saner ←
animation
Wisdom Omuya, Wed Nov 28 19:49:19: increased stack space
cleaned up pause
Wisdom Omuya, Wed Nov 28 18:52:08: less awkward layer
Wisdom Omuya, Wed Nov 28 18:43:43: layers working
Wisdom Omuya, Wed Nov 28 16:53:03: reverted for tests
Wisdom Omuya, Wed Nov 28 16:34:14: updated curve drawing ←
mechanism
Wisdom Omuya, Wed Nov 28 15:06:54: lists load correctly
Wisdom Omuya, Wed Nov 28 14:01:09: passing graphics test
Wisdom Omuya, Wed Nov 28 13:46:37: reverted to working ←
compile.nl
df10d3 ← Wisdom Omuya, Wed Nov 28 13:21:54: added pause to compile
Wisdom Omuya, Wed Nov 28 13:18:03: updated compile to open ←
graph
Wisdom Omuya, Wed Nov 28 13:14:20: added draw function
gitignore
Wisdom Omuya, Wed Nov 28 13:12:46: added test−curv1.cv
layer1.out
Wisdom Omuya, Wed Nov 28 12:56:40: updated makefile
Wisdom Omuya, Wed Nov 28 12:50:01: Merge branch 'master' of ←
bitbucket.org:John Chan/plt−project
Wisdom Omuya, Wed Nov 28 12:49:42: added layer test, changed ←
execute
Wisdom Omuya, Wed Nov 28 12:47:37: Merge branch 'master' of ←
bitbucket.org:John Chan/plt−project
Wisdom Omuya, Wed Nov 28 12:47:29: removed microc.nl
Wisdom Omuya, Wed Nov 28 12:16:47: added solocall test
Wisdom Omuya, Wed Nov 28 12:16:11: fixed single call to fn ←
stack inconsistency
Wisdom Omuya, Wed Nov 28 11:42:16: enumerated types, added a ←
return type, fixed tests to include return types
Wisdom Omuya, Wed Nov 28 11:06:34: changed testall script, ←
bug
Wisdom Omuya, Tue Nov 27 20:38:37: fixed bug
Wisdom Omuya, Tue Nov 27 15:08:57: merged
Wisdom Omuya, Tue Nov 27 15:03:58: moved some files
Wisdom Omuya, Tue Nov 27 14:53:23: removed drpe
Wisdom Omuya, Tue Nov 27 14:33:31: added test case
Wisdom Omuya, Tue Nov 27 14:16:27: return point works
Wisdom Omuya, Tue Nov 27 00:17:37: sizeofexpr
Wisdom Omuya, Mon Nov 26 18:59:07: added run
Wisdom Omuya, Mon Nov 26 16:30:45: bug
Wisdom Omuya, Mon Nov 26 15:04:04: getx gety work
Wisdom Omuya, Mon Nov 26 14:09:32: added gety
Wisdom Omuya, Mon Nov 26 13:46:15: getx works
Wisdom Omuya, Sun Nov 25 17:11:47: debugged stack op
Wisdom Omuya, Sun Nov 25 08:51:59: fixed size of id
Wisdom Omuya, Sat Nov 24 11:04:59: need to fix printing 2vals
Wisdom Omuya, Sat Nov 24 09:50:26: checkpoint
Wisdom Omuya, Sat Nov 24 09:27:19: checkpoint
Wisdom Omuya, Sat Nov 24 09:07:59: finished drp in stmt
Wisdom Omuya, Sat Nov 24 08:58:41: fixed rts
Wisdom Omuya, Fri Nov 23 17:31:40: ed
Wisdom Omuya, Fri Nov 23 09:18:34: edit
patched files

5b5cf80 John Chan Thu Nov 20 11:41:18: copied master to this

a8bcae2 John Chan Tue Nov 20 11:25:08: notes

d412c49 John Chan Mon Nov 19 00:50:46: removed cepr

3d35b04 John Chan Sat Nov 17 17:14:51: ed

3fa97b3 John Chan Sat Nov 17 15:46:36: bytecode

c96a4ef John Chan Sat Nov 17 11:16:36: notes

60c1a30 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 21:37:58: cleaner interpret

b06757c John Chan Fri Nov 16 10:52:36: updated lrm on jc branch

4760fa John Chan Fri Nov 16 01:53:37: fix two cepr errors (todo)

5f17e2f John Chan Fri Nov 16 00:51:47: added some more to parser, and scanner

d533b7f 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 gtest example

cf1c1d5 John Chan Wed Nov 14 16:09:56: notes

3759227 John Chan Wed Nov 14 15:25:14: slight edits to jc

75a470d John Chan Wed Nov 14 15:15:02: edit

14c4cb8 John Chan Wed Nov 14 14:12:14: added notes

b9993e3 John Chan Fri Nov 9 17:53:19: 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

caf89b1 John Chan Mon Nov 5 14:29:49: slight edit to scanner.

653f9a6 John Chan Mon Nov 5 14:23:59: reverted back to the old testall script.

6a81f98 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

c93690 Wisdom Omuya Thu Nov 1 17:18:45 2012 -0400: reverted to working condition

aaf253 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

24dc5f1 Wisdom Omuya Sat Oct 27 13:15:39 2012 -0400: updated object types section
Curve Final Report

January 19, 2012

bb7e5eb - 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
46ba099 - 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
7cc9d5f - 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
2ec6f2c - John Chan, Thu Oct 11 18:27:20 2012 -0400: added more information, and some examples
ee4c83f - John Chan, Thu Oct 11 18:24:38 2012 -0400: added more to curve2
fd98798 - John Chan, Thu Oct 11 00:06:48 2012 -0400: added some more to curve2
1ae7933 - Wisdom Omuya, Wed Sep 26 16:22:11 2012 -0400: reduced # pages
4f23a8a - David Hauskop, Wed Sep 26 16:16:29 2012 -0400: Added built-in types and operators in types and operators
fd2c2db - 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
c9d43bb - 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 sections owners
1405913 - John Chan, Sat Sep 22 17:55:26 2012 -0400: added more code examples
19f3bb5 - 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
9ed39d9 - John Chan, Thu Sep 20 12:00:21 2012 -0400: added contacts back in
f47d82e - Wisdom Omuya, Thu Sep 20 11:34:06 2012 -0400: rough initial proposal draft
2691e7f1 - 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

Page 38 of 83
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 Builtin 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
- OCaml yacc: 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 representations, 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.
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lit of int</td>
<td>Pushes a single integer value on to the top of the stack.</td>
</tr>
<tr>
<td>Drp</td>
<td>Pops the stack.</td>
</tr>
<tr>
<td>Bin of Ast.op</td>
<td>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.</td>
</tr>
<tr>
<td>Lod of int</td>
<td>Fetches the global variable specified by the int.</td>
</tr>
<tr>
<td>Str of int</td>
<td>Stores the global variable specified by the int.</td>
</tr>
<tr>
<td>Lfp of int</td>
<td>Loads a value indicated by the int, relative to the frame pointer.</td>
</tr>
<tr>
<td>Sfp of int</td>
<td>Puts the value from the top of the stack to a location indicated by the int, relative to the frame pointer.</td>
</tr>
<tr>
<td>Jsr of int</td>
<td>Moves the program counter to the location specified by the int. Used for jumping into a subroutine.</td>
</tr>
<tr>
<td>Ent of int</td>
<td>Prepares the stack for a new activation record by allocating appropriate number of bytes for local variables.</td>
</tr>
<tr>
<td>Rta</td>
<td>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.</td>
</tr>
<tr>
<td>Rts of int</td>
<td>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.</td>
</tr>
<tr>
<td>Beq of int</td>
<td>Branch if equal.</td>
</tr>
<tr>
<td>Bne of int</td>
<td>Branch not equal.</td>
</tr>
<tr>
<td>Bra of int</td>
<td>Branch.</td>
</tr>
<tr>
<td>Ind of int</td>
<td>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.</td>
</tr>
<tr>
<td>Ins of int</td>
<td>Similar to Ind, but the value on the top of the stack is stored at the location instead.</td>
</tr>
<tr>
<td>Hlt</td>
<td>Halt.</td>
</tr>
<tr>
<td>Ogr</td>
<td>Open Graph.</td>
</tr>
</tbody>
</table>

### 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 language 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**

```plaintext
int main()
{
   print(1);
}
```

**7.1.2 target – print.cv**

```plaintext
0 global variables
0 0gr
1 Jsr 113
2 Ent 0
3 Lfp -3
4 5 Rta
5 6 Rts 2
6 Ent 0
7 Lfp -2
8 Rta
9 10 Sfp -3
10 11 Rts 2
11 12 Ent 0
12 13 Ind 1
13 14 Ind 0
14 15 Rta
15 16 Sfp -10
16 17 Drp
17 18 Sfp -11
18 19 Rts 9
```
7.1.3 source – points.cv

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

7.1.4 target – points.cv

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

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:

```
0gr fp: 0 sp: 0 pc: 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
Est 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
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
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
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
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 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 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 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 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.

<table>
<thead>
<tr>
<th>File name</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>scanner.mll</td>
<td>62</td>
</tr>
<tr>
<td>ast.ml</td>
<td>43</td>
</tr>
<tr>
<td>parser.mly</td>
<td>147</td>
</tr>
<tr>
<td>bytecode.ml</td>
<td>62</td>
</tr>
<tr>
<td>compile.ml</td>
<td>366</td>
</tr>
<tr>
<td>execute.ml</td>
<td>101</td>
</tr>
<tr>
<td>semantic.ml</td>
<td>385</td>
</tr>
<tr>
<td>curve.ml</td>
<td>57</td>
</tr>
<tr>
<td>interpret.ml</td>
<td>222</td>
</tr>
<tr>
<td>stl.cv</td>
<td>564</td>
</tr>
<tr>
<td>Makefile</td>
<td>66</td>
</tr>
</tbody>
</table>

9.1 scanner.mll

```ml
{
  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 = [' ' '
' '	' '' '
']

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              |
```
9.2 ast.ml

type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater <
| Geq

type curvet =
| Literal
| Curvet
| Pointt
| Layert

type expr =
| Literal of int
| Dotop of string * string * expr list
| Curve of expr * expr * expr * expr * expr * expr * expr * expr
| Point of expr * expr
| Layer of string list
| Id of string
| Binop of expr * op * expr
| Assign of string * expr
| Call of string * expr list
| Noexpr

type stmt =
| Block of stmt list
| Expr of expr
| Return of expr
| If of expr * stmt * stmt
| For of expr * expr * expr * stmt
| While of expr * stmt

type var_decl = {
  t : curvet;
  name : string;
  value : int list;
}
9.3 parser.mly

```ml
| open Ast |
| token SEMI LPAREN RPAREN LBRACK RBRACK LBRACE RBRACE COMMA |
| token PLUS MINUS TIMES DIVIDE ASSIGN |
| token EQ NEQ LT LEQ GT GEQ |
| token RETURN IP ELSE FOR WHILE |
| token INT POINT CURVE DOT LAYER |
| token PLUSONE MINUSONE |
| token <int> LITERAL |
| token <string> LITERAL |
| token EOF |
| nonassoc NOELSE |
| nonassoc ELSE |
| left ASSIGN |
| left EQ NEQ |
| left LT GT LEQ GEQ |
| left PLUS MINUS |
| left TIMES DIVIDE |
| start program |
| type <<Ast-program>> program |
| |
| program: |
| /* nothing */ { [], [] } |
| program vdecl { $2 :: fst $1 }, snd $1 } |
| program fdecl { fst $1, ($2 :: snd $1) } |

fdecl: |
| INT ID LPAREN forms_opt RPAREN LBRACE vdecl_list stmt_list ← RBRACE |
| { |
| return = Literalt; |
| fname = $2; |
| forms = $4; |
| locals = List.rev $7; |
| body = List.rev $8 } } |
| POINT ID LPAREN forms_opt RPAREN LBRACE vdecl_list stmt_list ← RBRACE |
| { |
| return = Pointt; |
| fname = $2; |
| forms = $4; |
| locals = List.rev $7; |
| body = List.rev $8 } } |
| CURVE ID LPAREN forms_opt RPAREN LBRACE vdecl_list stmt_list ← RBRACE |
| { |
| return = Curvet; |
```

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

fname = $2$
formals = $4$
locals = List.rev $7$
body = List.rev $8$
| LAYER ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list \rightarrow RBRACE
|
| \{ return = Layert;
fname = $2$
formals = $4$
locals = List.rev $7$
body = List.rev $8$ \}
formals_opt:
/\* nothing */ \{ [] \}
| formals_list \{ List.rev $1$ \}
formal_list:
| \{ formal_list COMMA formal \{ $3 :: $1$ \} \}
formal:
| \{ INT ID \{ \{ t = Literalt; name = $2$; value = [0] \} \}
| \{ POINT ID \{ \{ t = Points; name = $2$; value = [0;0] \} \}
| \{ CURVE ID \{ \{ t = Curvet; name = $2$; value = [0;0;0;0;0; \rightarrow 0;0;0;0;0] \} \}
| \{ LAYER ID \{ \{ t = Layert; name = $2$; value = [] \} \}
vdecl_list:
/\* nothing */ \{ [] \}
| vdecl_list vdecl \{ $2$ :: $1$ \}
vdecl:
| \{ INT ID SEMI \{ \{ t = Literalt; name = $2$; value = [0] \} \}
| \{ POINT ID SEMI \{ \{ t = Points; name = $2$; value = [0;0] \} \}
| \{ CURVE ID SEMI \{ \{ t = Curvet; name = $2$; value = [0;0;0;0;0; \rightarrow 0;0;0;0;0] \} \}
| \{ LAYER ID SEMI \{ \{ t = Layert; name = $2$; value = [] \} \}
stmt_list:
/\* nothing */ \{ [] \}
| stmt_list stmt \{ $2$ :: $1$ \}
stmt:
expr SEMI \{ Expr($1$) \}
| RETURN expr SEMI \{ Return($2$) \}
| LBRACE stmt_list RBRACE \{ Block(List.rev $2$) \}
| IF LPAREN expr RPAREN stmt %prec NOELSE \{ If($3$; $5$; Block($[$]) \}
| IF LPAREN expr RPAREN stmt ELSE stmt \{ If($3$; $5$; $7$) \}
| FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt
| \{ For($3$; $5$; $7$; $9$) \}
| WHILE LPAREN expr RPAREN stmt \{ While($3$; $5$) \}
expr_opt:
/\* nothing */ \{ Noexpr \}
| expr \{ $1$ \}
expr:
| LITERAL \{ Literal($1$) \}
| ID DOT ID LPAREN actuals_opt RPAREN
| \{ Dotop($1$; $3$; $5$) \}
| LPAREN expr COMMA expr RPAREN
| LPAREN expr COMMA expr RPAREN
| LPAREN expr RPAREN
| LPAREN expr COMMA expr RPAREN
| LPAREN expr COMMA expr RPAREN
| \{ Curve($2$; $4$; $7$; $9$; $12$; $14$; $17$; $19$) \}
| LPAREN expr COMMA expr RPAREN
| LPAREN expr COMMA expr RPAREN
| LPAREN expr RPAREN
| LPAREN expr COMMA expr RPAREN
| \{ Curve($2$; $4$; $7$; $9$; $12$; $14$; $17$; $19$) \}
9.4 bytecode.ml

```ml
let type bstmt = 
  | Lit of int (* Push a literal *)
  | Drp (* Discard a value *)
  | Bin of Ast.op (* Perform arithmetic on top of stack *)
  | Ld of int (* Fetch global variable *)
  | Str of int (* Store global variable *)
  | Lfp of int (* Load frame pointer relative *)
  | Sfp of int (* Store frame pointer relative *)
  | Jsr of int (* Call function by absolute address *)
  | Ent of int (* Push FP, FP -> SP, SP := i *)
  | Rta (* sets new pc and fp *)
  | Rts of int (* Restore FP, SP, consume formals, push result *)
  | Beq of int (* Branch relative if top-of-stack is zero *)
  | Bne of int (* Branch relative if top-of-stack is non-zero *)
  | Bra of int (* Branch relative *)
  | Ins of int (* Stores value relative to the top of the stack *)
  | Hlt (* Terminate *)
  (* curve related instructions *)
  | 0gr (* Open graph *)

let type prog = {
  num Globals : int; (* Number of global variables *)
  text : bstmt array; (* Code for all the functions *)
```
9.5 compile.ml

let string_of_prog p =
  string_of_int p.numglobals " global variables
  "
let funca = Array.map
  (fun i s -> string_of_int i " " string_of_stmt s) p.text
in String.concat "\n" (Array.to_list funca)

open Ast
open Bytecode

module StringMap = Map.Make(String)

(* Symbol table: Information about all the names in scope *)
type env = {
  function_index : int StringMap.t; (* Index for each function *)
  global_index : int StringMap.t; (* "Address" for global variables *)
  local_index : int StringMap.t; (* FP offset for args, locals *)
}

(* val enum : int -> 'a list -> (int * 'a) list *)
let rec enum stride n = function
  | [] -> []
  | hd::tl -> (n, hd) :: enum stride (n+stride) tl

let sizeof Gil = function
  | Literal -> 1
  | Pointt -> 2
  | Curvet -> 8
  | Layert -> 81

let string_of_stmt = function
  | Ogr -> "Ogr"
  | Lit(i) -> "Lit" " string_of_int i
  | Drp -> "Drp"
  | Bin(Ast.Add) -> "Add"
  | Bin(Ast.Sub) -> "Sub"
  | Bin(Ast.Mult) -> "Mul"
  | Bin(Ast.Div) -> "Div"
  | Bin(Ast.Equal) -> "Eq!
  | Bin(Ast.Neq) -> "Neq"
  | Bin(Ast.Less) -> "Lt"
  | Bin(Ast.Leq) -> "Leq"
  | Bin(Ast.Greater) -> "Gt"
  | Bin(Ast.Geq) -> "Geq"
  | Lod(i) -> "Lod" " string_of_int i
  | Str(i) -> "Str" " string_of_int i
  | Lfp(i) -> "Lfp" " string_of_int i
  | Sfp(i) -> "Sfp" " string_of_int i
  | Jsr(i) -> "Jsr" " string_of_int i
  | Ent(i) -> "Ent" " string_of_int i
  | Rts(i) -> "Rts" " string_of_int i
  | Rta -> "Rta"
  | Bne(i) -> "Bne" " string_of_int i
  | Beq(i) -> "Beq" " string_of_int i
  | Bra(i) -> "Bra" " string_of_int i
  | Ind(i) -> "Ind" " string_of_int i
  | Ins(i) -> "Ins" " string_of_int i
  | Hlt -> "Hlt"
```ml
let sizeReq acc l =
  List.fold_left (fun ac el -> ac + (sizeof type t))
    acc l
let rec multiLfp index i n =
  if i = n then []
  else
    [Lfp (index + i)] @ (multiLfp index (i+1) n)
let rec multiLod index i n =
  if i = n then []
  else
    [Lod (index + i)] @ (multiLod index (i+1) n)
let rec multiSfp index i n =
  if i = n then []
  else
    [Drp] @ [Sfp (index + i)] @ (multiSfp index (i+1) n)

(* enum for variables, dir is direction *)
let rec enum_var dir n =
  if dir = [] then []
  else
    sizeOf type hd.t in
    if dir < 0 then
      (n-sz+1, hd.name) :: enum_var dir (n + dir+sz) tl
    else
      (n, hd.name) :: enum_var dir (n + dir+sz) tl

(* val string_map_pairs StringMap 'a -> (int * 'a) list -> StringMap 'a *)
let string_map_pairs map pairs =
  List.fold_left (fun m (i, a) -> StringMap.add n i m) map pairs
let vdeclmap map vlist =
  List.fold_left (fun m v -> StringMap.add v.name v n) map vlist
let fdeclmap map flist =
  List.fold_left (fun m f -> StringMap.add f.fname f.return m) map flist

(* Allocate "addresses" for each global variable *)
let global_indexes = string_map_pairs StringMap.empty
  (enum_var 1 0 globals) in
let funcmap = fdeclmap StringMap.empty functions in
let funcmap = StringMap.add "draw" (Layer) funcmap in
let funcmap = StringMap.add "pause" (Literal) funcmap in
let funcmap = StringMap.add "clear" (Literal) funcmap in
let funcmap = StringMap.add "random" (Literal) funcmap in
let funcmap = StringMap.add "getX" (Literal) funcmap in
let funcmap = StringMap.add "getY" (Literal) funcmap in
```
let funcmap = StringMap.add "setX" (Literal) funcmap in
let funcmap = StringMap.add "setY" (Literal) funcmap in
let funcmap = StringMap.add "setPoint" (Pointtt) funcmap in
let funcmap = StringMap.add "getCurve" (Curvet) funcmap in
let funcmap = StringMap.add "getPoint" (Pointtt) funcmap in
let funcmap = StringMap.add "getPoint" (Pointtt) funcmap in
let funcmap = StringMap.add "getPoint" (Pointtt) funcmap in
let funcmap = StringMap.add "getPoint" (Pointtt) funcmap in
let funcmap = StringMap.add "setCurve" (Curvet) funcmap in
let funcmap = StringMap.add "setCurve" (Literal) funcmap in
let globvmap = vdeclmap StringMap.empty globals in

(* Assign indexes to function names; built-in "print" is special *)
let graphics_functions = StringMap.add "print" (1) StringMap.empty in
let graphics_functions = StringMap.add "draw" (2) graphics_functions in
let graphics_functions = StringMap.add "pause" (3) graphics_functions in
let graphics_functions = StringMap.add "clear" (4) graphics_functions in
let graphics_functions = StringMap.add "random" (5) graphics_functions in
let built_in_functions = StringMap.add "getX" (1) graphics_functions in
let built_in_functions = StringMap.add "getY" (2) graphics_functions in
let built_in_functions = StringMap.add "getPoint" (3) built_in_functions in
let built_in_functions = StringMap.add "setPoint" (4) built_in_functions in
let built_in_functions = StringMap.add "getCurve" (5) built_in_functions in
let built_in_functions = StringMap.add "getPoint" (6) built_in_functions in
let built_in_functions = StringMap.add "setCurve" (7) built_in_functions in
let function_indexes = string_map_pairs built_in_functions
  (enum 1 ((StringMap.cardinal built_in_functions) -
    (StringMap.cardinal graphics_functions) + 1))
  (List.map (fun f -> f.fname) functions)) in

(* Translate a function in AST form into a list of bytecode statements *)
let translate env fdecl =
  (* Bookkeeping: FP offsets for locals and arguments *)
  let sz_formals = sizeReq 0 fdecl.formals in
  let szlocals = sizeReq 0 fdecl.locals in
  let sz_ret = sizeOfVar fdecl.return in
  let local_offsets = enum_var 1 1 fdecl.locals in
  let locvmap = vdeclmap StringMap.empty (fdecl.locals@fdecl.formals in)
  let formal_offsets = enum_var (1 - 1) (2 - 1) fdecl.formals in

let rec sizeofexpr = function
  | Literal i -> 1
  | Dotop (id, fn, e) -> sizeofexpr (Call(fn,[Id(id)]@e))
  | Curve (a,b,c,d,e,f,g,h) -> 8
  | Point (a,b) -> 2
  | Layer last -> 81
  | Id s ->
    (try
      sizeoftype (StringMap.find s locvmap).t
      with Not_found -> try
        sizeoftype (StringMap.find s globvmap).t
        with Not_found -> raise (Failure ("Undeclared variable " s))
      )
  | Binop (e1,op,e2) -> 1
  | Assign(s,e) -> sizeofexpr(Id(s))
  | Call(fname, arg) ->
    (let rett =
      (*
      
      *)
    )
  | (let rett =
      (*
      
      *)
    )
  | (let rett =
      (*
      
      *)
    )
Curve Final Report

December 19, 2012

(try
    (StringMap.find fname funcdmap)
    (match (Not_found => raise (Failure ("undefined function " ^ fname)))))
in sizeoftypeof rett)

let expr -> l in
let env = { env with local_index = string_map_pairs
            StringMap.empty (local_offsets 0 formal_offsets) } in

let rec expr = function
    | Literal i -> [Lit i]
    | Point (a,b) -> expr a @ expr b
    | Curve (a,b,c,d,e,f,g,h) ->
      expr a @
      expr b @
      expr c @
      expr d @
      expr e @
      expr f @
      expr g @
      expr h
    | Dotop (id, fn, e) -> expr (Call(fn, [Id(id)] @e))
    | Layer lst ->
      (let rec blanks i n =
        if i = n then
          []
        else
          [Lit (0)] @ (blanks (i+1) n) in
        blanks 0 (8*(10-(List.length lst))))
      @ List.concat (List.map (fun el -> expr (Id(el))) lst)
    @ [Lit (List.length lst)]
    | Id s ->
      (try
        let startg = (StringMap.find s env.local_index)
        and sz = sizeoftypeof (StringMap.find s locvdmap).t in
        multilp start 0 sz (* @ [Lit sz] *)
        with Not_found -> try
        let startg = (StringMap.find s env.global_index)
        and sz = sizeoftypeof (StringMap.find s globvomap).t in
        multilod startg 0 sz (* @ [Lit sz] *)
        with Not_found -> raise (Failure ("Undeclared variable " ^ sz))
      )
    | Binop (el, op, e2) -> expr el @ expr e2 @ [Bin op]
    | Assign (s, e) ->
      expr e @
      (try
        let startg = (StringMap.find s env.local_index) in
        let sz = sizeoftypeof (StringMap.find s locvdmap).t in
        List.rev (List.tl (multiSfp start 0 sz))
        with Not_found -> try
        let startg = (StringMap.find s env.global_index) in
        let sz = sizeoftypeof (StringMap.find s globvomap).t <-
        in
        List.rev (List.tl (multiStr startg 0 sz))
        with Not_found -> raise (Failure ("undeclared variable " ^ sz))
      )
    | Call (fname, actuals) -> (match fname with
      | "setX" -> expr (Assign(
        (match (List.hd actuals) with
          | Id (s)-> s | _ -> ^
          Point(
            List.hd (List.tl actuals), Call("getY", [(List.hd <-
            actuals)])
          )))
        | "setY" -> expr (Assign(
          (match (List.hd actuals) with
            | Id (s)-> s | _ -> ^
            Point(
          )))
      )
268 | "getPoint" ->
269     (* a b c d e f g h 8 8 a b *)
270 let indindx =
271     (Binop(Binop(Literal(4),Sub.(List.hd (List.tl actuals))←
272       Multi.Literal(2))) in
273     (try
274       expr (List.hd actuals) @
275       expr indindx @ expr indindx @
276       [Js (StringMap.find fn name env.function_index) ]
277     with Not_found ->
278       raise (Failure ("undefined function " → fnname))
279     )
280 | "setPoint" ->
281     (* a b c d e f g h 8 n1 n2 *)
282 let s = match (List.hd actuals) with
283     | Id (s)→ s | _ → "" in
284 let indindx =
285     (Binop(Binop(Literal(4),Sub.(List.hd (List.tl actuals))←
286       Multi.Literal(2))) in
287     (try
288       expr (List.hd actuals) @
289       expr indindx @ expr indindx @
290       expr (List.hd (List.rev actuals)) @
291       [Js (StringMap.find fn name env.function_index) ]
292     with Not_found ->
293       raise (Failure ("undefined function " → fnname))
294     )
295 | "setCurve" ->
296     (try
297       expr (List.hd actuals) @
298       expr (List.hd (List.tl actuals)) @
299       [Js (StringMap.find fn name env.function_index) ]
300     with Not_found ->
301       raise (Failure ("undefined function " → fnname))
302     )
303 | "getCurve" ->
304     (try
305       expr (List.hd actuals) @
306       expr (List.hd (List.tl actuals)) @
307       [Js (StringMap.find fn name env.function_index) ]
308     with Not_found ->
309       raise (Failure ("undefined function " → fnname))
310     )
311 | "getCurveh" ->
312     (try
313       expr (List.hd actuals) @
314       expr (List.hd (List.tl actuals)) @
315       expr (List.hd (List.rev actuals)) @
316       [Js (StringMap.find fn name env.function_index) ]
317     with Not_found ->
318       raise (Failure ("undefined function " → fnname))
319     )
with Not_found ->
    raise (Failure ("undefined function " ^ fname))
)

| "setCurve" ->
  expr (Assign{
    (match (List.hd actuals) with
       | Id (s) -> s | _ -> "")
      Call("setCurve", actuals)))
| _ ->
  try
    (List.concat (List.map expr (List.rev actuals))) @
    [Jar (StringMap.find fname env.function_index)]
  with Not_found ->
  raise (Failure ("undefined function " ^ fname)))
|
Kexpr -> []

in rec stmt = function
  Block sl -> List.concat (List.map stmt sl)
  Expr e -> {
    let rec drpsize a = match a with
      | Assign(s,e) -> 1
      | _ -> sizeofexpr a in
    let rec drpe i ex =
      if i = drpsize ex then
        []
      else
        [Drp] @ drpe (i+1) ex in
    expr e @ drpe 0 e
  | Return e ->
    (let rec loadretval i exp sargs =
      let sz = sizeofexpr exp in
      if i = sz then
        []
      else
        [Sfp (−sargs + sz −2 −i)] @ [Drp] @ loadretval (i+1) exp sargs
        in
      expr e @ [Rta] @ (loadretval 0 e sz_formals) @
        [Rts (sz_formals − (sizeofexpr e) + 1)]
    | If (p, t, f) -> let t' = stmt t and f' = stmt f in
      expr p @ [Seq (2 + List.length t')] @
      t' @ [Bra (1 + List.length f')] @ f'
    | For (e1, e2, e3, b) ->
      stmt (Block[(Expr(e1); While(e2, Block[(b; Expr(e3))]))])
    | While (e, b) ->
      let b' = stmt b and e' = expr e in
      [Bra (1 + List.length b’)] @ b’ @ e’ @
      [Sne (−(List.length b’ + List.length e’))]
    in
    [Ent (max sz_locals sz_ret)] @(* Entry: allocate space for ← locals *)
    stmt (Block fdecl.body) @ (* Body *)
    [Rta; Lit 0; Rts sz_formals] (* Default = return 0 *)
  in
let env = {
  function_index = function_indexes;
  global_index = global_index;
  local_index = StringMap.empty} in

(* Code executed to start the program: Jsr main; halt *)
let entry_function = try
  [Gqr; Jsr (StringMap.find "main" function_indexes); Hlt]
  with Not_found -> raise (Failure (" no \"main\" function"))
in
(* Compile the functions *)
let func_bodies = entry_function ::
9.6 execute.ml
let execute_prog_prog =
let debug = false in
let stack = Array.make (1024+1024) 0
and globals = Array.make prog.numGlobals 0
and new_fp = ref 0 and new_pc = ref 0 in

let sleep n = Thread.join(Thread.create(Thread.delay)(float_of_int <-
n /. 1000.0)) in

let random n = Random.self_init(); Random.int n in

let drawcurve m =
  Graphics.moveto (stack.(m)) (stack.(n+1));
  Graphics.curveto ((stack.(m+2)), stack.(m+3)) ((stack.(m+4)), <-
  stack.(n+5)) ((stack.(n+6)), stack.(n+7)) in

let rec exec fp sp pc =
  if debug then
    print_endline (Bytecode.string_of_stmt prog.text.(pc)) "$ fp: "
    string_of_int fp
    "$ sp: "
    string_of_int sp
    "$ pc: "
    string_of_int pc "
    String.concat "(printstack 0 sp"
    (Array.to_list (Array.sub stack 0 300)))
  else ();

match prog.text.(pc) with
  | Ugr -> Graphics.open_graph "1440x900"; exec fp (sp) (pc+1)
  | Lit i -> stack.(sp) <- i; exec fp (sp+1) (pc+1)
  | Drp -> exec fp (sp-1) (pc+1)
  | Bin sp -> let op1 = stack.(sp-2) and op2 = stack.(sp-1) in
    stack.(sp-2) <- (let boolean i = if i then 1 else 0 in
                     match op with
                     | Add -> op1 + op2
                     | Sub -> op1 - op2
                     | Mul -> op1 * op2
                     | Div -> op1 / op2
                     | Eqal -> boolean (op1 == op2)
                     | Nqeq -> boolean (op1 != op2)
                     | Less -> boolean (op1 < op2)
                     | Leq -> boolean (op1 <= op2)
                     | Greater -> boolean (op1 > op2)
                     | Geq -> boolean (op1 >= op2) ;
    exec fp (sp+1) (pc+1)
  | Lod i -> stack.(sp) <- globals.(i) ; exec fp (sp+1) (pc+1)
  | Str i -> globals.(i) <- stack.(sp-1) ; exec fp sp (pc+1)
  | Lfp i -> stack.(sp) <- stack.(fp+i) ; exec fp (sp+1) (pc+1)
  | Sfp i -> stack.(fp+i) <- stack.(sp-1) ; exec fp sp (pc+1)
  | Ind i -> stack.(sp) <- stack.(fp-stack.(fp-i-2)-i-2) ;
             exec fp (sp+1) (pc+1)
  | Ins i -> stack.(fp-stack.(fp-i-2)-i-2) <- stack.(sp-1) ;
             exec fp sp (pc+1)
  | Jsr(-1) -> print_endline (string_of_int stack.(sp-1));
  | exec fp sp (pc+1)
  | Jsr(-2) -> for j = 1 to stack.(sp-1) do
  |   ignore(drawcurve (sp-1 - (j * 8)))
  | done;
  | exec fp sp (pc+1)
  | Jsr(-3) -> sleep stack.(sp-1);
  | exec fp (sp) (pc+1)
  | Jsr(-4) -> Graphics.clear_graph ();
  | exec fp (sp+1) (pc+1)
  | Jsr(-5) -> stack.(sp-1) <- random stack.(sp-1);
             exec fp (sp) (pc+1)
  | Jsr i -> stack.(sp) <- pc + 1 ; exec fp (sp+1) i
  | Ext i -> stack.(sp) <- sp ; exec sp (sp+i+1) (pc+1)
  | Rta -> ignore(new_fp := stack.(fp));
  | ignore(new_pc := stack.(fp+1));
exec fp (sp) (pc+1)
| Rts i  -> ( * let new_fp = stack.(fp) and new_pc = stack.(fp-1) in
|   *)
|   (*stack.(fp-1) <- stack.(sp-1); *)
|   exec !new_fp (fp-1) !new_pc
| Beq i  -> exec fp (sp-1) (pc + if stack.(sp-1) = 0 then i else ←
| 1)
| Bne i  -> exec fp (sp-1) (pc + if stack.(sp-1) != 0 then i else ←
| 1)
| Bra i  -> exec fp sp (pc+i)
| Blt    -> Graphics.close_graph ()
in exec 0 0 0

9.7 semantic.ml

open Ast

module NameMap = Map.Make(struct
  type t = string
  let compare x y = Pervasives.compare x y
end)

exception ReturnException of curvet * var_decl NameMap.t

(* Main entry point: check a program *)
let rec check_layer cvs env count =
  let locals, globals = env in
  match cvs with
  | [] -> true
  | hd :: tl ->
    if count < 11 then
      if NameMap.mem hd locals && (NameMap.find hd locals).t = ←
        Curvet then
        check_layer tl env (count+1)
      else if NameMap.mem hd globals && (NameMap.find hd globals).t ←
        Curvet then
        check_layer tl env (count+1)
      else false
    else raise (Failure ("Too many curves in a layer!"))

let check_semantic (vars, funcsl =
  (* Put function declarations in a symbol table *)
  let func_decls = List.fold_left
    (fun funcs fdecl -> NameMap.add fdecl.fname fdecl funcsl)
    NameMap.empty funcsl
  in

  (* Invoke a function and return an updated global symbol table *)
  let rec call fdecl actuals globals checked =
    (* Evaluate an expression and return (value, updated environment) *)
    rec eval env = function
      | Literal(i) -> Literall, env;
      | Dotop(var, op, e) ->
        let (locals, globals) = env in
        if op = "getX" || op = "getY" then
          if NameMap.mem var locals then
            if (NameMap.find var locals).t = Pointt then

if List.length e = 0 then
  Literal, env
else
  raise (Failure ("Invalid parameter for ""op"))
else
  raise (Failure (var"" is not a Point")
else if NameMap.mem var globals then
  if (NameMap.find var globals).t = Pointt then
    if List.length e = 0 then
      Literal, env
else
  raise (Failure ("Invalid parameter for ""op"))
else
  raise (Failure (var"" is not a Point")
else
  raise (Failure ("Undeclared variable ""var")
else if op = "setX" || op = "setY" then
  if NameMap.mem var locals then
    if (NameMap.find var locals).t = Pointt then
      if List.length e = 1 then
        let 1, (locals, globals) = eval env (List.hd e) in
        if 1 = Literalt then
          Literal, env
else
  raise (Failure ("The parameter should be integer for ""op")
else
  raise (Failure ("Invalid parameter for ""op")
else
  raise (Failure (var"" is not a Point")
else
  raise (Failure ("Undeclared variable ""var")
else if op = "getPoint" then
  if NameMap.mem var locals then
    if (NameMap.find var locals).t = Curve then
      if List.length e = 1 then
        let 1, (locals, globals) = eval env (List.hd e) in
        if 1 = Literalt then
          Pointt, env
else
  raise (Failure ("Parameter should be 0-3 for ""op")
else
  raise (Failure ("Invalid parameter for ""op")
else
  raise (Failure (var"" is not a Curve")
else if NameMap.mem var globals then
  if (NameMap.find var globals).t = Curve then
    if List.length e = 1 then
      let 1, (locals, globals) = eval env (List.hd e) in
      if 1 = Literalt then
        Pointt, env
else
  raise (Failure ("Parameter should be 0-3 for ""op")
else
  raise (Failure ("Invalid parameter for ""op")
else
raise (Failure ("{} is not a Curve"))
else
raise (Failure ("{} is not a Curve"))
else if op = "setPoint" then
if NameMap.mem var locals then
if (NameMap.find var locals).t = Curvet then
if List.length e = 2 then
let 11, (locals, globals) = eval env (List.hd e) in
let 12, (locals, globals) = eval env (List.nth e 1) in
if 11 = Point then
if 12 = Point then
else
raise (Failure ("{} should be a Point for " "op"))
else if NameMap.mem var globals then
if (NameMap.find var globals).t = Curvet then
if List.length e = 2 then
let 11, (locals, globals) = eval env (List.hd e) in
let 12, (locals, globals) = eval env (List.nth e 1) in
if 11 = Literalt then
if 12 = Point then
else
raise (Failure ("{} should be a Point for " "op"))
else
raise (Failure ("{} should be 0-3 for " "op"))
else if NameMap.mem var globals then
if (NameMap.find var globals).t = Literalt then
if List.length e = 1 then
let 1, (locals, globals) = eval env (List.hd e) in
if 1 = Literalt then
Curvet, env
else
raise (Failure ("{} is not a Literalt"))
else
raise (Failure ("{} is not a Curve"))
else if NameMap.mem var locals then
if (NameMap.find var locals).t = Literalt then
if List.length e = 1 then
let 1, (locals, globals) = eval env (List.hd e) in
if 1 = Literalt then
Curvet, env
else
raise (Failure ("{} is not a Literalt"))
else
raise (Failure ("{} is not a Curve"))
else if NameMap.mem var globals then
if (NameMap.find var globals).t = Literalt then
if List.length e = 1 then
let 1, (locals, globals) = eval env (List.hd e) in
if 1 = Literalt then
Curvet, env
else
raise (Failure ("{} is not a Literalt"))
else
raise (Failure ("{} should be integer for " "op"))
else
raise (Failure ("{} is not an Integer"))
else
raise (Failure ("{} is not a Curve"))
else if NameMap.mem var locals then
if (NameMap.find var locals).t = Curvet then
if List.length e = 1 then
let 1, (locals, globals) = eval env (List.hd e) in
if 1 = Curvet then
else
raise (Failure ("{} is not a Curvet"))
else
  raise (Failure ("Undeclared variable "^var))
else if op = "setCurve" then
  if NameMap.memo var locals then
    if (NameMap.find var locals).t = Layert then
      if List.length e = 2 then
        let 11, (locals, globals) = eval env (List.hd e) in
        let 12, (locals, globals) = eval env (List.nth e 1) in
        if 11 = Literal then
          if 12 = Curvet then
            Curvet, env
          else
            raise (Failure ("The 2nd parameter should be a Curve ← for "^op))
        else
          raise (Failure ("The 1st parameter should be an integer ← for "^op))
      else
        raise (Failure ("Invalid parameter for "^op))
      else
        raise (Failure ("var" ^ is not a Layer"))
      else if NameMap.memo var globals then
        if (NameMap.find var globals).t = Layert then
          if List.length e = 2 then
            let 11, (locals, globals) = eval env (List.hd e) in
            let 12, (locals, globals) = eval env (List.nth e 1) in
            if 11 = Literal then
              if 12 = Curvet then
                Curvet, env
              else
                raise (Failure ("The 2nd parameter should be a Curve ← for "^op))
            else
              raise (Failure ("The 1st parameter should be an integer ← for "^op))
          else
            raise (Failure ("Invalid parameter for "^op))
        else
          raise (Failure ("var" ^ is not a Layer"))
      else
        raise (Failure ("Undeclared variable "^var))
else if op = "getSize" then
  if NameMap.memo var locals then
    if (NameMap.find var locals).t = Layert then
      if List.length e = 0 then
        Literal, env
      else
        raise (Failure ("Invalid parameter for "^op))
      else
        raise (Failure ("var" ^ is not a Layer"))
    else if NameMap.memo var globals then
      if (NameMap.find var globals).t = Layert then
        if List.length e = 0 then
          Literal, env
        else
          raise (Failure ("Invalid parameter for "^op))
        else
          raise (Failure ("var" ^ is not a Layer"))
    else
      raise (Failure ("Undeclared variable "^var))
else
  raise (Failure ("Undeclared dot operation "^op))
let 11, env = eval env e1 in
let 12, env = eval env e2 in
let 13, env = eval env e3 in
let 14, env = eval env e4 in
let 15, env = eval env e5 in
let 16, env = eval env e6 in
let 17, env = eval env e7 in
let 18, env = eval env e8 in
if List.map ((fun(x) -> if x = Literalt then 1 else 0) [1;1;1;1;1;1;1;1]) then
  Curvet, env
else raise (Failure ("Invalid curve definition."))

let 11, env = eval env e1 in
let 12, env = eval env e2 in
if 11 = Literalt && 12 = Literalt then
  Pointt, env
else raise (Failure ("Invalid point definition"))

let 11, env = eval env e1 in
let 12, env = eval env e2 in
if 11 = Literalt && 12 = Literalt then
  Layert, env
else raise (Failure ("Invalid layer definition"))

let 11, env = eval env e1 in
let 12, env = eval env e2 in
if 11 = Literalt && 12 = Literalt then
  Literalt, env
else raise (Failure ("Parameter should be integer for binary operation"))

let 1, (locals, globals) = eval env e in
if NameMap.mem var locals then
  (NameMap.find var locals).t, env
else if NameMap.mem var globals then
  (NameMap.find var globals).t, env
else raise (Failure ("Undeclared identifier \" var\")

let 11, env = eval env e1 in
let 12, env = eval env e2 in
if 11 = Literalt && 12 = Literalt then
  Literalt, env
else raise (Failure ("Parameter should be integer for binary operation"))

Call("draw", e) ->
if List.length e = 1 then
  let 1, env = eval env (List.hd e) in
  if 1 = Layert then
    (Literate, env)
  else raise (Failure ("The parameter of draw() function should be Layert))
else raise (Failure ("Invalid parameter for draw()"))

Call("print", e) ->
if List.length e = 1 then
  let 1, env = eval env (List.hd e) in
  if 1 = Literalt then
    (Literate, env)
  else raise (Failure ("The parameter of print() function should be integer"))
else raise (Failure ("Invalid parameter for print()"))

Call("pause", e) ->
if List.length e = 1 then
  let 1, env = eval env (List.hd e) in
  if 1 = Literalt then
    (Literate, env)
  else raise (Failure ("The parameter of pause() function should be integer"))
else raise (Failure ("Invalid parameter for pause()"))
<table>
<thead>
<tr>
<th>Call(&quot;random&quot;, e) -&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>if List.length e == 1 then</td>
</tr>
<tr>
<td>let l, env = eval env (List.hd e) in</td>
</tr>
<tr>
<td>if l = Literal then</td>
</tr>
<tr>
<td>(Literal, env)</td>
</tr>
<tr>
<td>else raise (Failure (&quot;The parameter of random() function should be integer&quot;).)</td>
</tr>
<tr>
<td>else raise (Failure (&quot;Invalid parameter for random()&quot;))</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>if List.length e == 0 then</td>
</tr>
<tr>
<td>(Literal, env)</td>
</tr>
<tr>
<td>else raise (Failure (&quot;Invalid parameter for clear()&quot;))</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>let fdecl =</td>
</tr>
<tr>
<td>try NameMap.find f func.decls;</td>
</tr>
<tr>
<td>with Not_found -&gt; raise (Failure (&quot;Undefined function &quot; f))</td>
</tr>
<tr>
<td>in</td>
</tr>
<tr>
<td>let actuals, env = List.fold_left</td>
</tr>
<tr>
<td>(fun (actuals, env) actual -&gt;</td>
</tr>
<tr>
<td>let v, env = eval env actual in v :: actuals, env)</td>
</tr>
<tr>
<td>([], env) (List.rev actuals)</td>
</tr>
<tr>
<td>in</td>
</tr>
<tr>
<td>let (locals, globals) = env in</td>
</tr>
<tr>
<td>try</td>
</tr>
<tr>
<td>if NameMap.mem f checked then</td>
</tr>
<tr>
<td>(NameMap.find f func.decls).return, (locals, globals)</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>let globals = call fdecl actuals globals (NameMap.add f l &lt;-&gt; checked)</td>
</tr>
<tr>
<td>in fdecl.return, (locals, globals)</td>
</tr>
<tr>
<td>with ReturnException(v, globals) -&gt;</td>
</tr>
<tr>
<td>if v = fdecl.return then</td>
</tr>
<tr>
<td>v, (locals, globals)</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>raise (Failure (&quot;Return type mismatch for function &quot; f))</td>
</tr>
<tr>
<td>in</td>
</tr>
<tr>
<td>(* Execute a statement and return an updated environment *)</td>
</tr>
<tr>
<td>let rec exec env = function</td>
</tr>
<tr>
<td>Block(stmts) -&gt; List.fold_left exec env stmts</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>if e1, e2, e3, s) -&gt;</td>
</tr>
<tr>
<td>let l, env = eval env e in</td>
</tr>
<tr>
<td>if l = List.fold_left</td>
</tr>
<tr>
<td>(Literal, List.fold_left exec env [s1;s2])</td>
</tr>
<tr>
<td>else raise (Failure (&quot;Invalid conditional statement.&quot;))</td>
</tr>
<tr>
<td>while (e, s) -&gt;</td>
</tr>
<tr>
<td>let l, env = eval env e in</td>
</tr>
<tr>
<td>if l = List.fold_left</td>
</tr>
<tr>
<td>exec env s</td>
</tr>
<tr>
<td>else raise (Failure (&quot;Invalid conditional statement.&quot;))</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>let _, env = eval env e1 in</td>
</tr>
<tr>
<td>let l, env = eval env e2 in</td>
</tr>
<tr>
<td>if l = List.fold_left</td>
</tr>
<tr>
<td>(Literal, exec env s) e3 in</td>
</tr>
<tr>
<td>exec env s</td>
</tr>
<tr>
<td>else raise (Failure (&quot;Invalid conditional statement.&quot;))</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>let v, (locals, globals) = eval env e in</td>
</tr>
<tr>
<td>ReturnException(v, globals))</td>
</tr>
<tr>
<td>in</td>
</tr>
<tr>
<td>(* Enter the function: bind actual values to formal arguments *)</td>
</tr>
<tr>
<td>let locals =</td>
</tr>
<tr>
<td>try List.fold_left2</td>
</tr>
<tr>
<td>(fun locals formal actual -&gt; if formal.t = actual then</td>
</tr>
</tbody>
</table>
365  NameMap.add formal.name { t = ↔
366  formal.t; name = formal.name; ↔
367  value = {0} } locals
368  else raise (Failure ("Type mismatch ←
369  for parameter " formal.name " →
370  in function " fdecl.fname))
371  NameMap.empty fdecl.formals actuals
372  with Invalid_argument(_) ->
373  raise (Failure ("wrong number of arguments passed to " fdecl.fname←
374  ))
375  in
376  let locals = List.fold_left
377  (fun locals local -> NameMap.add local.name { t = local.t; name = ↔
378  local.name; value = local.value } locals) locals fdecl.locals
379  in
380  (* Execute each statement in sequence, return updated global ←
381  symbol table *)
382  let env = List.fold_left exec (locals, globals) fdecl.body
383  in
384  if fdecl.fname = "main" then snd env
385  else raise (Failure ("No return statement for function " fdecl←
386  fname))
387  (* Run a program: initialize global variables to 0, find and run "→
388  main" *)
389  in
390  let globals = List.fold_left
391  (fun globals vdecl -> NameMap.add vdecl.name { t = vdecl.t; name = ↔
392  vdecl.name; value = vdecl.value } globals) NameMap.empty ←
393  vars
394  in
395  try
396  ignore (call (NameMap.find "main" func_decls) [] globals NameMap←
397  empty)
398  with Not_found -> raise (Failure ("did not find the main() function"←
399  ))

9.8 curve.ml

open Parsing
open Lexing

type action = Bytecode | Compile | Semantic | Bypass | NoStl

let read_file filename =
let lines = ref [] in
let chan = open_in filename in
try
  while true; do
    lines := input_line chan :: !lines
    done;
  with End_of_file ->
    close_in chan;
  List.rev !lines;;

let check prog = Semantic.check_semantic prog in
let argglength = Array.length Sys.argv in
let action = if arglengtgh > 2 then
  Listassoc Sys.argv.(1) [(
  "-b", Bytecode);
  ("-c", Compile);
  ("-y", Bypass);
  ("-m", NoStl);
  ("-s", Semantic)]
else Compile in
let lexbuf =
match action with
| NoStl ->
  Lexing.from_string
  (String.concat "\n"
   (read_file Sys.argv.(arglength - 1) ))
| _ ->
  Lexing.from_string
  (String.concat "\n"
   (read_file Sys.argv.(arglength - 1) @ read_file "stdlib/stl.cv")) in
let program = try Parser.program Scanner.token lexbuf
with Parse_error -> raise (Failure("Syntax error in program at 
  string_of_int((Parsing.symbol_start_pos()).pos_cnum))) in
match action with
| NoStl ->
  check program;
  let listing =
    Bytecode.string_of_prog (Compile.translate program)
  in
  print_endline listing
| Bypass ->
  Execute.execute_prog (Compile.translate program)
| Bytecode ->
  check program;
  let listing =
    Bytecode.string_of_prog (Compile.translate program)
  in
  print_endline listing
| Compile ->
  check program;
  Execute.execute_prog (Compile.translate program)
| Semantic ->
  check program

9.9 interpret.ml

open Ast
module NameMap = Map.Make(struct
type t = string
let compare x y = Pervasives.compare x y
end)
exception ReturnException of int list * int NameMap.t
let rec transform mat00 mat01 mat10 mat11 = function
| [] -> []
| [x] -> [x]
| x :: y :: t ->
  (x * mat00 + y * mat01)
:::
  (x * mat10 + y * mat11)
:::
  transform mat00 mat01 mat10 mat11 t
let explode s =
  let rec exp i l =
    if i < 0 then l else exp (i - 1) (s.[i] :: l) in
  exp (String.length s - 1) []
let implode l =
  let res = String.create (List.length l) in
  let rec imp i = function
  | [] -> res
  | c :: l -> res.[i] <- c; imp (i + 1) l in
  imp 0 l
let string_to_int_list s = List.map Char.code (explode s)@[−1]
let int_list_to_string l =
  let ascii_to_char = function |
  −1 -> ',' |
  x -> Char.chr x
in let rec char_to_string res temp = function |
  [] -> res |
  ['|': -> char_to_string ((implore (List.rev temp))::res) [] tl |
  x :: tl -> char_to_string (x::temp) tl |
  in List.rev (char_to_string [] [] (List.map ascii_to_char l))

(* Main entry point: run a program *)
let run (vars, func) =
  (* Put function declarations in a symbol table *)
  let func_decls = List.fold_left |
    (fun func fdecl -> NameMap.add fdecl.fname fdecl func) |
    NameMap.empty func
  in
  (* Invoke a function and return an updated global symbol table *)
  let rec call fdecl actuals globals =
    (* Evaluate an expression and return (value, updated environment) *)
    let rec eval env = function |
      Literal(i) -> [i], env |
      Dotop(var, op, [e]) ->
        let l1, (locals, globals) = eval env e in
        let v = List.hd l1 in
        if op = "nth" then
          [List.nth (NameMap.find var locals).value v], env |
        else if op = "getX" then
          let index = v + 2 in
          [List.nth (NameMap.find var locals).value index], env |
        else if op = "getY" then
          let index = v + 2 + 1 in
          [List.nth (NameMap.find var locals).value index], env |
        else raise (Failure ("undeclared dot operation "+ op))
    Curve(e1, e2, e3, e4, e5, e6, e7, e8) ->
      let l1, env = eval env e1 in
      let l2, env = eval env e2 in
      let l3, env = eval env e3 in
      let l4, env = eval env e4 in
      let l5, env = eval env e5 in
      let l6, env = eval env e6 in
      let l7, env = eval env e7 in
      let l8, env = eval env e8 in
      let v1 = List.hd l1 in
      let v2 = List.hd l2 in
      let v3 = List.hd l3 in
      let v4 = List.hd l4 in
      let v5 = List.hd l5 in
      let v6 = List.hd l6 in
      let v7 = List.hd l7 in
      let v8 = List.hd l8 in
      [v1; v2; v3; v4; v5; v6; v7; v8], env |
      Point(e1, e2) ->
        let l1, env = eval env e1 in
        let l2, env = eval env e2 in
        let v1 = List.hd l1 in
        let v2 = List.hd l2 in
        [v1; v2], env |
      Layer(cvs) ->
List.fold_left (fun result s -> result@string_to_int_list s) [] ←
cvs, env
| AddToList(cv, ly) ->
let locals, globals = env in
(NameMap.find ly locals).value@string_to_int_list cv, env
| Noexpr -> [1], env (* must be non-zero for the for loop ←
predicate *)
| Id(var) ->
let locals, globals = env in
if NameMap.neg var locals then
(NameMap.find var locals).value, env
else if NameMap.neg var globals then
((NameMap.find var globals)), env
else raise (Failure ("undeclared identifier " ^ var))
| Binop(e1, op, e2) ->
let l1, env = eval env e1 in
let l2, env = eval env e2 in
let v1 = List.hd l1 in
let v2 = List.hd l2 in
let boolean i = if i then 1 else 0 in
(match op with
Add -> [v1 + v2]
Sub -> [v1 - v2]
Mult -> [v1 * v2]
Div -> [v1 / v2]
Equal -> [boolean (v1 == v2)]
Neq -> [boolean (v1 != v2)]
Less -> [boolean (v1 < v2)]
Leq -> [boolean (v1 <= v2)]
Greater -> [boolean (v1 > v2)]
Geq -> [boolean (v1 >= v2)], env
| Assign(var, e) ->
let l, (locals, globals) = eval env e in
if NameMap.neg var locals then
1, (NameMap.add var { t = (NameMap.find var locals).t; name = ←
var; value = 1 }) locals, globals
else if NameMap.neg var globals then
1, (locals, NameMap.add var (List.hd l) globals)
else raise (Failure ("undeclared identifier " ^ var))
| Call("print", [e]) ->
let l, env = eval env e in
let v = List.hd l1 in
print_endline (string_of_int v);
[0], env
| Call("draw", [e]) ->
let l, env = eval env e in
print_endline "draw starts";
ignore (List.map (fun x -> print_endline (string_of_int x)) l);
print_endline "draw ends";
[0], env
| Call("pause", [e]) ->
let l, env = eval env e in
print_endline "pause starts";
ignore (List.map (fun x -> print_endline (string_of_int x)) l);
print_endline "pause ends";
[0], env
| Call("drawLayer", [e]) ->
let cvs = int_list_to_string l in
print_endline "draw layer starts";
ignore (List.map (fun x -> print_endline x) cvs);
print_endline "draw layer ends";
[0], env
| Call(f, actuals) ->
let fdecl = try NameMap.find f func_decls
with Not_found -> raise (Failure ("undefined function " ^ f))
in
let actuals, env = List.fold_left
  (fun (actuals, env) actual ->
    let v, env = env eval env actual in v :: actuals, env)
  ([], env) (List.rev actuals)
in
let (locals, globals) = env in
try
  let globals = call fdecl actuals globals
  in [0], (locals, globals)
with ReturnException(v, globals) -> v, (locals, globals)
in

(* Execute a statement and return an updated environment *)
let rec exec env = function
  | Expr(e) -> let _, env = eval env e in env
  | If(e, s1, s2) ->
    let l, env = eval env e in
    let v = List.hd l in
    exec env (if v != 0 then s1 else s2)
  | While(e, s) ->
    let rec loop env =
      if v != 0 then loop (exec env s) else env
    in loop env
  | For(e1, e2, e3, s) ->
    let _, env = eval env e1 in
    let rec loop env =
      if v != 0 then loop (exec env s) e3 in
    in loop env
  else
    env
  | Return(e) ->
    let v, (locals, globals) = eval env e in
    raise (ReturnException(v, globals))
in
(* Enter the function: bind actual values to formal arguments *)
let locals =
  try List.fold_left2
    (fun locals formal actual -> NameMap.add formal.name { t = formal.e
      t; name = formal.name; value = actual } locals)
  in NameMap.empty fdecl.formals actuals
with Invalid_argument(_) ->
  raise (Failure "wrong number of arguments passed to " ^ fdecl.fname ^
).)
in
(* Initialize local variables to 0 *)
let locals = List.fold_left
  (fun locals local -> NameMap.add local.name { t = local.t; name = local.name; value = local.value } locals) fdecl.locals
in
(* Execute each statement in sequence, return updated global symbol table *)
snd (List.fold_left exec (locals, globals) fdecl.body)
in
(* Run a program: initialize global variables to 0, find and run "main")
in let globals = List.fold_left
  (fun globalsvd ec1 actuals globals) NameMap.empty
    (List.map (fun x -> x.name vars)
  in try
    call (NameMap.bind "main" func_decls) [] globals
9.10 stl.cv

```c
int printp(Point p)
{
    print(p.getX());
    print(p.getY());
    return 0;
}

int printc(Curve c)
{
    int i;
    for(i = 0; i < 4; i = i+1)
    {
        printc(c.getPoint(i));
    }
    return 0;
}

int printl(Layer l)
{
    int sz;
    int i;
    Curve tmp;
    sz = l.getSize();
    for (i=0 ; i < sz; i = i+1)
    {
        tmp = l.getCurve(i);
        printc(tmp);
    }
    return 0;
}

int true()
{
    return 1;
}

int false()
{
    return 0;
}

int curveSize()
{
    return 4;
}

int maxLayerSize()
{
    return 10;
}

int nullI()
{
    return -1;
}

Point nullP()
{
```
return (-1, -1);
}

Curve nullC()
{
    return (-1, -1)(-1, -1)(-1, -1)(-1, -1);
}

Layer nullL()
{
    Curve c;
    Layer nullLayer;
    c = nullC();
    nullLayer = [c, c, c, c, c, c, c, c, c, c];
    return nullLayer;
}

int equalsP(Point p, Point q)
{
    if (p.getX() != q.getX()) {
        return false();
    }
    if (p.getY() != q.getY()) {
        return false();
    }
    return true();
}

int equalsC(Curve c, Curve d)
{
    int i;
    for (i = 0; i < curveSize(); i++) {
        if (equalsP(c.getPoint(i), d.getPoint(i)) == False) {
            return false();
        }
    }
    return true();
}

int equalsL(Layer l, Layer m)
{
    int i;
    int lsize;
    int msize;
    lsize = l.getSize();
    msize = m.getSize();
    if (lsize != msize) {
        return false();
    }
    for (i = 0; i < lsize; i++) {
        if (equalsC(l.getCurve(i), m.getCurve(i)) == False) {
            return false();
        }
    }
    return true();
}

Curve translateC(Curve c, int x, int y)
{
    int i;
    Point a;
    for (i = 0; i < 4; i = i + 1)
    {
        a = c.getPoint(i);
        }
```java
Curve Final Report December 19, 2012

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)(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  {
```
```java
Layer tri;
Curve s1;
Curve s2;
Curve s3;
s1 = lineP(p, q);
s2 = lineP(q, r);
s3 = lineP(r, p);
tri = [s1, s2, s3];
return tri;
}

Layer triangleXY(int x1, int y1, int x2, int y2, int x3, int y3) {
    Layer tri;
    Point p;
    Point q;
    Point r;
    p = (x1, y1);
    q = (x2, y2);
    r = (x3, y3);
    tri = triangleP(p, q, r);
    return tri;
}

Layer rectangleXY(int x, int y, int height, int width) {
    Curve s1;
    Curve s2;
    Curve s3;
    Curve s4;
    Layer rect;
    s1 = lineXY(x, y, x+width, y);
    s2 = lineXY(x+width, y, x+width, y+height);
    s3 = lineXY(x+width, y+height, x, y+height);
    s4 = lineXY(x, y+height, x, y);
    rect = [s1, s2, s3, s4];
    return rect;
}

Layer rectangleP(Point p, int height, int width) {
    Layer r;
    int x;
    int y;
    x = p.getX();
    y = p.getY();
    r = rectangleXY(x, y, height, width);
    return r;
}

int isLine(Curve c) {
    Point end1;
    Point middle1;
    Point middle2;
    Point end2;
    end1 = c.getPoint(0);
    middle1 = c.getPoint(1);
    middle2 = c.getPoint(2);
    end2 = c.getPoint(3);
    if (equalsP(end1, end2)) {
        return false();
    }
    return true();
}
```
if (equalsP(end1, middle1) == false) {
    if (equalsP(end2, middle1) == false) {
        return false;
    }
    if (equalsP(end1, middle2) == false) {
        return false;
    }
    return true;
}

int isRectangle(Layer r)
{
    Point p1;
    Point p2;
    Curve c1;
    Curve c2;
    Curve c3;
    Curve c4;
    int i;
    if (r.getSize() != 4)
        return 0;
    for (i = 0; i < 4; i++) {
        if (isLine(r.getCurve(i)) == false) {
            return false;
        }
    }
    c1 = r.getCurve(0);
    c2 = r.getCurve(1);
    c3 = r.getCurve(2);
    c4 = r.getCurve(3);
    p1 = c1.getP00()3);
    p2 = c2.getP00()3);
    if (equalsP(p1, p2) == false) {
        return false;
    }
    p1 = c2.getP00()3);
    p2 = c3.getP00()3);
    if (equalsP(p1, p2) == false) {
        return false;
    }
    p1 = c3.getP00()3);
    p2 = c4.getP00()3);
    if (equalsP(p1, p2) == false) {
        return false;
    }
    p1 = c4.getP00()3);
    p2 = c1.getP00()3);
    if (equalsP(p1, p2) == false) {
        return false;
    }
    return true();
}

Point getRectangleBase(Layer r)
{
    Curve c;
if (isRectangle(r) \equiv \text{false}) \{ \\
\quad \text{return nullP();} \\
\} \\
\quad c = r\text{.getCurve}(0); \\
\quad \text{return c\text{.getPoint}(0);} \\
\}

int getRectangleWidth(Layer r) 
{ 
\quad Curve c; 
\quad Point p1; 
\quad Point p2; 
\quad \text{if (isRectangle(r) \equiv \text{false}) \{ 
\quad \text{return nullI();} 
\} \\
\quad c = r\text{.getCurve}(0); 
\quad p1 = c\text{.getPoint}(0); 
\quad p2 = c\text{.getPoint}(3); 
\quad \text{return p2\text{.getX() \} \text{.getX}}() \text{.getX}() \text{.getX}()); 
\}

int getRectangleHeight(Layer r) 
{ 
\quad Curve c; 
\quad Point p1; 
\quad Point p2; 
\quad \text{if (isRectangle(r) \equiv \text{false}) \{ 
\quad \text{return nullI();} 
\} \\
\quad c = r\text{.getCurve}(1); 
\quad p1 = c\text{.getPoint}(0); 
\quad p2 = c\text{.getPoint}(3); 
\quad \text{return p2\text{.getY() \} \text{.getY}()} \text{.getY}()); 
\}

Layer fillRectangleL(Layer r, int lightness) 
{ 
\quad int i; 
\quad Curve line; 
\quad Layer lineLayer; 
\quad Layer rectangle; 
\quad Point left; 
\quad Point right; 
\quad Point top; 
\quad Point bottom; 
\quad Point p; 
\quad int height; 
\quad int width; 
\quad \text{if (isRectangle(r) \equiv \text{false}) \{ 
\quad \text{return nullL();} 
\} \\
\quad p = getRectangleBase(r); 
\quad \text{height = getRectangleHeight}(r); 
\quad \text{width = getRectangleWidth}(r); 
\quad \text{for (i = 0; i < height; i = i + lightness)} \{ 
\quad \text{left = (p\text{.getX()} + i);} 
\quad \text{right = (p\text{.getX()} + width)}; 
\quad \text{line = lineP(left, right);} 
\quad \text{lineLayer = [line];} 
\quad \text{draw(lineLayer);} 
\}
Curve Final Report

December 19, 2012

for (i = 0; i < width; i = i + lightness) {
    top = (p.getX()+i, p.getY());
    bottom = (p.getX()+i, p.getY()+height);
    line = lineP(top, bottom);
    lineLayer = [line];
    draw(lineLayer);
}

rectangle = rectangleP(p, height, width);
draw(rectangle);
return rectangle;

Layer fillRectangleP(Point p, int height, int width, int lightness) {
    int i;
    Curve line;
    Layer lineLayer;
    Layer rectangle;
    Point left;
    Point right;
    Point top;
    Point bottom;

    for (i = 0; i < height; i = i + lightness) {
        left = (p.getX()+i, p.getY() + i);
        right = (p.getX()+width, p.getY()+i);
        line = lineP(left, right);
        lineLayer = [line];
        draw(lineLayer);
    }

    for (i = 0; i < width; i = i + lightness) {
        top = (p.getX()+i, p.getY());
        bottom = (p.getX()+i, p.getY()+height);
        line = lineP(top, bottom);
        lineLayer = [line];
        draw(lineLayer);
    }

    rectangle = rectangleP(p, height, width);
    return rectangle;
}

Layer squareXY(int x, int y, int size) {
    Layer sq;
    sq = rectangleXY(x, y, size, size);
    return sq;
}

Layer squareP(Point p, int size) {
    Layer sq;
    int x;
    int y;
    sq = squareXY(x, y, size);
    return sq;
}

Layer quadP(Point p1, Point p2, Point p3, Point p4) {
    Layer quad;
    Curve s1;
    Curve s2;
    Curve s3;
    Curve s4;
}
Curve Final Report

Curve Final Report December 19, 2012

s1 = lineP(p1, p2);
s2 = lineP(p2, p3);
s3 = lineP(p3, p4);
s4 = lineP(p4, p1);
quad = [s1, s2, s3, s4];
return quad;

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*3/1000;
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);
bl = (x, r+y)(-ctrl+x, r+y)(-r+x, ctrl+y)(-r+x, y);
cir = [tr, br, bl, tl];
return cir;
}

Curve scaleC(Curve cv, int cx, int cy, int n, int d) {
    int i;
    Point p;
    Point m;
    for(i = 0; i < 4; i = i+1)
    {
        p = cv.getPoint(i);
        m.setX((p.getX()-cx)*n/d + cx);
        m.setY((p.getY()-cy)*n/d + cy);
        cv.setPoint(i, m);
    }
    return cv;
}

Layer scaleL(Layer l, int cx, int cy, int n, int d) {
    int sz;
    int i;
    Curve tmp;
    sz = l.getSize();
    for (i = 0; i < sz; i=i+1)
    {
        tmp = l.getCurve(i);
        tmp = scaleC(tmp, cx, cy, n, d);
        l.setCurve(i, tmp);
    }
    return l;
}

Curve rotateC(Curve cv, int cx, int cy, int s, int c, int d) {
    int i;
    Point p;
    Point m;
    for (i = 0; i < 4; i = i+1)
    {
        p = cv.getPoint(i);
    }
Curve Final Report  December 19, 2012

535 \texttt{m.setX((\texttt{p.getX()} - \texttt{cx}) \times c + (\texttt{p.getY()} - \texttt{cy}) \times s) + \texttt{cx} \times d) / d;}
536 \texttt{m.setY(((\texttt{p.getX()} - \texttt{cx}) \times (-s) + (\texttt{p.getY()} - \texttt{cy}) \times c) + \texttt{cy} \times d) / d;}
537 \texttt{cv.setPoint(i, n);}  
538 \texttt{return cv;}
539 \}
540 \}
541
542 \textbf{Layer rotateL(Layer l, int cx, int cy, int s, int c, int d)}
543 \{
544 \texttt{int sz;}
545 \texttt{Curve tmp;}
546 \texttt{sz = l.getSize();}
547 \texttt{for (i = 0; i < sz; i++)}
548 \{
549 \texttt{tmp = l.getCurve(i);}
550 \texttt{tmp = rotateC(tmp, cx, cy, s, c, d);}
551 \texttt{l.setCurve(i, tmp);}
552 \}
553 \texttt{return l;}
554 \}
555
556 \textbf{int exp(int x, int n)}
557 \{
558 \texttt{int i;}
559 \texttt{int acc;}
560 \texttt{acc = 1;}
561 \texttt{for (i = 0; i < n; i++)}
562 \{
563 \texttt{acc = acc \times x;}
564 \}
565 \texttt{return acc;}
566 \}

9.11 Makefile

\begin{verbatim}
OBJXS = ast.cmo parser.cmo scanner.cmo compile.cmo semantic.cmo ←
        bytecode.cmo execute.cmo curve.cmo
LIBS=$(WITHGRAPHICS) $(WITHUNIX) $(WITHTHREADS)
CONF=--I +threads
WITHUNIX =-lunix.cma -lunix -lunix -lunix
WITHSTR =-str.cma -lstr -lstr
WITHNUMS =-nums.cma -lnums -lnums
WITHTHREADS =threads.cma -lthreads -lthreads
WITHDBM =-dbm.cma -lthreads -lthreads -lthreads

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