Curve

Curve Ninjas

December 19, 2012

Ninjas

Shinobi

- Kun An
- John Chan
- David Mauskop
- Wisdom Omuya
- Zitong Wang

Overview

Simple, yet expressive

Overview

- 2D graphics and animations
- Minimal set of built-ins
- Easily tailored to more specific domains
- Static scoping, strongly-typed, call by value

Motivation

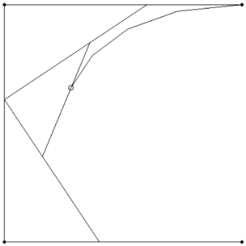
Bezier curves

Motivating Observation

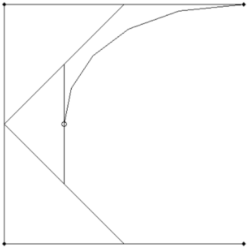
- All the geometric objects that form the building blocks of a graphics language generalize to Bezier curves
- A Bezier curve is defined by two “anchor” points and any number of “control” points (for us, two)

Bezier curve example

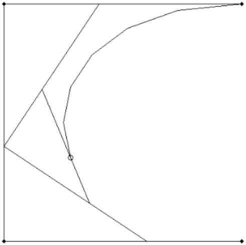
Screenshots from an animation written in Curve



$t = 4$



$t = 5$



$t = 6$

Basic syntax

```
// Declaration statements
Point p;
Curve c;
Layer l;
int i;

// Assignment statements
p = (x, y);
c1 = (p.getX(), y1)(x2, p.getY())(x3, y3)(x4, y4);
c2 = rectangleP(p, 100, 200);
l = [c1, c2];
```

Basic syntax

```
// Control flow
for (i = 0; i < 10; i++) { }
while (i < 10) {i++;}
if (i < 10) { } else { }

// Function declaration
Layer square(Point p, int size) { }

// Animation built-ins
draw(l);
pause(1000);
clear();
```

Example Program

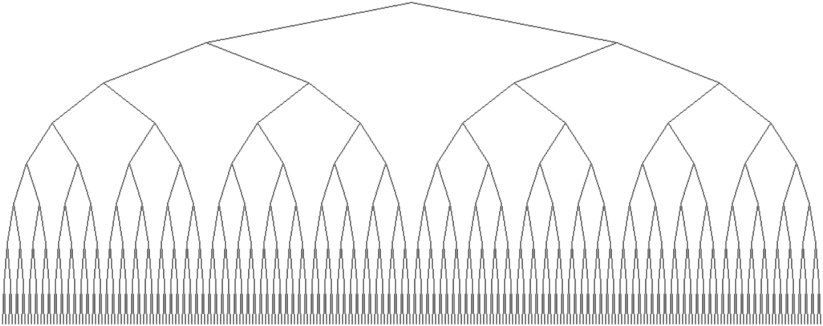
```
int drawTree(int x, int y, int n) {
  Curve left;
  Curve right;
  if (n == 0) return 1;

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

  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);
  return 1;
}
```

Result



Frontend

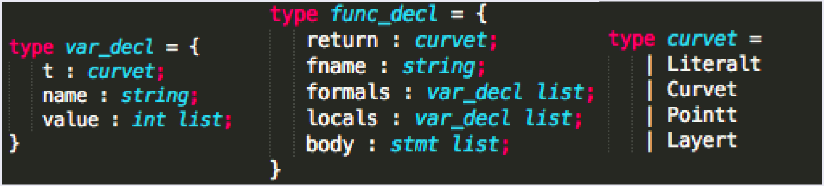
What is included

- Scanner
- Parser
- AST
- Interpreter
- Semantic checker

Frontend

AST

- Variable Declaration
- Function Declaration



```
type var_decl = {
    t : curvet;
    name : string;
    value : int list;
}
```

```
type func_decl = {
    return : curvet;
    fname : string;
    formals : var_decl list;
    locals : var_decl list;
    body : stmt list;
}
```

```
type curvet =
    | Literalt
    | Curvet
    | Pointt
    | Layert
```

Frontend

Interpreter

- Not part of final deliverable.
- Useful testing tool when implementing the scanner, parser, and AST.
- Easier to implement and modify compared with the compiler.

Frontend

Semantic checker

- All kinds of type mismatches including variable assignment, LHS & RHS of an assignment statement or binary operation, parameter of user-defined function, built-in function, standard library function, etc.
- Number of parameters mismatched with the definition of the function.
- Return type mismatches with the definition of the function's return type.
- Undeclared variables or functions.
- Lack of return statement for user-defined functions.

Backend

Bytecode

- Rta - Prepares for a Return
- Ind/Ins - Indirect Load/Store
- Ogr - Open Graph

Backend

Compiler

- Creates bytecode
- Record keeping - offsets for variables (global/local) and functions, return types, enumerates bytecode so subroutines have targets
- Built-ins

Backend

Execute - Bytecode interpreter

- Performs actions indicated by bytecode
- Initializes Graphics environment
- Maintains a stack
- Due to small size of instruction set, this code is terse

Lessons Learned

- Big groups aren't so bad
- Pacing is key
- Test, test, and test again