SFAL: Simple Flash Animation Language

Antonio Cruz
adc2104@columbia.edu

December 18, 2007

Contents

Appendix: Antlr and Java files

1. grammar-as-semant.g
2. Debug.java
3. Expr.java
4. Meta.java
5. SFAL.java
6. SymbolTable.java
7. Translator.java
8. Validator.java

1. An Introduction to SFAL

SFAL is a language for creating simple animations in Adobe Flash. As such, a brief introduction to Flash is necessary.

1.1 Background

Adobe Flash is the most pervasive rich media software platform for the Internet, "reaching 99% of Internet-enabled desktops in mature markets as well as a wide range of devices."[1]

Flash is particularly strong in the area of vector graphics -- lightweight, mathematically defined graphics that are well-suited for the bandwidth constraints of the Internet.

> "**Vector graphics** describe images using lines and curves, called vectors, that also include color and position properties. Each vector uses mathematical calculations, instead of bits, to describe the shape, which allows them be scaled without degrading in quality."[2]

In the past, creating Flash animations required use of the Flash Authoring Tool, a GUI application where various palettes of tools allowed users to create and manipulate objects on a drawing stage. Animations could be created via a timeline which sequenced objects into a series of frames.

Later, ActionScript, an ECMAScript[3] based scripting language for Flash, provided the ability to create animations dynamically in code, in conjunction with the GUI application and timeline.

[1] http://www.adobe.com/products/player_census/flashplayer/

[2] http://livedocs.adobe.com/flash/8/main/wwhelp/wwhimpl/common/html/wwhelp.htm?context=LiveDocs_Parts&file=00001702.html

[3] http://www.ecma-international.org/publications/standards/Ecma-262.htm

With the release of ActionScript 3.0 for Flash 9, the creation of graphical objects and animations in Flash may now be done completely in code, without the authoring GUI and timeline at all. Adobe provides an ActionScript 3.0 compiler, **`mxmlc`**[4], that takes ActionScript `.as` files and outputs a `.swf` file, which can be viewed in a Flash enabled browser or other `.swf` player.

However, ActionScript 3.0 is a fairly large and complex language with a considerable learning curve for programming novices.

For instance, the ActionScript required to create a colored circle with a radius of 10 pixels and place it on the drawing stage at x,y position 150, 150 requires 13 lines of code, as seen below.

```
package {
     import flash.display.Sprite;
     public class Circles extends Sprite {
            public function Circles() {
            var circle:Sprite = new Sprite();
            circle.graphics.beginFill(0x0000ff,100);
            circle.graphics.drawCircle(0,0,10);
            circle.graphics.endFill();
            circle.x = 150;
            circle.y = 150;
            }
     }
     }
```

SFAL does the same thing in 3 lines of code:

```
circle = makecircle(10, "yellow");
circle = setx(circle, 150);
circle = sety(circle, 150);
```

[4] http://livedocs.adobe.com/flex/2/docs/wwhelp/wwhimpl/common/html/wwhelp.htm?context=LiveDocs_Parts&file=00001481.html

1. 2 Goals

It is the goal of SFAL to be a simple language that allows users to quickly and easily create shapes, text, and groups of objects in a drawing area, and animate them. In addition to outputting `.as` files to be compiled by mxmlc into `.swf` files, the SFAL compiler will also output a corresponding `.html` to simplify publishing the compiled flash movie to the Internet or viewing locally in a Flash-enabled web browser.

2. SFAL Tutorial

SFAL uses simple syntax with imperative function calls to create text and graphics in the drawing area. Four simple programs will illustrate the ease-of-use and functionality of SFAL.

2.1 Tutorial Program One: "Hello World."

Here is the SFAL "Hello World" program.

```
hello = maketext("Hello World.");

hello = settextx(hello, 200);

hello = settexty(hello, 150);
```

Save this text in a document called "helloworld.sfal" and run through the SFAL compiler:

```
$ java SFAL HelloWorld helloworld.sfal
```

The first argument to the SFAL compiler is the basename you wish to give your `.as`, `.swf` and `.html` files. If you `ls HelloWorld.*` you'll see that SFAL has output an ActionScript file as it's Intermediate Representation. SFAL has also output a `HelloWorld.html` file, since it is easiest to view Flash movies in a Flash-enabled web browser such as Mozilla FireFox.

```
$ ls HelloWorld.*

HelloWorld.as     HelloWorld.html
```

The next step is to run the HelloWorld.as file through the Adobe `mxmlc` compiler.

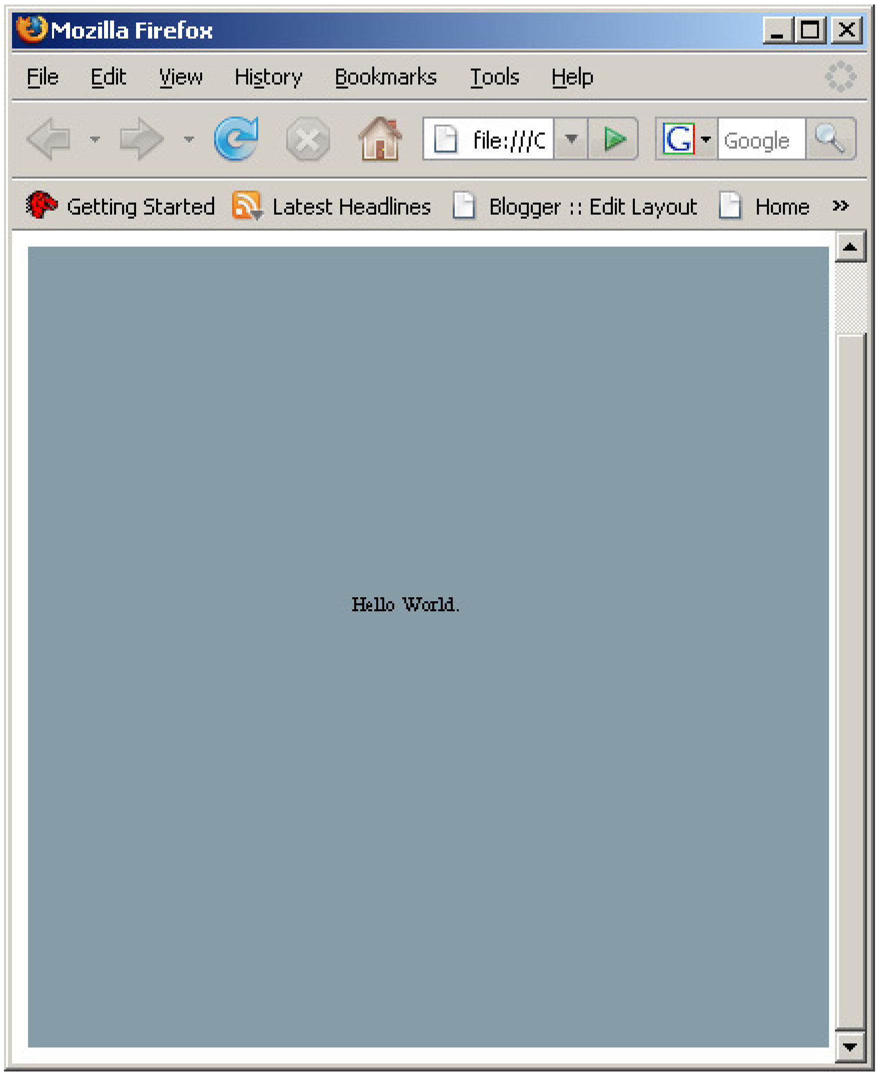
```
        $ mxmlc HelloWorld.as

Loading configuration file C:\flex\frameworks\flex-config.xml
```

```
C:\Documents and
Settings\admin\Desktop\rollback\plt\HelloWorld.swf (716 bytes)
```

Now, if you `ls HelloWorld.*` again you'll see that `mxmlc` has taken the `HelloWorld.as` file and compiled the `HelloWorld.swf` file, which can be loaded via the `HelloWorld.html` file into your Flash-enabled browser. Alternatively, the `HelloWorld.swf` file can just be drag-and-dropped into your browser.

```
$ ls HelloWorld.*

HelloWorld.as  HelloWorld.html  HelloWorld.swf
```

Here is the output:



Hello World.

2.2 Tutorial Program Two: "Three shapes in a row."

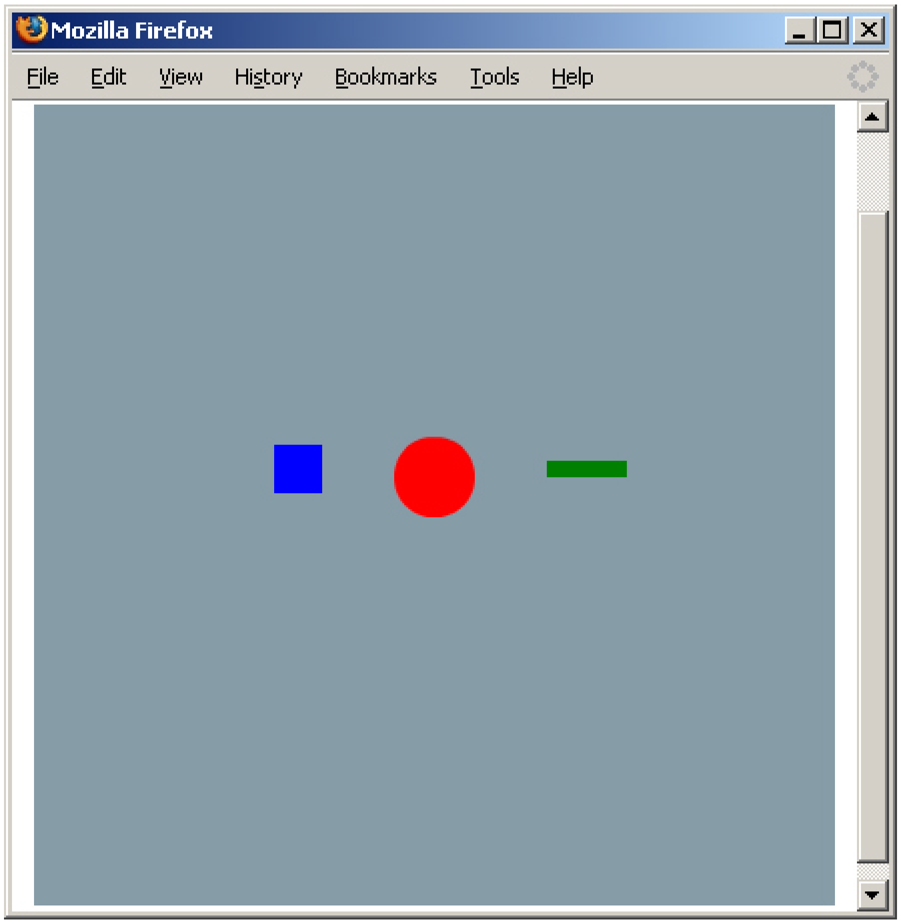
Here is a program to create and line up the main graphic shapes in SFAL:

```
---------------------------------------------------

square = makesquare(30, "blue");
square = setx(square, 150);
square = sety(square, 150);

circle = makecircle(25, "red");
circle = setx(circle, 250);
circle = sety(circle, 170);

rect = makerect(50, 10, "green");
rect = setx(rect, 320);
rect = sety(rect, 160);

---------------------------------------------------
```

Follow the same steps now from Tutorial Program 1. Save your `.sfal` file, e.g. as `shapes.sfal`, and run it through the SFAL compiler. Run the `Shapes.as` file through the `mxmlc` compiler, then load the `Shapes.html` file in your browser.

```
$ more shapes.sfal
square = makesquare(30, "blue");
square = setx(square, 150);
square = sety(square, 150);

circle = makecircle(25, "red");
circle = setx(circle, 250);
circle = sety(circle, 170);

rect = makerect(50, 10, "green");
rect = setx(rect, 320);
rect = sety(rect, 160);

$ java SFAL Shapes shapes.sfal

$ mxmlc Shapes.as
Loading configuration file C:\flex\frameworks\flex-
config.xml
C:\Documents and
Settings\admin\Desktop\rollback\plt\Shapes.swf (862 bytes)

$ ls Shapes.*
Shapes.as  Shapes.html  Shapes.swf
```

Here is the output:

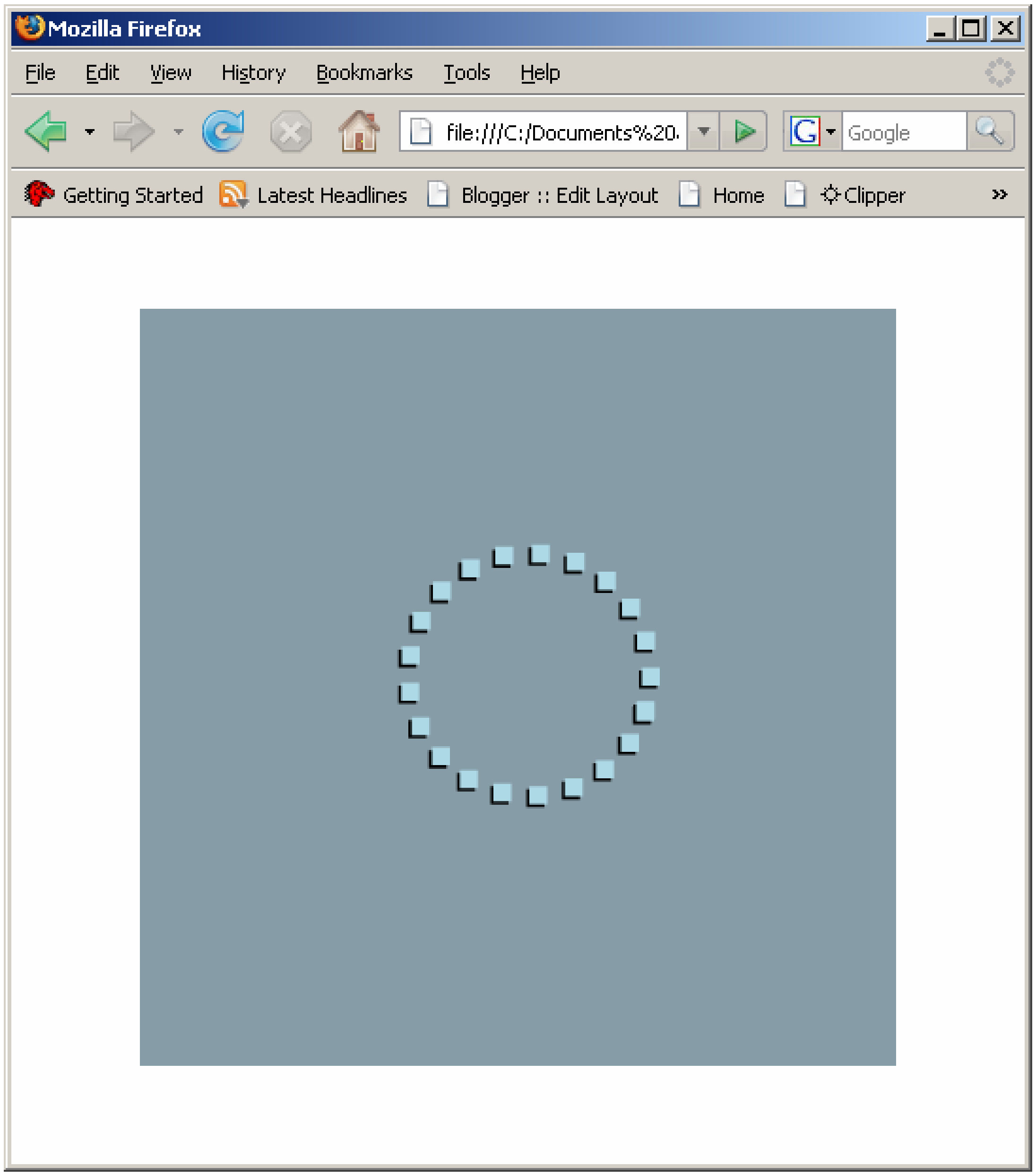


2.3 Tutorial Program Three: "A circle of squares."

SFAL has the ability to loop over a block of code. Here is a program that loops to create a group of squares in a circular pattern, with a drop shadow:

```
---------------------------------------------------

loop(1,23,k) {
      s = makesquare(12, "black");
      s2 = makesquare(12, "lightblue");

      // space the squares apart
      a = k * .3;

      cx = cos(a);
      sy = sin(a);

      // coordinates
      cx = cx * 80; cx = cx + 245;
      sy = sy * 80; sy = sy + 175;

      // place the first sprite
      s = setx(s,cx);
      s = sety(s,sy);

      cx = cx + 2;
      sy = sy - 2;

      // place the second sprite
      s2 = setx(s2,cx);
      s2 = sety(s2,sy);
}

---------------------------------------------------

$ java SFAL Loop loop.sfal

$ mxmlc Loop.as
Loading configuration file C:\flex\frameworks\flex-
config.xml
C:\Documents and
Settings\admin\Desktop\rollback\plt\Loop.swf (891 bytes)

$ ls Loop.*
Loop.as  Loop.html  Loop.swf
```

Here is the output:



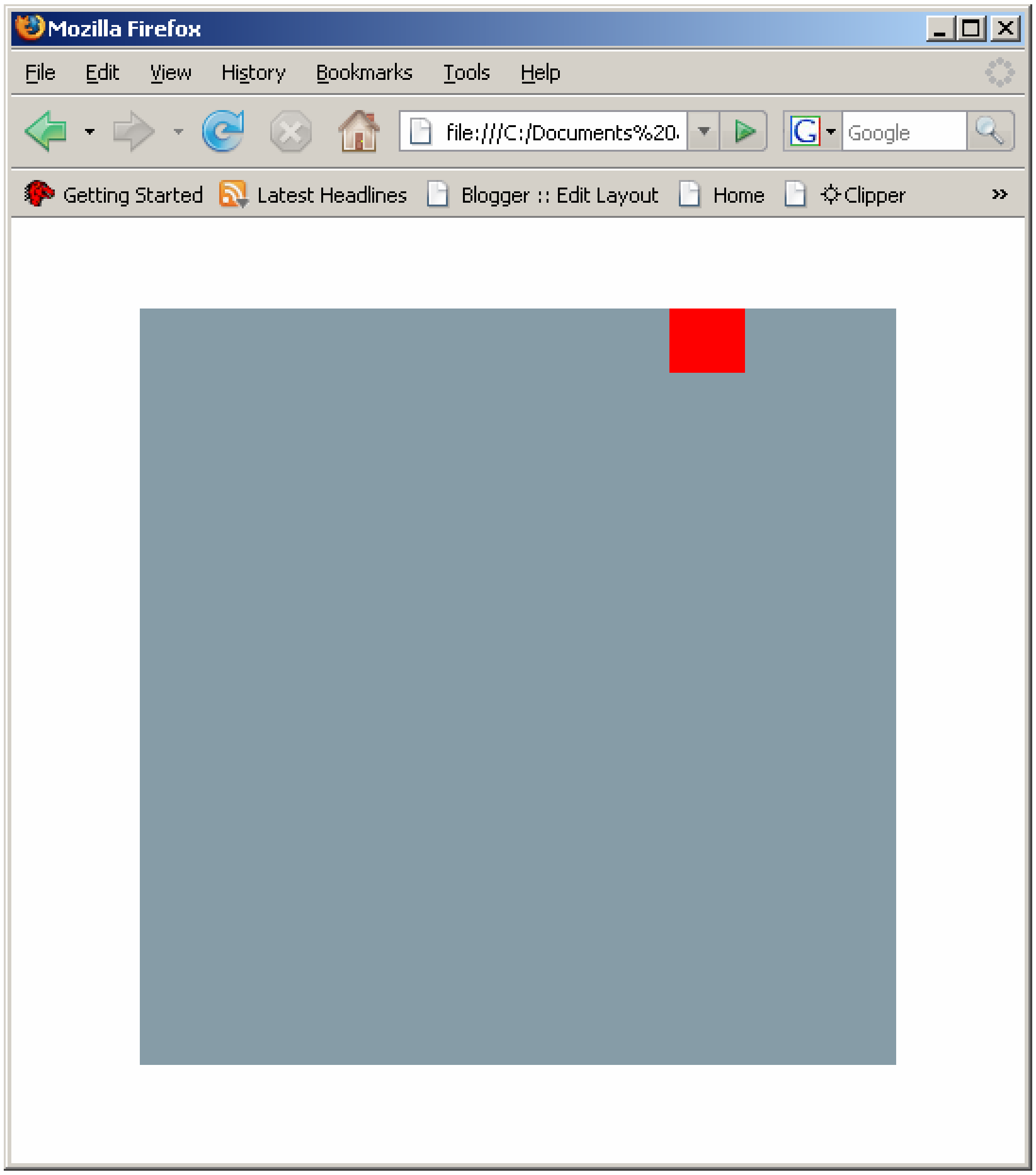
2.4 Tutorial Program Four: "Square animation."

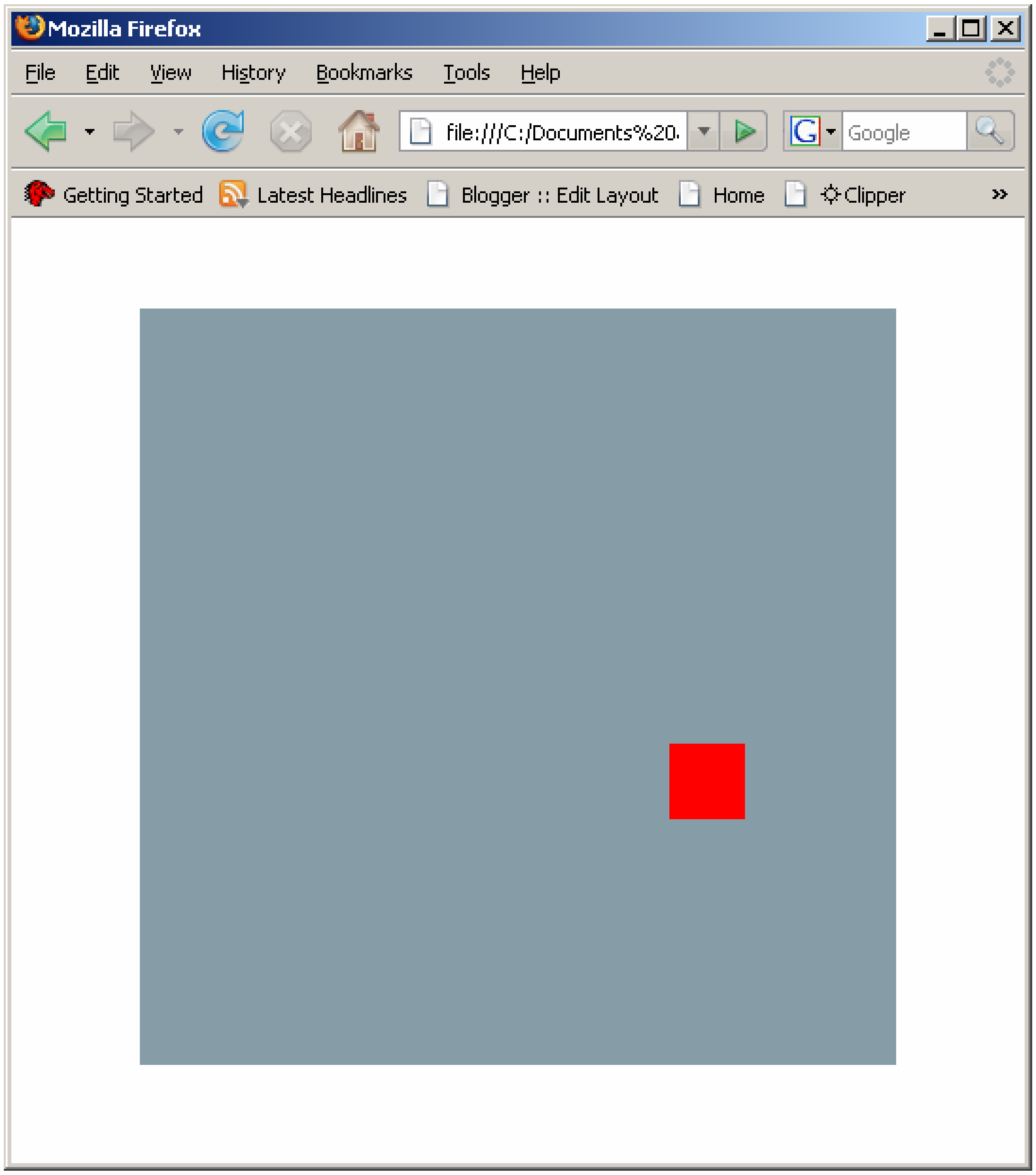
Finally, here is a simple animation of a red square moving down the drawing area and reappearing at the top after disappearing below the stage.

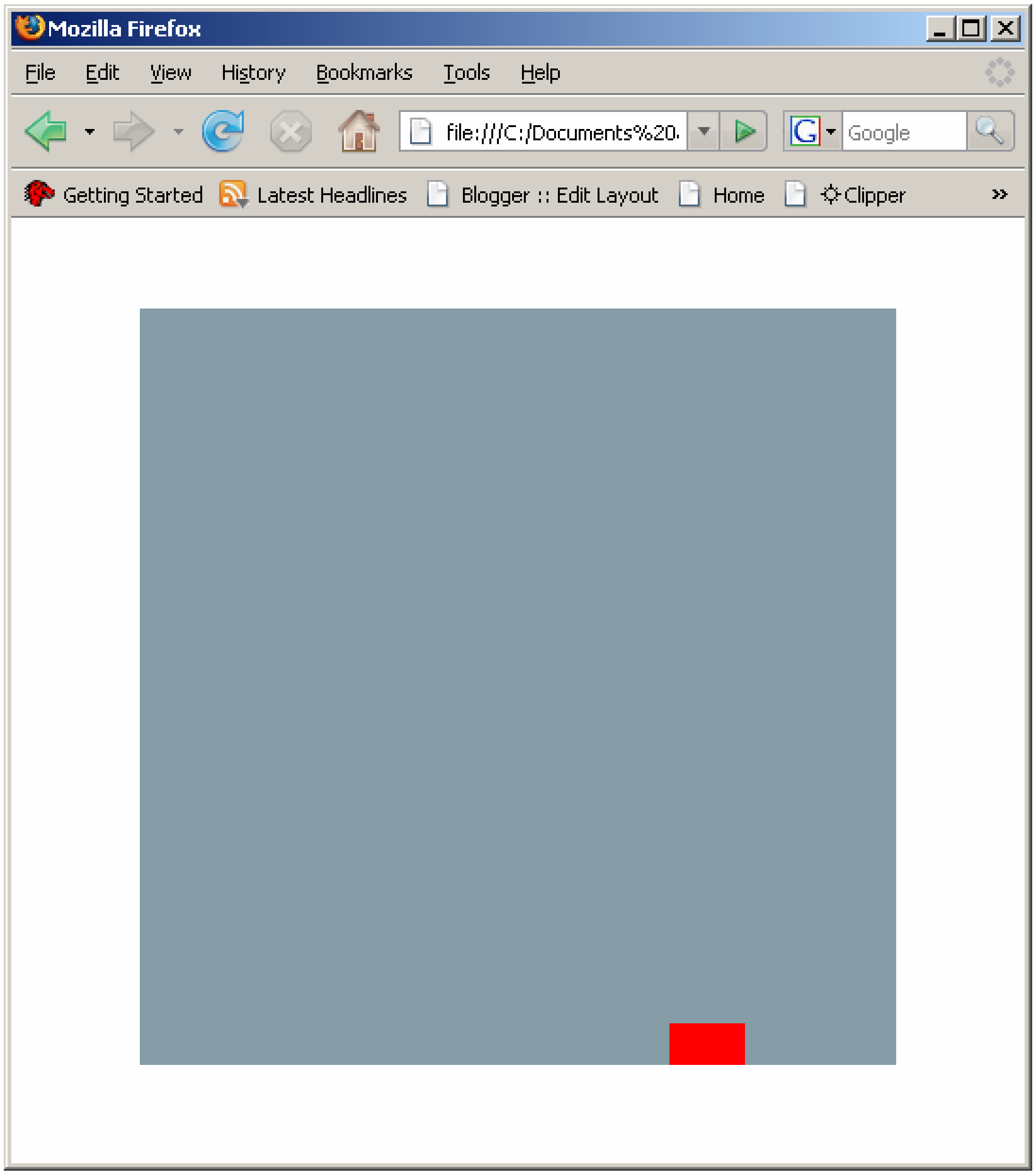
```
---------------------------------------------------

square = makesquare(50, "red");
square = setx(square, 350);
square = sety(square, 250);
animate {
      y = gety(square);
      if (y > 500) y = -175;
      y = y + 5;
      square = sety(square, y);
}

---------------------------------------------------
```

```
$ java SFAL SimpleMove simplemove.sfal

$ mxmlc SimpleMove.as
Loading configuration file C:\flex\frameworks\flex-
config.xml
C:\Documents and
Settings\admin\Desktop\rollback\plt\SimpleMove.swf (814
bytes)

$ ls SimpleMove.*
SimpleMove.as  SimpleMove.html  SimpleMove.swf
```

The output is illustrated over the next three graphics:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Documents%20
Google
Getting Started Latest Headlines Blogger :: Edit Layout Home Clipper


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Documents%20
Google
Getting Started Latest Headlines Blogger :: Edit Layout Home Clipper


Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help
file:///C:/Documents%20
Google
Getting Started
Latest Headlines
Blogger :: Edit Layout
Home
Clipper

3. SFAL Language Reference Manual

3.1 Tokens

SFAL tokens consist of: identifiers, keywords, constants, string literals and separators. Blanks, tabs, and newlines are ignored except as they serve to separate tokens.

Additionally, SFAL makes use of the following Tokens:

```
( )
{ }

*
+
-
/
%

;
=
,

>=
<=
>
<

==
!=

||
&&
```

3.1.1 Identifiers

Identifiers take the form:

ALPHA (ALPHA|DIGIT)*

where ALPHA and DIGIT are represented respectively:

```
protected
ALPHA   : 'a'..'z' | 'A'..'Z' | '_' ;

protected
DIGIT   : '0'..'9';
```

3.1.2 Comments

SFAL comments may be single line comments beginning with `//`, or multi-line comments beginning with `/*` and ending with `*/`.

3.1.3 Keywords

The following identifiers are reserved for use as keywords and may not be used otherwise:

```
if
else
true
false
not
loop
animate
```

3.1.4 Constants

SFAL supports hexadecimal, integer and floating-point constants.

```
protected
HEX         : '0'..'9' | 'A'..'F' | 'a'..'f' ;

HEXNUM      : "0x" HEX HEX HEX HEX HEX HEX ;


NUMBER  : (DIGIT)+ ('.' (DIGIT)* )?
      | '.' (DIGIT)+
      ;
```

3.1.5 String Literals

SFAL supports string literals, also called string constants, which are sequences of characters enclosed by double quotes, e.g. "Hello World".

3.2 Statements

Statements end with a semicolon and may take the following forms:

```
statement ->
      if-stmt |
      assignment |
      func-call-stmt |
      loop_stmt |
      animate
```

3.2.1 If Statement (Conditional Statement)

The conditional If statement takes the following forms:

```
if ( booleanExpression )
      Statement ;
```

or

```
if ( booleanExpression )
      Statement ;
else
      Statement ;
```

3.2.2 Assignment

Assignment takes the following form:

```
ID = RightValueExpression ;
```

3.2.3 Loop Statement (Iteration)

Iteration takes the form of the loop construct:

```
loop (startInteger, endInteger, indexVariable)
     Statement;
```

`indexVariable` is a local variable visible to the body of the loop statement which keeps track of which iteration the loop is on.

3.2.4 Function Call

SFAL includes a number of built-in functions which operate on and return SFAL objects.

```
makesquare(Number,(String|Hex))
makecircle(Number,(String|Hex));
makerect(Number,Number,(String|Hex));

makegroup();
addtogroup(groupID, memberID);

maketext(String);
addtotext(textID, String);

globalnumber();
```

```
setx(ID,Number);
sety(ID,Number);

getx(ID);
gety(ID);

settextx(textID,Number);
settexty(textID,Number);

gettextx(textID);
gettexty(textID);

sin(Number);
cos(Number);
```

4. Project Plan

4.1 Process

Once I decided on creating an animation language based on Flash, with ActionScript as the Intermediate Representation, I started familiarizing myself with ActionScript. After I had understanding of how to create objects and textfields in ActionScript, and then animate them, I sought to create a simpler, English-like language and syntax for accomplishing the same result.

I wrote several programs in ActionScript that encompassed the functionality I wished to achieve in SFAL and began working backwards from there.

The high level goals of SFAL were:

1. Easily create text fields.
2. Easily create basic shapes: Square, Circle, Rectangle. These shapes should have user defined size and color.
3. Easily create groups of text fields and shapes.
4. Easily animate text fields, shapes and groups.

4.2 Coding Conventions

As the sole developer for this project, I didn't strictly adhere to any coding style. I loosely followed some guidelines advocated by Sun's Code Conventions for the Java Programming Language[5], e.g. I tried to keep line lengths around 80 characters, used descriptive variable names and variable naming conventions, consistent formatting and indentation etc.

[5] http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

4.3 Timeline / Log

9/17	Leaning towards a graphical language
10/1	Explore programmatic output of .swf files
10/7	Learn simple ActionScript, explore AS compilers
10/17	Decide on using .as files as IR and compiling swf w/ mxmlc
10/24	LRM feedback from Dhivya (TA)
10/30	Lexer is working
11/6	Parser is working
11/13	Meet w/ Prof. Edwards to go over Walker
11/16	Walker is working
11/30	Validator, SymbolTable, start on Translator
12/7	Translator output simple programs
12/11	Translator output animations
12/14	Tutorial programs and test programs / Report
12/15	Report
12/18	Submit project

4.3 Development Environment

I used CVS for source control in Cygwin[6], a linux emulator for MS Windows. I consulted the example mx and dragonbook grammar.g files extensively, building the SFAL grammar.g using the the mx grammar.g as a guide. I used the development platform Eclipse[7] for developing my project, along w/ the Eclipse plug-in XmlBuddy[8]. Since ANTLR can output the Abstract Syntax

[6] http://www.cygwin.com/
[7] http://www.eclipse.org/
[8] http://www.xmlbuddy.com/

Tree (AST) in xml format, these tools were indispensable in understanding and developing the parser and walker.

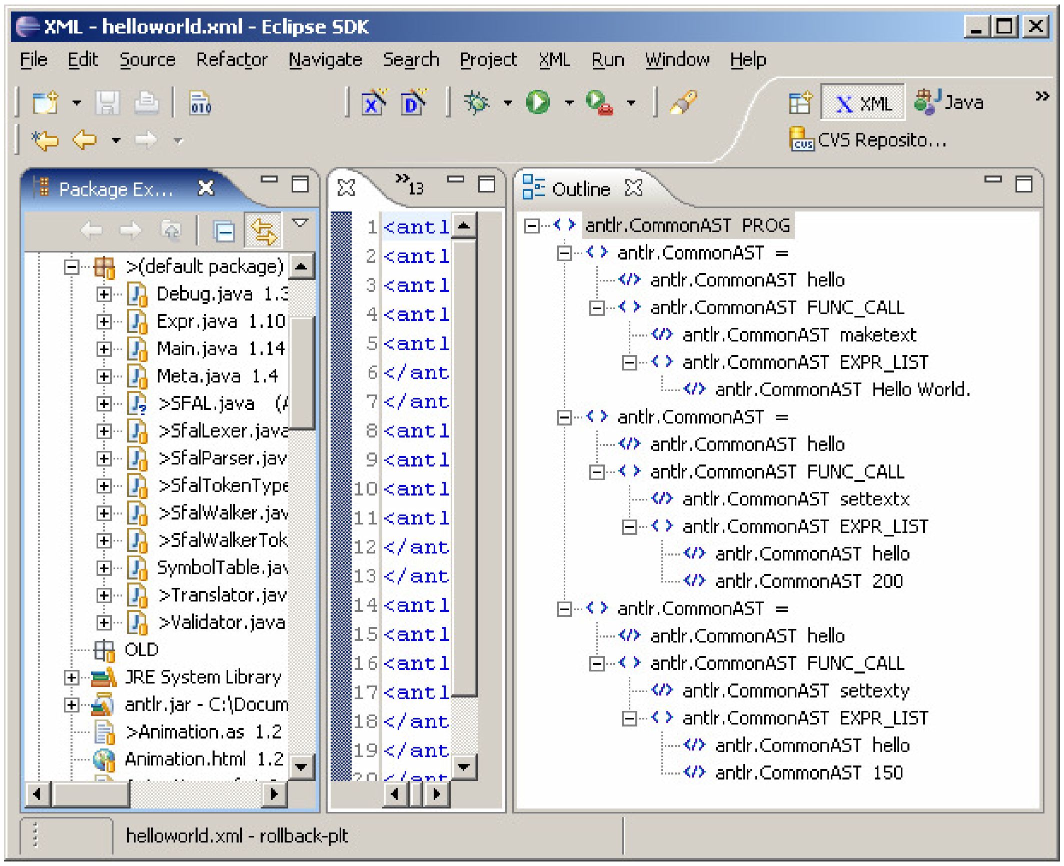
Here is code to have ANTLR output the parse tree in xml:

```
SfalParser parser = new SfalParser(lexer);
parser.program();

CommonAST tree = (CommonAST) parser.getAST();

Writer writer = new OutputStreamWriter(System.out);
tree.xmlSerialize(writer);
writer.write("\n");
writer.flush();
```

Here is the xml produced for the SFAL HelloWorld program:

```
<antlr.CommonAST text="PROG" type="34">
<antlr.CommonAST text="=" type="23">
<antlr.CommonAST text="hello" type="8"/><antlr.CommonAST
text="FUNC_CALL" type="37">
<antlr.CommonAST text="maketext" type="8"/><antlr.CommonAST
text="EXPR_LIST" type="38">
<antlr.CommonAST text="Hello World."
type="33"/></antlr.CommonAST>
</antlr.CommonAST>
</antlr.CommonAST>
<antlr.CommonAST text="=" type="23">
<antlr.CommonAST text="hello" type="8"/><antlr.CommonAST
text="FUNC_CALL" type="37">
<antlr.CommonAST text="settextx" type="8"/><antlr.CommonAST
text="EXPR_LIST" type="38">
<antlr.CommonAST text="hello" type="8"/><antlr.CommonAST
text="200" type="9"/></antlr.CommonAST>
</antlr.CommonAST>
</antlr.CommonAST>
<antlr.CommonAST text="=" type="23">
<antlr.CommonAST text="hello" type="8"/><antlr.CommonAST
text="FUNC_CALL" type="37">
<antlr.CommonAST text="settexty" type="8"/><antlr.CommonAST
text="EXPR_LIST" type="38">
<antlr.CommonAST text="hello" type="8"/><antlr.CommonAST
text="150" type="9"/></antlr.CommonAST>
</antlr.CommonAST>
</antlr.CommonAST>
</antlr.CommonAST>
```

This is what it looked like in XmlBuddy in Eclipse. The AST structure is clearly illustrated:

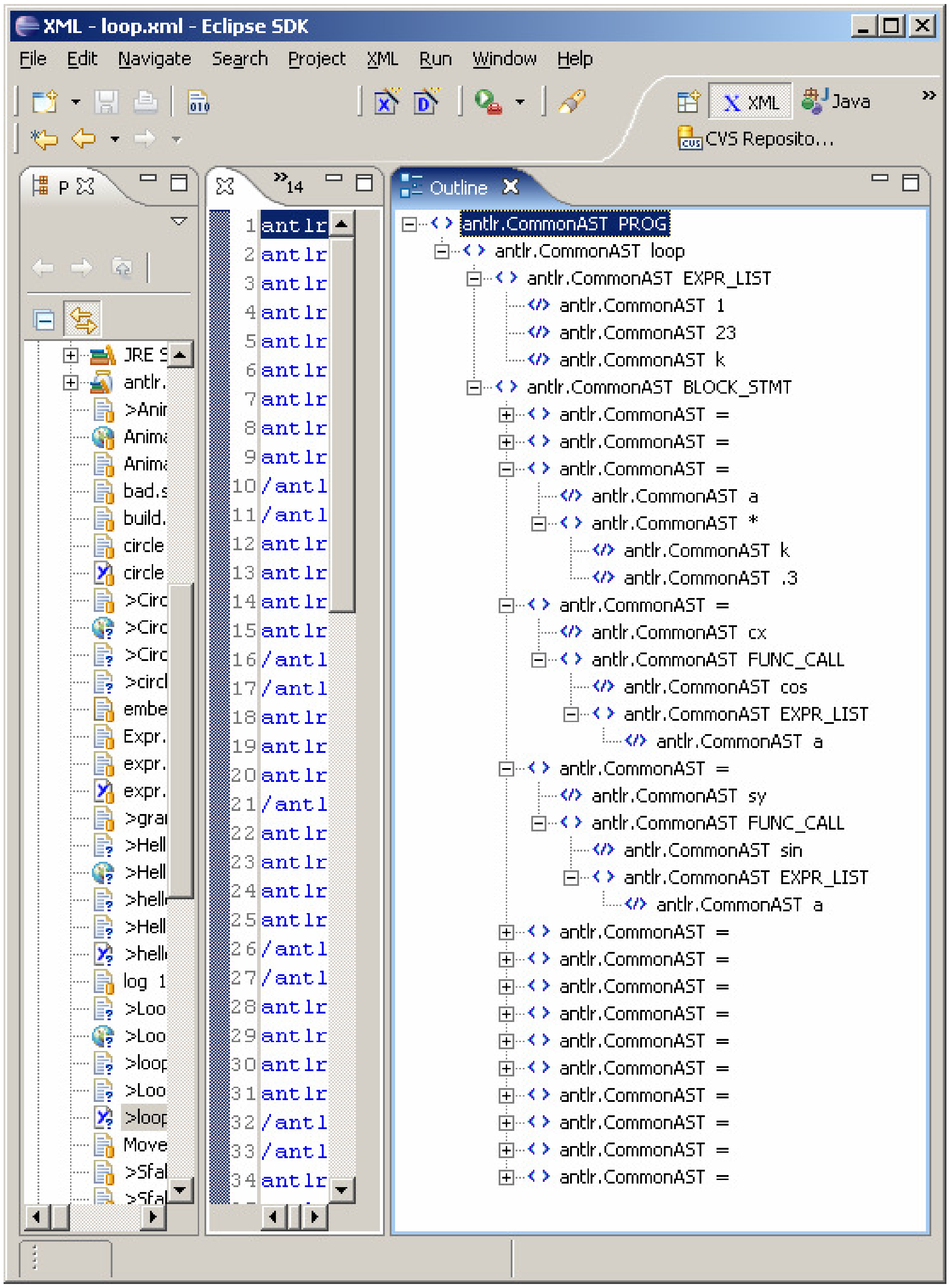

XML - helloworld.xml - Eclipse SDK
File Edit Source Refactor Navigate Search Project XML Run Window Help
XML
Java
CVS Reposito...
Package Ex...
Outline
helloworld.xml - rollback-plt

```
>(default package)
  Debug.java 1.3
  Expr.java 1.10
  Main.java 1.14
  Meta.java 1.4
  >SFAL.java (A
  >SfalLexer.java
  >SfalParser.jav
  >SfalTokenType
  >SfalWalker.jav
  >SfalWalkerTok
  SymbolTable.jav
  >Translator.jav
  >Validator.java
OLD
JRE System Library
antlr.jar - C:\Docum
>Animation.as 1.2
Animation.html 1.2
```

```
<antl
<antl
<antl
<antl
<antl
</ant
</ant
<antl
<antl
<antl
<antl
</ant
</ant
<antl
<antl
<antl
<antl
</ant
</ant
</ant
```

```
antlr.CommonAST PROG
  antlr.CommonAST =
    antlr.CommonAST hello
    antlr.CommonAST FUNC_CALL
      antlr.CommonAST maketext
      antlr.CommonAST EXPR_LIST
        antlr.CommonAST Hello World.
  antlr.CommonAST =
    antlr.CommonAST hello
    antlr.CommonAST FUNC_CALL
      antlr.CommonAST settextx
      antlr.CommonAST EXPR_LIST
        antlr.CommonAST hello
        antlr.CommonAST 200
  antlr.CommonAST =
    antlr.CommonAST hello
    antlr.CommonAST FUNC_CALL
      antlr.CommonAST settexty
      antlr.CommonAST EXPR_LIST
        antlr.CommonAST hello
        antlr.CommonAST 150
```

Here is another example of xml output in Eclipse with XmlBuddy: the SFAL example loop program:



```
antlr
antlr
antlr
antlr
antlr
antlr
antlr
antlr
antlr
/antl
/antl
antlr
antlr
antlr
antlr
/antl
/antl
antlr
antlr
antlr
/antl
antlr
antlr
antlr
antlr
/antl
/antl
antlr
antlr
antlr
antlr
/antl
/antl
antlr
```

```
antlr.CommonAST PROG
  antlr.CommonAST loop
    antlr.CommonAST EXPR_LIST
      antlr.CommonAST 1
      antlr.CommonAST 23
      antlr.CommonAST k
    antlr.CommonAST BLOCK_STMT
      antlr.CommonAST =
      antlr.CommonAST =
      antlr.CommonAST =
        antlr.CommonAST a
        antlr.CommonAST *
          antlr.CommonAST k
          antlr.CommonAST .3
      antlr.CommonAST =
        antlr.CommonAST cx
        antlr.CommonAST FUNC_CALL
          antlr.CommonAST cos
          antlr.CommonAST EXPR_LIST
            antlr.CommonAST a
      antlr.CommonAST =
        antlr.CommonAST sy
        antlr.CommonAST FUNC_CALL
          antlr.CommonAST sin
          antlr.CommonAST EXPR_LIST
            antlr.CommonAST a
      antlr.CommonAST =
      antlr.CommonAST =
      antlr.CommonAST =
      antlr.CommonAST =
      antlr.CommonAST =
      antlr.CommonAST =
      antlr.CommonAST =
      antlr.CommonAST =
      antlr.CommonAST =
      antlr.CommonAST =
```

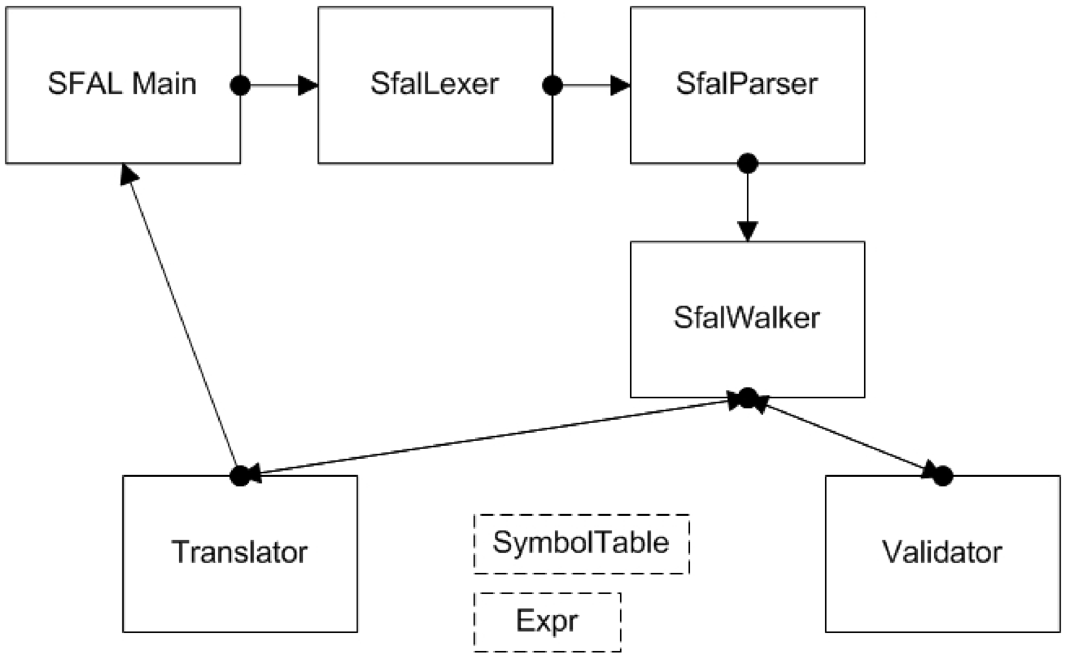
Additionally, the Eclipse debugger was extremely useful in tracing the execution of the mx and dragonbook example code covered in class.

5 Architectural Design

5.1 Block Diagram

SFAL implements a Validator and a Translator to integrate with the SfalWalker. As named, the Validator validates SFAL statements while the Translator translates them to the appropriate ActionScript. An Expr class represents SFAL expressions while the SymbolTable manages variables and scope. Expr objects and SymbolTable objects are passed between the SfalWalker, Validator and Translator via simple interfaces described in the next section.

Here is the block diagram illustrating the relationships between the SFAL compiler components:



5.2 Interfaces

The SfalWalker integrates with the Validator and Translator via simple interfaces. Following is the interface for `Validator.java`.

Validator.java
boolean **validate**(int root, Expr e, SymbolTable top) This method is responsible for validating SFAL statements with one expression. A return value of false results in a RecognitionException being thrown in the SfalWalker.
boolean **validate**(int root, Expr e1, Expr e2, SymbolTable top) This method is responsible for validating SFAL statements with two expressions. A return value of false results in a RecognitionException being thrown in the SfalWalker.
boolean **validateLoop**(List l) This method is responsible for validating SFAL loop statements. A return value of false results in a RecognitionException being thrown in the SfalWalker.

Before examining the SFAL Translator, it is necessary to review the structure of an ActionScript program that incorporates animation.

```
/* Must begin with package declaration, which may
optionally be named.*/
package {

      // Must import objects to be used.
      import flash.display.Sprite;
      import flash.text.TextField;
      import flash.events.Event;

      /* Class name must match <File>.as name and
            extend Sprite */
      public class Animation extends Sprite {

            /* ActionScript constructor,
                  must have same basename as <File>.as and
                  class declaration. */
            public function Animation() {

                  (... make objects to animate here)

                  globalSprite.doSomething;

                  /* some local sprite to be drawn
                        or animated */
                  var localSprite:Sprite = new Sprite();

                  localSprite.doSomething;

                  /* Register animate() function with
                  EventBroadcaster */
                  addEventListener(
                        Event.ENTER_FRAME, animate);
            }

            /* Here is the method enabling the animation
                  it is event driven and called on
                  Event.ENTER_FRAME */

            public function animate(event:Event):void {
                  (... code to animate objects in
                  constructor here)
            }

            // e.g. some global sprite to be animated
            public var globalSprite:Sprite = new Sprite();

            /* e.g. some arbitrary variables used
                  in the animation */
            public var angle:Number = 0;
            public var inc:Number = 0;
      }
}
```

The ActionScript example above can be broken into three nested blocks.

The outermost block is the `package` block which contains `import` statements and the nested `class` block. Nested within the `class` block is the `class` constructor, along with an `EventListener` function -- the `animate` function -- and global variable declarations.

Given the strict layout of the ActionScript program, SFAL statements cannot simply be translated to ActionScript in the order they appear in the SFAL program. Certain SFAL block statements, like `animate,` must be rendered to the ActionScript file in a distinct location. Additionally, ActionScript global and local variables have different declarations, with global variables distinguished with the keyword `public` and not contained in another block in the `class` block. Thus the Translator employs a program buffer and several sub-buffers to ensure animate blocks, local and global variable declarations etc. are rendered in ActionScript in the prescribed sequence.

Translator.java
String **translate**(int type, String s) This method translates SFAL statements into their ActionScript equivalents.
String **translateAnimator**() Given the structure of an ActionScript program with animation, it is necessary to translate animate blocks separately.
void **buildAnimator**(String s) Given the structure of an ActionScript program with animation, animate blocks are buffered before before being output.
void **buildProgramBuffer**(String s)

Given the structure of ActionScript programs, SFAL translations are buffered so that portions of the ActionScript are output in the correct order.
String **translateSymbolTableLocal**() SFAL is an untyped language but ActionScript is not. Both local and global variables are typed and must be declared in the proper scope.
String **translateSymbolTable**() SFAL is an untyped language but ActionScript is not. Both local and global variables are typed and must be declared in the proper scope.

6 Test Plan

6.1 Build

I used a simple build script (`build.sh`) for clean builds everytime I modified .java or .g files:

```
#!/bin/bash
rm -rf Sfal* *.class
echo "...cleaning"

echo "...java antlr.Tool grammar-as-semant.g"
java antlr.Tool grammar-as-semant.g

javac -source 1.4 *.java
echo "...compiling"

echo "...done"
```

6.2 Automated Testing

For automated testing I used a `Makefile` modeled after the one used for the `dragonbook.tar` example. There are two `check` targets. The first, `check_mxmlc`, will compare the compiler output with a "gold" file, and then run the `mxmlc` compiler on the output .as file. The second, `check`, just compares the compiler output with the "gold" file, but does not assume the user has the `mxmlc` compiler installed.

There are six test cases that check functionality of the SFAL compiler. Each test case focuses on one major area of functionality.

```
$ more Makefile

TESTCASES = Test1 Test2 Test3 Test4 Test5 Test6

check_mxmlc : SFAL.class

        @for basename in $(TESTCASES) ; do \

        echo -n $$basename"..."; \

        java SFAL $$basename $$basename.sfal; \

        (diff -b -q $$basename.gold $$basename.as && echo
"OK") \

          || echo "FAILED"; \

          mxmlc $$basename.as; \

        done

check : SFAL.class

        @for basename in $(TESTCASES) ; do \

        echo -n $$basename"..."; \

        java SFAL $$basename $$basename.sfal; \
        (diff -b -q $$basename.gold $$basename.as && echo
"OK") \

          || echo "FAILED"; \

        done



$ make check

Test1...OK

Test2...OK

Test3...OK

Test4...OK

Test5...OK

Test6...OK



$ make check_mxmlc

Test1...OK

Loading configuration file C:\flex\frameworks\flex-
config.xml

C:\Documents and
Settings\admin\Desktop\rollback\plt\Test1.swf (718 bytes)

Test2...OK

Loading configuration file C:\flex\frameworks\flex-
config.xml

C:\Documents and
Settings\admin\Desktop\rollback\plt\Test2.swf (861 bytes)

Test3...OK

Loading configuration file C:\flex\frameworks\flex-
config.xml

C:\Documents and
Settings\admin\Desktop\rollback\plt\Test3.swf (888 bytes)

Test4...OK
```

```
Loading configuration file C:\flex\frameworks\flex-
config.xml

C:\Documents and
Settings\admin\Desktop\rollback\plt\Test4.swf (803 bytes)

Test5...OK

Loading configuration file C:\flex\frameworks\flex-
config.xml

C:\Documents and
Settings\admin\Desktop\rollback\plt\Test5.swf (1264 bytes)

Test6...OK

Loading configuration file C:\flex\frameworks\flex-
config.xml

C:\Documents and
Settings\admin\Desktop\rollback\plt\Test6.swf (1270 bytes)
```

To view the resultant `.swf` file in a Flash-enabled browser, double-click or open the corresponding `.html` file.

6.2.1 Test Case 1 - Hello World

Test1.sfal:

```
hello = maketext("Hello World.");
hello = settextx(hello, 200);
hello = settexty(hello, 150);
```

Compiler output: Test1.as:

```
package {
      import flash.display.Sprite;
      import flash.text.TextField;
      import flash.events.Event;
      public class Test1 extends Sprite {
            public function Test1() {
                  hello.text="Hello World.";
                  addChild(hello);
                  hello.x=200;
                  hello.y=150;
                  addEventListener(Event.ENTER_FRAME, animate);
            }
            public function animate(event:Event):void {
            }
            public var hello:TextField = new TextField();
      }
}
```

6.2.2 Test Case 2 - Simple Shapes

Test2.sfal:

```
square = makesquare(30, "blue");
square = setx(square, 150);
square = sety(square, 150);

circle = makecircle(25, "red");
circle = setx(circle, 250);
circle = sety(circle, 170);

rect = makerect(50, 10, "green");
rect = setx(rect, 320);
rect = sety(rect, 160);
```

Compiler output: Test2.as:

```
package {
      import flash.display.Sprite;
      import flash.text.TextField;
      import flash.events.Event;
      public class Test2 extends Sprite {
            public function Test2() {
                  square.graphics.beginFill(0x0000FF,100);
                  square.graphics.drawRect(0,0,30,30);
                  square.graphics.endFill();
                  square.x = 0;
                  square.y = 0;
                  addChild(square);
                  square.x=150;
                  square.y=150;
                  circle.graphics.beginFill(0xFF0000,100);
                  circle.graphics.drawCircle(0,0,25);
                  circle.graphics.endFill();
                  circle.x = 0;
                  circle.y = 0;
                  addChild(circle);
                  circle.x=250;
                  circle.y=170;
                  rect.graphics.beginFill(0x008000,100);
                  rect.graphics.drawRect(0,0,50,10);
                  rect.graphics.endFill();
                  rect.x = 0;
                  rect.y = 0;
                  addChild(rect);
                  rect.x=320;
                  rect.y=160;
                  addEventListener(Event.ENTER_FRAME, animate);
            }
            public function animate(event:Event):void {
            }
            public var square:Sprite = new Sprite();
            public var circle:Sprite = new Sprite();
            public var rect:Sprite = new Sprite();
      }
}
```

6.2.3 Test Case 3 - Loop

Test3.sfal:

```
loop(1,23,k) {
      s = makesquare(12, "black");
      s2 = makesquare(12, "lightblue");

      // space the squares apart
      a = k * .3;

      cx = cos(a);
      sy = sin(a);

      // coordinates
      cx = cx * 80; cx = cx + 245;
      sy = sy * 80; sy = sy + 175;

      // place the first sprite
      s = setx(s,cx);
      s = sety(s,sy);

      cx = cx + 2;
      sy = sy - 2;

      // place the second sprite
      s2 = setx(s2,cx);
      s2 = sety(s2,sy);
}
```

Compiler output: Test3.as:

```
package {
      import flash.display.Sprite;
      import flash.text.TextField;
      import flash.events.Event;
      public class Test3 extends Sprite {
            public function Test3() {
                  for (var k:int = 1;
                  k < 23;
                  k++){
                        var s:Sprite = new Sprite();
                        s.graphics.beginFill(0x000000,100);
                        s.graphics.drawRect(0,0,12,12);
                        s.graphics.endFill();
                        s.x = 0;
                        s.y = 0;
                        addChild(s);
                        var s2:Sprite = new Sprite();
                        s2.graphics.beginFill(0xADD8E6,100);
                        s2.graphics.drawRect(0,0,12,12);
                        s2.graphics.endFill();
                        s2.x = 0;
```

```
                s2.y = 0;
                addChild(s2);
                var a:Number = 0;
                a=k*.3;
                var cx:Number = 0;
                cx=Math.cos(a);
                var sy:Number = 0;
                sy=Math.sin(a);
                cx=cx*80;
                cx=cx+245;
                sy=sy*80;
                sy=sy+175;
                s.x=cx;
                s.y=sy;
                cx=cx+2;
                sy=sy-2;
                s2.x=cx;
                s2.y=sy;
            }
            addEventListener(Event.ENTER_FRAME, animate);
        }
        public function animate(event:Event):void {
        }
    }
}
```

6.2.4 Test Case 4 - Simple Animation and Conditional Statement

Test4.sfal:

```
square = makesquare(50, "red");
square = setx(square, 350);
square = sety(square, 250);

animate {
      inc=5;
      y = gety(square);
      if (y > 500) y = -175;
      y = y + inc;
      square = sety(square, y);
}
```

Compiler output: Test4.as:

```
package {
      import flash.display.Sprite;
      import flash.text.TextField;
      import flash.events.Event;
      public class Test4 extends Sprite {
            public function Test4() {
                  square.graphics.beginFill(0xFF0000,100);
                  square.graphics.drawRect(0,0,50,50);
                  square.graphics.endFill();
                  square.x = 0;
                  square.y = 0;
                  addChild(square);
                  square.x=350;
                  square.y=250;
                  addEventListener(Event.ENTER_FRAME, animate);
            }
            public function animate(event:Event):void {
                  var inc:Number = 0;
                  inc=5;
                  var y:Number = 0;
                  y=square.y;
                  if (y>500) {
                        y=-175;
                  }
                  y=y+inc;
                  square.y=y;
            }
            public var square:Sprite = new Sprite();
      }
}
```

6.2.5 Test Case 5 - Multiple Animations

Test5.sfal:

```
square = makesquare(50,"red");
square = setx(square, 350);
square = sety(square, 250);

circle = makecircle(10,"yellow");
circle = setx(circle, 150);
circle = sety(circle, 150);

rect = makerect(50, 20,"green");
rect = setx(rect, 400);
rect = sety(rect, 400);

circlesquare = makegroup();
circlesquare = addtogroup(circlesquare, circle);

angle=globalnumber();
angle2=globalnumber();
inc=globalnumber();

loop(1,23,k) {
      s = makesquare(7, "darkgray");
      s2 = makesquare(7, "lightblue");

      a = k * .3;

      cx = cos(a);
      sy = sin(a);

      // coordinates
      cx = cx * 80; cx = cx + 200;
      sy = sy * 80; sy = sy + 125;

      // place the first sprite
      s = setx(s,cx);
      s = sety(s,sy);

      cx = cx - 2;
      sy = sy - 2;

      // place the second sprite
      s2 = setx(s2,cx);
      s2 = sety(s2,sy);
}

animate {
    inc=5;

    x=getx(circlesquare);
    y=gety(circlesquare);

    angle=angle+.1;
    cx = cos(angle); cx = cx * 100;
    sy = sin(angle); sy = sy * 100;

    circlesquare=setx(circlesquare,cx);
    circlesquare=sety(circlesquare,sy);

    x=getx(rect);
    y=gety(rect);

    angle2=angle2-.2;
    cx = cos(angle2); cx=cx*50; cx=cx+200;
    sy = sin(angle2); sy=sy*50; sy=sy+200;

    rect=setx(rect,cx);
    rect=sety(rect,sy);

    y=gety(square);

    if (y > 500) y=-175;

      y = y + inc;

    square=sety(square,y);
}
```

Compiler output: Test5.as:

```
package {
      import flash.display.Sprite;
      import flash.text.TextField;
      import flash.events.Event;
      public class Test5 extends Sprite {
            public function Test5() {
                  square.graphics.beginFill(0xFF0000,100);
                  square.graphics.drawRect(0,0,50,50);
                  square.graphics.endFill();
                  square.x = 0;
                  square.y = 0;
                  addChild(square);
                  square.x=350;
                  square.y=250;
                  circle.graphics.beginFill(0xFFFF00,100);
                  circle.graphics.drawCircle(0,0,10);
                  circle.graphics.endFill();
                  circle.x = 0;
                  circle.y = 0;
                  addChild(circle);
                  circle.x=150;
                  circle.y=150;
                  rect.graphics.beginFill(0x008000,100);
                  rect.graphics.drawRect(0,0,50,20);
                  rect.graphics.endFill();
                  rect.x = 0;
                  rect.y = 0;
                  addChild(rect);
                  rect.x=400;
                  rect.y=400;
                  addChild(circlesquare);
                  circlesquare.addChild(circle);
                  for (var k:int = 1;
                  k < 23;
                  k++){
                        var s:Sprite = new Sprite();
                        s.graphics.beginFill(0xA9A9A9,100);
                        s.graphics.drawRect(0,0,7,7);
                        s.graphics.endFill();
                        s.x = 0;
                        s.y = 0;
                        addChild(s);
                        var s2:Sprite = new Sprite();
                        s2.graphics.beginFill(0xADD8E6,100);
                        s2.graphics.drawRect(0,0,7,7);
                        s2.graphics.endFill();
                        s2.x = 0;
                        s2.y = 0;
                        addChild(s2);
                        var a:Number = 0;
                        a=k*.3;
                        var cx:Number = 0;
                        cx=Math.cos(a);
                        var sy:Number = 0;
                        sy=Math.sin(a);
                        cx=cx*80;
                        cx=cx+200;
                        sy=sy*80;
                        sy=sy+125;
                        s.x=cx;
                        s.y=sy;
                        cx=cx-2;
                        sy=sy-2;
                        s2.x=cx;
                        s2.y=sy;
                  }
                  addEventListener(Event.ENTER_FRAME, animate);
            }
            public function animate(event:Event):void {
                  inc=5;
                  var x:Number = 0;
                  x=circlesquare.x;
                  var y:Number = 0;
                  y=circlesquare.y;
                  angle=angle+.1;
                  var cx:Number = 0;
                  cx=Math.cos(angle);
                  cx=cx*100;
                  var sy:Number = 0;
                  sy=Math.sin(angle);
                  sy=sy*100;
                  circlesquare.x=cx;
                  circlesquare.y=sy;
                  x=rect.x;
                  y=rect.y;
                  angle2=angle2-.2;
                  cx=Math.cos(angle2);
                  cx=cx*50;
                  cx=cx+200;
                  sy=Math.sin(angle2);
                  sy=sy*50;
                  sy=sy+200;
                  rect.x=cx;
                  rect.y=sy;
                  y=square.y;
                  if (y>500) {
                        y=-175;
                  }
                  y=y+inc;
                  square.y=y;
            }
            public var square:Sprite = new Sprite();
            public var angle2:Number = 0;
            public var angle:Number = 0;
            public var inc:Number = 0;
            public var circle:Sprite = new Sprite();
            public var rect:Sprite = new Sprite();
            public var circlesquare:Sprite = new Sprite();
      }
}
```

6.2.6 Test Case 6 - Animations with Groups

Test6.sfal:

```
square = makesquare(50,"red");
square = setx(square, 350);
square = sety(square, 250);

circle = makecircle(10,"yellow");
circle = setx(circle, 150);
circle = sety(circle, 150);

rect = makerect(50, 20,"green");
rect = setx(rect, 400);
rect = sety(rect, 400);

circlesquare = makegroup();
circlesquare = addtogroup(circlesquare, circle);

angle=globalnumber();
angle2=globalnumber();
inc=globalnumber();

loop(1,23,k) {
      s = makesquare(7, "darkgray");
      s2 = makesquare(7, "lightblue");

      a = k * .3;

      cx = cos(a);
      sy = sin(a);

      // coordinates
      cx = cx * 80; cx = cx + 200;
      sy = sy * 80; sy = sy + 125;

      // place the first sprite
      s = setx(s,cx);
      s = sety(s,sy);

      cx = cx - 2;
      sy = sy - 2;

      // place the second sprite
      s2 = setx(s2,cx);
      s2 = sety(s2,sy);

circlesquare = addtogroup(circlesquare, s);
circlesquare = addtogroup(circlesquare, s2);
}

animate {
      inc=5;

    x=getx(circlesquare);
    y=gety(circlesquare);

    angle=angle+.1;

    cx = cos(angle); cx = cx * 100;
    sy = sin(angle); sy = sy * 100;

    circlesquare=setx(circlesquare,cx);
    circlesquare=sety(circlesquare,sy);

    x=getx(rect);
    y=gety(rect);

    angle2=angle2-.2;
    cx = cos(angle2); cx=cx*50; cx=cx+200;
    sy = sin(angle2); sy=sy*50; sy=sy+200;

    rect=setx(rect,cx);
    rect=sety(rect,sy);

    y=gety(square);

    if (y > 500) y=-175;

      y = y + inc;

    square=sety(square,y);
}
```

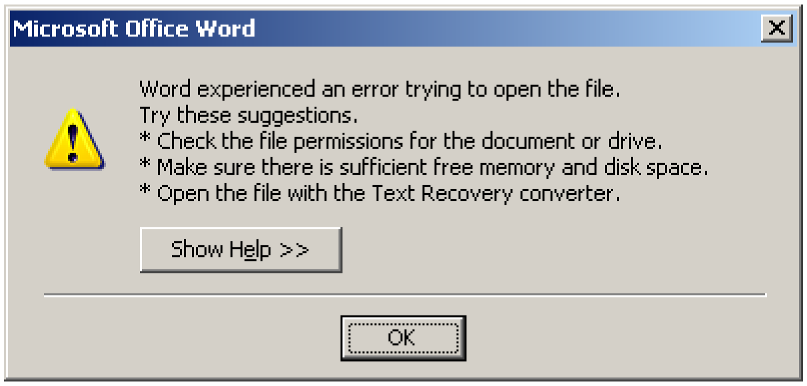
Compiler output: Test6.as

```
package {
      import flash.display.Sprite;
      import flash.text.TextField;
      import flash.events.Event;
      public class Test6 extends Sprite {
            public function Test6() {
                  square.graphics.beginFill(0xFF0000,100);
                  square.graphics.drawRect(0,0,50,50);
                  square.graphics.endFill();
                  square.x = 0;
                  square.y = 0;
                  addChild(square);
                  square.x=350;
                  square.y=250;
                  circle.graphics.beginFill(0xFFFF00,100);
                  circle.graphics.drawCircle(0,0,10);
                  circle.graphics.endFill();
                  circle.x = 0;
                  circle.y = 0;
                  addChild(circle);
                  circle.x=150;
                  circle.y=150;
                  rect.graphics.beginFill(0x008000,100);
                  rect.graphics.drawRect(0,0,50,20);
                  rect.graphics.endFill();
```

```
        rect.x = 0;
        rect.y = 0;
        addChild(rect);
        rect.x=400;
        rect.y=400;
        addChild(circlesquare);
        circlesquare.addChild(circle);
        for (var k:int = 1;
        k < 23;
        k++){
              var s:Sprite = new Sprite();
              s.graphics.beginFill(0xA9A9A9,100);
              s.graphics.drawRect(0,0,7,7);
              s.graphics.endFill();
              s.x = 0;
              s.y = 0;
              addChild(s);
              var s2:Sprite = new Sprite();
              s2.graphics.beginFill(0xADD8E6,100);
              s2.graphics.drawRect(0,0,7,7);
              s2.graphics.endFill();
              s2.x = 0;
              s2.y = 0;
              addChild(s2);
              var a:Number = 0;
              a=k*.3;
              var cx:Number = 0;
              cx=Math.cos(a);
              var sy:Number = 0;
              sy=Math.sin(a);
              cx=cx*80;
              cx=cx+200;
              sy=sy*80;
              sy=sy+125;
              s.x=cx;
              s.y=sy;
              cx=cx-2;
              sy=sy-2;
              s2.x=cx;
              s2.y=sy;
              circlesquare.addChild(s);
              circlesquare.addChild(s2);
        }
        addEventListener(Event.ENTER_FRAME, animate);
  }
  public function animate(event:Event):void {
        inc=5;
        var x:Number = 0;
        x=circlesquare.x;
        var y:Number = 0;
        y=circlesquare.y;
        angle=angle+.1;
        var cx:Number = 0;
        cx=Math.cos(angle);
        cx=cx*100;
        var sy:Number = 0;
        sy=Math.sin(angle);
```

```
                  sy=sy*100;
                  circlesquare.x=cx;
                  circlesquare.y=sy;
                  x=rect.x;
                  y=rect.y;
                  angle2=angle2-.2;
                  cx=Math.cos(angle2);
                  cx=cx*50;
                  cx=cx+200;
                  sy=Math.sin(angle2);
                  sy=sy*50;
                  sy=sy+200;
                  rect.x=cx;
                  rect.y=sy;
                  y=square.y;
                  if (y>500) {
                        y=-175;
                  }
                  y=y+inc;
                  square.y=y;
            }
            public var square:Sprite = new Sprite();
            public var angle2:Number = 0;
            public var angle:Number = 0;
            public var inc:Number = 0;
            public var circle:Sprite = new Sprite();
            public var rect:Sprite = new Sprite();
            public var circlesquare:Sprite = new Sprite();
      }
}
```

7. Lessons Learned


Microsoft Office Word

Word experienced an error trying to open the file.
Try these suggestions.
* Check the file permissions for the document or drive.
* Make sure there is sufficient free memory and disk space.
* Open the file with the Text Recovery converter.

It is difficult to overstate the importance of versioning every file in your project, including the final report. After many pages and long nights, I went to output my report to `.pdf` and received the dialog box above from MS Word. I tried opening the file on several different machines and OSes to no avail. Instructions on how to repair damaged word docs (how did I damage it?) from the Microsoft website[9] did not work. Fortunately, I had put my report in CVS and rolled back to a previous version, and only lost a few hours of work. In the future, it may be better to just avoid writing large, important reports in MS Word altogether.

In addition to regular advice like "start early" and "plan ahead", I would add "ask questions" early also. While the lexer and parser for my project came together quickly, I struggled with the walker for a couple of weeks. After a brief office hours visit with Professor Edwards, I was able to get the walker pretty-printing my test programs in a couple of days.

Study the mx and dragon book ANTLR examples. I added breakpoints and ran programs from both in Eclipse with the debugger on which was extremely helpful in understanding the program flow. Outputting the parse trees to xml and viewing the structure was also critical for my project.

[9] http://support.microsoft.com/kb/826864

But at the end of day, be sure to put everything in version control.

Appendix

Antonio Cruz
adc2104@columbia.edu

1. ANTLR grammar file

grammar-as-semant.g

```
/************************************************************
 * SfalLexer
 * @author Antonio Cruz adc2104@columbia.edu
 * @version $Id: grammar-as-semant.g,v 1.28 2007/12/16 07:27:54 admin
Exp $
 ************************************************************/

class SfalLexer extends Lexer;

options {
    k = 2;
    charVocabulary = '\3'..'\377';
    testLiterals = false;
    exportVocab = Sfal;
}

protected
ALPHA   : 'a'..'z' | 'A'..'Z' | '_' ;

protected
DIGIT   : '0'..'9';

protected
HEX         : '0'..'9' | 'A'..'F' | 'a'..'f' ;

HEXNUM      : "0x" HEX HEX HEX HEX HEX HEX ;

ID  options { testLiterals = true; }
        : ALPHA (ALPHA|DIGIT)*
        ;

NUMBER  : (DIGIT)+ ('.' (DIGIT)* )?
            | '.' (DIGIT)+
            ;

WS      : (' ' | '\t')+
                { $setType(Token.SKIP); }
        ;
```

```
NL      : ('\n' | ('\r' '\n') => '\r' '\n' | '\r')
                  { $setType(Token.SKIP); newline(); }
        ;

COMMENT : (
                  "/*"
                        ( options {greedy=false;}
                        : (NL)
                        | ~( '\n' | '\r' ))*
                  "*/"

            | "//" (~( '\n' | '\r' ))* (NL)

          ) { $setType(Token.SKIP); }
        ;

LPAREN      : '(';
RPAREN  : ')';
LBRACE  : '{';
RBRACE  : '}';

MULT    : '*';
PLUS    : '+';
MINUS   : '-';
DIV     : '/';
MOD     : '%';

SEMI    : ';';
ASGN    : '=';
COMMA   : ',';

GE      : ">=";
LE      : "<=";
GT      : '>';
LT      : '<';

EQ      : "==";
NEQ     : "!=";

OR          : "||";
AND   : "&&";


/* mx */
STRING  : '"'!
             (  ~('"' | '\n' | '#')
              | ('"'!'"')
             )*
          '"'!
        ;

/* Esterel
StringConstant    : '"'!
                              ( ~('"' | '\n')
                              | ('"'! '"')
                              )*
                              '"'!
                        ;
*/

/*************************************************************

      SfalParser

      program -> (statement)*

      statement ->
            if-stmt |
            assignment |
            func-call-stmt |
            loop_stmt |
            animator ;

      if-stmt -> 'if' '(' expression ')' block-stmt ( 'else' block-stmt
)?

      block-stmt -> '{' (statement)* '}'

      assignment -> l-value ('=') expression ';'

      func-call-stmt -> func-call ';'

      func-call -> ID '(' expr-list ')'

      expr-list -> expression (',' expression)* | e

      animator -> 'animate' block-stmt

      loop_stmt -> 'loop' '(' expression ')' block-stmt

      expression -> logic-term ( '||' logic-term )*

      logic-term -> logic-factor ( '&&' logic-factor )*

      logic-factor -> ('not')? relat-expr

      relat-expr -> arith-expr ( ('>=' | '<=' | '>' | '<' | '==' |
'!=') arith-expr )?

      arith-expr -> arith-term ( ('+' | '-') arith-term )*

      arith-term -> arith-factor ( ('*' | '/' | '%') arith-factor )*

      arith-factor -> ('-') r-value

      r-value -> l-value | func-call | constant

      l-value -> ID

      constant -> NUMBER | HEXNUM

 *************************************************************/
```

```
class SfalParser extends Parser;

options {
    k = 2;
    buildAST = true;
}

tokens {
   PROG;
   STATEMENT;
   BLOCK_STMT;
   FUNC_CALL;
   EXPR_LIST;
   UPLUS;
   UMINUS;
}


program
        : ( statement )*
          {#program = #([PROG, "PROG"], program); }
        ;

statement
        : if_stmt
        | assignment
        | func_call_stmt
        | loop_stmt
        | animator
        ;

if_stmt
        : "if"^ LPAREN! expression RPAREN! (block_stmt | statement)
            (options {greedy = true;}: "else"! (block_stmt | statement)
)?
        ;

block_stmt
        : LBRACE! (statement)* RBRACE!
            { #block_stmt = #([BLOCK_STMT, "BLOCK_STMT"], block_stmt);
}
        ;

assignment
        : l_value ( ASGN^ ) expression SEMI!
        ;

func_call_stmt
        : func_call SEMI!
        ;

func_call
        : ID LPAREN! expr_list RPAREN!
            {#func_call = #([FUNC_CALL, "FUNC_CALL"], func_call); }
        ;

expr_list
```

```
        : expression ( COMMA! expression )*
            {#expr_list = #([EXPR_LIST, "EXPR_LIST"], expr_list); }

        | /* necessary for function with no arguments */
            {#expr_list = #([EXPR_LIST,"EXPR_LIST"], expr_list); }
        ;

animator
        : "animate"^ block_stmt
        ;

loop_stmt
        : "loop"^ LPAREN! expr_list RPAREN! block_stmt
        ;


/*****************************************************************

      Expressions

 *****************************************************************/

expression
        : logic_term ( OR^ logic_term )*
        ;

logic_term
        : logic_factor ( AND^ logic_factor )*
        ;

logic_factor
        : ("not"^)? relat_expr
        ;

relat_expr
        : arith_expr ( (GE^ | LE^ | GT^ | LT^ | EQ^ | NEQ^) arith_expr
)?
        ;

arith_expr
        : arith_term ( (PLUS^ | MINUS^) arith_term )*
        ;

arith_term
        : arith_factor ( (MULT^ | LDV^ | MOD^ | DIV^) arith_factor )*
        ;

arith_factor
        : PLUS! r_value
            {#arith_factor = #([UPLUS, "UPLUS"], arith_factor); }

        | MINUS! r_value
            {#arith_factor = #([UMINUS, "UMINUS"], arith_factor); }

        | r_value
        ;
```

```
r_value
        : l_value
        | func_call
        | NUMBER
        | STRING
        | "true"
        | "false"
        | HEXNUM
        | LPAREN! expression RPAREN!
        ;

l_value
        : ID
        ;


/**************************************************************

      TreeWalker

 **************************************************************/
class SfalWalker extends TreeParser;

options {
    importVocab = Sfal;
}

{
  SymbolTable top = null;
  Validator validator = new Validator(true);
  Translator translator = null;
  Debug log = new Debug(false);
  boolean isAnimator = false;
  boolean isIfBlock = false;
  StringBuffer programBuffer = new StringBuffer();
}

program
{
      String s = null;
      /*programBuffer.append("package {\n" +
            "import flash.display.Sprite;\n" +
        "import flash.text.TextField;\n" +
        "import flash.events.Event;\n\n" +
        "public class {0} extends Sprite {\n\n" +
        "public function {0}() {\n"
        );*/
    }

      : #(PROG
            {
                  SymbolTable saved_environment = top; top = new
SymbolTable(top);
                  translator = new Translator(top);
            }

            (s=stmt)*
            { top = saved_environment; }
            )
      {

      programBuffer.append("addEventListener(Event.ENTER_FRAME,
animate);");
      programBuffer.append("\n}");


      programBuffer.append("public function animate(event:Event):void
{");

      programBuffer.append(translator.translateAnimator());

      programBuffer.append("\n}");

      // System.out.println("GRAMMAR:TOP: " + top);

      programBuffer.append(translator.translateSymbolTable());


      programBuffer.append("\n} ");
      programBuffer.append("\n} ");
      }
      ;

stmt returns [String s]
      {
            s = null;
            String s1=null, s2=null, s3=null; // e=null;
            Expr e1, e2;
            java.util.LinkedList l = new java.util.LinkedList();
      }

      :     #(ASGN e1=expr e2=expr )
            {
                  /* ID = FUNC_CALL | ID | LITERAL | EXPR */

                  // System.err.println("ASGN: E1: " + e1.toString() +
" E2: "  + e2.toString());

                  //--------------------------------------------------
                  // Handle SymbolTable
                  //--------------------------------------------------

                  if (e2.type.equals("FUNC_CALL")) {

                        // System.out.println("in symtable already: " +
e1.value + ": " + top.get(e1.value));


                        /* local variable declarations */
                        boolean shoulddeclare =
                              /* variable e1.value not in symboltable
yet */
                              (top.get(e1.value) == null)
                              /* This means we're in a new scope,
declare
                              local variables if necessary */
                              && (top.outer != null);

                        /* Map ID to the return type of the function
call */
                        top.put(e1.value,
                              (String) Meta.FUNC_TO_TYPE.get(e2.value),
null);

                        if (shoulddeclare) {

                              if (!isAnimator) {

      programBuffer.append(translator.translateSymbolTableLocal());
                              }
                              else {

      translator.buildAnimator(translator.translateSymbolTableLocal());
                              }

                        }
                  }
                  else {


                        /*
                        Map ID to "type" associated to the node name,
e.g. "Number", "Sprite"
                        If e2 is an ID, get the type from the
SymbolTable
                        */

                        if (e2.type.equals("ID")) {
                              SymbolTable.SymTabEntry t =
(SymbolTable.SymTabEntry) top.get(e2.value);
                              if (t==null) throw new
RecognitionException("undeclared right value ID: "
                                    + e2.value);
                        }
                        else {
                              top.put(e1.value, (String)
Meta.NODE_TO_TYPE.get(e2.type), e2.value);

                              if (!isAnimator) {

      programBuffer.append(translator.translateSymbolTableLocal());
                              }
                              else {

      translator.buildAnimator(translator.translateSymbolTableLocal());
                              }

                        }
                  }

                  // System.out.println("GRAMMAR:TOP: " + top);

                  //-------------------------------------------------
                  // END Handle SymbolTable
                  //-------------------------------------------------

                  if (isIfBlock) {/* do nothing */}
                  else {
                        if (!isAnimator) {
                              programBuffer.append(
                              translator.translate(ASGN, e1.toString()
+ e2.toString()));
                        }
                        else {
                              translator.buildAnimator(
                              translator.translate(ASGN, e1.toString()
+ e2.toString()));
                        }
                  }


                  if (!validator.validate(ASGN, e1, e2, top))
                  throw new RecognitionException("BAD Assignment: " +
e1.error);


                  // return this for if statement
                  s = translator.translate(ASGN, e1.toString() +
e2.toString());
                  log.trace(top);
            }
      |     #("if"      {
                        isIfBlock=true;
                        StringBuffer ifBuffer = new StringBuffer("if
(");
                        }
                  e1=expr {ifBuffer.append(e1).append(") {");}
                  s2=stmt
                  {     ifBuffer.append(s2).append('}');
                        if (_t != null) {
                              ifBuffer.append(" else {");
                        }
                  }
                  (s3=stmt)? ) {
                        /*if (_t != null) {
                              ifBuffer.append("}");
                        }*/

                  if (!isAnimator) {
                        programBuffer.append(
                        ifBuffer.toString());
                  }
                  else {
                        translator.buildAnimator(
                        ifBuffer.toString());
                  }
                  isIfBlock=false;

                  }

      |     #("loop" { }
                  l=expr_list {

                  if (!validator.validateLoop(l))
            throw new RecognitionException("BAD LOOP format");

                  programBuffer.append(translator.translate(1001,
                        ((Expr)l.get(0)).toString() +
                        ((Expr)l.get(1)).toString() +
                        ((Expr)l.get(2)).toString())); }

                  s2=stmt )

      |     #(BLOCK_STMT      {if (!isAnimator)
programBuffer.append("{");}

                  {
                  SymbolTable saved_environment = top;
                  top = new SymbolTable(top);
                  /* Update the translator w/ the SymbolTable for the
new scope,
                  in case we need to declare local variables */
                  translator.symTable=top;
                  }

                  (s=stmt)* ) {if (!isAnimator)
programBuffer.append("}");}

                  {
                  top = saved_environment;
                  /* Update the translator with the saved SymbolTable
*/
                  translator.symTable=top;
                  }

      |     #("animate" {isAnimator=true;}
                  s2=stmt  ) {isAnimator=false;}
      ;


expr returns [Expr e]
      {
            e = null;
            String s;
            Expr a, b;
            java.util.LinkedList l;
      }

      :     #(FUNC_CALL ID l=expr_list)
            {
                  e = new Expr("FUNC_CALL");
                  e.value = #ID.getText();
                  e.values = l;

                  if (!validator.validate(FUNC_CALL, e, top))
                        throw new RecognitionException("BAD FUNC_CALL:
" + e.value);
            }
      |     #(MULT a=expr b=expr)
            {
                  e = new Expr("*");
                  e.child1=a;
            e.child2=b;

            // System.out.println("MULT: " + e);
                  // System.out.println("Top: " + top);

                  if (!validator.validate(MULT, e, top))
                        throw new RecognitionException("RE: *: " +
e.error);
            }
    |       #(DIV a=expr b=expr)
      {
            e = new Expr("/");
                  e.child1=a;
            e.child2=b;
                  if (!validator.validate(DIV, e, top))
                        throw new RecognitionException("RE: /: " +
e.error);
            }
    |       #(MOD a=expr b=expr)
      {
            e = new Expr("%");
                  e.child1=a;
            e.child2=b;
            if (!validator.validate(MOD, e, top))
                        throw new RecognitionException("RE: %: " +
e.error);
            }
    |       #(PLUS a=expr b=expr)
      {
            e = new Expr("+");
            e.child1=a;
            e.child2=b;
            if (!validator.validate(PLUS, e, top))
                        throw new RecognitionException("RE: +: " +
e.error);
            }
      |     #(MINUS a=expr b=expr)
            {
                  e = new Expr("-");
                  e.child1=a;
            e.child2=b;
                  /* if (!validator.validate(MINUS, e, top))
                        throw new RecognitionException("RE: -: " +
e.error); */
            }
      |     #(UMINUS a=expr)
```

```
            {
                  e = new Expr("U-");
                  e.child1=a;

                  /* fix
                  if (!validator.validate(UMINUS, e, top))
                        throw new RecognitionException("RE: U-: " +
e.error); */
            }
      |     #(AND a=expr b=expr)
            {
                  e = new Expr("&&");
                  e.child1=a;
            e.child2=b;
                  if (!validator.validate(AND, e, top))
                        throw new RecognitionException("RE: &&: " +
e.error);
            }
      |     #(OR a=expr b=expr)
            {
                  e = new Expr("||");
                  e.child1=a;
            e.child2=b;
                  if (!validator.validate(OR, e, top))
                        throw new RecognitionException("RE: ||: " +
e.error);
            }
      |     #(GT a=expr b=expr)
            {
                  e = new Expr(">");
                  e.child1=a;
            e.child2=b;

            /* fix
                  if (!validator.validate(GT, e, top))
                        throw new RecognitionException("RE: >: " +
e.error); */
            }
      |     #(LT a=expr b=expr)
            {
                  e = new Expr("<");
                  e.child1=a;
            e.child2=b;
            /* fix
                  if (!validator.validate(LT, e, top))
                        throw new RecognitionException("RE: <: " +
e.error); */
            }
      |     #(EQ a=expr b=expr)
            {
                  e = new Expr("==");
                  e.child1=a;
            e.child2=b;
                  if (!validator.validate(EQ, e, top))
                        throw new RecognitionException("RE: ==: " +
e.error);
            }
```

```
        |      #(NEQ a=expr b=expr)
               {
                     e = new Expr("!=");
                     e.child1=a;
               e.child2=b;
                     if (!validator.validate(NEQ, e, top))
                           throw new RecognitionException("RE: !=: " +
e.error);
               }
        |      #(GE a=expr b=expr)
               {
                     e = new Expr(">=");
                     e.child1=a;
               e.child2=b;
                     if (!validator.validate(GE, e, top))
                           throw new RecognitionException("RE: >=: " +
e.error);
               }
        |      #(LE a=expr b=expr)
               {
                     e = new Expr("<=");
                     e.child1=a;
               e.child2=b;
                     if (!validator.validate(LE, e, top))
                           throw new RecognitionException("RE: <=: " +
e.error);
               }

        |      num:NUMBER
               {
                     e = new Expr("Number");
                     e.value=num.getText();
               }
        |      str:STRING
               {
                     e = new Expr("String");
                     e.value=str.getText();
               }
        |      hex:HEXNUM
               {
                     e = new Expr("Hex");
                     e.value=hex.getText();
               }
        |      "true"
               {
                     e = new Expr("true");
                     e.value="true";
               }
        |      "false"
               {
                     e = new Expr("false");
                     e.value = "false";
               }
        |      id:ID
               {
                     /* since untyped do this in the ASGN stmt
```

```
                  s = top.get(id.getText()) == null ? null :
id.getText();
                  if (s == null) {
                        /* don't want to do this with an expression
list
                        there, we are only reading the id
                        otherwise, we can be declaring it

                        top.put(id.getText(), "ID");
                  }*/

                  e = new Expr("ID");
                  e.value = id.getText();

                  /*
                  System.out.println("SymbolTable: ");
                  System.out.println(top);*/
            }

      ;


expr_list returns [java.util.LinkedList l]
{
      l = new java.util.LinkedList();
      Expr e = null;
}
      :     #(EXPR_LIST
            (e=expr     { l.addLast(e); })* )
      ;
```

2. Debug.java

Debug.java

```
/**
 * Debug
 *
 * @author Antonio Cruz adc2104@columbia.edu
 * @version $Id: Debug.java,v 1.3 2007/12/09 17:14:06 admin Exp $
 */
public class Debug {

	public Debug(boolean b) {
		debug = b;
	}

	boolean debug;

	void trace(Object s) {
		if (debug) System.out.println(s);
	}
}
```

3. Expr.java

Expr.java

```
/**
 * Represent an SFAL expression.
 *
 * @author Antonio Cruz adc2104@columbia.edu
 * @version $Id: Expr.java,v 1.11 2007/12/16 07:30:45 admin Exp $
 */

import java.util.LinkedList;
import java.util.Iterator;


public class Expr {
      /**
       * String representation of expression node,
       * e.g. "+", "FUNC_CALL", "ID".
       */
      String type;

      /**
       * Value of expression node for "ID" and "FUNC_CALL",
       * e.g. "x", "makecircle".
       */
      String value;

      /**
       * List of expression node's children, e.g.
       * for "+": Expr operand1, Expr operand2.
       */
      LinkedList values = new LinkedList();


      public Expr(String t) {
            type = t;
      }

      public String toString() {
            StringBuffer sb = new StringBuffer();

            if (value != null) {
                  sb.append(value).append(',');
            }

            /* For operations w/ 2 operands */
            if (child1 != null) {
                  if (type.equals("U-")) sb.append('-');
                  sb.append(child1.value);
            }
            if (child1 != null && child2 != null) sb.append(type);
            if (child2 != null) {
                  sb.append(child2.value);
            }


            /* For expression lists */
            for (Iterator itr = values.iterator(); itr.hasNext(); ) {
                  Expr e = (Expr) itr.next();
                  sb.append(e.toString());
            }

            return sb.toString();
      }

      /* For operations w/ 2 operands */
      Expr child1;
      Expr child2;
      String error;
}
```

4. Meta.java

Meta.java

```
/**
 * Static mappings
 *
 * @author Antonio Cruz adc2104@columbia.edu
 * @version $Id: Meta.java,v 1.7 2007/12/16 08:06:53 admin Exp $
 */

import java.util.HashMap;
import java.util.Map;

public class Meta {

      /** Map Sfal functions to arg Number and types
       * This is used for validating functions.
       */
      static final Map SFAL_FUNCTIONS = new HashMap();
      static {
            SFAL_FUNCTIONS.put("makesquare", "Number,String|Hex");
            SFAL_FUNCTIONS.put("makecircle", "Number,String|Hex");
            SFAL_FUNCTIONS.put("makerect", "Number,Number,String|Hex");

            SFAL_FUNCTIONS.put("makegroup", "None");
            SFAL_FUNCTIONS.put("addtogroup", "Sprite,Sprite");

            SFAL_FUNCTIONS.put("maketext", "String");
            SFAL_FUNCTIONS.put("addtotext", "TextField,String");

            SFAL_FUNCTIONS.put("makemovie", "String");
            SFAL_FUNCTIONS.put("globalnumber", "None");

            SFAL_FUNCTIONS.put("setx", "Sprite,Number");
            SFAL_FUNCTIONS.put("sety", "Sprite,Number");

            SFAL_FUNCTIONS.put("settextx", "TextField,Number");
            SFAL_FUNCTIONS.put("settexty", "TextField,Number");

            SFAL_FUNCTIONS.put("getx", "Sprite");
            SFAL_FUNCTIONS.put("gety", "Sprite");

            SFAL_FUNCTIONS.put("gettextx", "TextField");
            SFAL_FUNCTIONS.put("gettexty", "TextField");

            SFAL_FUNCTIONS.put("sin", "Number");
            SFAL_FUNCTIONS.put("cos", "Number");
      }
      /**
       * Map non-function call node types to corresponding ActionScript
objects
       */
      public static final Map NODE_TO_TYPE = new HashMap();
      static {
            NODE_TO_TYPE.put("*", "Number");
            NODE_TO_TYPE.put("+", "Number");
            NODE_TO_TYPE.put("-", "Number");
            NODE_TO_TYPE.put("/", "Number");
            NODE_TO_TYPE.put("%", "Number");

            // Number token literal
            NODE_TO_TYPE.put("Number", "Number");

            // String token literal
            NODE_TO_TYPE.put("String", "String");
      }

      /**
       * Map Sfal function return types to corresponding ActionScript
objects
       *
       */
      public static final Map FUNC_TO_TYPE = new HashMap();
      static {
            FUNC_TO_TYPE.put("makemovie", "");
            FUNC_TO_TYPE.put("globalnumber", "Number");

            FUNC_TO_TYPE.put("makesquare", "Sprite");
            FUNC_TO_TYPE.put("makecircle", "Sprite");
            FUNC_TO_TYPE.put("makerect", "Sprite");

            FUNC_TO_TYPE.put("maketext", "TextField");
            FUNC_TO_TYPE.put("addtotext", "TextField");

            FUNC_TO_TYPE.put("makegroup", "Sprite");
            FUNC_TO_TYPE.put("addtogroup", "Sprite");

            FUNC_TO_TYPE.put("setx", "Sprite");
            FUNC_TO_TYPE.put("sety", "Sprite");

            FUNC_TO_TYPE.put("getx", "Number");
            FUNC_TO_TYPE.put("gety", "Number");

            FUNC_TO_TYPE.put("settextx", "TextField");
            FUNC_TO_TYPE.put("settexty", "TextField");

            FUNC_TO_TYPE.put("gettextx", "Number");
            FUNC_TO_TYPE.put("gettexty", "Number");

            FUNC_TO_TYPE.put("sin", "Number");
            FUNC_TO_TYPE.put("cos", "Number");
      }


      /**
```

```
 * Basic web colors name to hex mappings.
 * From site: http://www.htmlgoodies.com
 */
public static final Map WEB_COLORS = new HashMap();
static {
      WEB_COLORS.put("Aliceblue".toLowerCase(), "0xF0F8FF");
      WEB_COLORS.put("Antiquewhite".toLowerCase(), "0xFAEBD7");
      WEB_COLORS.put("Aqua".toLowerCase(), "0x00FFFF");
      WEB_COLORS.put("Aquamarine".toLowerCase(), "0x7FFFD4");
      WEB_COLORS.put("Azure".toLowerCase(), "0xF0FFFF");
      WEB_COLORS.put("Beige".toLowerCase(), "0xF5F5DC");
      WEB_COLORS.put("Bisque".toLowerCase(), "0xFFE4C4");
      WEB_COLORS.put("Black".toLowerCase(), "0x000000");
      WEB_COLORS.put("Blanchedalmond".toLowerCase(), "0xFFEBCD");
      WEB_COLORS.put("Blue".toLowerCase(), "0x0000FF");
      WEB_COLORS.put("Blueviolet".toLowerCase(), "0x8A2BE2");
      WEB_COLORS.put("Brown".toLowerCase(), "0xA52A2A");
      WEB_COLORS.put("Burlywood".toLowerCase(), "0xDEB887");
      WEB_COLORS.put("Cadetblue".toLowerCase(), "0x5F9EA0");
      WEB_COLORS.put("Chartreuse".toLowerCase(), "0x7FFF00");
      WEB_COLORS.put("Chocolate".toLowerCase(), "0xD2691E");
      WEB_COLORS.put("Coral".toLowerCase(), "0xFF7F50");
      WEB_COLORS.put("Cornflowerblue".toLowerCase(), "0x6495ED");
      WEB_COLORS.put("Cornsilk".toLowerCase(), "0xFFF8DC");
      WEB_COLORS.put("Crimson".toLowerCase(), "0xDC143C");
      WEB_COLORS.put("Cyan".toLowerCase(), "0x00FFFF");
      WEB_COLORS.put("Darkblue".toLowerCase(), "0x00008B");
      WEB_COLORS.put("Darkcyan".toLowerCase(), "0x008B8B");
      WEB_COLORS.put("Darkgoldenrod".toLowerCase(), "0xB8860B");
      WEB_COLORS.put("Darkgray".toLowerCase(), "0xA9A9A9");
      WEB_COLORS.put("Darkgreen".toLowerCase(), "0x006400");
      WEB_COLORS.put("Darkkhaki".toLowerCase(), "0xBDB76B");
      WEB_COLORS.put("Darkmagenta".toLowerCase(), "0x8B008B");
      WEB_COLORS.put("Darkolivegreen".toLowerCase(), "0x556B2F");
      WEB_COLORS.put("Darkorange".toLowerCase(), "0xFF8C00");
      WEB_COLORS.put("Darkorchid".toLowerCase(), "0x9932CC");
      WEB_COLORS.put("Darkred".toLowerCase(), "0x8B0000");
      WEB_COLORS.put("Darksalmon".toLowerCase(), "0xE9967A");
      WEB_COLORS.put("Darkseagreen".toLowerCase(), "0x8FBC8F");
      WEB_COLORS.put("Darkslateblue".toLowerCase(), "0x483D8B");
      WEB_COLORS.put("Darkslategray".toLowerCase(), "0x2F4F4F");
      WEB_COLORS.put("Darkturquoise".toLowerCase(), "0x00CED1");
      WEB_COLORS.put("Darkviolet".toLowerCase(), "0x9400D3");
      WEB_COLORS.put("Deeppink".toLowerCase(), "0xFF1493");
      WEB_COLORS.put("Deepskyblue".toLowerCase(), "0x00BFFF");
      WEB_COLORS.put("Dimgray".toLowerCase(), "0x696969");
      WEB_COLORS.put("Dodgerblue".toLowerCase(), "0x1E90FF");
      WEB_COLORS.put("Firebrick".toLowerCase(), "0xB22222");
      WEB_COLORS.put("Floralwhite".toLowerCase(), "0xFFFAF0");
      WEB_COLORS.put("Forestgreen".toLowerCase(), "0x228B22");
      WEB_COLORS.put("Fuchsia".toLowerCase(), "0xFF00FF");
      WEB_COLORS.put("Gainsboro".toLowerCase(), "0xDCDCDC");
      WEB_COLORS.put("Ghostwhite".toLowerCase(), "0xF8F8FF");
      WEB_COLORS.put("Gold".toLowerCase(), "0xFFD700");
      WEB_COLORS.put("Goldenrod".toLowerCase(), "0xDAA520");
      WEB_COLORS.put("Gray".toLowerCase(), "0x808080");
      WEB_COLORS.put("Green".toLowerCase(), "0x008000");
```

```
            WEB_COLORS.put("Greenyellow".toLowerCase(), "0xADFF2F");
            WEB_COLORS.put("Honeydew".toLowerCase(), "0xF0FFF0");
            WEB_COLORS.put("Hotpink".toLowerCase(), "0xFF69B4");
            WEB_COLORS.put("Indianred".toLowerCase(), "0xCD5C5C");
            WEB_COLORS.put("Indigo".toLowerCase(), "0x4B0082");
            WEB_COLORS.put("Ivory".toLowerCase(), "0xFFFFF0");
            WEB_COLORS.put("Khaki".toLowerCase(), "0xF0E68C");
            WEB_COLORS.put("Lavender".toLowerCase(), "0xE6E6FA");
            WEB_COLORS.put("Lavenderblush".toLowerCase(), "0xFFF0F5");
            WEB_COLORS.put("Lawngreen".toLowerCase(), "0x7CFC00");
            WEB_COLORS.put("Lemonchiffon".toLowerCase(), "0xFFFACD");
            WEB_COLORS.put("lightblue".toLowerCase(), "0xADD8E6");
            WEB_COLORS.put("Lightcoral".toLowerCase(), "0xF08080");
            WEB_COLORS.put("Lightcyan".toLowerCase(), "0xE0FFFF");
            WEB_COLORS.put("Lightgoldenrodyellow".toLowerCase(),
"0xFAFAD2");
            WEB_COLORS.put("Lightgreen".toLowerCase(), "0x90EE90");
            WEB_COLORS.put("Lightgrey".toLowerCase(), "0xD3D3D3");
            WEB_COLORS.put("Lightpink".toLowerCase(), "0xFFB6C1");
            WEB_COLORS.put("Lightsalmon".toLowerCase(), "0xFFA07A");
            WEB_COLORS.put("Lightseagreen".toLowerCase(), "0x20B2AA");
            WEB_COLORS.put("Lightskyblue".toLowerCase(), "0x87CEFA");
            WEB_COLORS.put("Lightslategray".toLowerCase(), "0x778899");
            WEB_COLORS.put("Lightsteelblue".toLowerCase(), "0xB0C4DE");
            WEB_COLORS.put("Lightyellow".toLowerCase(), "0xFFFFE0");
            WEB_COLORS.put("Lime".toLowerCase(), "0x00FF00");
            WEB_COLORS.put("Limegreen".toLowerCase(), "0x32CD32");
            WEB_COLORS.put("Linen".toLowerCase(), "0xFAF0E6");
            WEB_COLORS.put("Magenta".toLowerCase(), "0xFF00FF");
            WEB_COLORS.put("Maroon".toLowerCase(), "0x800000");
            WEB_COLORS.put("Mediumauqamarine".toLowerCase(),
"0x66CDAA");
            WEB_COLORS.put("Mediumblue".toLowerCase(), "0x0000CD");
            WEB_COLORS.put("Mediumorchid".toLowerCase(), "0xBA55D3");
            WEB_COLORS.put("Mediumpurple".toLowerCase(), "0x9370D8");
            WEB_COLORS.put("Mediumseagreen".toLowerCase(), "0x3CB371");
            WEB_COLORS.put("Mediumslateblue".toLowerCase(),
"0x7B68EE");
            WEB_COLORS.put("Mediumspringgreen".toLowerCase(),
"0x00FA9A");
            WEB_COLORS.put("Mediumturquoise".toLowerCase(),
"0x48D1CC");
            WEB_COLORS.put("Mediumvioletred".toLowerCase(),
"0xC71585");
            WEB_COLORS.put("Midnightblue".toLowerCase(), "0x191970");
            WEB_COLORS.put("Mintcream".toLowerCase(), "0xF5FFFA");
            WEB_COLORS.put("Mistyrose".toLowerCase(), "0xFFE4E1");
            WEB_COLORS.put("Moccasin".toLowerCase(), "0xFFE4B5");
            WEB_COLORS.put("Navajowhite".toLowerCase(), "0xFFDEAD");
            WEB_COLORS.put("Navy".toLowerCase(), "0x000080");
            WEB_COLORS.put("Oldlace".toLowerCase(), "0xFDF5E6");
            WEB_COLORS.put("Olive".toLowerCase(), "0x808000");
            WEB_COLORS.put("Olivedrab".toLowerCase(), "0x688E23");
            WEB_COLORS.put("Orange".toLowerCase(), "0xFFA500");
            WEB_COLORS.put("Orangered".toLowerCase(), "0xFF4500");
            WEB_COLORS.put("Orchid".toLowerCase(), "0xDA70D6");
            WEB_COLORS.put("Palegoldenrod".toLowerCase(), "0xEEE8AA");
            WEB_COLORS.put("Palegreen".toLowerCase(), "0x98FB98");
            WEB_COLORS.put("Paleturquoise".toLowerCase(), "0xAFEEEE");
            WEB_COLORS.put("Palevioletred".toLowerCase(), "0xD87093");
            WEB_COLORS.put("Papayawhip".toLowerCase(), "0xFFEFD5");
            WEB_COLORS.put("Peachpuff".toLowerCase(), "0xFFDAB9");
            WEB_COLORS.put("Peru".toLowerCase(), "0xCD853F");
            WEB_COLORS.put("Pink".toLowerCase(), "0xFFC0CB");
            WEB_COLORS.put("Plum".toLowerCase(), "0xDDA0DD");
            WEB_COLORS.put("Powderblue".toLowerCase(), "0xB0E0E6");
            WEB_COLORS.put("Purple".toLowerCase(), "0x800080");
            WEB_COLORS.put("Red".toLowerCase(), "0xFF0000");
            WEB_COLORS.put("Rosybrown".toLowerCase(), "0xBC8F8F");
            WEB_COLORS.put("Royalblue".toLowerCase(), "0x4169E1");
            WEB_COLORS.put("Saddlebrown".toLowerCase(), "0x8B4513");
            WEB_COLORS.put("Salmon".toLowerCase(), "0xFA8072");
            WEB_COLORS.put("Sandybrown".toLowerCase(), "0xF4A460");
            WEB_COLORS.put("Seagreen".toLowerCase(), "0x2E8B57");
            WEB_COLORS.put("Seashell".toLowerCase(), "0xFFF5EE");
            WEB_COLORS.put("Sienna".toLowerCase(), "0xA0522D");
            WEB_COLORS.put("Silver".toLowerCase(), "0xC0C0C0");
            WEB_COLORS.put("Skyblue".toLowerCase(), "0x87CEEB");
            WEB_COLORS.put("Slateblue".toLowerCase(), "0x6A5ACD");
            WEB_COLORS.put("Slategray".toLowerCase(), "0x708090");
            WEB_COLORS.put("Snow".toLowerCase(), "0xFFFAFA");
            WEB_COLORS.put("Springgreen".toLowerCase(), "0x00FF7F");
            WEB_COLORS.put("Steelblue".toLowerCase(), "0x4682B4");
            WEB_COLORS.put("Tan".toLowerCase(), "0xD2B48C");
            WEB_COLORS.put("Teal".toLowerCase(), "0x008080");
            WEB_COLORS.put("Thistle".toLowerCase(), "0xD8BFD8");
            WEB_COLORS.put("Tomato".toLowerCase(), "0xFF6347");
            WEB_COLORS.put("Turquoise".toLowerCase(), "0x40E0D0");
            WEB_COLORS.put("Violet".toLowerCase(), "0xEE82EE");
            WEB_COLORS.put("Wheat".toLowerCase(), "0xF5DEB3");
            WEB_COLORS.put("White".toLowerCase(), "0xFFFFFF");
            WEB_COLORS.put("Whitesmoke".toLowerCase(), "0xF5F5F5");
            WEB_COLORS.put("Yellow".toLowerCase(), "0xFFFF00");
            WEB_COLORS.put("YellowGreen".toLowerCase(), "0x9ACD32");
      }
}
```

5. SFAL.java

SFAL.java

```
/**
 * Main class
 *
 * @author Antonio Cruz adc2104@columbia.edu
 * @version $Id: SFAL.java,v 1.3 2007/12/16 07:30:46 admin Exp $
 */

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.StringReader;
import java.io.Writer;
import java.text.*;

import antlr.CommonAST;

public class SFAL {
    static boolean DEBUG = false;

    public static void main(String[] args) {
        try {
            SfalLexer lexer = null;

            if (args.length != 2) {
                throw new Exception("Usage: java SFAL <MovieName>
<file.sfal>");
            }
            else {
                FileInputStream filename = new
FileInputStream(args[1]);
                DataInputStream input = new DataInputStream(filename);
                lexer = new SfalLexer(new DataInputStream(input));
            }

            SfalParser parser = new SfalParser(lexer);
            parser.program();

            CommonAST tree = (CommonAST) parser.getAST();

            if (DEBUG) {
                Writer writer = new OutputStreamWriter(System.out);
                tree.xmlSerialize(writer);
                writer.write("\n");
                writer.flush();
            }

            SfalWalker walker = new SfalWalker();
            walker.program(parser.getAST());

            StringBuffer headerBuffer = new StringBuffer();

            headerBuffer.append("package {")
            .append("import flash.display.Sprite;")
            .append("import flash.text.TextField;")
            .append("import flash.events.Event;")
            .append("public class ").append(args[0])
            .append(" extends Sprite {")
            .append("public function ").append(args[0])
            .append("() {");

            walker.programBuffer.insert(0, headerBuffer.toString());

            String program = walker.programBuffer.toString();

            char[] chars = program.toCharArray();

            // Pretty print the ActionScript file
            StringBuffer sb = new StringBuffer();
            for (int i = 0; i < chars.length; i++) {
                char c = chars[i];
                if (c == '\n') continue;

                sb.append(c);

                if (c == ';' || c == '{' || c == '}')
                    sb.append('\n');
            }

            BufferedReader in = new BufferedReader(new
StringReader(sb.toString()));
            BufferedWriter out = new BufferedWriter(new
FileWriter(args[0] + ".as"));

            String tabs = "";

            String line = null;
            try {
                while ((line = in.readLine()) != null) {
                    line = line.trim();

                    if (line.length() == 0) continue;

                    if (line.charAt(line.length()-1) == '}') {
                        tabs=tabs.substring(0,tabs.length()-1);
                    }

                    out.write(tabs); out.write(line); out.write('\n');

                    if (line.charAt(line.length()-1) == '{') {
                        tabs += "\t";
                    }
                }
            }
            catch (IOException e) {
                System.err.println(e);
            }
            finally {
                try {
                    if (out != null) {
                        out.flush();
                        out.close();
                    }
                }
                catch (IOException e) {}
            }

            in = new BufferedReader(new FileReader("embedder.txt"));
            out = new BufferedWriter(new FileWriter(args[0] +
".html"));

            try {
                  out.write("<!-- AUTO-GENERATED -->\n");
                  while ((line = in.readLine()) != null) {
                        out.write(MessageFormat.format(line,args));
                        out.write("\n");
                  }
                  out.flush();
                  out.close();
            }
            catch (IOException e) {
                  System.err.println(e);
            }
            finally {
                try {
                    if (out != null) {
                        out.flush();
                        out.close();
                    }
                }
                catch (IOException e) {}
            }
        }
        catch (Exception e) {
            System.err.println(e.getMessage());
            e.printStackTrace();
        }
    }
}
```

6. SymbolTable.java

SymbolTable.java

```
/**
 * SymbolTable and SymbolTable entry
 *
 * @author Antonio Cruz adc2104@columbia.edu
 * @version $Id: SymbolTable.java,v 1.13 2007/12/16 07:32:41 admin Exp
$
 */

import java.util.*;


public class SymbolTable {
      HashMap table;

      SymbolTable outer;

      public SymbolTable(SymbolTable st) {
            table = new HashMap();
            outer = st;
      }

      class SymTabEntry {
            SymTabEntry(String t, String y, String v) {
                  token=t; type=y; value=v;
            }
            // type here is AS type, e.g. Sprite, Number, TextField
            String token, type, value;
            boolean declared;
            public String toString() {
                  return token + "," + type + "," + value + "," +
declared;
            }
      }

      public void put(String token, String type, String value) {
            if (!alreadyDeclared(token) && table.get(token) == null) {
                  table.put(token, new SymTabEntry(token, type,
value));
            }
      }

      public SymbolTable.SymTabEntry get(String token) {
            for (SymbolTable table = this; table != null; table =
table.outer) {
                  SymbolTable.SymTabEntry entry =
(SymbolTable.SymTabEntry) (table.table.get(token));
                  if (entry != null)
                        return entry;
            }
            return null;
      }

      public boolean alreadyDeclared(String token) {
            boolean declared = false;
            int depth = 0;
            for (SymbolTable table = this; table != null; table =
table.outer) {
                  SymbolTable.SymTabEntry entry =
(SymbolTable.SymTabEntry) (table.table.get(token));
                  declared = entry != null && depth > 0;
                  depth++;
            }
            return declared;
      }

      public String toString() {

            StringBuffer sb = new StringBuffer();
            String tabs = "";
            for (SymbolTable st = this; st != null; st = st.outer) {
                  sb.append(tabs).append("SymbolTable: \n");
                  Set entries = st.table.entrySet();
                  for (Iterator itr = entries.iterator();
itr.hasNext(); ) {
                        Map.Entry entry = (Map.Entry) itr.next();
                        sb.append(tabs)
                              .append(entry.getKey()).append(": ")
                              .append(entry.getValue().toString())
                              .append('\n');
                  }
                  tabs += "\t";
            }
            return sb.toString();
      }


}
```

7. Translator.java

Translator.java

```
/**
 * Translate SFAL statements to ActionScript 3.0.
 *
 * @author Antonio Cruz adc2104@columbia.edu
 * @version $Id: Translator.java,v 1.14 2007/12/16 07:30:46 admin Exp $
 */

import java.util.*;
import java.text.*;


public class Translator implements SfalWalkerTokenTypes {

	public Translator(SymbolTable st) { symTable = st; }
	SymbolTable symTable = null;

	static HashMap TRANSLATIONS = new HashMap();
	static {

		TRANSLATIONS.put("makesquare",
				"{0}.graphics.beginFill({3},100);\n" +
				"{0}.graphics.drawRect(0,0,{2},{2});\n" +
				"{0}.graphics.endFill();\n" +
				"{0}.x = 0;\n" +
				"{0}.y = 0;\n" +
				"addChild({0});\n");

		TRANSLATIONS.put("makerect",
				"{0}.graphics.beginFill({4},100);\n" +
				"{0}.graphics.drawRect(0,0,{2},{3});\n" +
				"{0}.graphics.endFill();\n" +
				"{0}.x = 0;\n" +
				"{0}.y = 0;\n" +
				"addChild({0});\n");

		TRANSLATIONS.put("makecircle",
				"{0}.graphics.beginFill({3},100);\n" +
				"{0}.graphics.drawCircle(0,0,{2});\n" +
				"{0}.graphics.endFill();\n" +
				"{0}.x = 0;\n" +
				"{0}.y = 0;\n" +
				"addChild({0});\n");

		TRANSLATIONS.put("makemovie",
				"package {\n" +
				"import flash.display.Sprite;\n" +
			"import flash.text.TextField;\n" +
			"import flash.events.Event;\n\n" +
                    "public class {2} extends Sprite {\n\n" +
                    "public function {2}() {\n");

            TRANSLATIONS.put("maketext",
                        "{0}.text=\"{2}\";\n" +
                        "addChild({0});");

            TRANSLATIONS.put("loop",
                  "for (var {2}:int = {0}; {2} < {1}; {2}++)");

            TRANSLATIONS.put("setx", "{0}.x={3};\n");
            TRANSLATIONS.put("sety", "{0}.y={3};\n");
            TRANSLATIONS.put("getx", "x={2}.x;\n");
            TRANSLATIONS.put("gety", "y={2}.y;\n");

            TRANSLATIONS.put("settextx", "{0}.x={3};\n");
            TRANSLATIONS.put("settexty", "{0}.y={3};\n");
            TRANSLATIONS.put("gettextx", "x={2}.x;\n");
            TRANSLATIONS.put("gettexty", "y={2}.y;\n");
            TRANSLATIONS.put("addtotext",
"{0}.appendText(\"{3}\");\n");

            TRANSLATIONS.put("makegroup", "addChild({0});");
            TRANSLATIONS.put("addtogroup", "{0}.addChild({3});\n");

            TRANSLATIONS.put("globalnumber", "");

            TRANSLATIONS.put("cos", "{0}=Math.cos({2});");
            TRANSLATIONS.put("sin", "{0}=Math.sin({2});");
      }

      /*
       *
       */
      void checkForColor(String[] args) {
            String func=args[1], color, newcolor;
            if (func.equals("makesquare") || func.equals("makecircle"))
{
                  if (args.length < 4) return;
                  color = args[3];
                  newcolor = (String)
Meta.WEB_COLORS.get(color.toLowerCase());
                  if (newcolor == null) return;
                  else args[3] = newcolor;
            }
            else if (func.equals("makerect")) {
                  if (args.length < 5) return;
                  color = args[4];
                  newcolor = (String)
Meta.WEB_COLORS.get(color.toLowerCase());
                  if (newcolor == null) return;
                  else args[4] = newcolor;
            }
            else return;
      }

      public String translate(int type, String s) {
            // System.err.println("TR: string: " + s);

            StringBuffer sb = new StringBuffer();

            switch(type) {
            case ASGN:
                  String[] args = s.split(",");

                  if (args.length < 1) return null;

                  checkForColor(args);

                  String translation = (String)
TRANSLATIONS.get(args[1]);

                  if (translation != null) {
                        sb.append(MessageFormat.format(translation,
args));
                  }
                  else {
                        sb.append(args[0] + "=" + args[1] + ";");
                  }
                  break;
            case 1001: // loop
                  translation = (String) TRANSLATIONS.get("loop");
                  sb.append(MessageFormat.format(translation,
s.split(",")));
                  break;

            }

            return sb.toString();
      }

      StringBuffer aniBlock = new StringBuffer();

      /**
       *
       * @return
       */
      public String translateAnimator() {
            return aniBlock.toString();
      }


      /**
       *
       * @param s
       */
      public void buildAnimator(String s) {
            aniBlock.append(s).append('\n');
      }


    StringBuffer programBuffer = new StringBuffer();

      /**
```

```
         *
         * @return
         */
        public String getProgramBuffer() {
            return programBuffer.toString();
    }


        /**
         *
         * @param s
         */
        public void buildProgramBuffer(String s) {
          programBuffer.append(s).append('\n');
        }


        /**
         * Local variable declarations.
         *
         */
        public String translateSymbolTableLocal() {
              return writeVariableDeclarations("");
        }

        /**
         * Global variable declarations.
         *
         */
        public String translateSymbolTable() {
              return writeVariableDeclarations("public ");
        }

        public String writeVariableDeclarations(String scope) {
              Set entries = symTable.table.entrySet();
              StringBuffer sb = new StringBuffer();
              for (Iterator itr = entries.iterator(); itr.hasNext(); ) {
                    Map.Entry entry = (Map.Entry) itr.next();

                    SymbolTable.SymTabEntry ste =
(SymbolTable.SymTabEntry) entry.getValue();

                    if (ste.declared == true) continue;

                    if (ste.type != null ) {
                          if (ste.type.equals("Sprite")) {
                                sb.append(scope + "var " +
                                      entry.getKey() + ":Sprite = new " +
ste.type + "();")
                                      .append('\n');
                          }
                          else if (ste.type.equals("TextField")) {
                                sb.append(scope + "var " +
                                      entry.getKey() + ":TextField = new
" + ste.type + "();")
                                      .append('\n');
                          }
```

```
                else if (ste.type.equals("Number")) {
                              sb.append(scope + "var " + entry.getKey()
+ ":Number = 0;")
                              .append('\n');
                        }
                  }

                  ste.declared = true;
            }
            return sb.toString();
      }
}
```

8. Validator.java

Validator.java

```
/**
 * Validate SFAL expressions and statements.
 *
 * @author Antonio Cruz adc2104@columbia.edu
 * @version $Id: Validator.java,v 1.16 2007/12/17 02:49:53 admin Exp $
 */

import java.util.*;
import antlr.*;


public class Validator extends Meta implements SfalWalkerTokenTypes {

      boolean isValidating = false;
      public Validator(boolean validating) {
            isValidating = validating;
      }

      /**
       * Validate a statement with one expression
       * @param root - Expression node type
       * @param nodes - A list of the Expression node's children nodes.
       * @return
       */
      boolean validate(int root, Expr e, SymbolTable top)
      throws RecognitionException
      {
            if (!isValidating) return true;
            return validate(root, e, null, top);
      }

      /**
       * Validate a statement with two expressions
       * @param root
       * @param e
       * @param e2
       * @param top
       * @return
       */
      boolean validate(int root, Expr e1, Expr e2, SymbolTable top)
      throws RecognitionException
      {
            if (!isValidating) return true;

            switch(root) {
                  case ASGN:
                        /* Make sure e1 and e2 AS types are equal
```

```
                         * Depending on node type, get AS type
accordingly from SymbolTable,
                         * function mappings, or node mappings.
                         */

                        if (e1 == null || e2 == null) return false;

                        // SFAL node types, and AS types
                        String t1 = e1.type, t2 = e2.type, t1_AS_type,
t2_AS_type;

                        if (t1.equals("ID")) {
                              SymbolTable.SymTabEntry entry =
                                    (SymbolTable.SymTabEntry)
top.get(e1.value);
                              t1_AS_type = entry.type;
                        }
                        else if (t1.equals("FUNC_CALL")) {
                              t1_AS_type = (String)
Meta.FUNC_TO_TYPE.get(e1.value);
                        }
                        else {
                              t1_AS_type = (String)
Meta.NODE_TO_TYPE.get(t1);
                        }

                        if (t2.equals("ID")) {
                              SymbolTable.SymTabEntry entry =
                                    (SymbolTable.SymTabEntry)
top.get(e2.value);
                              t2_AS_type = entry.type;
                        }
                        else if (t2.equals("FUNC_CALL")) {
                              t2_AS_type = (String)
Meta.FUNC_TO_TYPE.get(e2.value);
                        }
                        else {
                              t2_AS_type = (String)
Meta.NODE_TO_TYPE.get(t2);
                        }


                        if (t1_AS_type == null
                                    || t2_AS_type == null
                                    || !t1_AS_type.equals(t2_AS_type))
                        {

                              /* This is a BUG */
                              if ((t1_AS_type != null && t2_AS_type !=
null) &&
                                    !(t1_AS_type.equals("Sprite") &&
t2_AS_type.equals("Number")))
                              {
                                    e1.error="Incompatible types: e1: "
                                          + e1.type  + " e2: " +
e2.type;
                              }
                        }



                        return e1.error==null;

                    case MULT:
                    case DIV:
                    case MOD:
                    case PLUS:
                    case MINUS:

                        if (e1 != null && e2 == null) {

                            /* rvalue here is an arith expression,
e.g. 2+2 */

                            if (!(getType(e1.type).equals("Number")))
{
                                e1.error = "BAD ARITH r_value: " +
e1.toString();
                                return false;
                            }
                        }
                        else if (e2 != null) {

                            // SFAL node types, and AS types
                            t1 = e1.type;
                            t2 = e2.type;
                            t1_AS_type = null;
                            t2_AS_type = null;

                            if (t1.equals("ID")) {
                                SymbolTable.SymTabEntry entry =
                                    (SymbolTable.SymTabEntry)
top.get(e1.value);
                                t1_AS_type = entry.type;
                            }
                            else if (t1.equals("FUNC_CALL")) {
                                t1_AS_type = (String)
Meta.FUNC_TO_TYPE.get(e1.value);
                            }
                            else {
                                t1_AS_type = (String)
Meta.NODE_TO_TYPE.get(t1);
                            }

                            if (t2.equals("ID")) {
                                SymbolTable.SymTabEntry entry =
                                    (SymbolTable.SymTabEntry)
top.get(e2.value);
                                t2_AS_type = entry.type;
                            }
                            else if (t2.equals("FUNC_CALL")) {
                                t2_AS_type = (String)
Meta.FUNC_TO_TYPE.get(e2.value);
                            }
```

```
                              else {
                                    t2_AS_type = (String)
Meta.NODE_TO_TYPE.get(t2);
                              }


                              if (!t1_AS_type.equals(t2_AS_type)) {
                                    // if (t1_AS_type != null &&
t2_AS_type != null) {
                                          e1.error="Incompatible types:
e1: "
                                                      + e1.type  + "
e2: " + e2.type;
                                    // }
                              }
                        }

                        return e1.error==null;

                  case FUNC_CALL:
                         /* Make sure the function exists, and there
are the right number
                 * and type of function arguments.
                 */

                        String functionArgs = (String)
SFAL_FUNCTIONS.get(e1.value);

                        String[] args = null;
                if (functionArgs.indexOf(",") != -1) args =
functionArgs.split(",");
                else {
                  // handle makegroup and globalnumber which have no
args
                  if (!functionArgs.equals("None")) {

                        // functions w/ one arg
                        args = new String[] {functionArgs};
                  }
                  else {
                        args = new String[] {};
                  }
                }

                        return functionArgs!=null && (args.length ==
e1.values.size());

            }
            return false;
      }

      /**
     * Validate a loop statement
     * @param l
     * @return
     */
    boolean validateLoop(List l) {
      if (!isValidating) return true;


        try {
            return l.size()==3
            && ((Expr)l.get(0)).type.equals("Number")
            && ((Expr)l.get(1)).type.equals("Number")
            && ((Expr)l.get(2)).type.equals("ID");
        }
        catch (Exception e) {
            System.err.println(e);
            return false;
        }
    }

      /* Try and get Node type mapping first */
      String getType(String t) {
            if (t==null) return "";
            String mapping = (String) NODE_TO_TYPE.get(t);
            if (mapping != null) return mapping;
            else return t;
      }
}
```