ELMO Loves Manipulating Objects (ELMO): A Graphics Object Manipulation Language

Jeffrey Cua
jmc2108@columbia.edu

Stephen Lee
sl2285@columbia.edu

Erik Peterson
edp2002@columbia.edu

John Waugh
jrw2005@columbia.edu

December 21, 2004

Contents

Chapter 1

Introduction

Many programmers invest substantial effort into expressing three dimensional objects using API's like OpenGL in C/C++, Java3D, and Direct3D also in C/C++. Languages like C/C++ and Java are Swiss Army knives which work well for a wide variety of applications, but can still become unwieldy.

OpenGL is enormously powerful, but it can become quite cumbersome for something as simple as a rotation. The objective of ELMO is not to augment the functionality offered by OpenGL, but provide an additional way to express the same set of operations and primitives that OpenGL grafts on to C/C++. Likewise, Java3D which may be easier to use than OpenGL, suffers from the same fundamental flaw: it is grafted onto an existing general purpose language.

ELMO abstracts many of the details related to the matrix manipulation that OpenGL or other 3D API's reveal to the programmer. When a programmer wants to do a simple rotation, the programmer might call glRotate3f(), a function in OpenGL, which initializes a 4x4 matrix that is placed on one of the various transformation stacks that OpenGL employs to keep track of the various operations conducted on the 3D objects in the program.

ELMO aims to be intuitive to the lay programmer, allowing him or her to quickly express everything from simple spheres to more complex objects.

1.1 Capabilities

- **3D Geometry**
  The core of ELMO is its manipulation of three dimensional graphical objects. While supporting a wealth of options and user-defined properties, ELMO aims to be easy to code and understand conceptually.
- **Basic Transformations**
  All the primitive transformations you'd expect are supported within the language. Translation, rotation, and scaling are all handled under the surface with no need to learn matrix multiplication, fret over vectors, or understand homogeneous coordinate systems.
- **Grouping and Cloning**
  On top of its primitive operations, ELMO allows for convenient creation of more complex objects from simpler ones through an amalgamation process more readily known as grouping. This allows basic cylinders and spheres to become more impressive graphical creations such as dendritic growths, it trees. Having created these complex objects, ELMO also allows for their convenient cloning through an internal process of copying such that an initial conceptual model can be instatiated time and time again as fits the programmer's intentions.
- **Parameterized Objects**
  An important feature that differentiates ELMO from OpenGL and Java3D is the ability to quickly

script and generate complex and unique objects using OBJ input from OBJ files directly. By giving the user the ability to involve random numbers easily into object manipulation, projects like generating complex yet unique structures, such as a forest of trees, becomes much less of a hassle. Now all a modeler needs to do is create a tree script and include randomness. Run the script as many times as you need unique trees and voila! A diverse forest emerges.

1.2 Implementation

1.2.1 Input/Output

Essentially, an ELMO program will take some "building block" geometry, perform various transformations on these objects or copies of these objects, and output the resultant 3D scene to a file. We have chosen the Alias/Wavefront OBJ file format to describe both the inputs and the outputs of an ELMO program. Thus, the operation of an ELMO program will look like the following:

```
OBJ input(s) ---> ELMO execution ---> OBJ output
```

1.2.2 Types

ELMO will support several built-in types and automatic conversions between them, where applicable. The list of built-in types, with a short description of each, is as follows: For types with attributes, those attributes are referenced with C++/Java-style "dot" syntax. Examples:

```
myvector.x
```

Note that any operation performed on a group can also be performed on an object. Objects are inherently groups, and thus, the same group operations can be applied to objects. Internally, an implicit group, containing only myobject, would be used.

1.2.3 Functions

Applying functions to variables takes the following general form:

```
<action> <target> <preposition> <input>
```

For instance,

```
rotate myobject around <1,-1,1>
move myobject along myvector
```

Several built-in functions will enable the easy use of geometric transformations. These include:

```
rotate
move
scale
copy
```

1.2.4 Random Numbers

In the automatic generation of fractals and interesting 3D objects in general, it is often desirable to use random numbers for some inputs. In ELMO, writing

```
[5..10]
```

means "a random number between 5 and 10, inclusive."

1.2.5 Flow Control

Flow control happens much as it does in C/C++/Java, though the number of such structures available in ELMO is fewer. Braces define code blocks, as in C "if" and "for". Control structures are identical in structure to C. There is no "switch" statement or "try/catch" exception handling. Additionally, there is a "foreach" statement available with syntax as follows:

```
foreach myobject in mygroup {
    <perform some actions on myobject, etc>
}
```

Chapter 2

Tutorial

A variety of quality 3D graphical viewers exist across all popular platforms, so to build another would be redundant. Instead, ELMO serves as a way to script 3D manipulations outside the often cumbersome framework of a full graphical language such as OpenGL. ELMO accomplishes this by operating directly on the object's geometric representation and by encapsulating the technical aspects of 4x4 matrix manipulation for ease of use. The .OBJ format was chosen for its ubiquitous usage as a platform-independent representation of graphical data. ELMO operates on the format by reading in a file, manipulating the data as scripted, and then generating a file with the resultant data. Because input and output share the same domain, ELMO programs can be used as modules, much like stages along an assembly line, in any batch operation. Now, with this conceptualization, let us turn to the language itself.

Unlike Java, the filename can be arbitrary as long as it uses the extension `.elmo`. For this example, we'll use the appropriate name `curl.elmo`.

Before getting into the meat of the program, we're going to want a function to simplify our code. We open the function definition like a C-style function. We'll want an integer value to limit the recursion, an object to output, and an axis about which to rotate as we move.

```
void curl ( int counter, object sphere, vector axis ) {
    /* We start out by stamping a copy of our object so that
    it will show up in the output. */
    stamp sphere;

    /* Now, if the counter is 0, we're done and the recursion ends.
    Otherwise, we decrement once, rotate and move along the axis, and
    scale down so the next iteration is smaller than this one before
    we recursively call curl again. */
    if ( counter != 0 ) {
        counter--;
        rotate sphere around axis by PI/6;
        move sphere along axis by 4;
        scale sphere around <0,0,0> by 0.9;
        curl( counter=counter, sphere=&sphere, axis=axis );
    }

    /* Finally, don't forget to close the function definition. */
}

/* Now we're ready to write the core of the program.  Let's start
by setting our object.  We do this by simply assigning the filename
of an .OBJ file to our object variable.  The file import and
```

```
initialization of data structures is all taken care of internally
for programming convenience. */
object sphere = "sphere.obj";

/* We'll want another copy of the object, so we can define another
variable and assign to it the first.  sphere2 is now a clone of
sphere. */
object sphere2 = sphere;

/* We move sphere a bit along the x-axis and sphere2 a bit in the
opposite direction. */
move sphere along <1,0,0> by 3;
move sphere2 along <1,0,0> by 0-3;

/* Now, set an integer to the recursive depth desired. */
int counter = 20;

/* All that's left is to repeatedly call our curl function with
either sphere or sphere2 and various axes to produce our curls out
from the center. */
curl( counter=counter, sphere=sphere, axis=<0,1,0> );
curl( counter=counter, sphere=sphere2, axis=<0,1,0> );
curl( counter=counter, sphere=sphere, axis=<0,0,1> );
curl( counter=counter, sphere=sphere2, axis=<0,0,1> );
curl( counter=counter, sphere=sphere, axis=<0,0-1,0> );
curl( counter=counter, sphere=sphere2, axis=<0,0-1,0> );
curl( counter=counter, sphere=sphere, axis=<0,0,0-1> );
curl( counter=counter, sphere=sphere2, axis=<0,0,0-1> );

/* And, we're done.  Now, save the file and run the interpreter
from the commandline. */
$> java -jar elmoc.jar curl.elmo output.obj

/* The output .OBJ file with your manipulated objects will now be
available to you under the name "output.obj". */
```

Chapter 3

Language Reference Manual

3.1 Lexical Conventions

There are five kinds of tokens: identifiers, keywords, constants, expression operators, and other separators. In general blanks, tabs, newlines, and comments as described below are ignored except as they serve to separate tokens. At least one of these characters is required to separate otherwise adjacent identifiers, constants, and certain operator-pairs.

3.1.1 Comments

The characters `/*` introduce a comment, which terminates with the characters `*/`. Text starting with `//` and continuing to the end of the line also denotes a comment line.

3.1.2 Identifiers (Names)

An identifier is a sequence ofletters and digits; the first character must be alphabetic. The underscore counts as alphabetic. Upper and lowercase letters are considered different. No more than the first eight characters are significant, and only the first seven for external identifiers.

3.1.3 Keywords

```
PI
float int vector group object
rotate move scale stamp attach remove
around along by to deg rad from
if else for foreach in break return void
print inherit
```

3.1.4 Constants

Vector Constants

A vector constant is a `<` character followed by a `>` with expressions in between separated by commas. Vectors only support three dimensions. (e.g. `<0, 1.2, a>` or `<1+2,3,4>`)

Numerical Constants

A int or float constant is a sequence of digits. The digits can be interrupted by one decimal only. These will be implicitly assigned and used just as a number variable. (e.g. `512`, `0`, `0.50`, `35.56`, or `PI`)

Object Constants

An object constant is the name of either a OBJ file. If a file does not exist, a run-time error will occur and the ELMO program that referenced it will halt.
`<args>` takes the same format as for user-defined functions

```
"tree.obj"
```

object constant loaded from OBJ file tree.obj

3.2 Meaning of Identifiers

3.2.1 Types

- *int*
- *float*
- *vector*
- *group*
- *object* (every object is also a group)

Numerical Variables

A `int` and `float` variable is explicitly assigned by another `int` or `float` variable or a `int` and `float` constant.

```
int i = 5;
```

`i` is a declared `int` variable
`5` is an implicit `int` constant

Implicit numbers can be given with random-number syntax as well

```
float i = [-1..3] //i is a random number between -1 and 3
int j = [4] //j is a int between 0 and 4
```

vector Variables

A vector variable is explicitly assigned by another vector variable or a vector constant. Vectors are simply a collection of three number variables locally associated as `x`, `y`, and `z`. These are accessed with vectorname.x, vectorname.y, vectorname.z respectively.

group Variables

Groups are variables that contain objects and can have transformations applied to them. A group is defined by three vectors (axis) `X`, `Y`, and `Z`. These are accessed with `groupname.X`, `groupname.Y`, `groupname.Z` respectively. All of the vectors are **READ ONLY**.

object Variables

An object variable is a data structure that represents a 3D object. These objects are either primitives or imported OBJ or ELMO files. Objects are implicitly also a group that allows a user to directly apply transformations. An object also has three attributes (axis) `X`, `Y`, and `Z`. These vectors are accessed with `objectname.X`, `objectname.Y`, `objectname.Z` respectively. All of the vectors are **READ ONLY**.

```
object o = "tree.obj"
```

`o` is assigned to the value of an object constant, `"tree.obj"`

3.3 Declarations

declaration:
  type *identifier* assigment-expression
  type *identifier* function-args-declaration

  type:
  `int`
  `float`
  `vector`
  `group`
  `object`
  `void`

  function-args-declaration:
  `(` (function-input-declaration)* `)`

  function-input-declaration:
  type *identifier* assignment-expression

3.4 Statements

statement:
  (assignment-expression)? `;`
  compound-statement
  iteration-statement
  selection-statement
  interrupt-statement
  group-statement

A sequence of statements is executed one after another, resolving the current one before beginning the next. Each statement has an effect on the current data and/or generates a value but has no value itself. Statements can be subdivided into multiple categories.

3.4.1 Compound Statements

compound-statement:
  `{` (statement)* `}`
  (*declaration-list statement-list*)?
declaration-list:
  declaration
  declaration-list declaration
statement-list:
  statement
  statement-list statement

Otherwise known as a block statement, a compound statement allows for the declaration of local variables and the execution of a sequence of statements.

3.4.2 Iteration Statements

  `for` `(` (declaration OR assignment-expression) `;` expression `;` assignment-expression `)` statement
  `foreach` identifier `in` identifier statement

The `for` and `foreach` statements allow for the repeated execution of a sequence of statements. The first production takes three expressions which are executed once at the beginning, once before each loop, and once after each loop, respectively. The second expression's evaluation as true or false determines whether the statements are executed or the loop exited. The typical format is to initialize some sentinel value, test that it is within some range, and then increment or decrement its value for each iteration of the loop.

The second production serves as a short-cut for executing the same statements on each highest-level element in the specified group. "foreach" loops do not recurse to groups within groups. The current element is stored to the identifier specified. For recursive descent into nested groups, a "foreach" loop can be nested within another and executed using a selection statement to first test whether the element is a group or an individual object.

3.4.3 Selection Statements

`if (` expression `)` statement
`if (` expression `)` statement `else` statement

These statements allow for the conditional execution of a sequence of statements. The expression is first evaluated, and the output determines which statement is executed. In the top production, a result of true or non-zero result leads to the execution of the following statements. In the other production, a true or non-zero result leads to the execution of the immediately following sequence of statements whereas a false or zero result leads to the execution of the statement sequence following the "else" keyword.

3.4.4 Interrupt Statements

`break`
`return` (assignment-expression)?

3.4.5 Group Statements

group-statement:
`move` *identifier* `along` *assignment-expression* (`by` *assignment-expression*)?
`rotate` *identifier* `around` *assignment-expression* (`by` *assignment-expression units-modifier*)?
`scale` *identifier* `around` *assignment-expression* (`by` *assignment-expression*)?
`stamp` *assignment-expression*
`attach` *identifer* `to` *identifier*
`remove` *identifier* `from` *identifier* (`inherit`)?
units-modifier:
`deg` (degrees)
`rad` (radians)

This are really the main operators in the ELMO language. None of these operators will return anything. They will solely operate on the contents of the first argument. The following are some examples of these operators in action.

```
move myobject along myobject.x by 3.2;     //ok - every object is also a group
move mygroup.x along <1,2,3>;              //ILLEGAL - myobject.x is READ-ONLY!
move myvector along <1,2,3>;               //ok
rotate mygroup around mygroup.x by PI rad;
rotate myvector myOtherVector 30;
scale tree 5;                              //make a big tree!
scale "tree.obj" by 3.2;                   //ILLEGAL - "tree.obj" is not a
                                           //declared group
stamp "mytree.obj";
stamp myTree;                               //output myTree wherever it happens
                                           //to be right now it will forever
                                           //have that visual info in the output
                                           //file even if we move myTree later.

remove myTree from myForest;                //might make myTree jump back to the
                                           //origin, if that's where it started

remove eyeball head inherit;                //keeps the eyeball where it used to
                                           //be, just not further affected by
                                           //head's transformations
```

3.5 Expressions

3.5.1 Constant Expressions

constant-expresion:
 vector-constant
 int-constant
 float-constant
 object-constant

3.5.2 Random Number Expressions

random-number-expression:
 `[` *assignment-expression* `..` *assignment-expression* `]`

3.5.3 Primary Expressions

primary-expression:
 identifier
 constant-expression
 random-number-expression

For the most part, the operators in ELMO work have the same precedence and associativity as those in the C Language Specification. However, ELMO does not have the exact same concept as pointers as C. Instead, ELMO uses references transparently, like Java. Moreover, ELMO lacks bitwise and shift operators.

3.5.4 Parentheses Expressions

paren-expresion:
 primary-expression
 `(` *paren-expression* `)`

3.5.5 Vector Length Expressions

bars-expresion:
 `|` *paren-expression* `|`

3.5.6 Function Invokation

function-invoke-expresion:
  bars-expression
  identifier `(` (*argument-expression-list*)? `)`
argument-expression-list:
  assignment-expression `,` *argument-expression-list*

3.5.7 Suffix Expression

suffix-expresion:
  function-invoke-expression
  function-invoke-expression `.X`
  function-invoke-expression `.Y`
  function-invoke-expression `.Z`
  function-invoke-expression `.x`
  function-invoke-expression `.y`
  function-invoke-expression `.z`

3.5.8 Prefix Expressions

prefix-expression:
  suffix-expression
  `++` *prefix-expression*
  `--` *prefix-expression*
  `-` *prefix-expression*

3.5.9 Postfix Expressions

postfix-expression:
  prefix-expression
  postfix-expression `++`
  postfix-expression `--`

3.5.10 Power Expressions

power-expression:
  postfix-expression
  postfix-expression (`^` *power-expression*)?

Multiplicative Operators

multiplicative-expression:
  power-expression
  multiplicative-expression `*` *power-expression*
  multiplicative-expression `/` *power-expression*

Additive Operators

additive-expression:
  multiplicative-expression
  multiplicative-expression `+` *additive-expression*
  multiplicative-expression `-` *additive-expression*

Assignment Expressions

assignment-expression:
  additive-expression (assignment-operator assignment-expression)?

assignment-operator: one of
  `= =& *= /= += -=`

The basic `=` will create a copy of the right side and assign it to a variable named on the left side. The `=&` will create a reference of the right side and assign it to the variable named on the left side.

Logical Not Expression

logical-not-expression:
  assignment-expression
  `!` *logical-not-expression*

Logical and Relational Operators

logical-relational-expression:
  logical-not-expression
  logical-not-expression `<` *logical-not-expression*
  logical-not-expression `>` *logical-not-expression*
  logical-not-expression `<=` *logical-not-expression*
  logical-not-expression `>=` *logical-not-expression*
  logical-not-expression `==` *logical-not-expression*

3.6 External Declarations

3.6.1 Function Defintion

Function declarations are similar to those in C, but with thorough support of default variables (even more so than C++). Example:

```
vector myfunc(vector v1=<0,0,0>, vector v2=<1,1,1>)
{
    return v1 + v2
}
```

The above function, if called with no parameters, would return `<0,0,0> + <1,1,1>` (in other words, `<1,1,1>`) Example:

```
vector v = myfunc();    //v now equals <1,1,1>
```

The caller specifies arguments like so:

```
function_name([input1=value1] [,input2=value2] ... )
```

So for our example, we could have the following code segment:

```
vector v = myfunc(v1=<2,2,2>)       //leave v2 as its default <1,1,1>
                                    //- so v is <3,3,3> now
vector q = myfunc(v2=v)         //now q = <0,0,0> + v, which is just v
```

Note that since each input is referred to as a key=value pair, some C++ issues regarding default parameters do not arise. For example, in C++ if one wrote:

```
int myfunc(int foo=0, int bar=1)
{
    return foo+bar;
}
```

There would be no way for the caller to specify a value for bar but leave foo default

```
int q = myfunc(5);      //5 is assigned to foo -
                        //there is no way to make the 5 "bind" to bar
```

Of course, defaults are not required in ELMO. The following function is valid:

```
vector myfunc(vector v1=<0,0,0>, vector v2)
{
    return v1 - v2;
}
```

In such a case, v2 MUST be specified by the caller.

```
vector v = myfunc(v1=<2,2,2>) //ERROR - v2 not specified and has no default
```

Note also that, unlike with C++, default values can be assigned to any input, not just those at the end of the list of inputs.

One potential disadvantage to ELMO's scheme of defaults is that there could be some confusion with variable names. For example:

```
vector v1 = <9,9,9>
vector v = myfunc(v1=v1) //v = <9,9,9> + <1,1,1> = <10,10,10>
```

The `v1=v1` input to myfunc is not ambiguous, but it is perhaps not entirely intuitive. The name on the left of `=` is in the function's scope, while the name on the right is in the caller's scope.

Using `input=value`, the value, input will be a copy of value. This means that the original variable's value cannot be changed by the function. Note that while in a language such as Java, one can pass an Object by value, and then alter the contents of that Object using the "dot" syntax (internally dereferencing a pointer). This is not how ELMO behaves. If an a complex data type (such as an object or vector) is passed by value, the function can not alter the contents of the original input.

If by-reference functionality is desired, use the `=&` operator. example:

```
void myfunc(vector vec)     // note: no default value for vec -
                            // compiler will yell at you if
                            // you don't supply a value for vec
{
    vec = <1,1,1>
}
//elsewhere in the program...
vector v = <3,3,3>
myfunc(vec=&v)      //now v has value <1,1,1> since the original
                    //was altered by the function
```

The `=&` operator can be used for any type of inpus, such as numbers.

Chapter 4

Project Plan

4.1 Planning and Development

The project began with an idea. The team wanted to create a language for accomplishing complex graphical manipulations without the user needing to learn technical jargon or work with 4x4 matrices and associated homogenous coordinates. We asked ourselves what abilities we would want in such a language. We concluded that ELMO developers would want to import a grqphics object, transform rigid bodies, group objects and transofmrations, and save the transformed objects in multiple forms.

With a foundational structure in place and a general syntactic rules in mind, we broke the various tasks into related parts and distributed them among the team members. Working individually or in pairs, we built up each of the components, maintaining a running current version of the source online via CVS. As modules were added, the code could be reviewed, critiqued, and modified by the other team members.

Weekly meetings were used to adjust progress and responsibilities and the project took shape. A formal dependence graph was not necessary, but when one component's completion was held back by another's, priorities shifted to maintain parallelism within the team.

4.2 Style

The biggest challenge of any multiperson project is maintaining internal consensus and consistency. Basic software engineering techniques were followed to compartmentalize and encapsulate unrelated sections of code. With single programmers working on intimately related components, the necessary level of consistency was achieved while giving each individual coder the freedom to work as was personally comfortable.

A few conventions were used by each of the team members including simple naming conventions such as the prefix ELMO- before class names. Generous spacing was used in function definitions to promote readability, and comments were included liberally not only for reference but to mark areas which required future refinement.

Separating classes out into individual files also aided flexibility as one team member's changes rarely affected a file simultaneously being modified by another team member. Moreover, we divided our functionality into classes which computed graphics transforms, language specific constructs, and group/object manipulation.

We wanted functional and correct code at each stage of development, so as much as possible, we worked from the bottom up, building the lexer, then the parser, then the tree-walker, etc. as shown by our intermediate milestones below.

4.3 Timeline

- Late September:
  - Brainstorming on Subject of Language
  - White Paper
- Early October:
  - Lexer
  - Language Reference Manual
- Mid-October: Parser
- Late October: Tree-Walker
  - Support for integers/float
  - Support for vector
- Early November:
  - Support for group/object
  - Support for move/rotate/scale
- Mid-November: Support for looping/conditionals
- Late November: Support for .OBJ file import/export
- Early December:
  - Sample Programs
  - Final Report
  - Demonstration

4.4 Roles and Responsibilites

- **Erik** Project Manager, Tester, Java Coder
- **Stephen** ANTLR/Java Coder, Documentation
- **John** ANTLR/Java Coder, Tester
- **Jeff** Documentation

4.5 Software Development Environment

We used the following tools to build ELMO.

- **Ant**
  Ant used to execute the various parts that comprise ELMO. It invoked ANTLR to generate the Java code for the top-level ELMO interpreter. Moreover, it ran the Java compiler to compile both the ANTLR generated Java code and our own language-specific Java code.
- **ANTLR**
  We used ANTLR primarily to create the lexer, parser, and tree walker.
- **Java**

We used both 3D Studio Max and Maya to create and render the input and output used and produced by an ELMO script.

4.6 Project Log

NOTE: Our project log does not correspond directly with our timeline. We did not start using CVS until November.

See the CVS Log Appendix D.

Chapter 5

Architectural Design

We followed a very standard implementation for our translator as shown in Figure 5.1. These three components, at a high-level, are implemented using ANTLR. Erik worked on the lexer and interfaced with John, who primarily worked on the parser. Stephen primarily worked on the walker and interfaced with John's parser. Stephen's walker interfaced with Erik's underlying logic for doing the geometric transformations and group/object manipulation.

5.1 Walker (Stephen/John)

The walker shown in Figure 5.2 traverses a program which consists of declarations and/or statements. A declaration is either a variable declaration or a function declaration. It defines a symbol (identifier) linked to a segment of code or initialized by an expression. A statement is a compound statement, return statement, an expression encapsulated within a statement, etc. The parameters of each of these statements are either expressions or additional nested statements. Many of the ELMO constructs function in a similar fashion as their C counterparts. However, the notable exceptions are the group statements and the various types of expressions encapsulated within a statement.

Declarations require a reference to the symbol table used in context. They make additions to the current symbol table.

Statements also require a reference to the symbol table used in context. Thus, when the walker is initialized, a top-level symbol table, without a parent reference, is also instantiated. Each recursive call passes this reference down to its descendants until a For/Foreach or Function is encountered. Each of these generate a new child symbol table which allows for child scopes. Moreover, functions can be nested within each other.

The top-level symbol table defines a single namespace shared by functions and variables. A symbol table consists of a hash table for local variable/function definitions, a reference to its parent symbol table, and another hash table for reference to child scopes. When a symbol is referenced, the symbol table looks up the symbol locally and checks its parents recursively until a definition is found for this symbol. Furthermore, retrieving the symbol works in the same bottom-up fashion, and consequently, the symbol table does shallow

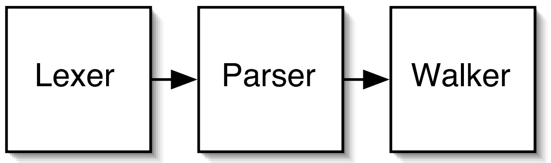


Figure 5.1: Top-level Components of the ELMO Interpreter

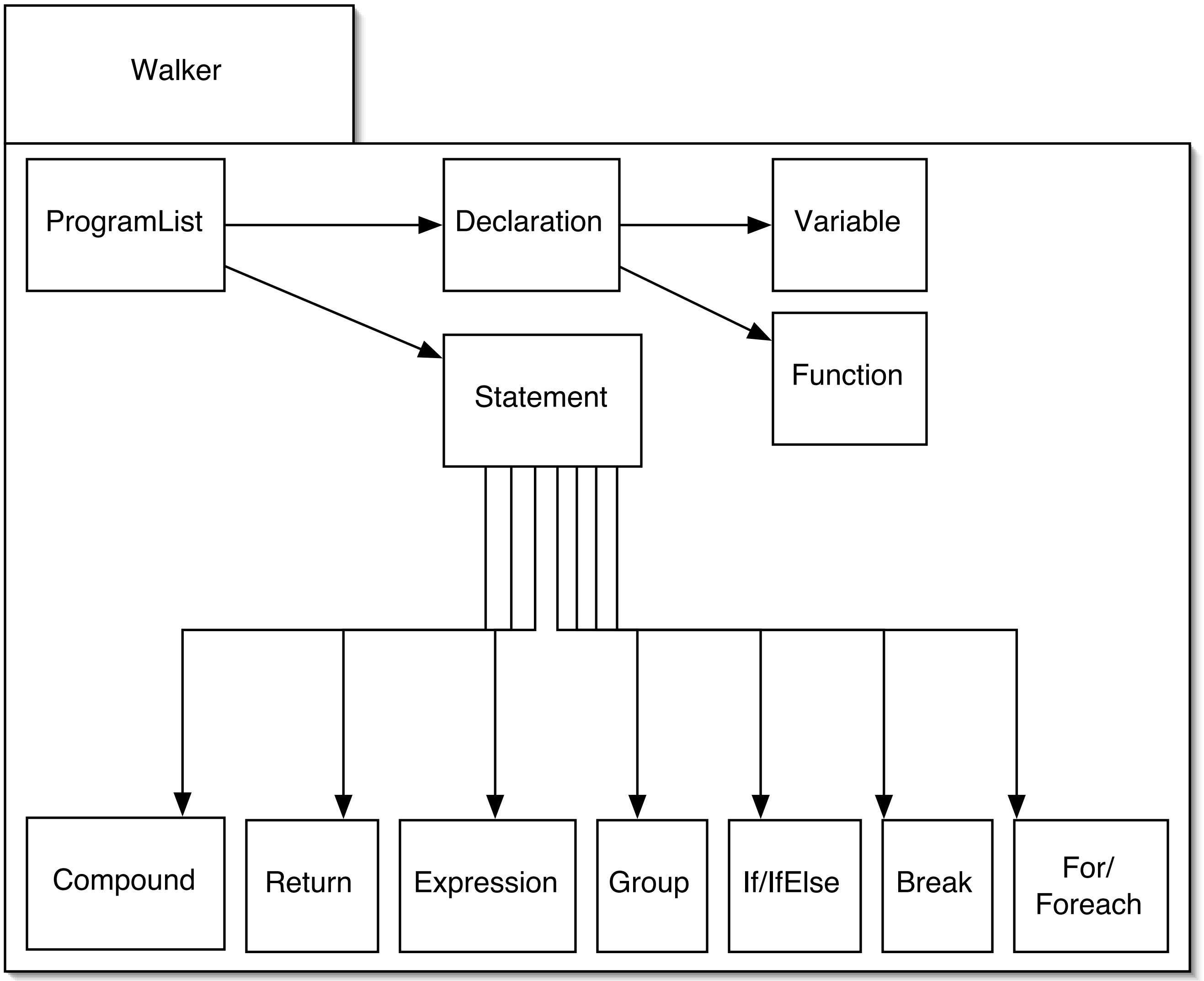


Figure 5.2: Walker's Composition

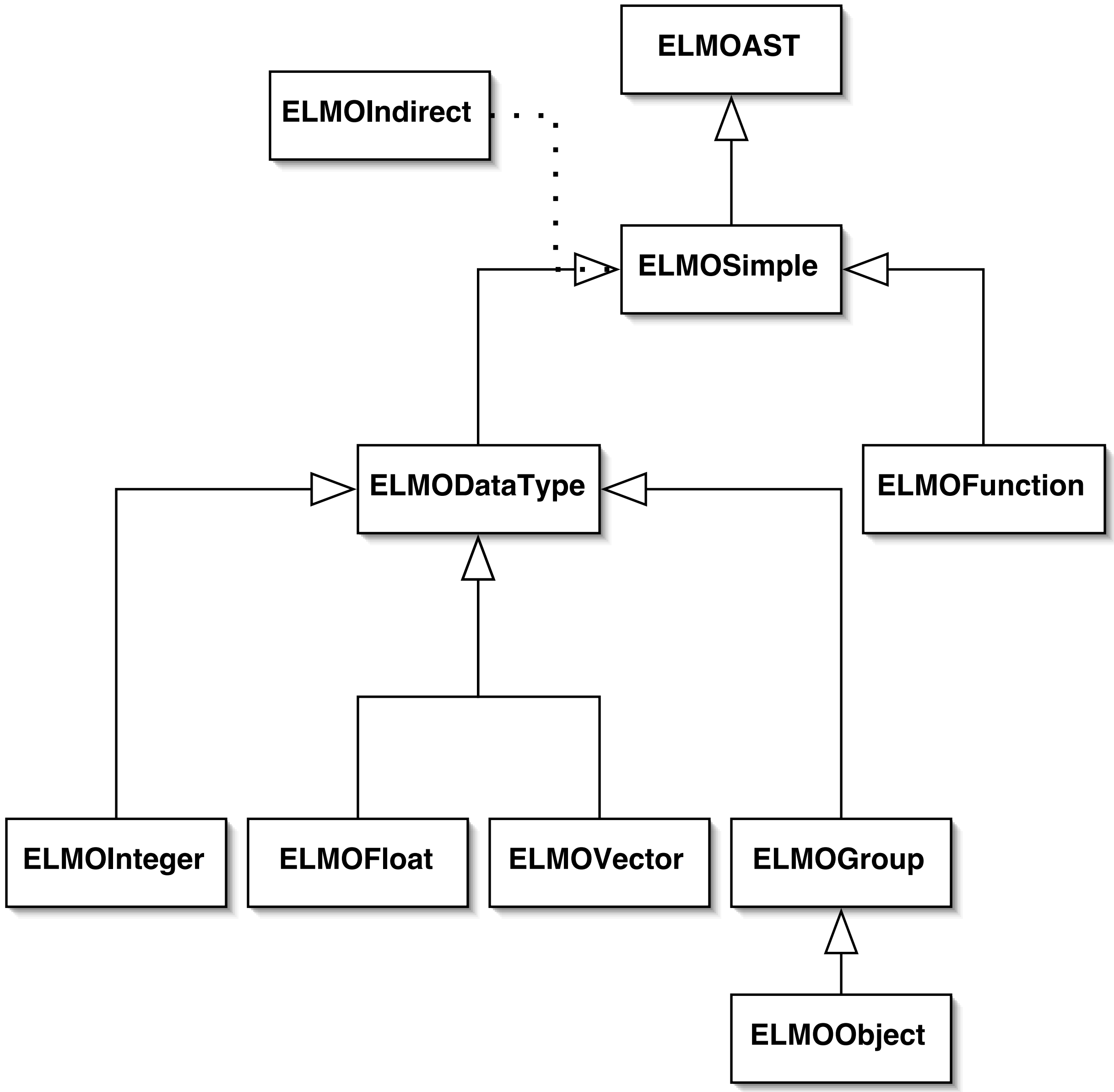


Figure 5.3: Basic ELMO Data Types

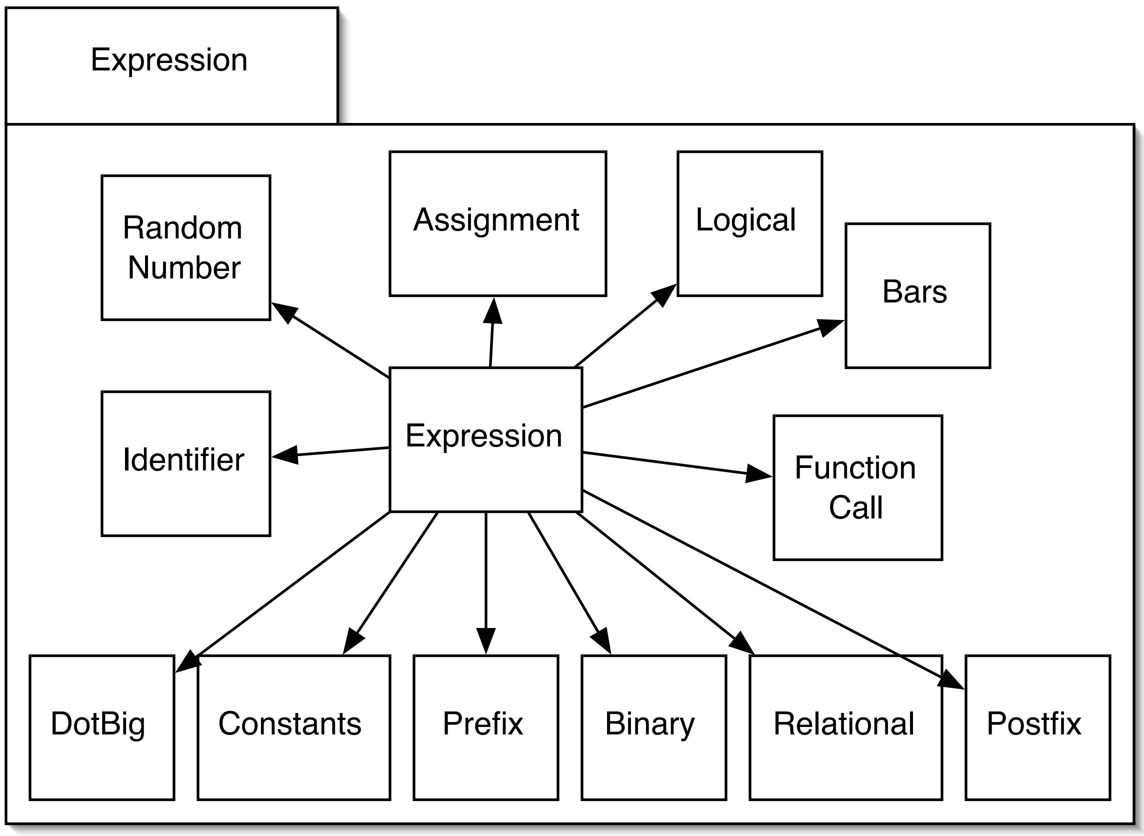


Figure 5.4: Expression Rules

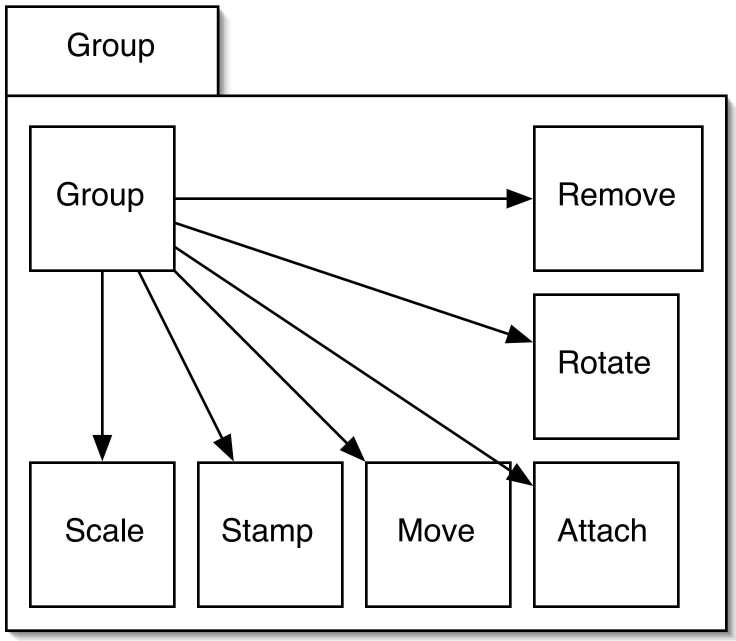


Figure 5.5: Group Statement Rules

binding for symbol retrieval and lookup.

The For/Foreach and Function define new scopes. Specifically, For/Foreach define a new scope for the duration of the iteration. However, Function defines a new scope each time it is invoked. Without creating a new symbol table for each function invocation and instead instantiating one just for the function declaration, we would be unable to recursively call a function.

5.1.1 Data Types (Stephen/John/Erik)

ELMO consists of many classes, but the ones in Figure 5.3 are the foundation which everything else depends upon. ELMOAST is simply a class to maintain the line and column number for where in the original source code one of the underlying types was instantiated. While the other classes in the diagram represent superclass and subclass relationships, ELMOIndirect simply references an ELMOSimple object. This relationship is denoted by a dotted line. ELMOIndirect objects are put into the symbol table and point to either a ELMODataType or an ELMOFunction. ELMOFunction maintains a pointer to its parent symbol table, a hash table of input parameters, and the body of the function, which is not semantically checked until the function is actually invoked.

ELMOInteger and ELMOFloat more or less implement the same operators. ELMOVector implements scaling and inverse scaling for multiplication and division respectively. ELMOGroup and consequently ELMOObject almost implement a completely disjoint set of operations, when compared to ELMOInteger, ELMOFloat, and ELMOVector. The ELMOGroup correspond directly with those described by the various ELMO specific group commands.

5.1.2 Expressions (Stephen/John)

ELMODataType as shown in Figure 5.4 is the abstract data type which integers, floats, vectors, groups, and objects derive from. The generic ELMODataType defines a common set of operations, but throws exceptions for each one of them. Each subclass selectively implements various operations, type checks its operands, and returns appropriate ELMODataType objects. Otherwise, an exception is thrown and caught at a higher level.

We used the ELMOIndirect class to wrap around objects of class type ELMOFunction or subclasses of ELMODataType. This is particularly important for assignment by reference (`=&`). We can simply look the symbol, wrapped in an ELMOIndirect object, and change the ELMOIndirect object's reference when assignment by reference is used. Otherwise, we can call an underlying copy function for assignment by value.

Identifiers use the current symbol to retrieve symbols referenced by expressions. Consequently, it uses the lookup and retrieval method prescribed earlier.

Other expression operators were broken into Java classes for groups of similar operators. Operations like addition, subtraction, multiplication, etc., where each was keyed by a string corresponding to the actual operation, were grouped under ELMOBinaryOperatorAST, which was instantiated by the parser. The similar pattern was applied to relational, logical, and prefix operators.

Another notable implementation detail is Postfix operators, which we defined as only occurring once the outermost expression is evaluated. Consequently, when the walker recognizes a Postfix operation, the walker returns the referenced variable and queues the requested action. Once an expression depth counter is reset to zero, the queued postfix operations are executed.

5.1.3 Groups (Erik)

Group statements as shown in Figure 5.5 are used to change the state of a group or object, which are both basic data types of ELMO. They are considered statements and not expressions because they were designed not to return anything. Scale, Rotate, and Move correspond to general rigid body transformations on a single group or object. Attach and Remove operate apply one group or object to another. Stamp adds the group or object to the output of the ELMO program. Generally, when each of these commands are recognized by the walker and the subjects of the command are identified as a group or object, the underlying ELMOGroup or ELMOObject method is called.

Chapter 6

Testing

Testing was conducted throughout the programming process. Initially, we used internal tests to make sure that the Java code for various functions was working as intended. This was done by importing and exporting sample .OBJ files and running snippets of code through the lexer and parser, validating their outputs by hand. Later tests were automated, checking output against a string of the desired output. By running such test scripts after each modification, consistency was maintained with previous versions of the code.

After the interpreter became functional, testing took the form of individual case studies. As functionality was added, such as the support of various transformations or grouping operations, full ELMO programs were run and compared to maintain consistency.

The following is a test for the postfix operator.

```
int x = 3;
print x;

x++;
print x;

x++;
print x;

print ( (x++ + 3) + x++ );
print x;

int a = 1;
int b = a++;

print b;
print a;
```

The above example should produce:

```
x[3]
x[4]
x[5]
integer5[13]
x[7]
b[2]
a[2]
```

The following is a test for the function recursion.

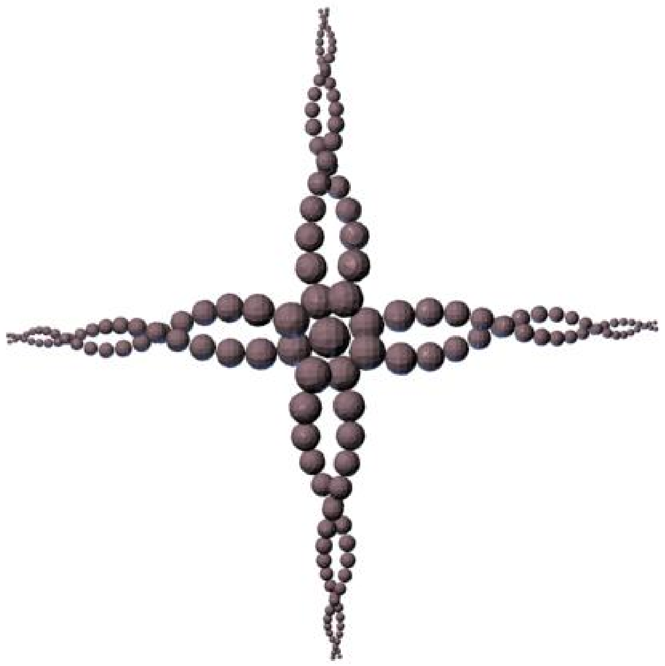


Figure 6.1: Output of curl.elmo test

```
int sum( int a ) {
    print a;

    if( a == 0 ) {
        return 0;
    }

    return a + sum( a=a-1 );
}
int s = sum( a=5 );
print s;
```

The example above outputs:

```
a[5]
a[4]
a[3]
a[2]
a[1]
a[0]
s[15]
```

The following is a test for the object manipulation functionality. It takes a sphere object and produces a helix like object shown in Figure 6.1.

```
void curl ( int counter, object sphere, vector axis ) {
    stamp sphere;
    if ( counter != 0 ) {
        counter --;
        rotate sphere around axis by PI/6;
        move sphere along axis by 4;
        scale sphere around <0,0,0> by 0.9;
        curl( counter=counter, sphere=&sphere, axis=axis );
    }
}

object sphere = "tests/sphere.obj";
print sphere;

object sphere2 = sphere;

move sphere along <1,0,0> by 3;
move sphere2 along <1,0,0> by 0-3;
int counter = 20;
curl( counter=counter, sphere=sphere, axis=<0,1,0> );
curl( counter=counter, sphere=sphere2, axis=<0,1,0> );
curl( counter=counter, sphere=sphere, axis=<0,0,1> );
curl( counter=counter, sphere=sphere2, axis=<0,0,1> );
curl( counter=counter, sphere=sphere, axis=<0,0-1,0> );
curl( counter=counter, sphere=sphere2, axis=<0,0-1,0> );
curl( counter=counter, sphere=sphere, axis=<0,0,0-1> );
curl( counter=counter, sphere=sphere2, axis=<0,0,0-1> );
```

Chapter 7

Lessons Learned

7.1 Jeffrey Cua

For programming issues, the best advice I have is to start early (obviously). The more time you have to work out the details, the more time you'll have to get things working.

For organization, I ran into two main issues.

1. **CVS**
   Using some sort of version control is necessary for group work, but being familiar with such a system is also useful for when it may come in handy in the future. I ran into two problems not knowing the ins and outs of CVS.

   The first came from not correctly updating. I thought I'd updated the entire project, but had actually only updated the files existing on my local repository. As a result, new files created by my team weren't visible, and I was working from old code using older versions of functions and their signatures. This required some reworking of my code after getting the correct files, but resulted in wasted time.

   The second problem was with committing changes. I had written some new code into an existing file and uploaded to the repository, or so I thought. I hadn't, and this lead to another team member writing code I'd already written. I then had to retrofit his work with the rest of my changes. The result was double coding and wasted time.

2. **Group Work**
   Besides the annoyances that came with my unfamiliarity with the systems being used, I allowed myself to lose touch with the project's progress. While still reading group emails and attending meetings, when other work diverted my attention, I found myself somewhat out of the loop. In small group there's little or no firm hierarchy, so if I was distracted, no one was there to pull me back in. As a result, I didn't quite "click" with the progressing structure. I made a number of small changes, finding local coding errors and adding in some simple functionality to existing code, but was not in a position to substantially alter the core of the project. This was not a problem in my case because the rest of the group had more coding experience anyway, but it meant playing catch up at times.

   To the individual, I advise you to stay involved as much as possible, leaving yourself enough time amongst your other courses to keep up with the project. To the group, I advise you to keep each other involved as much as possible. If one team member falls behind, it becomes harder not only for him/her to catch up, but for the rest of the team to operate as planned. Help each other out for your sake and the sake of the project as a whole.

7.2 Stephen Lee

Before implementation, early and thorough planning is the most important aspect of a language. An incomplete design can prove time-consuming to resolve late in the game. Without the foresight required by a thorough design, implementation can be very difficult to change since too much may have already been implemented to facilitate a design change.

Teamwork couples tightly with good design and planning. Without a precisely defined interface between each component and developer's domains, the danger of confusion and overlapping work grows substantially. While I feel like most of our project resulted in a strong teamwork, I also think that certain components could have been broken down into smaller modules and interfaced properly. Teamwork also requires some understanding of other developer's work and reasoning. Sometimes, we independently made design and implementation choices without discussing them with the rest of the team. Later, when another teammate made changes to the same module, code would be misplaced or removed completely because of general confusion.

In the end, we could have saved significant time if we had spent more time up front, rather than gloss over some necessary and substantial details.

7.2.1 Advice for Future Groups

- Start early!
- Write an extremely thorough Language Reference Manual. This will expose you to potential issues and trouble spots when implementing your actual language. Ultimately, the design will be a lot simpler when these little, but potentially explosive issues are resolved early before a single line of code is written.
- Get your hands dirty with ANTLR, or whichever lexer/parse you decide to use, early in the project. ANTLR aims to be a direct translation of the grammars and general methods taught in class, but it also has a lot of subtleties.
- Break big milestones into smaller objectives, but avoid getting bogged down in just planning.

7.3 Erik Peterson

In the planning stages of group projects it is important to let everyone express themselves so that the group feels comfortable about sharing new ideas and gets motivated by the project idea.

The largest set back our group faced was the inactivity of one of the members. Despite a constant effort on my part to reintegrate him into the working structure, he never put the effort forward to make any positive contributions to the code.

The one thing I learned through him was that optimism is always important. By never given up on the member the group was able to get some productivity out of him by the end through the compiling of both the presentation and the final document.

Next time I will make sure to address and confront inactive members early and give the individual less opportunity to slack off through excuse.

7.4 John Waugh

7.4.1 What Went Right

- Started fairly early

- Good description of our language (a good concept of what we wanted to do) done early
- Got the parser and lexer done pretty early
- Learned ANTLR pretty fast - that didn't end up causing too much trouble, after the initial learning curve

7.4.2 Room for Improvement

- We didn't keep up our rigor as the project wore on
- We implemented changes somewhat helter-skelter rather than discussing the deeper ramifications with each other (or not enough, at least)
- Didn't have enough meetings, in general
- Some form of online forum or wiki would have been nice - to discuss ideas when we couldn't meet physically
- It was a fairly ambitious project, and we didn't divide up the work very clearly until late in the game.
- Need more/better error reporting in the language.

Appendix A

Ant Project Build (Stephen/John)

```
<?xml version="1.0"?>

<project name="ELMO" default="compile_antlr" basedir=".">

<!-- build directory -->
<property name="build" value="./build" />                  <!-- all the ant output -->
<property name="buildsrc" value="${build}/src" />          <!-- .java files files -->
<property name="buildclasses" value="${build}/classes" />   <!-- .class files -->

<!-- src directory -->
<property name="src" value="./src" />
<property name="antlrsrc" value="${src}/antlr" />          <!-- .g files and .txt files -->
<property name="javasrc" value="${src}/java" />            <!-- .java files -->

<target name="init">
    <tstamp/>
    <mkdir dir="${buildsrc}" />
    <mkdir dir="${buildclasses}" />
</target>

<!-- run antlr on grammar files to generate java code -->
<target name="antlr" depends="antlr_walker" />

<target name="antlr_walker" depends="antlr_parser">
    <antlr target="${antlrsrc}/ELMO_Walker.g" outputdirectory="${buildsrc}"  dir="${antlrsrc}"/>
    <!-- since both the .java and .txt files will be put in ${buildsrc}, we need to copy the .txt files back -->
    <move todir="${antlrsrc}">
        <fileset dir="${buildsrc}">
          <include name="*.txt"/>
        </fileset>
    </move>
</target>
<target name="antlr_parser" depends="antlr_lexer">
    <antlr target="${antlrsrc}/ELMO_Parser.g" outputdirectory="${buildsrc}"  dir="${antlrsrc}"/>
    <!-- since both the .java and .txt files will be put in ${buildsrc}, we need to copy the .txt files back -->
    <move todir="${antlrsrc}">
        <fileset dir="${buildsrc}">
          <include name="*.txt"/>
        </fileset>
    </move>
</target>
<target name="antlr_lexer" depends="init">
    <antlr target="${antlrsrc}/ELMO_Lexer.g" outputdirectory="${buildsrc}" dir="${antlrsrc}"/>
    <!-- since both the .java and .txt files will be put in ${buildsrc}, we need to copy the .txt files back -->
    <move todir="${antlrsrc}">
        <fileset dir="${buildsrc}">
          <include name="*.txt"/>
        </fileset>
    </move>
</target>

<!-- old build rule, broken into finer granularity above
<target name="antlr" depends="init">
    <antlr target="${antlrsrc}/ELMO_Lexer.g" outputdirectory="${buildsrc}" />
    <antlr target="${antlrsrc}/ELMO_Parser.g" outputdirectory="${buildsrc}" />
    <antlr target="${antlrsrc}/ELMO_Walker.g" outputdirectory="${buildsrc}" />
</target>
//-->
```

```
<target name="compile_antlr" depends="compile_walker" />

<target name="compile_walker" depends="antlr,compile_parser">
    <javac debug="yes" srcdir="${buildsrc}" destdir="${buildclasses}" />
</target>

<target name="compile_parser" depends="antlr_parser,compile_lexer">
    <javac debug="yes" srcdir="${buildsrc}" destdir="${buildclasses}" />
</target>

<target name="compile_lexer" depends="antlr_lexer">
    <copy todir="${buildsrc}">
        <fileset dir="${javasrc}" />
    </copy>
    <javac debug="yes" srcdir="${buildsrc}" destdir="${buildclasses}" />
</target>

<target name="compile_testparser" depends="compile_parser">
    <copy todir="${buildsrc}" file="${javasrc}/TestParser.java" />
    <javac debug="yes" srcdir="${buildsrc}" destdir="${buildclasses}" />
</target>
<target name="compile_test" depends="compile_walker">
    <copy todir="${buildsrc}" file="${javasrc}/Tester.java" />
    <javac debug="yes" srcdir="${buildsrc}" destdir="${buildclasses}" />
</target>

<!--   TODO: have a 'build rule' but also rules to just build the test, walker, parser, lexer, etc -->

<target name="elmoc_compile" depends="compile_antlr">
    <copy todir="${buildsrc}" file="${javasrc}/Elmoc.java" />
    <javac debug="yes" srcdir="${buildsrc}" includes="elmoc.java" destdir="${buildclasses}" />
</target>

<target name="elmoc_build" depends="elmoc_compile">
    <jar destfile="${build}/elmoc.jar" basedir="${buildclasses}">
        <manifest>
            <attribute name="Main-Class" value="Elmoc" />
            <attribute name="Class-Path" value="antlr.jar" />
        </manifest>
    </jar>
    <copy todir="${build}" file="${src}\antlr.jar" />
</target>

<target name="elmoc" depends="elmoc_build">
    <fail message="Must specify -Dinput=[script.elmo] -Doutput=[output.obj]" unless="input" />
    <fail message="Must specify -Dinput=[script.elmo] -Doutput=[output.obj]" unless="output" />
    <java fork="yes" jar="${build}/elmoc.jar" maxmemory="512m">
        <arg file="${input}" />
        <arg file="${output}" />
    </java>
</target>
<target name="test_elmoc" depends="elmoc_compile">
    <fail message="Must specify -Dinput=[script.elmo] -Doutput=[output.obj]" unless="input" />
    <fail message="Must specify -Dinput=[script.elmo] -Doutput=[output.obj]" unless="output" />
    <java fork="yes" classname="Elmoc" maxmemory="512m">
        <classpath>
            <pathelement path="${java.class.path}" />
            <pathelement path="${buildclasses}" />
        </classpath>
        <arg file="${input}" />
        <arg file="${output}" />
    </java>
</target>

<target name="test" depends="compile_test">
    <fail message="Must specify -Dinput=[input.txt]!" unless="input" />
    <java fork="yes" classname="Tester" input="${input}">
        <classpath>
            <pathelement path="${java.class.path}" />
            <pathelement path="${buildclasses}" />
        </classpath>
    </java>
</target>
<target name="testparser" depends="compile_testparser">
    <java fork="yes" classname="TestParser" input="tests/script.txt">
        <classpath>
            <pathelement path="${java.class.path}" />
            <pathelement path="${buildclasses}" />
        </classpath>
    </java>
</target>
```

```
<target name="clean">
    <delete dir="${build}" />
    <delete file="${buildtarget}" />
</target>

<target name="javadoc">
</target>

</project>
```

Appendix B

ANTLR

B.1 Lexer (Erik)

```
class ELMOLexer extends Lexer;
options {
    testLiterals = false; // By default, don't check tokens against keywords
    k = 2;                // Need to decide when strings literals end
    charVocabulary = '\u0000'..'\u00FF';    //all ASCII 8 bit characters
    //exportVocab = VLLexer;    //default
}
tokens
{
    //these are tokens that aren't matched, but are returned by the lexer
    FLOAT_CONSTANT;
    INT_CONSTANT;

    //these are tokens that are keywords (IDs cannot have these names)
    /* flow control */
    IF          = "if";
    ELSE        = "else";
    FOR         = "for";
    FOREACH     = "foreach";
    IN          = "in";
//  SWITCH      = "switch";
    BREAK       = "break";

    RETURN      = "return";

    /* commands */
    PRINT       = "print";
    MOVE        = "move";
    ROTATE      = "rotate";
    SCALE       = "scale";
    STAMP       = "stamp";
    ATTACH      = "attach";
    REMOVE      = "remove";

//  CLONE       = "clone";

    /* command extras */
    INHERIT     = "inherit";    //modifies the remove command
    AROUND      = "around";
    ALONG       = "along";
    BY          = "by";
    TO          = "to";
    FROM        = "from";
    DEGREES     = "deg";
    RADIANS     = "rad";

    /* constants */
    PI          = "PI";

    /* types */
    TYPE_INTEGER = "int";
    TYPE_FLOAT  = "float";
    TYPE_OBJECT = "object";
    TYPE_GROUP  = "group";
    TYPE_VECTOR = "vector";
    TYPE_VOID   = "void";
    /* string modifiers */
//  OBJ         = "obj";        //not currently used
//  ELMO        = "elmo";

}


/* OPERATORS */
DOTX        : ".X";
DOTY        : ".Y";
DOTZ        : ".Z";
DOTx        : ".x";
DOTy        : ".y";
DOTz        : ".z";
DOTDOT      : "..";

/* punctuation */
protected DOT           : '.';
SEMI        : ';';
COMMA       : ',';

/* matching thingies */
LPAREN
options { paraphrase = "'('";}
            : '(';
RPAREN
options { paraphrase = "')'";}
            : ')';
LBRACKET
options { paraphrase = "'['";}
            : '[';
RBRACKET
options { paraphrase = "']'";}
            : ']';
LCURLY
options { paraphrase = "'{'";}
            : '{';

RCURLY
options { paraphrase = "'}'";}
            : '}';

//for abs-value operator
PIPE        : '|';

//assignment
ASSIGN_COPY     : '=';
ASSIGN_REF      : "=&";

//comparison
EQUAL           : "==";
NOT_EQUAL       : "!=";
LTE             : "<=";
LT              : "<";
GTE             : ">=";
GT              : ">";

//logical
LAND            : "&&";
LNOT            : '!';
LOR             : "||";

//arithmetic
POWER           : '^';
DIV             : '/';
DIV_ASSIGN      : "/=";
PLUS            : '+';
PLUS_ASSIGN     : "+=";
INC             : "++";
MINUS           : '-';
MINUS_ASSIGN    : "-=";
DEC             : "--";
TIMES           : '*';
TIMES_ASSIGN    : "*=";
//MOD           : '%';
//MOD_ASSIGN    : "%=";


protected LETTER        : ( 'a'..'z' | 'A'..'Z' );
protected DIGIT         : '0'..'9';
protected START_MULTICOMMENT    : "/*";
protected END_MULTICOMMENT      : "*/";
protected START_SINGLECOMMENT   : "//";
//protected DIGIT_NZ    : '1'..'9';

ID
        options
                {
                testLiterals = true;
                }
        :       ( LETTER | '_' )
                ( LETTER | '_' | DIGIT )*
//      {System.out.println("lex: ID");}
        ;

NUMBER :    (DIGIT)+
            (
                (DOT (options{greedy=true;}:DIGIT)+)=>
                (DOT (options{greedy=true;}:DIGIT)+)
                    {
                        _ttype = FLOAT_CONSTANT;
                //      System.out.println("lex: float");
                    }
                | /* nothing */
                    {
                        _ttype = INT_CONSTANT;
                //      System.out.println("lex: int");
                    }
            )
            ;


STRING  :   '"'! ( '"' '"'! | ~('"'))*  '"'!
//          {System.out.println("string");}
            ;

Comment
    :
    (
        (
            START_MULTICOMMENT
            CommentContents
            END_MULTICOMMENT
        )
        |(
            START_SINGLECOMMENT
            (~('\n'|'\r'))*
            NEWLINE
        )
    )
    { $setType(Token.SKIP); }
    ;

protected CommentContents
    :
    ( // Prevent .* from eating the whole file
        options {greedy=false;}:
        (
            NEWLINE { newline(); }
            | ~('\n'|'\r')
        )
    )*
    ;


WHITESPACE  :
                (   WHITESEP
                    | NEWLINE
                    { newline(); }
                ) { $setType(Token.SKIP); }
                ;

protected NEWLINE
    :
    (
        ("\r\n") => "\r\n"      //DOS
        | '\n'                  //UNIX
        | '\r'                  //MAC
    )
    ;
```

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

B.2 Parser (John)

```
class ELMOParser extends Parser;
options {
    buildAST = true;    // Enable AST building
    k = 3;              // Need 2 to distinguish between ID by itself and ID ASSIGN
                        // Need 3 to do Declarations (with optional assignment following)
    importVocab = ELMOLexer;

    defaultErrorHandler=false;

}

tokens {
    ProgramList;

//declarations
    VariableDeclaration;
    VariableDeclaration_Initialized;
    VariableDeclaration_InitializedRef;

    FunctionDeclaration;
    FunctionInputList;
    FunctionInputDeclaration;
    FunctionInputDeclaration_Initialized;

    FunctionBody;
    FunctionCall;
    FunctionCallArgVal;
    FunctionCallArgRef;


//Leaf nodes
    //ID    already in lexer
    ELMOVector;
    ELMOInteger<AST=ELMOInteger>;
    ELMOFloat<AST=ELMOFloat>;
//Quotes Operator
    ELMOObject<AST=ELMOObject>;
    ElmoProgram;

//Binary Operators
    BinaryOperator<AST=ELMOBinaryOperatorAST>;

//Relational Operators
    RelationalOperator<AST=ELMORelationalOperatorAST>;
    LogicalOperator<AST=ELMOLogicalOperatorAST>;
    LogicalNot<AST=ELMOLogicalOperatorAST>;

//Prefix Operators
//  PrefixPlus;     //not used currently
    PostfixOperator<AST=ELMOPostfixOperatorAST>;
    PrefixOperator<AST=ELMOPrefixOperatorAST>;

//Group Accessor Operators
    DotBig<AST=ELMODotBigAST>;

//Component Accessor Operators
    DotLittle<AST=ELMODotLittleAST>;

//Random Number Operator
    RandomNumberRange;
    RandomNumberNoRange;

//Bars Operator
    Bars;

//Angle Units
//  DEGREES     //declared in the lexer's tokens section
//  RADIANS
//Object/Vector Operators
    GroupVecRotate;         //rotate ID around expr [by expr (deg|rad)]
    GroupVecScale;          //scale ID around expr [by expr]
    GroupVecScaleAroundZero;        //the 'around vec' was omitted
    GroupVecMove;           //move ID along expr [by expr]

//Group Operators
    GroupStamp;         //stamp expr
    GroupAttach;            //attach ID to ID       ? attach expr to ID
    GroupRemove;            //remove ID from ID
    GroupInheritRemove;     //remove ID from ID inherit

//Assignment Operators
    AssignmentOperator<AST=ELMOAssignmentOperatorAST>;

// other types of constant expressions
    // add in constant vector

//Iterator Statements
    ForIter;
    ForEachIter;

//other kinds of statement things
    Statement;
    CompoundStatement;

    ReturnStatement;
    BreakStatement;
    PrintStatement;     //print expr

//flow control
    If;
    IfElse;
    //Switch;
}
{
    public static final int FLAG_ERROR      =   0x80000000;
    public static final int FLAG_CONSTANT_ONLY  =   0x00000002;
    public static final int FLAG_GENERAL        =   0x01000000;
    public int  myFlags = 0;
    public int  iterDepth = 0;
    public int  funcDepth = 0;
    private void myParseError(String s, RecognitionException e) throws RecognitionException, TokenStreamException
    {
        System.out.println("Parse Error: line "+e.getLine()+": "+s);
        System.out.println('\t'+e.getMessage());

        /*
            the following two lines are copied from ANTLR's default error handlers
            according to documentation, they
            "sync to the follow set of the rule, and return from that rule"

            important thing is they allow parsing to continue
            if they're not there, infinite errors result (because the error-causer is never consumed)
        */
        consume();
        consumeUntil(_tokenSet_0);
//      throw e;
        myFlags |= FLAG_ERROR;
    }

}
program
{
    getASTFactory().setASTNodeClass("ELMOAST");
}
    :   (
            declaration
            exception
            catch[RecognitionException e]
            {
                myParseError("invalid declaration",e);
            }

            | statement
            exception
            catch[RecognitionException e]
            {
                myParseError("invalid statement",e);
            }
        )+
        (
            EOF!
            { ## = #([ProgramList,"ProgramList"], ##); }
            exception
            catch[RecognitionException e]
            {
                myParseError("unexpected end of file",e);
            }
        )

    ;
    exception
    catch[RecognitionException e]
    {
        myParseError("empty files are not valid ELMO programs",e);
    }
declaration
    : type t:ID
    (
        ((
            ASSIGN_COPY! assignExpr { ## = #( [VariableDeclaration_Initialized, "VarDecl_Initd" ], ## );}
            | ASSIGN_REF! assignExpr    { ## = #( [VariableDeclaration_InitializedRef, "VarDecl_Initd_Ref" ], ## ); }
            |   /* nothing */           { ## = #( [VariableDeclaration, "VarDecl" ], ## ); }
        ) SEMI!
        )
        exception
        catch[RecognitionException e]
        {
            myParseError("error in variable declaration",e);
        }
        |   functionArgsDeclaration
            {
                funcDepth++;
            }
            compoundStatement {
                ## = #( [FunctionDeclaration, "FuncDecl" ], ## );
                funcDepth--;
            }
    )
    {
        ((ELMOAST)##).setInfo(t);
    }

    ;
    exception
    catch[RecognitionException e]
    {
        myParseError("declaration error",e);
    }

type
    :
    t1:TYPE_FLOAT       {((ELMOAST)##).setInfo(t1);}
    | t2:TYPE_INTEGER   {((ELMOAST)##).setInfo(t2);}
    | t3:TYPE_OBJECT    {((ELMOAST)##).setInfo(t3);}
    | t4:TYPE_GROUP     {((ELMOAST)##).setInfo(t4);}
    | t5:TYPE_VECTOR    {((ELMOAST)##).setInfo(t5);}
    | t6:TYPE_VOID      {((ELMOAST)##).setInfo(t6);}

    ;
functionArgsDeclaration
    : LPAREN!
    (
        (functionInputDeclaration (COMMA! functionInputDeclaration)*)
        | //nothing
    )
    RPAREN! {
        ## = #( [ FunctionInputList, "FunctionInputList" ], ## );
    }
    ;
functionInputDeclaration
    : type t:ID
        (
            (
                ASSIGN_COPY! { myFlags |= FLAG_CONSTANT_ONLY; } primaryExpression {
                    ## = #([FunctionInputDeclaration_Initialized,"FuncInputDecl_Initd"],##);
                    myFlags &= ~FLAG_CONSTANT_ONLY;
                }
            )
            | /* nothing */                         { ## = #([FunctionInputDeclaration,"FuncInputDecl"],##); }
        )
    {((ELMOAST)##).setInfo(t);}
    ;
statement
```

```
    :   (( assignExpr )? SEMI! {
            ## = #( [ Statement, "Statement" ], ## );
        }
        exception
        catch[RecognitionException e]
        {
            myParseError("Error in expression",e);
        }
        )
    |   compoundStatement
    |   iterationStatement
    |   selectionStatement
    |   commandStatement SEMI!
    |   returnStatement s1:SEMI!
        {
            if(funcDepth==0)
                throw new SemanticException("return statement only allowed in function body","<file>",s1.getLine(),s1.getColumn());
        }
    |   breakStatement s2:SEMI!
        {
            if(iterDepth==0)
                throw new SemanticException("break statement only allowed in iteration body","<file>",s2.getLine(),s2.getColumn());
        }
    ;

compoundStatement
    :   LCURLY! ( ( statement | declaration )+ )? RCURLY!
        { ## = #( [ CompoundStatement, "CompoundStatement" ], ## ); }
    ;
    exception
    catch[RecognitionException e]
    {
        myParseError("error in compound statement",e);
    }

expression : compareExpr;
compareExpr
    : notExpr
        (
            (
                t1:EQUAL!           { ## = #([RelationalOperator,"=="],##); ((ELMOAST)##).setInfo(t1);}
                |t2:NOT_EQUAL!          { ## = #([RelationalOperator,"!="],##); ((ELMOAST)##).setInfo(t2);}
                |t3:LTE!            { ## = #([RelationalOperator,"<="],##); ((ELMOAST)##).setInfo(t3);}
                |t4:LT!             { ## = #([RelationalOperator,"<"],##); ((ELMOAST)##).setInfo(t4);}
                |t5:GTE!            { ## = #([RelationalOperator,">="],##); ((ELMOAST)##).setInfo(t5);}
                |t6:GT!             { ## = #([RelationalOperator,">"],##); ((ELMOAST)##).setInfo(t6);}
                |t7:LAND!           { ## = #([LogicalOperator,"&&"],##); ((ELMOAST)##).setInfo(t7);}
                |t8:LOR!            { ## = #([LogicalOperator,"||"],##); ((ELMOAST)##).setInfo(t8);}
            )
            notExpr
        )*

    ;
notExpr
    : assignExpr
    | t:LNOT! notExpr                   { ## = #([LogicalNot,"!"],##); ((ELMOAST)##).setInfo(t);}
    ;
assignExpr
    : addExpr
        (
            (
                t1:ASSIGN_COPY!         { ## = #([AssignmentOperator,"="],##); ((ELMOAST)##).setInfo(t1);}
                |t2:ASSIGN_REF!         { ## = #([AssignmentOperator,"=&"],##); ((ELMOAST)##).setInfo(t2);}
                |t3:PLUS_ASSIGN!        { ## = #([AssignmentOperator,"+="],##); ((ELMOAST)##).setInfo(t3);}
                |t4:MINUS_ASSIGN!       { ## = #([AssignmentOperator,"-="],##); ((ELMOAST)##).setInfo(t4);}
                |t5:TIMES_ASSIGN!       { ## = #([AssignmentOperator,"*="],##); ((ELMOAST)##).setInfo(t5);}
                |t6:DIV_ASSIGN!         { ## = #([AssignmentOperator,"/="],##); ((ELMOAST)##).setInfo(t6);}
            )
            assignExpr  //right recursion for right-associativity
        )?
    ;
addExpr
    : multExpr
        (
            (
                t1:PLUS!            { ## = #([BinaryOperator,"+"],##); ((ELMOAST)##).setInfo(t1);}
                |t2:MINUS!          { ## = #([BinaryOperator,"-"],##); ((ELMOAST)##).setInfo(t2);}
            )multExpr
        )*
    ;
multExpr
```

```
    : powExpr
        (
            (
                t1:TIMES!           { ## = #([BinaryOperator,"*"],##); ((ELMOAST)##).setInfo(t1);}
                |t2:DIV!            { ## = #([BinaryOperator,"/"],##); ((ELMOAST)##).setInfo(t2);}
            )powExpr
        )*
    ;

powExpr
    //right-recursion means right-associative
    : postOpExpr
        (
            t:POWER!                { ## = #([BinaryOperator,"^"],##); ((ELMOAST)##).setInfo(t);}
            powExpr
        )?
    ;

postOpExpr
    : prefixExpr
        ( options { greedy=false; }:
            t1:INC!                 { ## = #([PostfixOperator,"++"],##); ((ELMOAST)##).setInfo(t1);}
            |t2:DEC!                { ## = #([PostfixOperator,"--"],##); ((ELMOAST)##).setInfo(t2);}

        )*
    ;
prefixExpr
    : suffixExpr
    | t1:MINUS! prefixExpr      { ## = #([PrefixOperator,"-"],##); ((ELMOAST)##).setInfo(t1);}
    | t2:INC! prefixExpr        { ## = #([PrefixOperator,"++"],##); ((ELMOAST)##).setInfo(t2);}
    | t3:DEC! prefixExpr        { ## = #([PrefixOperator,"--"],##); ((ELMOAST)##).setInfo(t3);}
    ;
suffixExpr
    : functionExpression
    (
        (
            t1:DOTX!        { ## = #([DotBig,".X"],##); ((ELMOAST)##).setInfo(t1);}
            |t2:DOTY!       { ## = #([DotBig,".Y"],##); ((ELMOAST)##).setInfo(t2);}
            |t3:DOTZ!       { ## = #([DotBig,".Z"],##); ((ELMOAST)##).setInfo(t3);}
        )
        (
            t4:DOTx!        { ## = #([DotLittle,".x"],##); ((ELMOAST)##).setInfo(t4);}
            |t5:DOTy!       { ## = #([DotLittle,".y"],##); ((ELMOAST)##).setInfo(t5);}
            |t6:DOTz!       { ## = #([DotLittle,".z"],##); ((ELMOAST)##).setInfo(t6);}
        )?
        |(
            t7:DOTx!        { ## = #([DotLittle,".x"],##); ((ELMOAST)##).setInfo(t7);}
            |t8:DOTy!       { ## = #([DotLittle,".y"],##); ((ELMOAST)##).setInfo(t8);}
            |t9:DOTz!       { ## = #([DotLittle,".z"],##); ((ELMOAST)##).setInfo(t9);}
        )
    )?
    ;

functionExpression
    :
    (ID functionArguments)=>
    (t:ID functionArguments)        { ## = #([FunctionCall,"FunctionCall"],##); ((ELMOAST)##).setInfo(t);}
    | barExpression
    ;

functionArguments
    : LPAREN!
    (
        (functionInput (COMMA! functionInput)*)
        | //nothing
    )
    RPAREN!
    ;
functionInput
    : t:ID
    (
        (
            ASSIGN_COPY!        { ## = #([FunctionCallArgVal,"="],##); }
            |ASSIGN_REF!        { ## = #([FunctionCallArgRef,"=&"],##); }
        )
        assignExpr
    )
    {
        ((ELMOAST)##).setInfo(t);
    }
    ;
```

```
barExpression
    : t:PIPE! expression PIPE!      { ## = #([Bars,"||"],##); ((ELMOAST)##).setInfo(t);}
    | parenExpression
    ;
parenExpression
    : LPAREN! expression RPAREN!
    | primaryExpression
    ;
primaryExpression
    :
    (
        id:ID {
            if( ( myFlags & FLAG_CONSTANT_ONLY ) != 0 ) {
                throw new SemanticException( "constant value expected!","<file>",id.getLine(),id.getColumn() );
            }
        }
        | randomNumber
        | vector
        | i:INT_CONSTANT   { ## = #([ELMOInteger,i.getText()]); ((ELMOAST)##).setInfo(i);}
        | f:FLOAT_CONSTANT { ## = #([ELMOFloat,f.getText()]); ((ELMOAST)##).setInfo(f);}
        | o:STRING          {
                                ## = #([ELMOObject,o.getText()]); ((ELMOAST)##).setInfo(o);
                                if(((ELMOObject)##).filefound==false)
                                    throw new SemanticException("file '"+((ELMOObject)##).filename+"' does not exist!","<file>",o.getLine(),o.getC
                            }
        | builtinConstant
    )
    ;

vector
    :t:LT! assignExpr COMMA! assignExpr COMMA! assignExpr GT!
    { ## = #([ELMOVector,"vec"],##); ((ELMOAST)##).setInfo(t);}
    ;

randomNumber
    :
    t:LBRACKET! assignExpr
    (
        (DOTDOT! assignExpr  RBRACKET!)     { ## = #([RandomNumberRange, "[..]"],##); }
        |RBRACKET!              { ## = #([RandomNumberNoRange, "[.]"],##); }
    )
    {
        ((ELMOAST)##).setInfo(t);
    }
    ;
builtinConstant
    : f:PI  {
            f.setText( String.valueOf( Math.PI ) );
            ## = #([ELMOFloat, f.getText()]);
            ((ELMOAST)##).setInfo(f);
        }
    ;
iterationStatement
    { iterDepth++; }
    :   (
            t1:FOR! LPAREN! ( declaration | assignExpr SEMI ! ) expression SEMI! assignExpr RPAREN!
            statement           {## = #( [ ForIter, "for" ], ## );((ELMOAST)##).setInfo(t1);}

            |t2:FOREACH! ID IN! ID statement    {## = #( [ ForEachIter, "foreach" ], ## );((ELMOAST)##).setInfo(t2);}
        )

    { iterDepth--; }
    ;
selectionStatement
    : ifStatement
//  | switchStatement
    ;
ifStatement
//this works, but uses the syntactic predicate (=>)
    : t1:IF! LPAREN! expression RPAREN! statement
    (
        (ELSE)=>        //solve the dangling else problem
        (t2:ELSE! statement)            {## = #( [ IfElse, "if-else" ], ## );((ELMOAST)##).setInfo(t1);}
        |   //nothing
        {## = #( [ If, "if" ], ## );((ELMOAST)##).setInfo(t2);}
    )
    ;

commandStatement
    :
```

```
    //TODO: fill in with rotate, scale, etc.
    t1:PRINT! expression                            { ## = #( [ PrintStatement, "print" ], ## ); ((ELMOAST)##).setInfo(t1);}
    | t2:ROTATE! ID AROUND! assignExpr (BY! assignExpr (DEGREES|RADIANS)?)? { ## = #( [ GroupVecRotate, "rotate" ], ## ); ((ELMOAST)##).setInfo(t2
    | t3:SCALE! ID  (
                (AROUND! assignExpr)
                | // nothing
                {
                    myFlags |= FLAG_GENERAL;
                }

            ) BY! assignExpr
            {
                if((myFlags&FLAG_GENERAL)==0)
                {
                    ## = #( [ GroupVecScale, "scale" ], ## );
                    ((ELMOAST)##).setInfo(t3);
                }
                else
                {
                    myFlags &= ~FLAG_GENERAL;
                    ## = #( [ GroupVecScaleAroundZero, "scale around 0" ], ## );
                    ((ELMOAST)##).setInfo(t3);
                }
            }

    | t4:MOVE! ID ALONG! assignExpr (BY! assignExpr)?                  { ## = #( [ GroupVecMove, "move" ], ## ); ((ELMOAST)##).setInfo(t4);}
    | t5:STAMP! assignExpr                                 { ## = #( [ GroupStamp, "stamp" ], ## ); ((ELMOAST)##).setInfo(t5);}
    | t6:ATTACH! ID TO! ID                                 { ## = #( [ GroupAttach, "attach" ], ## ); ((ELMOAST)##).setInfo(t6);}
    | t7:REMOVE! ID FROM! ID
                (
                    INHERIT!            { ## = #( [ GroupInheritRemove, "remove-inherit" ], ## ); ((ELMOAST)##).setInfo(t7);}
                    | /* nothing */         { ## = #( [ GroupRemove, "remove" ], ## ); ((ELMOAST)##).setInfo(t7);}
                )
    ;

returnStatement
    :
        t:RETURN! (assignExpr)?     { ## = #( [ ReturnStatement, "return" ], ## ); ((ELMOAST)##).setInfo(t);}
    ;
breakStatement
    :
        t:BREAK!            { ## = #( [ BreakStatement, "break" ], ## ); ((ELMOAST)##).setInfo(t);}
    ;
```

B.3 Walker (Stephen)

```
{
    import java.io.*;
    import java.util.*;
    import antlr.collections.ASTEnumeration;
}
class ELMOWalker extends TreeParser;
options {
    //buildAST = true;
    importVocab = ELMOParser;
    defaultErrorHandler = false;
    ASTLabelType = "ELMOAST";
}
{
    boolean hadBreak=false;
    boolean hadRet = false;
    // initialize stuff here
    private ELMOSymbolTable symbolTable = new ELMOSymbolTable( "GLOBAL", null );
    private int exprRecurseDepth = 0;
}

program
{
    ELMOIndirect a;
}
    :   #( ProgramList ( declaration[symbolTable] | a=stmt[symbolTable] )* )
    ;
//{{{ declaration
declaration[ ELMOSymbolTable symt ]
{
    ELMOIndirect a;
    int var = -1;
    HashMap b;
    ELMODataType d;
}
```

```
    :   #( VariableDeclaration_Initialized var=type id1:ID a=expr[symt] {
            if( symt.isSymbolDefined( id1.getText() ) ) {
                throw( new SemanticException( "object " + id1.getText() +
                    " is being redefined!","<file>",#VariableDeclaration_Initialized.getLine(),#VariableDeclaration_Initialized.getColumn() ) );
            }

            try{
                d = ELMODataType.getInstance( var );
                d.setCopy((ELMODataType)a.data);
                d.setName( id1.getText() );
                ELMOIndirect newSymbol = new ELMOIndirect(d);
                symt.put( id1.getText(), newSymbol );
            }
            catch(ClassCastException e) {
                throw new SemanticException("cannot assign function to variable","<file>",0,0);
            }
        })

    |   #( VariableDeclaration_InitializedRef var=type id2:ID a=expr[symt] ) {
            if( symt.isSymbolDefined( id2.getText() ) ) {
                throw( new SemanticException( "object " + id2.getText() +
                    " is being redefined!","<file>",0,0 ) );
            }

            try{

                if(var != ((ELMODataType)a.data).getTypeCode() ) {
                    throw( new SemanticException( "incompatible reference initialization " +
                        "for variable " + id2.getText(),"<file>",0,0) );
                }
                symt.put( id2.getText(), new ELMOIndirect( a.data ) );  //new name for an old variable
            }
            catch(ClassCastException e) {
                throw new SemanticException("cannot assign function reference to variable","<file>",0,0);
            }
        }

    |   #( VariableDeclaration var=type id3:ID ) {
            if( symt.isSymbolDefined( id3.getText() ) ) {
                throw( new SemanticException( "object " + id3.getText() +
                    " is being redefined!","<file>",0,0 ) );
            }

            ELMOIndirect newInst = new ELMOIndirect(ELMODataType.getInstance( var ));
            newInst.data.setName(id3.getText());
            symt.put( id3.getText(), newInst );
        }
    |   #( FunctionDeclaration var=type id4:ID b=functionArgs[symt] fbody:. ) {
            if( symt.isSymbolDefined( id4.getText() ) ) {
                throw( new SemanticException( "object " + id4.getText() +
                    " is being redefined!","<file>",0,0 ) );
            }

            //ELMOSymbolTable childScope = symt.addChildScope( id4.getText() );
            ELMOFunction func = new ELMOFunction( var, b, this,
                #fbody, symt );
                //#fbody, childScope );
            func.setName( id4.getText() );
            symt.put( id4.getText(), new ELMOIndirect(func) );
        }
    ;
    exception
    catch[SemanticException e]
    {
        throw new SemanticException(e.getMessage(),"<file>",#declaration.getLine(),#declaration.getColumn());
    }
//}}}
//{{{ functionArgs
functionArgs [ELMOSymbolTable symt] returns [ HashMap r ]
{
    ELMOArgument a;
    r = new HashMap();
}
    :   #( FunctionInputList ( a=functionArg[symt] { r.put( a.getName(), a ); } )* )
    ;
//}}}
//{{{ functionArg
functionArg [ELMOSymbolTable symt] returns [ ELMOArgument r ]
{
    ELMOIndirect a;
    int var = -1;
    r = null;
}
    :   #( FunctionInputDeclaration_Initialized var=type id1:ID a=expr[symt] ) {

            try{
                if(var != ((ELMODataType)a.data).getTypeCode()) {
                    throw( new SemanticException( "Incompatible initialization " +
                        "for variable " + id1.getText(),"<file>",0,0 ) );
                }
                //since it's initialized, the expr a is guaranteed to NOT be a reference to actual vars
                r = new ELMOArgument( ((ELMODataType)a.data).getTypeCode(), id1.getText(), a );
            }
            catch(ClassCastException e) {
                throw new SemanticException("cannot pass function as argument","<file>",0,0);
            }
        }
    |   #( FunctionInputDeclaration var=type id2:ID ) {
            r = new ELMOArgument( var, id2.getText() );
        }
    ;
//}}}
//{{{ stmt
stmt [ELMOSymbolTable symt] returns [ELMOIndirect retValue]
{
    ELMOIndirect a=null,b=null;
    retValue = null;
}
    :   #( Statement a=expr[symt])
    |   a=cmpdstmt[symt] { retValue = a; }
    |   a=iteration[symt] { retValue = a; }
    |   #( PrintStatement a=expr[symt] {
            System.out.println( a.data.toString() );
        } )
    |   #( ReturnStatement {
            if( #ReturnStatement.getNumberOfChildren() > 0 ) {
                retValue = expr( #ReturnStatement.getFirstChild(), symt );
            }
            else {
                retValue = new ELMOIndirect( new ELMOVoid() );
            }
            hadRet = true;
        } )
    |   #( BreakStatement  {
            hadBreak = true;
        })
    |   #( If a=expr[symt] {
            if( a.data instanceof ELMOInteger ) {
                ELMOInteger condition = ( ELMOInteger )a.data;

                if( condition.getValue() > 0 ) {
                    retValue = this.stmt( _t, symt );
                }
            }
        } )
    |   #( IfElse a=expr[symt] {
            if( a.data instanceof ELMOInteger ) {
                ELMOInteger condition = ( ELMOInteger )a.data;

                if( condition.getValue() > 0 ) {
                    retValue = this.stmt( _t, symt );
                }
                else {
                    retValue = this.stmt( _t.getNextSibling(), symt );
                }
            }
        } )

        |   #( GroupVecRotate id:ID a=expr[symt] b=expr[symt] {
            ELMOIndirect dt = this.expr( id, symt );

            if( !( dt.data instanceof ELMOGroup ) ) {
                throw( new SemanticException( "expected group","<file>",0,0 ) );
            }

            ELMOGroup group = ( ELMOGroup )dt.data;
            group.rotate( ( ELMOVector )a.data, ( ELMOFloat )b.data );
        } )
    |   #( GroupVecScale id1:ID a=expr[symt] b=expr[symt] {
            ELMOIndirect dt = this.expr( id1, symt );
            if( !( dt.data instanceof ELMOGroup ) ) {
                throw( new SemanticException( "expected group","<file>",0,0 ) );
            }
            ELMOGroup group = ( ELMOGroup )dt.data;
            ELMOVector s;
            if(b.data instanceof ELMOVector)
                s = (ELMOVector)(b.data);
            else if(b.data instanceof ELMOFloat)
                s = new ELMOVector((ELMOFloat)b.data,(ELMOFloat)b.data,(ELMOFloat)b.data);
            else if(b.data instanceof ELMOInteger)
                s = new ELMOVector((ELMOInteger)b.data,(ELMOInteger)b.data,(ELMOInteger)b.data);
            else
                throw new SemanticException("scale factor must be either a number or a vector","<file>",0,0);
            group.scale( ( ELMOVector )a.data, s );
        } )
    |   #( GroupVecScaleAroundZero id99:ID b=expr[symt] {
            ELMOIndirect dt = this.expr( id99, symt );
            if( !( dt.data instanceof ELMOGroup ) ) {
                throw( new SemanticException( "expected group" ) );
            }
            ELMOGroup group = ( ELMOGroup )dt.data;
            ELMOVector s;
            if(b.data instanceof ELMOVector)
                s = (ELMOVector)(b.data);
            else if(b.data instanceof ELMOFloat)
                s = new ELMOVector((ELMOFloat)b.data,(ELMOFloat)b.data,(ELMOFloat)b.data);
            else if(b.data instanceof ELMOInteger)
                s = new ELMOVector((ELMOInteger)b.data,(ELMOInteger)b.data,(ELMOInteger)b.data);
            else
                throw new SemanticException("scale factor must be either a number or a vector","<file>",0,0);
            group.scale( new ELMOVector(0,0,0), s );
        } )
    |   #( GroupVecMove id2:ID a=expr[symt] (b=expr[symt])? {
            ELMOIndirect dt = this.expr( id2, symt );
            if( !( dt.data instanceof ELMOGroup ) ) {
                throw( new SemanticException( "expected group","<file>",0,0 ) );
            }

            ELMOVector trans;
            if(b!=null)
            {
                if(b.data instanceof ELMOInteger)
                {
                    trans = ((ELMOVector)a.data).mult((ELMOInteger)b.data);
                }
                else if(b.data instanceof ELMOFloat)
                {
                    trans = ((ELMOVector)a.data).mult((ELMOFloat)b.data);
                }
                else
                {
                    trans = new ELMOVector(((ELMOVector)a.data).getX().getValue(),
                                ((ELMOVector)a.data).getY().getValue(),
                                ((ELMOVector)a.data).getZ().getValue());
                }
            }
            else
            {
                trans = new ELMOVector(((ELMOVector)a.data).getX().getValue(),
                                ((ELMOVector)a.data).getY().getValue(),
                                ((ELMOVector)a.data).getZ().getValue());
            }

            ELMOGroup group = ( ELMOGroup )dt.data;
            group.translate( trans );
        } )
    |   #( GroupAttach a=expr[symt] id3:ID {
            ELMOIndirect dt = this.expr( id3, symt );

            if( !( dt.data instanceof ELMOGroup ) ) {
                throw( new SemanticException( "expected group","<file>",0,0 ) );
            }

            ELMOGroup group = ( ELMOGroup )dt.data;
            group.attach( ( ELMOGroup )a.data );
        } )
    |   #( GroupRemove a=expr[symt] id4:ID {
            ELMOIndirect dt = this.expr( id4, symt );

            if( !( dt.data instanceof ELMOGroup ) ) {
                throw( new SemanticException( "expected group","<file>",0,0 ) );
            }
```

```
            ELMOGroup group = ( ELMOGroup )dt.data;
            group.remove( ( ELMOGroup )a.data );
        } )
    |   #( GroupInheritRemove a=expr[symt] id5:ID {
            ELMOIndirect dt = this.expr( id5, symt );

            if( !( dt.data instanceof ELMOGroup ) ) {
                throw( new SemanticException( "expected group","<file>",0,0 ) );
            }

            ELMOGroup group = ( ELMOGroup )dt.data;
            group.removeWithInheritance( ( ELMOGroup )a.data );
        } )

    |   #( GroupStamp id6:ID {
            ELMOIndirect dt = this.expr( id6, symt );

            if( !( dt.data instanceof ELMOGroup ) ) {
                throw( new SemanticException("expected group","<file>",0,0 ) );
            }
            ELMOGroup group = ( ELMOGroup )dt.data;
            group.stamp();
        } )
    ;
    exception
    catch[SemanticException e]
    {
        throw new SemanticException(e.getMessage(),"<file>",#stmt.getLine(),#stmt.getColumn());
    }
//}}}
//{{{ cmpdstmt
cmpdstmt [ELMOSymbolTable symt] returns [ELMOIndirect retValue]
{
    ELMOIndirect a;
    retValue = null;
}
    :   #( CompoundStatement
            (

                (
                    (
                        declaration[symt]
                        |a=stmt[symt]
                        {
                            if(hadRet)
                            {
                                retValue=a;
                                return retValue;
                            }
                        }
                    )
                    {
                        if(hadBreak)
                        {
                            //hadBreak = false;
                            return null;
                        }
                    }
                )*
            )
            {
                //check whether a return statement was hit
            }
        )
    ;
//}}}
//{{{ iteration
iteration [ELMOSymbolTable symt] returns [ELMOIndirect retValue]
{
    ELMOIndirect a, b, c, d;
    retValue = null;
}
    :   #( ForIter {
            ELMOSymbolTable forsymt = symt.addChildScope();
        } ( declaration[forsymt] | a=expr[forsymt] ) {
            b = this.expr( _t, forsymt );

            if( !( b.data instanceof ELMOInteger ) ) {
                throw( new SemanticException( "condition must evaluate to an integer","<file>",0,0 ) );
            }
            ELMOInteger condition = ( ELMOInteger )b.data;
            AST incNode = _t.getNextSibling();
            AST cmpStmtNode = incNode.getNextSibling();

            while( condition.getValue() > 0) {
                retValue = this.stmt( cmpStmtNode, forsymt );
                this.expr( incNode, forsymt );
                if(hadBreak||hadRet)
                {
                    hadBreak = false;
                    break;
                }
                condition = ( ELMOInteger )(this.expr( _t, forsymt ).data);
            }

            symt.removeScope( forsymt );
        } )
    |   #( ForEachIter {
            ELMOSymbolTable forsymt = symt.addChildScope();
        } id1:ID id2:ID {
            ELMODataType dt = ( ELMODataType )(this.expr( id2, symt ).data);

            if( !( dt instanceof ELMOGroup ) ) {
                throw( new SemanticException( "expected group","<file>",0,0 ) );
            }

            ELMOGroup g = ( ELMOGroup )dt;
            Iterator groupIter = g.iterator();
            ELMOGroup iterVar = null;

            while( groupIter.hasNext()) {
                iterVar = ( ELMOGroup )groupIter.next();
                forsymt.put( id1.getText(), new ELMOIndirect(iterVar) );
                retValue = this.stmt( _t, forsymt );
                if(hadBreak||hadRet)
                {
                    hadBreak = false;
                    break;
                }
            }

            symt.removeScope( forsymt );
        } )
    ;
//}}}
//{{{ identifier
identifier [ELMOSymbolTable symt] returns [ELMOIndirect r]
{
    r=null;
}
    :
    #(i1:ID
    {
        //check to see if i1 is in the symbol table
        r = symt.lookupSymbol(i1.getText());
        if(r==null)
        {
            throw( new SemanticException( "no symbol named " + i1.getText()
                + " is defined","<file>",0,0 ) );
        }
    })
    |   #( DotLittle r=identifier[symt] {
            r = ( ( ELMODotLittleAST )#DotLittle ).value( ( ELMODataType )r.data );
        } )
    ;

//}}}
//{{{ lhs
lhs [ELMOSymbolTable symt] returns [ELMOIndirect r]
{
    r = null;
}
    :
    r=identifier[symt]
    {
        if(!(r.data instanceof ELMODataType))   //not a function, etc
            r = null;
    }
    | r=constantExpr    //this is only here to generate an error
    {
        //need this silly if(true) because otherwise the java code
        //generated by the grammar has an unreachable 'break' statement
        //which javac complains about
        if(true)
            throw new SemanticException("left-hand-side may not be a constant","<file>",0,0);
    }
    ;
//}}}
//{{{ expr
expr [ELMOSymbolTable symt] returns [ ELMOIndirect r ]
{
    Integer tmp;
    ELMOIndirect a, b;
    ELMOIndirect a1, a2, a3;
    ELMOArgument arg;
    ELMOIndirect sym;
    r = null;
    exprRecurseDepth++;
}
    // assignment operators
    :   (
    #( AssignmentOperator a=lhs[symt] b=expr[symt] {
            r = ( ( ELMOAssignmentOperatorAST )#AssignmentOperator ).assign( a, b );
        } )

    // prefix operators
    |   #( PrefixOperator a=expr[symt] {
            r = new ELMOIndirect( ( ( ELMOPrefixOperatorAST )
                #PrefixOperator ).eval( ( ELMODataType )a.data ) );
        } )
    // postfix operators
    |   #( PostfixOperator r=lhs[symt] {
            ( ( ELMOPostfixOperatorAST )#PostfixOperator ).freeze(
                ( ELMOInteger )r.data
            );
        } )

    // binary operators
    |   #( BinaryOperator a=expr[symt] b=expr[symt] ) {
            try{
                r = new ELMOIndirect( ( ( ELMOBinaryOperatorAST )#BinaryOperator ).value( (ELMODataType)a.data, (ELMODataType)b.data ));
            }
            catch(ClassCastException e) {
                throw new SemanticException("expected vector","<file>",0,0);
            }
        }

    |   #( FunctionCall funcName:ID { HashMap funcCallArgs = new HashMap(); }
        ( arg=functionCallArg[symt]
                {
                    funcCallArgs.put( arg.getName(), arg );
                }
        )* {

            ELMOIndirect o = symt.lookupSymbol( funcName.getText() );

            if( o == null ) {
                throw( new SemanticException( "function " + funcName.getText() +
                    " not found","<file>",0,0 ) );
            }

            if( !( o.data instanceof ELMOFunction ) ) {
                throw( new SemanticException( funcName.getText() + " is not a function","<file>",0,0 ) );
            }
            exprRecurseDepth--;
            r = (( ELMOFunction )o.data).invoke( funcCallArgs );
            hadRet = false;
            exprRecurseDepth++;
        } )

    // relational operators
    |   #( RelationalOperator a=expr[symt] b=expr[symt] {
            try {
                r = new ELMOIndirect( ( ( ELMORelationalOperatorAST )#RelationalOperator ).value(
                    ( ELMODataType )a.data, ( ELMODataType )b.data )
                );
            }
            catch( ClassCastException e ) {
                throw( new SemanticException( "can't use function in comparison","<file>",0,0 ) );
            }
        } )
    |   #( LogicalOperator a=expr[symt] b=expr[symt] {
            try {
                r = new ELMOIndirect( ( ( ELMOLogicalOperatorAST )#LogicalOperator ).value(
                    ( ELMODataType )a.data, ( ELMODataType )b.data )
                );
            }
            catch( ClassCastException e ) {
                throw( new SemanticException( "can't use function in comparison","<file>",0,0 ) );
            }
        } )
    |   #( LogicalNot a=expr[symt] {
            try {
                r = new ELMOIndirect( ( ( ELMOLogicalOperatorAST )#LogicalNot ).value(
                    ( ELMODataType )a.data )
                );
            }
            catch( ClassCastException e ) {
                throw( new SemanticException( "can't use function in comparison","<file>",0,0 ) );
            }
        } )
    // dot BIG (X|Y|Z)
    |   #( DotBig a=expr[symt] {
            r = new ELMOIndirect( ( ( ELMODotBigAST )#DotBig ).value( ( ELMODataType )a.data ) );
        } )

    // random number stuff
    |   #( RandomNumberRange a=expr[symt] b=expr[symt] {
            r = new ELMOIndirect( new ELMORandomFloat( ( ELMODataType )a.data,
                ( ELMODataType )b.data ) );
        } )
    |   #( RandomNumberNoRange a=expr[symt] {
            r = new ELMOIndirect( new ELMORandomFloat( ( ELMODataType )a.data ) );
        } )

    // vector specific expressions
    |   #( Bars a=expr[symt] {
            if( a.data instanceof ELMOVector ) {
                r = new ELMOIndirect( ( ( ELMOVector )a.data ).length());
            }
        } )

    |   r=identifier[symt]
    // basic types
    |   r=constantExpr
    |   #( ELMOVector a1=expr[symt] a2=expr[symt] a3=expr[symt] ) {
        try{
            r = new ELMOIndirect( new ELMOVector( (ELMODataType)a1.data, (ELMODataType)a2.data, (ELMODataType)a3.data ));
        }
        catch(ClassCastException e)
        {
            throw new SemanticException("vector components may not be functions","<file>",0,0);
        }
        }
    ) {
        exprRecurseDepth--;
        if( exprRecurseDepth == 0 ) {
            ELMOPostfixOperatorAST.evalAll();
        }
    }
    ;
///}}}
//{{{ constantExpr
constantExpr returns [ ELMOIndirect r ]
{
    //ELMODataType a1, a2, a3;
    r = null;
}
    :   #( ELMOInteger {
            r = new ELMOIndirect( ( ELMOInteger )#ELMOInteger );
        } )
    |   #( ELMOFloat {
            r = new ELMOIndirect( ( ELMOFloat )#ELMOFloat );
        } )
    |   #( ELMOObject {
            r = new ELMOIndirect( ( ELMOObject )#ELMOObject );
        } )
    ;

//}}}
//{{{ functionCallArg
functionCallArg [ ELMOSymbolTable symt ] returns [ ELMOArgument r ]
{
    ELMOIndirect a;
    r = null;
}
    :   #( FunctionCallArgVal name1:ID a=expr[symt] {
            try{
                ELMODataType var = ELMODataType.getInstance( ((ELMODataType)a.data).getTypeCode() );
                var.setName(name1.getText());

                var.setCopy((ELMODataType)a.data);
                r = new ELMOArgument( ((ELMODataType)a.data).getTypeCode(), name1.getText(), new ELMOIndirect(var) );
            }
            catch(ClassCastException e) {
                throw new SemanticException("can't use function as a function argument","<file>",0,0);
            }
        } )
    |   #( FunctionCallArgRef name2:ID a=expr[symt] {
            try{
                r = new ELMOArgument( ((ELMODataType)a.data).getTypeCode(), name2.getText(), a );
            }
            catch(ClassCastException e) {
                throw new SemanticException("can't use function as a function argument","<file>",0,0);
            }
        } )
    ;
//}}}
//{{{ type
type returns [ int r ]
{
    r = -1;
}
    :   TYPE_INTEGER {
            r = ELMODataType.TYPE_CODE_INTEGER;
        }
    |   TYPE_FLOAT {
            r = ELMODataType.TYPE_CODE_FLOAT;
        }
    |   TYPE_VECTOR {
            r = ELMODataType.TYPE_CODE_VECTOR;
        }
    |   TYPE_GROUP {
            r = ELMODataType.TYPE_CODE_GROUP;
            }
    |   TYPE_OBJECT {
            r = ELMODataType.TYPE_CODE_OBJECT;
        }
    |   TYPE_VOID {
            r = ELMODataType.TYPE_CODE_VOID;
        }
    ;
//}}}
```

Appendix C

Java

C.1 Generic ELMO Type Exception

```
import antlr.SemanticException;

class BadELMOTypeException extends SemanticException
{
    BadELMOTypeException(int lhs_type, int rhs_type, String filename, int line, int column)
    {
        super("cannot cast "+ELMODataType.typeString(rhs_type)+" to "+ELMODataType.typeString(lhs_type),filename,line,column);
    }
    BadELMOTypeException(String message)
    {
        super(message);
    }
}
```

C.2 Walker Operator AST Classes (Stephen/John)

```
import antlr.CommonAST;
import antlr.Token;
import antlr.SemanticException;

public class ELMOAssignmentOperatorAST extends ELMOAST {
    public ELMOAssignmentOperatorAST() {
        super();
    }

    public ELMOAssignmentOperatorAST( Token tok ) {
        super( tok );
    }

    public ELMOIndirect assign( ELMOIndirect a, ELMOIndirect b )
        throws SemanticException {
        try {
            if( getText().equals( "=" ) ) {
                ( ( ELMODataType )a.data ).setCopy( ( ELMODataType )b.data );
                return( a );
            }
            else if( getText().equals( "=&" ) ) {
                if( ( ( ELMODataType )a.data ).getTypeCode() !=
                    ( ( ELMODataType )b.data ).getTypeCode() ) {
                    throw( new BadELMOTypeException(
                        ( ( ELMODataType )a.data ).getTypeCode(),
                        ( ( ELMODataType )b.data ).getTypeCode(),
                        "<file>",0,0
                    ) );
                }

                //this alters the actual internal pointer in the symbol table
                //(so if b changes, a.data changes too)
                a.data = b.data;
                return( a );
            }
            else if( getText().equals( "+=" ) ) {
                ( ( ELMODataType )a.data ).setCopy(
                    ( ( ELMODataType )a.data ).plus(
                    ( ELMODataType )b.data ) );
                return( a );
            }
            else if( getText().equals( "-=" ) ) {
                ( ( ELMODataType )a.data ).setCopy(
                    ( ( ELMODataType )a.data ).minus(
                    ( ELMODataType )b.data ) );
                return( a );
            }
            else if( getText().equals( "*=" ) ) {
                ( ( ELMODataType )a.data ).setCopy(
                    ( ( ELMODataType )a.data ).star(
                    ( ELMODataType )b.data ) );
                return( a );
            }
            else if( getText().equals( "/=" ) ) {
                ( ( ELMODataType )a.data ).setCopy(
                    ( ( ELMODataType )a.data ).slash(
                    ( ELMODataType )b.data ) );
                return( a );
            }
        }
        catch( ClassCastException e ) {
            throw( new SemanticException( "function not allowed in assignment","<file>",0,0 ) );
        }
        catch( BadELMOTypeException e ) {
            throw( new SemanticException( e.getMessage(),"<file>",0,0 ) );
        }

        throw( new SemanticException( "'" + getText() + "' is an unknown " +
            "assignment operator","<file>",0,0 ) );
    }
}

import antlr.CommonAST;
import antlr.Token;
import antlr.SemanticException;

public class ELMOBinaryOperatorAST extends ELMOAST {
    public ELMOBinaryOperatorAST() {
        super();
    }

    public ELMOBinaryOperatorAST( Token tok ) {
        super(tok);
    }

    public ELMODataType value( ELMODataType a, ELMODataType b )
        throws SemanticException {
        if( getText().equals( "+" ) ) {
            return( a.plus( b ) );
        }
        else if( getText().equals( "-" ) ) {
            return( a.minus( b ) );
        }
        else if( getText().equals( "*" ) ) {
            return( a.star( b ) );
        }
        else if( getText().equals( "/" ) ) {
            return( a.slash( b ) );
        }
        else if( getText().equals( "^" ) ) {
            return( a.carat( b ) );
        }

        throw( new SemanticException( "unknown binary operator" ) );
    }
}

import antlr.CommonAST;
import antlr.Token;
import antlr.SemanticException;

public class ELMODotBigAST extends ELMOAST {
    public ELMODotBigAST() {
        super();
    }

    public ELMODotBigAST( Token tok ) {
        super( tok );
    }
    public ELMOVector value( ELMODataType a )
        throws SemanticException {
        if( !( a instanceof ELMOGroup ) ) {
            throw( new BadELMOTypeException( "group expected" ) );
        }

        ELMOGroup g = ( ELMOGroup )a;

        if( getText().equals( ".X" ) ) {
            return( g.X() );
        }
        else if( getText().equals( ".Y" ) ) {
            return( g.Y() );
        }
        else if( getText().equals( ".Z" ) ) {
            return( g.Z() );
        }

        throw( new SemanticException( "expected .X .Y or .Z" ) );
    }
}

import antlr.CommonAST;
import antlr.Token;
import antlr.SemanticException;

public class ELMODotLittleAST extends ELMOAST {
    public ELMODotLittleAST() {
        super();
    }

    public ELMODotLittleAST( Token tok ) {
        super( tok );
    }

    public ELMOIndirect value( ELMODataType a )
        throws SemanticException {
        if( !( a instanceof ELMOVector ) )  {
            throw( new BadELMOTypeException( "expected a vector" ) );
        }

        ELMOVector v = ( ELMOVector )a;

        if( getText().equals( ".x" ) ) {
            return( v.getXind() );
        }
        else if( getText().equals( ".y" ) ) {
            return( v.getYind() );
        }
        else if( getText().equals( ".z" ) ) {
            return( v.getZind() );
        }

        throw( new SemanticException( "expected .x, .y, or .z" ) );
    }
}

import antlr.CommonAST;
import antlr.Token;
import antlr.SemanticException;

public class ELMOLogicalOperatorAST extends ELMOAST {
    public ELMOLogicalOperatorAST() {
        super();
    }

    public ELMOLogicalOperatorAST( Token tok ) {
        super(tok);
    }

    public ELMOInteger value( ELMODataType a, ELMODataType b )
        throws SemanticException {
        if( getText().equals( "&&" ) ) {
            return( a.ampamp( b ) );
        }
        else if( getText().equals( "||" ) ) {
            return( a.pipepipe( b ) );
        }

        throw( new SemanticException( "unknown logical operator!" ) );
    }
    public ELMOInteger value( ELMODataType a ) throws SemanticException {
        if( getText().equals( "!" ) ) {
            return( a.bang() );
        }

        throw( new SemanticException( "unknown logical operator!" ) );
    }
}

import antlr.CommonAST;
import antlr.Token;
import antlr.SemanticException;
import java.util.Vector;
import java.util.Enumeration;

public class ELMOPostfixOperatorAST extends ELMOAST {
    private static Vector _frozenVars = new Vector();
    private ELMOInteger _frozenVar;

    public ELMOPostfixOperatorAST() {
        super();
    }

    public ELMOPostfixOperatorAST( Token tok ) {
        super( tok );
    }

    public void freeze( ELMOInteger v ) {
        _frozenVar = v;
        _frozenVars.add( this );
    }

    public void eval() throws SemanticException {
        if( _frozenVar != null ) {
            if( getText().equals( "++" ) ) {
                _frozenVar.increment();
                return;
            }
            else if( getText().equals( "--" ) ) {
                _frozenVar.decrement();
                return;
            }

            throw( new SemanticException( "unknown postfix operator!" ) );
        }

        throw( new SemanticException( "variable expected in postfix operation" ) );
    }

    public static void evalAll() throws SemanticException {
        Enumeration postfixEnum = _frozenVars.elements();

        while( postfixEnum.hasMoreElements() ) {
            ELMOPostfixOperatorAST pfNode =
                ( ELMOPostfixOperatorAST )postfixEnum.nextElement();
            pfNode.eval();
        }

        _frozenVars.removeAllElements();
    }
}

import antlr.CommonAST;
import antlr.Token;
import antlr.SemanticException;

public class ELMOPrefixOperatorAST extends ELMOAST {
    public ELMOPrefixOperatorAST() {
        super();
    }

    public ELMOPrefixOperatorAST( Token tok ) {
        super( tok );
    }

    public ELMODataType eval( ELMODataType a ) throws SemanticException {
        if( getText().equals( "-" ) ) {
            return( a.minus() );
        }
        else if( getText().equals( "++" ) ) {
            if( a instanceof ELMOInteger ) {
                ( ( ELMOInteger )a ).increment();
                return( a );
            }
        }
        else if( getText().equals( "--" ) ) {
            if( a instanceof ELMOInteger ) {
                ( ( ELMOInteger )a ).decrement();
                return( a );
            }
        }

        throw( new SemanticException( "unknown prefix operator" ) );
    }
}

import antlr.CommonAST;
import antlr.Token;
import antlr.SemanticException;

public class ELMORelationalOperatorAST extends ELMOAST {
    public ELMORelationalOperatorAST() {
        super();
    }

    public ELMORelationalOperatorAST( Token tok ) {
        super(tok);
    }

    public ELMOInteger value( ELMODataType a, ELMODataType b )
        throws SemanticException {
        if( getText().equals( "<" ) ) {
            return( a.lt( b ) );
        }
        else if( getText().equals( "<=" ) ) {
            return( a.leq( b ) );
        }
        else if( getText().equals( ">" ) ) {
            return( a.gt( b ) );
        }
        else if( getText().equals( ">=" ) ) {
            return( a.geq( b ) );
        }
        else if( getText().equals( "==" ) ) {
            return( a.eq( b ) );
        }
        else if( getText().equals( "!=" ) ) {
            return( a.neq( b ) );
        }

        throw( new SemanticException( "unknown relational operator" ) );
    }
}

import java.util.HashMap;
import java.util.HashSet;
import java.util.Stack;
import java.util.*;

public class ELMOSymbolTable extends HashMap {
    private static int _anonymousContext = 0;

    private ELMOSymbolTable _parent;
    private HashSet _children;
    private String _name;

    public ELMOSymbolTable( String name, ELMOSymbolTable parent ) {
        _name = name;
        _parent = parent;
        _children = new HashSet();
    }

    public ELMOSymbolTable addChildScope() {
        return( addChildScope( "<ANON_" + _anonymousContext++ + ">" ) );
    }

    public ELMOSymbolTable addChildScope( String name ) {
        ELMOSymbolTable childScope = new ELMOSymbolTable( name, this );
        _children.add( childScope );
        return( childScope );
    }
```

```
public void removeScope( ELMOSymbolTable symt ) {
    _children.remove( symt );
}

protected ELMOSymbolTable getParent() {
    return( _parent );
}

public boolean isSymbolDefined( String name ) {
    // shallow binding -- start at current symbol table
    ELMOSymbolTable symbolTable = this;

    do {
        if( symbolTable.containsKey( name ) ) {
            return( true );
        }

        symbolTable = symbolTable.getParent();
    }
    while( symbolTable != null );

    return( false );
}

public String getName() {
    return( _name );
}

public ELMOIndirect lookupSymbol( Object key ) {
    String name = ( String )key;

    // shallow binding -- start at current symbol table
    ELMOSymbolTable symbolTable = this;

    do {
        if( symbolTable.containsKey( name ) ) {
            return( ( ELMOIndirect )symbolTable.get( name ) );
        }

        symbolTable = symbolTable.getParent();
    }
    while( symbolTable != null );

    return( null );
}

private static void _outputScope( ELMOSymbolTable symbolTable, String name ) {
    ELMOSymbolTable p = symbolTable;
    Stack scopes = new Stack();

    while( p != null ) {
        scopes.push( p.getName() );
        p = p.getParent();
    }

    StringBuffer scopesString = new StringBuffer();

    while( !scopes.empty() ) {
        String n = ( String )scopes.pop();
        scopesString.append( n ).append( "::" );
    }

    scopesString.append( name );
    System.out.println( scopesString.toString() );
}

public void putSymbol( String key, ELMOIndirect s ) {
    if( !isSymbolDefined( key ) ) {
        put( key, s );
        return;
    }

    // if symbol is already defined but not locally,
    // march up to the where it was defined
    ELMOSymbolTable symbolTable = this;

    do {
        if( symbolTable.containsKey( key ) ) {
            symbolTable.put( key, s );
            return;
            }

            symbolTable = symbolTable.getParent();
        }
        while( symbolTable != null );
    }
    public void printSymbols()
    {
        //print my stuff
        Iterator myit = this.keySet().iterator();
        String s;

        System.out.print("scope: "+_name);
        if(_parent!=null)
        {
            System.out.println("(child of "+_parent.getName()+")");
        }
        else
        {
            System.out.println("(root)");
        }

        while(myit.hasNext())
        {
            s = (String)myit.next();
            System.out.print("key: "+s);
            System.out.println(", val: "+((ELMOIndirect)this.get(s)).data);
        }
        //print children stuff
        myit = _children.iterator();
        while(myit.hasNext())
        {
            ((ELMOSymbolTable)myit.next()).printSymbols();
        }

    }
}

class ELMOIndirect
{
    public ELMOIndirect(ELMOSimple d)
    {
        data = d;
    }
    public ELMOSimple data;
}
```

C.3 ELMO Data Types and ELMO Functions (John/Stephen)

```
import antlr.*;
import antlr.collections.*;

public class ELMOAST extends CommonAST
{
    int line;
    int column;


    public ELMOAST() {
    }
    public ELMOAST(Token tok) {
        super(tok);
//      line = tok.getLine();
//      column = tok.getColumn();
    }
    public void initialize(AST t) {
        super.initialize(t);
        if (t instanceof ELMOAST)
        {
            line = ((ELMOAST)t).getLine();
            column = ((ELMOAST)t).getColumn();
        }
    }
    public void initialize(int t, java.lang.String txt)
    {
        super.initialize(t,txt);
    }
    public void initialize(Token tok) {
        super.initialize(tok);
        line = tok.getLine();
        column = tok.getColumn();

    }


    public int getLine()
    {return line;}
    public int getColumn()
    {return column;}
    public void setLine(int l)
    {line =  l;}
    public void setColumn(int c)
    {column = c;}

    public void setInfo(Token t)
    {
        if(t!=null)
        {
            line = t.getLine();
            column = t.getColumn();
        }
    }
}

import antlr.CommonAST;
import antlr.Token;

public abstract class ELMOSimple extends ELMOAST {
    private String _name = null;

    public ELMOSimple() {
        super();
    }
    public ELMOSimple(String name) {
        super();
        setName(name);
    }

    public ELMOSimple( Token t ) {
        super( t );
    }

    public String name() {
        return( _name );
    }

    public void setName( String name ) {
        _name = name;
    }

    public String toString()
    {
        if(_name==null)
            return new String("ELMOSimple-nullname");
        else
            return new String(_name);
        //return new String(_name);
    }
}

import antlr.SemanticException;
import antlr.Token;

public abstract class ELMODataType extends ELMOSimple {
    protected static final int TYPE_CODE_INTEGER = 0;
    protected static final int TYPE_CODE_FLOAT   = 1;
    protected static final int TYPE_CODE_VECTOR  = 2;
    protected static final int TYPE_CODE_GROUP   = 3;
    protected static final int TYPE_CODE_OBJECT  = 4;
    protected static final int TYPE_CODE_VOID    = 5;

    public static final int TYPE_CODE = -1;

    public static final String[] typeStrings = new String[]
                                {
                                    "integer",
                                    "float",
                                    "vector",
                                    "group",
                                    "object",
                                    "void"
```

```
                            };

//track how many names have been generated for each type
private static final int[] counters = new int[]
                            {
                                0,  //integer
                                0,  //float
                                0,  //vector
                                0,  //group
                                0,  //object
                                0   //void
                            };

public static String typeString(int type)
{
    return new String(typeStrings[type]);
}
public static String generateName(int type)
{
    return new String(typeString(type).concat(String.valueOf(counters[type]++)));
}

public ELMODataType() {
    super();
}
public ELMODataType(String name)
{
    super(name);
}

public ELMODataType( Token t ) {
    super( t );
}

public int getTypeCode() {
    if( this instanceof ELMOInteger ) {
        return( TYPE_CODE_INTEGER );
    }
    else if( this instanceof ELMOFloat ) {
        return( TYPE_CODE_FLOAT );
    }
    else if( this instanceof ELMOVector ) {
        return( TYPE_CODE_VECTOR );
    }
    else if( this instanceof ELMOObject ) {
        return( TYPE_CODE_OBJECT );
    }
    else if( this instanceof ELMOGroup ) {
        return( TYPE_CODE_GROUP );
    }
    else if( this instanceof ELMOVoid ) {
        return( TYPE_CODE_VOID );
    }

    return( TYPE_CODE );
}

//must be defined by each subclass
//return a new object, with internal values the same as this
abstract public ELMODataType copy();
//set the internal values of this to be a copy of input
abstract public void setCopy(ELMODataType input) throws BadELMOTypeException;


public static ELMOFace[] arrayCopy(ELMOFace[] in)
{

    ELMOFace[] ret = new ELMOFace[in.length];
    for(int i=0;i<in.length;i++)
    {
        ret[i] = ( ELMOFace )in[i].copy();
    }
    return ret;
}

public static ELMOVector[] arrayCopy(ELMOVector[] in)
{

    ELMOVector[] ret = new ELMOVector[in.length];
    for(int i=0;i<in.length;i++)
    {
        ret[i] = ( ELMOVector )in[i].copy();
    }
    return ret;
}


public static final ELMODataType getInstance( int type ) {
    return getInstance(type,"<unnamed>");
}
public static final ELMODataType getInstance (int type, String name) {
    switch( type ) {
        case TYPE_CODE_INTEGER:
            return( new ELMOInteger() );
        case TYPE_CODE_FLOAT:
            return( new ELMOFloat() );
        case TYPE_CODE_VECTOR:
            return( new ELMOVector() );
        case TYPE_CODE_OBJECT:
            return ( new ELMOObject() );
        case TYPE_CODE_GROUP:
            return ( new ELMOGroup() );
    }

    return( new ELMOVoid() );
}

// binary operators
public ELMODataType plus( ELMODataType a )
    throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMODataType minus() throws SemanticException {
    throw( new SemanticException( "negation only applies to int, float or vector","<file>",getLine(),getColumn() ) );
}

public ELMODataType minus( ELMODataType a )
    throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMODataType star( ELMODataType a )
    throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMODataType slash( ELMODataType a )
    throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMODataType carat( ELMODataType a )
    throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMOInteger lt( ELMODataType a ) throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMOInteger leq( ELMODataType a ) throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMOInteger gt( ELMODataType a ) throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMOInteger geq( ELMODataType a ) throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMOInteger eq( ELMODataType a ) throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMOInteger neq( ELMODataType a ) throws BadELMOTypeException {
    throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
}

public ELMOInteger ampamp( ELMODataType a ) throws BadELMOTypeException {
        throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
    }

    public ELMOInteger pipepipe( ELMODataType a ) throws BadELMOTypeException {
        throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",getLine(),getColumn() ) );
    }

    public ELMOInteger bang() throws SemanticException {
        throw( new SemanticException( "logical not can only be applied to int","<file>",getLine(),getColumn() ) );
    }

    public String toString()
    {
        return name();
    }
}

import antlr.SemanticException;
import antlr.Token;
import antlr.collections.AST;

public class ELMOInteger extends ELMODataType {
    private int _value;

    public ELMOInteger(int v, String name)
    {
        super(name);
        setValue(v);
    }

    public ELMOInteger() {
        this(0,ELMODataType.generateName(ELMODataType.TYPE_CODE_INTEGER));
    }

    public ELMOInteger( int v ) {
        this(v,ELMODataType.generateName(ELMODataType.TYPE_CODE_INTEGER));
    }

    public ELMOInteger( boolean b ) {
        super();
        setValue( b ? 1 : 0 );
    }

    public ELMOInteger( Token t ) {
        super( t );
    }

    public ELMOInteger( double v ) {
        this((int)v,ELMODataType.generateName(ELMODataType.TYPE_CODE_INTEGER));
    }

    public void initialize( AST a ) {
        super.initialize( a );
        _value = Integer.parseInt( a.getText() );
    }

    public void initialize( int t, String txt ) {
        super.initialize( t, txt );
        _value = Integer.parseInt( txt );
    }

    public void initialize( Token t ) {
        super.initialize( t );
        _value = Integer.parseInt( t.getText() );
    }

    public void setCopy( ELMODataType d) throws BadELMOTypeException
    {
        switch(d.getTypeCode())
        {
            case ELMODataType.TYPE_CODE_INTEGER:
                setValue(((ELMOInteger)d).getValue());
            break;
            case ELMODataType.TYPE_CODE_FLOAT:
                setValue((int)((ELMOFloat)d).getValue());
            break;
            default:
                throw new BadELMOTypeException(getTypeCode(),d.getTypeCode(),"<file>",0,0);
        }
    }

    public void setValue( int v ) {
```

```
    _value = v;
}

public int getValue() {
    return( _value );
}


public void increment() {
    _value++;
}
public void decrement(){
    _value--;
}

public ELMODataType copy() {
    return new ELMOInteger(_value);
}

public boolean equals( Object o ) {
    if( o instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )o;
        return( i.getValue() == getValue() );
    }

    return( false );
}

public ELMODataType plus( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        return( new ELMOInteger( getValue() + ( ( ELMOInteger )a ).getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        return( new ELMOFloat( getValue() + ( ( ELMOFloat )a ).getValue() ) );
    }

    return( super.plus( a ) );
}

public ELMODataType minus() throws BadELMOTypeException {
    return( new ELMOInteger( -getValue() ) );
}

public ELMODataType minus( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        return( new ELMOInteger( getValue() - ( ( ELMOInteger )a ).getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        return( new ELMOFloat( ( double )getValue() -
            ( ( ELMOFloat )a ).getValue() ) );
    }

    return( super.minus( a ) );
}

public ELMODataType star( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        return( new ELMOInteger( getValue() * ( ( ELMOInteger )a ).getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        return( new ELMOFloat( ( double )getValue() *
            ( ( ELMOFloat )a ).getValue() ) );
    }

    return( super.star( a ) );
}

public ELMODataType slash( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        return( new ELMOInteger( getValue() / ( ( ELMOInteger )a ).getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        return( new ELMOFloat( ( double )getValue() /
            ( ( ELMOFloat )a ).getValue() ) );
    }

    return( super.slash( a ) );
}

public ELMODataType carat( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        return( new ELMOFloat( Math.pow( getValue(),
            ( ( ELMOInteger )a ).getValue() ) )
        );
    }
    else if( a instanceof ELMOFloat ) {
        return( new ELMOFloat( Math.pow( ( double )getValue(),
            ( ( ELMOFloat )a ).getValue() ) )
        );
    }

    return( super.carat( a ) );
}

public ELMOInteger lt( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )a;
        return( new ELMOInteger( getValue() < i.getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        ELMOFloat f = ( ELMOFloat )a;
        return( new ELMOInteger( getValue() < f.getValue() ) );
    }

    return( super.lt( a ) );
}

public ELMOInteger leq( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )a;
        return( new ELMOInteger( getValue() <= i.getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        ELMOFloat f = ( ELMOFloat )a;
        return( new ELMOInteger( getValue() <= f.getValue() ) );
    }

    return( super.leq( a ) );
}

public ELMOInteger gt( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )a;
        return( new ELMOInteger( getValue() > i.getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        ELMOFloat f = ( ELMOFloat )a;
        return( new ELMOInteger( getValue() > f.getValue() ) );
    }

    return( super.gt( a ) );
}

public ELMOInteger geq( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )a;
        return( new ELMOInteger( getValue() >= i.getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        ELMOFloat f = ( ELMOFloat )a;
        return( new ELMOInteger( getValue() >= f.getValue() ) );
    }

    return( super.geq( a ) );
}

public ELMOInteger eq( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )a;
        return( new ELMOInteger( getValue() == i.getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        ELMOFloat f = ( ELMOFloat )a;
        return( new ELMOInteger( getValue() == f.getValue() ) );
    }

    return( super.eq( a ) );
}

public ELMOInteger neq( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )a;
```

```
            return( new ELMOInteger( getValue() != i.getValue() ) );
        }
        else if( a instanceof ELMOFloat ) {
            ELMOFloat f = ( ELMOFloat )a;
            return( new ELMOInteger( getValue() != f.getValue() ) );
        }

        return( super.neq( a ) );
    }

    public ELMOInteger ampamp( ELMODataType a ) throws BadELMOTypeException {
        if( a instanceof ELMOInteger ) {
            ELMOInteger i = ( ELMOInteger )a;
            return( new ELMOInteger( i.getValue() > 0 && getValue() > 0 ) );
        }

        return( super.ampamp( a ) );
    }

    public ELMOInteger pipepipe( ELMODataType a ) throws BadELMOTypeException {
        if( a instanceof ELMOInteger ) {
            ELMOInteger i = ( ELMOInteger )a;
            return( new ELMOInteger( i.getValue() > 0 || getValue() > 0 ) );
        }

        return( super.pipepipe( a ) );
    }

    public ELMOInteger bang() throws BadELMOTypeException {
        return( new ELMOInteger( getValue() > 0 ? 0 : 1 ) );
    }

    public String toString() {
        if( name() == null ) {
            return( "int[" + _value + "]" );
        }

        return( name() + "[" + _value + "]" );
    }
}

import antlr.SemanticException;
import antlr.Token;

public class ELMOFloat extends ELMODataType {
    public static final int TYPE_CODE = ELMODataType.TYPE_CODE_FLOAT;

    protected double _value;

    public ELMOFloat() {
        super();
        setValue(0);
    }

    public ELMOFloat(double v, String name)
    {
        super(name);
        setValue(v);
    }

    public ELMOFloat( double v ) {
        this(v,ELMODataType.generateName(ELMODataType.TYPE_CODE_FLOAT));
    }
    public ELMOFloat(ELMOFloat v)
    {
        this(v.getValue(),ELMODataType.generateName(ELMODataType.TYPE_CODE_FLOAT));
    }

    public ELMOFloat(ELMODataType in) throws BadELMOTypeException
    {
        setCopy(in);
    }

    public void initialize( int t, String txt ) {
        super.initialize( t, txt );
        _value = Double.parseDouble( txt );
    }

    public void initialize( Token t ) {
        super.initialize( t );
        _value = Double.parseDouble( t.getText() );
    }
```

```
public void setCopy( ELMODataType d) throws BadELMOTypeException
{
    switch(d.getTypeCode())
    {
        case ELMODataType.TYPE_CODE_FLOAT:
            setValue(((ELMOFloat)d).getValue());
            break;
        case ELMODataType.TYPE_CODE_INTEGER:
            setValue((double)((ELMOInteger)d).getValue());
            break;
        default:
            throw new BadELMOTypeException(getTypeCode(),d.getTypeCode(),"<file>",0,0);
    }
}

public ELMOFloat( Token t ) {
    super();
    _value = Double.parseDouble( t.getText() );
}

public void setValue( double v ) {
    _value = v;
}

public double getValue() {
    return( _value );
}

public ELMODataType copy()  {
    return new ELMOFloat(_value);
}

public String toString() {
    if( name() == null ) {
        return( "float[" + _value + "]" );
    }

    return( name() + "[" + _value + "]" );
}

public boolean equals( Object o ) {
    if( o instanceof ELMOFloat ) {
        ELMOFloat i = ( ELMOFloat )o;
        return( i.getValue() == getValue() );
    }

    return( false );
}

public ELMODataType plus( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        return( new ELMOFloat( getValue() + ( ( ELMOInteger )a ).getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        return( new ELMOFloat( getValue() + ( ( ELMOFloat )a ).getValue() ) );
    }

    throw( new BadELMOTypeException(getType(),a.getType(),"<file>",0,0) );
}

public ELMODataType minus() throws BadELMOTypeException {
    return( new ELMOFloat( -getValue() ) );
}

public ELMODataType minus( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        return( new ELMOFloat( getValue() -
            ( double )( ( ELMOInteger )a ).getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        return( new ELMOFloat( getValue() - ( ( ELMOFloat )a ).getValue() ) );
    }

    throw( new BadELMOTypeException(getType(),a.getType(),"<file>",0,0) );
}

public ELMODataType star( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        return( new ELMOFloat( getValue() *
            ( double )( ( ELMOInteger )a ).getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        return( new ELMOFloat( getValue() * ( ( ELMOFloat )a ).getValue() ) );
    }

    throw( new BadELMOTypeException(getType(),a.getType(),"<file>",0,0) );
}

public ELMODataType slash( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        return( new ELMOFloat( getValue() /
            ( double )( ( ELMOInteger )a ).getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        return( new ELMOFloat( getValue() / ( ( ELMOFloat )a ).getValue() ) );
    }

    throw( new BadELMOTypeException(getType(),a.getType(),"<file>",0,0) );
}

public ELMODataType carat( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        return( new ELMOFloat( Math.pow( getValue(),
            ( double )( ( ELMOInteger )a ).getValue() ) )
        );
    }
    else if( a instanceof ELMOFloat ) {
        return( new ELMOFloat( Math.pow( getValue(),
            ( ( ELMOFloat )a ).getValue() ) )
        );
    }

    throw( new BadELMOTypeException(getType(),a.getType(),"<file>",0,0) );
}

public ELMOInteger lt( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )a;
        return( new ELMOInteger( getValue() < i.getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        ELMOFloat f = ( ELMOFloat )a;
        return( new ELMOInteger( getValue() < f.getValue() ) );
    }

    return( super.lt( a ) );
}

public ELMOInteger leq( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )a;
        return( new ELMOInteger( getValue() <= i.getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        ELMOFloat f = ( ELMOFloat )a;
        return( new ELMOInteger( getValue() <= f.getValue() ) );
    }

    return( super.leq( a ) );
}

public ELMOInteger gt( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )a;
        return( new ELMOInteger( getValue() > i.getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
        ELMOFloat f = ( ELMOFloat )a;
        return( new ELMOInteger( getValue() > f.getValue() ) );
    }

    return( super.gt( a ) );
}

public ELMOInteger geq( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOInteger ) {
        ELMOInteger i = ( ELMOInteger )a;
        return( new ELMOInteger( getValue() >= i.getValue() ) );
    }
    else if( a instanceof ELMOFloat ) {
            ELMOFloat f = ( ELMOFloat )a;
            return( new ELMOInteger( getValue() >= f.getValue() ) );
        }

        return( super.geq( a ) );
    }

    public ELMOInteger eq( ELMODataType a ) throws BadELMOTypeException {
        if( a instanceof ELMOInteger ) {
            ELMOInteger i = ( ELMOInteger )a;
            return( new ELMOInteger( getValue() == i.getValue() ) );
        }
        else if( a instanceof ELMOFloat ) {
            ELMOFloat f = ( ELMOFloat )a;
            return( new ELMOInteger( getValue() == f.getValue() ) );
        }

        return( super.eq( a ) );
    }

    public ELMOInteger neq( ELMODataType a ) throws BadELMOTypeException {
        if( a instanceof ELMOInteger ) {
            ELMOInteger i = ( ELMOInteger )a;
            return( new ELMOInteger( getValue() != i.getValue() ) );
        }
        else if( a instanceof ELMOFloat ) {
            ELMOFloat f = ( ELMOFloat )a;
            return( new ELMOInteger( getValue() != f.getValue() ) );
        }

        return( super.neq( a ) );
    }
}

import antlr.SemanticException;
import antlr.Token;

public class ELMORandomFloat extends ELMOFloat {
    public ELMORandomFloat() {
        super();
    }

    public ELMORandomFloat(double v, String name)
    {
        super(v,name);
    }

    public ELMORandomFloat( double v ) {
        super( v );
    }

    public ELMORandomFloat(ELMORandomFloat v)
    {
        super( v );
    }

    public ELMORandomFloat( ELMODataType a )
        throws BadELMOTypeException {
        if( a instanceof ELMOInteger ) {
            _value = Math.random() * ( double )( ( ELMOInteger )a ).getValue();
        }
        else
        if( a instanceof ELMOFloat ) {
            _value = Math.random() * ( ( ELMOFloat )a ).getValue();
        }
        else {
            throw( new BadELMOTypeException( "expected float or int" ) );
        }
    }

    public ELMORandomFloat( ELMODataType a, ELMODataType b )
        throws BadELMOTypeException {
        double diff = 0;

        if( a instanceof ELMOInteger &&
            b instanceof ELMOInteger ) {
            double low = ( double )( ( ELMOInteger )a ).getValue();
            diff = ( double )( ( ELMOInteger )b ).getValue() - low;
            _value = Math.random() * diff + low;
        }
        else
        if( a instanceof ELMOInteger &&
            b instanceof ELMOFloat ) {
            double low = ( double )( ( ELMOInteger )a ).getValue();
            diff = ( ( ELMOFloat )b ).getValue() - low;
            _value = Math.random() * diff + low;
        }
        else
        if( a instanceof ELMOFloat &&
            b instanceof ELMOInteger ) {
            double low = ( ( ELMOFloat )a ).getValue();
            diff = ( double )( ( ELMOInteger )b ).getValue() - low;
            _value = Math.random() * diff + low;
        }
        else
        if( a instanceof ELMOFloat &&
            b instanceof ELMOFloat ) {
            double low = ( ( ELMOFloat )a ).getValue();
            diff = ( ( ELMOFloat )b ).getValue() - low;
            _value = Math.random() * diff + low;
        }
        else {
            throw( new BadELMOTypeException( "expected float or int" ) );
        }
    }

    public ELMORandomFloat( Token t ) {
        super( t );
    }

    public void initialize( int t, String txt ) {
        super.initialize( t, txt );
    }

    public void initialize( Token t ) {
        super.initialize( t );
    }
}

import antlr.SemanticException;

public class ELMOVector extends ELMODataType {
    /*
    public ELMOFloat x;
    public ELMOFloat y;
    public ELMOFloat z;
    */
    private ELMOIndirect x;
    private ELMOIndirect y;
    private ELMOIndirect z;

    public ELMOVector( double x, double y, double z, String name) {
        super(name);
        this.x = new ELMOIndirect(new ELMOFloat(x));
        this.y = new ELMOIndirect(new ELMOFloat(y));
        this.z = new ELMOIndirect(new ELMOFloat(z));
        updateComponentNames();
    }
    public ELMOVector() {
        this(0,0,0,ELMODataType.generateName(ELMODataType.TYPE_CODE_VECTOR));
    }
    public ELMOVector( double x, double y, double z ) {
        this(x,y,z,ELMODataType.generateName(ELMODataType.TYPE_CODE_VECTOR));
    }
    public ELMOVector( ELMOFloat a1, ELMOFloat a2, ELMOFloat a3 ) {
        this(a1.getValue(),a2.getValue(),a3.getValue(),ELMODataType.generateName(ELMODataType.TYPE_CODE_VECTOR));
    }
    public ELMOVector( ELMODataType a1, ELMODataType a2, ELMODataType a3 ) throws BadELMOTypeException
    {
        this(0,0,0,ELMODataType.generateName(ELMODataType.TYPE_CODE_VECTOR));
        ((ELMOFloat)x.data).setCopy(a1);
        ((ELMOFloat)y.data).setCopy(a2);
        ((ELMOFloat)z.data).setCopy(a3);
    }
    public void setName(String name)
    {
        super.setName(name);
        updateComponentNames();
    }
    public void updateComponentNames()
    {
        if(x!=null && x.data !=null)
            x.data.setName(name()+".x");
        if(y!=null && y.data !=null)
```

```
            y.data.setName(name()+".y");
        if(z!=null && z.data !=null)
            z.data.setName(name()+".z");
    }


    public ELMOFloat getX()
    {
        return (ELMOFloat)x.data;
    }
    public ELMOFloat getY()
    {
        return (ELMOFloat)y.data;
    }
    public ELMOFloat getZ()
    {
        return (ELMOFloat)z.data;
    }
    public void setX(ELMOFloat f)
    {
        x.data = f;
    }
    public void setY(ELMOFloat f)
    {
        y.data = f;
    }
    public void setZ(ELMOFloat f)
    {
        z.data = f;
    }

//{{{ edit the ELMOIndirect pointer - use with caution */
    public ELMOIndirect getXind()
    {
        return x;
    }
    public ELMOIndirect getYind()
    {
        return y;
    }
    public ELMOIndirect getZind()
    {
        return z;
    }
    public void setXind(ELMOIndirect i)
    {
        x = i;
    }
    public void setYind(ELMOIndirect i)
    {
        y = i;
    }
    public void setZind(ELMOIndirect i)
    {
        z = i;
    }
//}}}

    public ELMODataType copy()  {
        return new ELMOVector(getX(),getY(),getZ());
    }

    public void setCopy(ELMODataType d) throws BadELMOTypeException
    {
        if(d.getTypeCode()!=ELMODataType.TYPE_CODE_VECTOR)
        {
            throw new BadELMOTypeException(getTypeCode(),d.getTypeCode(),"<file>",0,0);
        }
        else
        {
            set((ELMOVector)d);
        }
    }

    public void set(ELMOVector other)
    {
        getX().setValue(other.getX().getValue());
        getY().setValue(other.getY().getValue());
        getZ().setValue(other.getZ().getValue());
    }
    public ELMOVector negate() {
```

```
    return( new ELMOVector( -getX().getValue(), -getY().getValue(),-getZ().getValue() ) );
}
public ELMOVector normalize() {
    return( new ELMOVector( getX().getValue()/this.length().getValue(),
                                    getY().getValue()/this.length().getValue(),
                                    getZ().getValue()/this.length().getValue() ) );
}
public ELMOVector add( ELMOVector v3d ) {
    return( new ELMOVector( getX().getValue() + v3d.getX().getValue(),
                            getY().getValue() + v3d.getY().getValue(),
                            getZ().getValue() + v3d.getZ().getValue() ) );
}

public ELMOVector sub( ELMOVector v3d ) {
    return( new ELMOVector( getX().getValue() - v3d.getX().getValue(),
                            getY().getValue() - v3d.getY().getValue(),
                            getZ().getValue() - v3d.getZ().getValue() ) );
}

public ELMOVector mult( ELMOInteger s ) {
    return( new ELMOVector( getX().getValue() * s.getValue(),
                            getY().getValue() * s.getValue(),
                            getZ().getValue() * s.getValue() ) );
}

public ELMOVector mult( ELMOFloat s ) {
    return( new ELMOVector( getX().getValue() * s.getValue(),
                            getY().getValue() * s.getValue(),
                            getZ().getValue() * s.getValue() ) );
}

public static ELMOVector mult( ELMOMatrix m, ELMOVector v ) {
    //TODO: fill this in
    //note: do we want to allow vector * matrix?  should only need matrix * vector

    return new ELMOVector(  new ELMOFloat(v.getX().getValue()*m.m[0][0]+v.getY().getValue()*m.m[0][1]+v.getZ().getValue()*m.m[0][2]+m.m[0][3])
                new ELMOFloat(v.getX().getValue()*m.m[1][0]+v.getY().getValue()*m.m[1][1]+v.getZ().getValue()*m.m[1][2]+m.m[1][3]),
                new ELMOFloat(v.getX().getValue()*m.m[2][0]+v.getY().getValue()*m.m[2][1]+v.getZ().getValue()*m.m[2][2]+m.m[2][3]));
}

/*
public ELMOVector div( ELMOInteger s ) {
    return( new ELMOVector( getX().getValue() / s.getValue(),
                            getY().getValue() / s.getValue(),
                            getZ().getValue() / s.getValue() ) );
}

public ELMOVector div( ELMOFloat s ) {
    return( new ELMOVector( getX().getValue() / s.getValue(),
                            getY().getValue() / s.getValue(),
                            getZ().getValue() / s.getValue() ) );
}
*/

public ELMOFloat length() {
    return( new ELMOFloat( Math.sqrt( Math.pow( getX().getValue(), 2 ) +
                                    Math.pow( getY().getValue(), 2 ) +
                                    Math.pow( getZ().getValue(), 2 ) ) ) );
}

public String toString() {
    StringBuffer buffer = new StringBuffer( name() + "<" );
    buffer.append( getX().getValue() ).append( ", " );
    buffer.append( getY().getValue() ).append( ", " );
    buffer.append( getZ().getValue() ).append( ">" );
    return( buffer.toString() );
}

public boolean equals( Object o ) {
    if( o instanceof ELMOVector ) {
        ELMOVector i = ( ELMOVector )o;
        return( i.getX().getValue() == getX().getValue()
                && i.getY().getValue() == getY().getValue()
                && i.getZ().getValue() == getZ().getValue() );
    }

    return( false );
}

public ELMODataType plus( ELMODataType a ) throws BadELMOTypeException {
    if( a instanceof ELMOVector ) {
            ELMOVector vec = ( ELMOVector )a;
            return( new ELMOVector( getX().plus(vec.getX()), getY().plus(vec.getY()), getZ().plus(vec.getZ()) ) );
        }

        throw( new BadELMOTypeException(getType(),a.getType(),"<file>",0,0) );
    }

    public ELMODataType minus() throws BadELMOTypeException {
        return( new ELMOVector( getX().minus(), getY().minus(), getZ().minus() ) );
    }

    public ELMODataType minus( ELMODataType a ) throws BadELMOTypeException {
        if( a instanceof ELMOVector ) {
            ELMOVector vec = ( ELMOVector )a;
            return( new ELMOVector( getX().minus(vec.getX()),getY().minus(vec.getY()), getZ().minus(vec.getZ()) ) );
        }

        throw( new BadELMOTypeException(getType(),a.getType(),"<file>",0,0) );
    }

    public ELMODataType star( ELMODataType a ) throws BadELMOTypeException {
        if( a instanceof ELMOInteger ) {
            double s = ( double )( ( ELMOInteger )a ).getValue();
            return( new ELMOVector( s * getX().getValue(), s * getY().getValue(), s * getZ().getValue() ) );
        }
        else if( a instanceof ELMOFloat ) {
            double s = ( ( ELMOFloat )a ).getValue();
            return( new ELMOVector( s * getX().getValue(), s * getY().getValue(), s * getZ().getValue() ) );
        }

        throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",0,0 ) );
    }

    public ELMODataType slash( ELMODataType a ) throws BadELMOTypeException {
        if( a instanceof ELMOInteger ) {
            double s = ( double )( ( ELMOInteger )a ).getValue();
            return( new ELMOVector( getX().getValue() / s, getY().getValue() / s, getZ().getValue() / s ) );
        }
        else if( a instanceof ELMOFloat ) {
            double s = ( ( ELMOFloat )a ).getValue();
            return( new ELMOVector( getX().getValue() / s, getY().getValue() / s, getZ().getValue() / s ) );
        }

        throw( new BadELMOTypeException( getTypeCode(),a.getTypeCode(),"<file>",0,0 ) );
    }

    public ELMODataType carat( ELMODataType a ) throws BadELMOTypeException {
        return( new ELMOVector( getX().carat(a), getY().carat(a), getZ().carat(a) ) );
    }

    public ELMOInteger eq( ELMODataType a ) throws BadELMOTypeException {
        if( a instanceof ELMOInteger ) {
            ELMOVector v = ( ELMOVector )a;
            return( new ELMOInteger( x == v.x && y == v.y && z == v.z ) );
        }

        return( super.eq( a ) );
    }

    public ELMOInteger neq( ELMODataType a ) throws BadELMOTypeException {
        if( a instanceof ELMOInteger ) {
            ELMOVector v = ( ELMOVector )a;
            return( new ELMOInteger( x != v.x || y != v.y || z != v.z ) );
        }

        return( super.neq( a ) );
    }
}

import antlr.SemanticException;

public final class ELMOVoid extends ELMODataType {

    public ELMODataType copy() {
        return new ELMOVoid();
    }
    public void setCopy(ELMODataType d) throws BadELMOTypeException
    {
        if(d.getType()!=ELMODataType.TYPE_CODE_VOID)
            throw new BadELMOTypeException(getType(),d.getType(),"<file>",0,0);

        //do nothing - it's void
    }

    public ELMOVoid() {
        super();
    }
}
```

C.3.1 Function-specific Classes (Stephen)

```
public class ELMOArgument {
    private ELMOIndirect _value;
    private String _name;
    private int _type;

    public ELMOArgument( int type, String name, ELMOIndirect value ) {
        _type = type;
        _value = value;
        _name = name;
    }

    public ELMOArgument( int type, String name ) {
        this(type,name,null);
    }

//  public ELMODataType getValue() {
//      return( (ELMODataType)_value.data );
//  }
    public ELMOIndirect getValue() {
        return _value;
    }

    public String getName() {
        return( _name );
    }

    public int hashCode() {
        return( _name.hashCode() );
    }

    public boolean equals( Object o ) {
        if( o instanceof ELMODataType ) {
            ELMODataType dt = ( ELMODataType )o;

            if( _value == null ) {
                return( dt.getTypeCode() == _type );
            }

            return( dt.getTypeCode() == _type && _value.equals( o ) );
        }

        return( false );
    }
}

import antlr.collections.AST;
import antlr.SemanticException;
import antlr.RecognitionException;
import java.util.HashMap;
import java.util.Iterator;

public class ELMOFunction extends ELMOSimple {
    //private ELMOSymbolTable _symbolTable;
    private ELMOSymbolTable _parentSymbolTable;
    private int _retType;
    private HashMap _args;
    private AST _body;
    private ELMOWalker _walker;

    public ELMOFunction( int retType, HashMap a, ELMOWalker walker, AST b,
        ELMOSymbolTable s ) {
        super();
        _retType = retType;
        _args = a;
        _body = b;
        //_symbolTable = s;
        _parentSymbolTable = s;
        _walker = walker;
    }

    public ELMOIndirect invoke( HashMap input ) throws RecognitionException {
        ELMOSymbolTable childScope = _parentSymbolTable.addChildScope();
        Iterator argsIter = _args.keySet().iterator();

        while( argsIter.hasNext() ) {
            String argName = ( String )argsIter.next();
            ELMOArgument arg = ( ELMOArgument )_args.get( argName );

            if( input.containsKey( argName ) ) {
                ELMOArgument inpParam = ( ELMOArgument )input.get( argName );
                childScope.put( argName, inpParam.getValue() );
            }
            else {
                if( arg.getValue() != null ) {
            /* will this ever happen? */
                    childScope.put( argName, arg.getValue() );
                }
                else {
                    // if an argument w/o a default value is undefined,
                    throw( new SemanticException( "parameter " + argName + " is undefined" ) );
                }
            }
        }

        // check for parameters defined by the invokation but not in the function definition
        Iterator inpIter = input.keySet().iterator();

        while( inpIter.hasNext() ) {
            String argName = ( String )inpIter.next();

            if( !_args.containsKey( argName ) ) {
                throw( new SemanticException( "parameter " + argName + " is not in the function definition" ) );
            }
        }
        ELMOIndirect retVal = _walker.cmpdstmt( _body, childScope );
        _parentSymbolTable.removeScope( childScope );
        /*
        // remove function input parameters from symbol table
        argsIter = _args.keySet().iterator();

        while( argsIter.hasNext() ) {
            String argName = ( String )argsIter.next();
            _symbolTable.remove( argName );
        }
        */

        if(_retType==ELMODataType.TYPE_CODE_VOID)
        {
            return null;
        }

        if( (retVal.data instanceof ELMODataType)
            && ((ELMODataType)retVal.data).getTypeCode() == _retType ) {
            return( retVal );
        }

        // throw exception since return type doesn't match
        throw( new SemanticException( "return type "+ELMODataType.typeString(_retType)+" does not match that of the function declaration" ) );
    }
}
```

C.4 Group and Objects Classes (Erik)

```
/*
 * ELMOFace.java
 *
 * Created on December 6, 2004, 3:57 PM
 */

/**
 *
 * @author  Erik
 */
public class ELMOFace extends ELMODataType {
    //the list of vertex data; each triplet is (vert_index,tex_index,norm_index)
    public ELMOVector[] p;
    private boolean normals = false, texture = false;

    /** Creates a new instance of ELMOFace */
    public ELMOFace(){}
    public ELMOFace( ELMOVector[] faces ) {
        p = faces;
    }

    public void normals( boolean n ) {
        normals = n;
    }
    public void texture( boolean n ) {
        texture = n;
    }
    public boolean normals() {
    return normals;
    }
    public boolean texture() {
    return texture;
    }

    public ELMODataType copy()
    {
        ELMOFace ret = new ELMOFace();
        ret.p = ELMODataType.arrayCopy(p);
        ret.texture = texture;
        ret.normals = normals;
        return ret;
    }
    public void setCopy(ELMODataType input) throws BadELMOTypeException
    {
        ELMOFace t = (ELMOFace)input.copy();
        p = t.p;
        texture = t.texture;
        normals = t.normals;
    }

}

/*
 * ELMOGroup.java
 *
 * Created on November 2, 2004, 3:43 PM
 */
import java.util.HashSet;
import java.util.Iterator;
import antlr.collections.AST;
import antlr.SemanticException;
import antlr.Token;
/**
 *
 * @author  Erik
 */
public class ELMOGroup extends ELMODataType {
    private ELMOGroup _parent;
    private HashSet groups;
    private HashSet objects;
    private ELMOMatrix m = ELMOMatrix.ID();

    public ELMOGroup() {
        super();
        groups = new HashSet();
        objects = new HashSet();
        _parent = null;
    }

    public ELMOGroup( Token t ) {
        super( t );
    }

    public ELMOGroup( ELMOGroup parent ) {
        super();
        groups = new HashSet();
        objects = new HashSet();
        _parent = parent;
    }

    public ELMOMatrix getTransformationMatrix() {
                ELMOMatrix n = new ELMOMatrix(m.m);
        return( n );
    }

    public ELMOMatrix getInheritedTM() {
        ELMOMatrix t = m;
        ELMOGroup g = _parent;

        while( g != null ) {
            t = ELMOMatrix.mult( t, g.getTransformationMatrix() );
            g = g.getParent();
        }
        return t;
    }

    public Iterator iterator() {
        HashSet tmp = new HashSet( groups );
        tmp.addAll( objects );
        return( tmp.iterator() );
    }

    public void setParent( ELMOGroup e ) {
        _parent = e;
    }

    public ELMOGroup getParent() {
        return( _parent );
    }

    public void translate( ELMOVector v ) throws SemanticException {
    ELMOMatrix t = ELMOMatrix.ID();
    t.m[0][3] = v.getX().getValue();
    t.m[1][3] = v.getY().getValue();
        t.m[2][3] = v.getZ().getValue();
    multiply(t);

    }

    public void transform( ELMOMatrix n ) {
        m = ELMOMatrix.mult(n,m);
    }

    public void rotate( ELMOVector v, ELMOFloat f ) {
        ELMOMatrix t = new ELMOMatrix();
        t.rot( f.getValue(), v );
        multiply( t );
    }

    public void scale( ELMOVector center, ELMOVector s )
        throws SemanticException {
        translate( center.negate() );
        ELMOMatrix t = ELMOMatrix.ID();
        t.scale( s.getX().getValue(),s.getY().getValue(),s.getZ().getValue() );
        multiply( t );
        translate( center );
    }

    public void multiply( ELMOMatrix n ) {
    m = ELMOMatrix.mult(n,m);

    }

    public void setCopy(ELMODataType d) throws BadELMOTypeException
    {
        if(d.getTypeCode()!=ELMODataType.TYPE_CODE_GROUP)
        {
            throw new BadELMOTypeException(getTypeCode(),d.getTypeCode(),"<file>",0,0);
        }
        else
        {
            ELMOGroup g = (ELMOGroup)d.copy();
                        _parent = g.getParent();
                        multiply(g.getTransformationMatrix());
        }
    }

    public ELMOVector X() {
            ELMOVector x = new ELMOVector(1,0,0);
            ELMOMatrix t = this.getInheritedTM();
            x = ELMOVector.mult(t, x);
            return x;
    }

    public ELMOVector Y() {
            ELMOVector y = new ELMOVector(0,1,0);
            ELMOMatrix t = this.getInheritedTM();
            y = ELMOVector.mult(t, y);
            return y;
    }

    public ELMOVector Z() {
            ELMOVector z = new ELMOVector(0,0,1);
            ELMOMatrix t = this.getInheritedTM();
            z = ELMOVector.mult(t, z);
            return z;
    }

    public ELMODataType copy() {
            ELMOGroup ret = new ELMOGroup();
            ret._parent = _parent;
            ret.multiply(m);
            return ret;
    }

    public void remove( ELMOGroup a ) {
        if( a instanceof ELMOObject ) {
            objects.remove( a );
        }
        else if( a instanceof ELMOGroup ) {
            groups.remove( a );
        }

        a.setParent( null );
    }

    public void removeWithInheritance( ELMOGroup a ) {
        if( a instanceof ELMOObject ) {
            objects.remove( a );
        }
        else if( a instanceof ELMOGroup ) {
            groups.remove( a );
        }

        ELMOMatrix t = this.getInheritedTM();
        a.setParent( null );
        a.multiply( t );
    }

    public void stamp() {
        stamp( ELMOMatrix.ID() );
    }

    public void stamp( ELMOMatrix t ) {
        Iterator childrenIter = objects.iterator();
        t = ELMOMatrix.mult( t, m );

                if( this instanceof ELMOObject ) {
                    ELMOObjectWriter.write( (ELMOObject)this, t );
                }
        while( childrenIter.hasNext() ) {
            ELMOObject o = ( ELMOObject )childrenIter.next();
            o.stamp(t);
        }

        childrenIter = groups.iterator() ;
        while( childrenIter.hasNext() ) {
            ELMOGroup g  = ( ELMOGroup )childrenIter.next();
            g.stamp( t );
        }
    }

    public void attach( ELMOGroup a ) {
        // only add if a has no parent and won't form a cycle
        // alternative is to remove the group from its existing parent and then attach it here
        if (a._parent==null && !isAncestor(a)) {
            if( a instanceof ELMOObject ) {
                objects.add( a );
            }
            else if( a instanceof ELMOGroup ) {
                groups.add( a );
            }

            a.setParent( this );
            //throw( new SemanticException( "Invalid type" ) );
        }
    }

    public boolean isAncestor( ELMOGroup a ) {  // ELMOGroup a is about to be added
        if (this == a) {                        // is the new group the same as this group?
            return true;
        } else if (_parent == null) {           // is this group the root?
            return false;
        } else {                                    // is the new group an ancestor of our parent group
            return _parent.isAncestor(a);
        }
    }

}

/*
 * ELMOMatrix.java
 *
 * Created on November 2, 2004, 2:10 PM
 */

/**
 *
 * @author  Erik
 */
public class ELMOMatrix {
    public double m[][];

    public ELMOMatrix() {
        m = new double[4][4];
    }
    public ELMOMatrix( double n[][] ) {
        m = n;
    }
    public void mult( double s ) {
        for ( int i = 0; i < 4; i ++ ) {
            for ( int j = 0; j < 4; j ++ ) {
                m[i][j] *= s;
            }
        }
    }
    public static ELMOMatrix mult( ELMOMatrix x, ELMOMatrix y ) {
        ELMOMatrix t = new ELMOMatrix();
        for ( int i = 0; i < 4; i ++ ) {
            for ( int j = 0; j < 4; j ++ ) {
                for ( int k = 0; k < 4; k ++ ) {
                    t.m[i][j] = t.m[i][j]+x.m[i][k]*y.m[k][j];
                }
            }
        }
       return t;
    }
    public static ELMOMatrix add( ELMOMatrix x, ELMOMatrix y ) {
       ELMOMatrix t = new ELMOMatrix();
       for ( int i = 0; i < 4; i ++ ) {
           for ( int j = 0; j < 4; j ++ ) {
               t.m[i][j] = x.m[i][j]+y.m[i][j];
           }
       }
       return t;
    }

    public static ELMOMatrix ID () {
       double p[][] = new double[4][4];
       for ( int i = 0; i < 4; i ++ ) {
           for ( int j = 0; j < 4; j ++ ) {
               if ( i == j ) p[i][j] = 1;
               else p[i][j] = 0;
           }
       }
       return new ELMOMatrix(p);
    }

    public void scale( ELMOVector v ) {
        scale( v.getX().getValue(), v.getY().getValue(), v.getZ().getValue() );
    }

    public void scale( double s ) {
        scale( s, s, s );
    }

    public void scale( double x, double y, double z ) {
        for ( int i = 0; i < 4; i ++ ) {
            for ( int j = 0; j < 4; j ++ ) {
                if( i == j ) {
                    switch( i ) {
                        case 0:
                            m[i][j] *= x;
                            break;
                        case 1:
                            m[i][j] *= y;
                            break;
                        case 2:
                            m[i][j] *= z;
                            break;
                    }
                }
            }
        }
    }

    public void rot( double radians, ELMOVector _axis ) {
        ELMOVector axis = _axis.normalize();
        double cos = (double)Math.cos(radians);
        double sin = (double)Math.sin(radians);

        ELMOMatrix n1 = ELMOMatrix.ID();
        n1.mult( cos );

        ELMOMatrix n2 = ELMOMatrix.ID();
        double f1 = 0, f2 = 0;
        for ( int i = 0; i < 4; i ++ ) {
           for ( int j = 0; j < 4; j ++ ) {
               f1 = 0;
               f2 = 0;
               switch (i) {
                   case 0: f1 = axis.getX().getValue(); break;
                   case 1: f1 = axis.getY().getValue(); break;
                   case 2: f1 = axis.getZ().getValue(); break;
               }
               switch (j) {
                   case 0: f2 = axis.getX().getValue(); break;
                   case 1: f2 = axis.getY().getValue(); break;
                   case 2: f2 = axis.getZ().getValue(); break;
               }
               n2.m[i][j] = f1*f2;
           }
        }
        n2.m[3][3] = 1;
        n2.mult(1-cos);

        ELMOMatrix n3 = ELMOMatrix.ID();
        n3.m[0][1] = -1*axis.getZ().getValue();
        n3.m[1][0] = axis.getZ().getValue();
        n3.m[0][2] = axis.getY().getValue();
        n3.m[2][0] = -1*axis.getY().getValue();
        n3.m[2][1] = axis.getX().getValue();
        n3.m[1][2] = -1*axis.getX().getValue();

        n3.m[0][0] = 0;
        n3.m[1][1] = 0;
        n3.m[2][2] = 0;
        n3.mult(sin);

        m = (ELMOMatrix.add(n1,ELMOMatrix.add(n2,n3))).m;
        m[3][3] = 1;
    }

    public void print () {
        for ( int i = 0; i < 4; i ++ ) {
           for ( int j = 0; j < 4; j ++ ) {
               System.out.print( m[j][i]+"\t" );
           }
           System.out.println();
        }
    }

}

/*
 * Obj.java
 *
 * Created on November 2, 2004, 3:03 PM
 */
import antlr.SemanticException;
import antlr.collections.AST;
import antlr.Token;
import java.io.*;
/**
 *
 * @author  Erik
 */
```

```
public class ELMOObject extends ELMOGroup {
    public static final int TYPE_CODE = ELMODataType.TYPE_CODE_OBJECT;

    public ELMOVector[] v;  //vertices
    public ELMOVector[] n;  //normals
    public ELMOFace[] f;    //faces
    public ELMOVector[] t;  //texture
    public String filename;
    public boolean filefound=true;

    public ELMOObject() {
        super();
    }

    public ELMOObject( String filename ) throws FileNotFoundException{
        super();
        this.filename = filename;
        OBJFile.importOBJ(filename,this);
    }

    public ELMOObject( Token t ) {
        super( t );
    }

    public void initialize( int t, String txt ){
        super.initialize( t, txt );
        filename = txt;

        try{
            OBJFile.importOBJ(filename,this);
        }
        catch( FileNotFoundException e)
        {
            filefound=false;
        }
    }

    public void initialize( AST a ) {
        super.initialize( a );
        filename = a.getText();
    }

    public void initialize( Token t ) {
        super.initialize( t );
        filename = t.getText();
    }

    public void setCopy(ELMODataType d) throws BadELMOTypeException
    {
        if(d.getTypeCode()!=ELMODataType.TYPE_CODE_OBJECT)
        {
            throw new BadELMOTypeException(getTypeCode(),d.getTypeCode(),"<file>",0,0);
        }
        else
        {
            ELMOObject obj = (ELMOObject)d.copy();
            //just steal the copy's data
            v = obj.v;  //vertices
            n = obj.n;  //normals
            f = obj.f;    //faces
            t = obj.t;  //texture
                        multiply(obj.getTransformationMatrix());
            filename = obj.filename; //filename
        }
    }

    public ELMODataType copy()
    {
        try{
            ELMOObject ret = new ELMOObject();

            ret.v = ELMODataType.arrayCopy(v);
            ret.f = ELMODataType.arrayCopy(f);
            ret.n = ELMODataType.arrayCopy(n);
            ret.t = ELMODataType.arrayCopy(t);

                        ret.multiply(getTransformationMatrix());
            return ret;
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        //finally   //this caused null to always return :(
        {
            return null;
        }
    }

    public String toString()
    {
        return filename;
    }

}

import java.util.HashSet;
import java.util.ArrayList;

public class ELMOObjectWriter {

    private static ArrayList verts = new ArrayList(128);
        private static ArrayList norms = new ArrayList(128);
        private static ArrayList text = new ArrayList(128);
        private static ArrayList faces = new ArrayList(512);

    public static void write( ELMOObject o ) {
        write(o,ELMOMatrix.ID());
    }

    public static void write( ELMOObject o, ELMOMatrix m ) {
            int vlength = verts.size();
            int nlength = norms.size();
            int tlength = text.size();
            int flength = faces.size();

            for ( int i = 0; i < o.v.length; i ++ ) {
        verts.add(ELMOVector.mult(m,o.v[i]));
            }

        for ( int i = 0; i < o.n.length; i ++ ) {
                norms.add(ELMOVector.mult(m,o.n[i]));
            }

            for ( int i = 0; i < o.t.length; i ++ ) {
                text.add(o.t[i]);
            }


        ELMOFace newface;
            for ( int i = 0; i < o.f.length; i ++ ) {
        newface= new ELMOFace();
        newface.p = new ELMOVector[o.f[i].p.length];
                for ( int j = 0; j < o.f[i].p.length; j ++ ) {
            newface.p[j] = new ELMOVector();
            newface.p[j].setX( new ELMOFloat(vlength + o.f[i].p[j].getX().getValue()));
            if(o.f[i].texture())
            newface.p[j].setY(new ELMOFloat(nlength + o.f[i].p[j].getY().getValue()));
            if(o.f[i].normals())
            newface.p[j].setZ(new ELMOFloat(tlength + o.f[i].p[j].getZ().getValue()));
            newface.normals(o.f[i].normals());
            newface.texture(o.f[i].texture());
            faces.add(newface);
                }
            }
    }
        public static void flush( String filename ) {
            OBJFile.exportOBJ( filename, (ELMOVector[])verts.toArray(new ELMOVector[0]), (ELMOVector[])norms.toArray(new ELMOVector[0]), (ELMOVect
        }
}

/*
 * ELMOOBJ.java
 *
 * Created on November 21, 2004, 5:38 PM
 */

import java.util.StringTokenizer;
import java.util.Vector;
import java.io.*;

/**
```

```
 * @author  Erik
 */
public class OBJFile {

    public static void importOBJ ( String filename, ELMOObject o ) throws FileNotFoundException {

        Vector v = new Vector();
        Vector f = new Vector();
        Vector vn = new Vector();
        Vector vt = new Vector();

        File thefile = new File(filename);
        if(!thefile.exists())
            throw new FileNotFoundException("could not find file "+filename);

        StringFile file = new StringFile(filename,"USAGE");
        file.reader();
        String temp = file.readLine();
        String next = "";
        while ( temp != null && next != null ) {
            next = "";
//            System.out.println( "temp: "+temp );
            StringTokenizer token = new StringTokenizer(temp);
            if ( token.hasMoreTokens()) next = token.nextToken();
//            System.out.print( "next: "+next );
            if ( !next.startsWith("#")) {
                if ( next.startsWith("v")) {
                    if ( next.startsWith("vn")) {
                        double x = Double.parseDouble(token.nextToken());
                        double y = Double.parseDouble(token.nextToken());
                        double z = Double.parseDouble(token.nextToken());
                        ELMOVector tmp = new ELMOVector(x,y,z);
                        vn.add(tmp);
                    } else if ( next.startsWith("vt")) {
                        double x = Double.parseDouble(token.nextToken());
                        double y = Double.parseDouble(token.nextToken());
                        ELMOVector tmp = new ELMOVector(x,y,0);
                        vt.add(tmp);
                    } else if ( next.equals("v")) {
                        double x = Double.parseDouble(token.nextToken());
//                        System.out.print(" x: "+x);
                        double y = Double.parseDouble(token.nextToken());
//                        System.out.print(" y: "+y);
                        double z = Double.parseDouble(token.nextToken());
//                        System.out.print(" z: "+z+"\n");
                        ELMOVector tmp = new ELMOVector(x,y,z);
                        v.add(tmp);
                    }
                } else if ( next.startsWith("f")) {
                    ELMOFace ftmp = new ELMOFace();
                    Vector compile = new Vector();

                    while ( token.hasMoreTokens() ) {
                        next = token.nextToken();
//                        System.out.println("faces: "+next);

                        StringTokenizer faces = new StringTokenizer(next,"/");
                        String s = faces.nextToken();
                        int x = -1, y = -1, z = -1;
                        x = Integer.parseInt(s);
                        if ( faces.hasMoreTokens()) {
                            s = faces.nextToken();
                            y = Integer.parseInt(s);
                            ftmp.texture(true);
                        }
                        if ( faces.hasMoreTokens()) {
                            s = faces.nextToken();
                            z = Integer.parseInt(s);
                            ftmp.normals(true);
                        }
                        ELMOVector tmp = new ELMOVector(x,y,z);
                        compile.add(tmp);
                    }
                    ftmp.p = new ELMOVector[compile.size()];
                    for ( int i = 0; i < compile.size(); i ++ ) {
                        ftmp.p[i] = (ELMOVector)compile.get(i);
                    }
                    compile = null;
                    f.add(ftmp);
                }
            }
```

```
                temp = file.readLine();
            }

            o.v = new ELMOVector[v.size()];
            for ( int i = 0; i < v.size(); i ++ ) {
                o.v[i] = (ELMOVector)v.get(i);
            }
            v = null;

            o.n = new ELMOVector[vn.size()];
            for ( int i = 0; i < vn.size(); i ++ ) {
                o.n[i] = (ELMOVector)vn.get(i);
            }
            vn = null;

            o.t = new ELMOVector[vt.size()];
            for ( int i = 0; i < vt.size(); i ++ ) {
                o.t[i] = (ELMOVector)vt.get(i);
            }
            vt = null;

            o.f = new ELMOFace[f.size()];
            for ( int i = 0; i < f.size(); i ++ ) {
                o.f[i] = (ELMOFace)f.get(i);
            }
            f = null;
        }
        public static void exportOBJ ( String filename, ELMOVector[] v, ELMOVector[] vn, ELMOVector[] vt, ELMOFace[] f ) {

    System.out.println("creating file");
            StringFile file = new StringFile(filename,"USAGE");
    System.out.println("writer");
            file.writer();

            file.writeString("# Created by ELMO\n");
            file.writeString("# Erik, Stephen, John, and not so much Jeff\n"); // <--- oh, thanks, guys!
            file.writeString("g default\n\n");

            for ( int i = 0; i < v.length; i ++ ) {
                file.writeString("v "+format(v[i].getX().getValue())+" "+format(v[i].getY().getValue())+" "+format(v[i].getZ().getValue())+"\n");
            }
            System.out.println("v out");
            for ( int i = 0; i < vt.length; i ++ ) {
                file.writeString("vt "+format(vt[i].getX().getValue())+" "+format(vt[i].getY().getValue())+"\n");
            }
            System.out.println("vt out");
            for ( int i = 0; i < vn.length; i ++ ) {
                file.writeString("vn "+format(vn[i].getX().getValue())+" "+format(vn[i].getY().getValue())+" "+format(vn[i].getZ().getValue())+"\n");
            }
            System.out.println("vn out");
            for ( int i = 0; i < f.length; i ++ ) {
                file.writeString("f ");
                for ( int j = 0; j < f[i].p.length; j ++ ) {
                    file.writeString(String.valueOf((int)f[i].p[j].getX().getValue()));
            //System.out.println("texture: "+f[i].texture());
            //System.out.println("normal: "+f[i].normals());
                    if ( f[i].texture())
                file.writeString("/"+(int)f[i].p[j].getY().getValue());
                    if ( f[i].normals())
                file.writeString("/"+(int)f[i].p[j].getZ().getValue());
            file.writeString(" ");
                }
                file.writeString("\n");
            }

    //System.out.println("writeString");
    //        file.writeString(out);
    System.out.println("writeSave");
            file.writeSave();
    System.out.println("write finished");
        }
        public static String format ( double d ) {
            return format( ""+d );
        }
        public static String format ( String s ) {
            boolean negative = false;
            if ( s.startsWith("-")) {
                negative = true;
                s = s.substring(1);
            }
            if ( s.indexOf('E') != -1 ) {
```

```
            double d = Double.parseDouble(s);
            s = "";
            for ( int i = 0; i < 7; i ++ ) {
                s += ""+(int)(d/Math.pow(10,-i));
                if ( i == 0 ) s += ".";
            }
        } else {
            int length = s.length();
            if ( length < 8 ) {
                length = 8 - length;
                for ( int i = 0; i < length; i ++ ) {
                    s += "0";
                }
            }
        }
        if ( negative ) s = "-"+s;
        return s;
    }
}

import java.io.*;

public class StringFile {
    private static final String DEFAULT_USAGE = "DEFINE ME";

    private String file, usage;
    private FileReader rd;
    private FileWriter wt;
    private BufferedReader read;
    private BufferedWriter write;

    public StringFile( String name ) {
        file = name;
        usage = DEFAULT_USAGE;
    }

    public StringFile ( String name, String err ) {
        file = name;
        usage = err;
    }
    public int reader() {
        try {
            rd = new FileReader(file);
            read = new BufferedReader(rd);
            return 1;
        } catch ( IOException x ){
            System.out.println(x+"\n"+usage);
            return -1;
        }
    }
    public String readLine() {
        try {
            return read.readLine();
        } catch ( IOException x ){
            return null;
        }
    }
    public int writer() {
        try {
            wt = new FileWriter(file);
            write = new BufferedWriter(wt);
            return 1;
        } catch ( IOException x ){
            System.out.println(x+"\n"+usage);
            return -1;
        }
    }
    public void writeLine(String s) {
        try {
            write.write(s + "\n");
        } catch ( IOException x ){
        }
    }
        public void writeString(String s) {
        try {
            write.write(s);
        } catch ( IOException x ){
        }
    }
        public void writeSave(){
                try {
            write.close();
        } catch ( IOException x ){
        }
        }
}
```

C.5 Front-Ends (John)

```
import java.io.*;
import antlr.CommonAST;
import antlr.collections.AST;
import antlr.debug.misc.ASTFrame;
import antlr.TokenStreamException;
import antlr.Token;
import antlr.RecognitionException;
import antlr.SemanticException;

public class Tester
{
    public static void main(String[] args)
    {
        ASTFrame frame=null;
        ELMOLexer lexer=null;
        ELMOParser parser=null;
        ELMOWalker walker=null;

        try{
            DataInputStream input = new DataInputStream(System.in);

            // Create the lexer and parser and feed them the input

            lexer = new ELMOLexer(input);
        /*
            //this code manually runs the lexer for debugging purposes
            System.out.println("running lexer");
            while(true)
            {
                try
                {
                    if(lexer.nextToken().getType()==Token.EOF_TYPE)
                        break;
                }
                catch(TokenStreamException tse)
                {
                    System.err.println("token error: "+tse);
                    //keep reading (or attempt to)
                    lexer.consume();
                }
            }

            System.out.println("done lexing");
        */

            parser = new ELMOParser(lexer);


            System.out.println("parsing file");
            parser.program(); // "file" is the main rule in the parser
            System.out.println("done parsing");
        }
        catch(SemanticException e)
        {
            System.out.print("Semantic Error: at ");
            System.out.println(e.toString());
            //e.printStackTrace();
//          frame.dispose();
        }
        catch(RecognitionException e)
        {
            System.out.print("Syntax Error: at ");
            System.out.println(e.toString());
//          frame.dispose();
        }
        catch(TokenStreamException e)
        {
            System.out.print("Lexer Error: at ");
            System.out.println(e.toString());
//          frame.dispose();
        }
        catch(Exception e)
        {
            System.out.println("General Error: at ");
            System.out.println(e.toString());
            e.printStackTrace();
//          frame.dispose();
        }


        try
        {
            if((parser.myFlags & parser.FLAG_ERROR)!=0)
            {
                System.out.println("The file did not parse correctly.");
                System.out.println("Please fix it and try again.");
            }
            else
            {
                // Get the AST from the parser
                CommonAST parseTree = (CommonAST)parser.getAST();
                if(parseTree!=null)
                {
                    // Print the AST in a human-readable format
                    System.out.println(parseTree.toStringList());

                    // Open a window in which the AST is displayed graphically
                    frame = new ASTFrame("AST from the ELMO parser", parseTree);
                    frame.setVisible(true);
                    System.out.println("time to walk it: ");

                    walker = new ELMOWalker();
                    walker.program(parseTree);
                    ELMOObjectWriter.flush("output.obj");

                    System.out.println("done walking");
                }
                else
                    System.out.println("null parse tree!");
            }
        }
        catch(SemanticException e)
        {
            System.out.print("Semantic Error: line ");
            System.out.println(e.getLine()+":"+e.getMessage());
//          frame.dispose();
        }
        catch(RecognitionException e)
        {
            System.out.print("Syntax Error: line ");
            System.out.println(e.getLine()+":"+e.getMessage());
        }
        catch(Exception e)
        {
            System.out.println("General Error: ");
            System.out.println(e.getMessage());
        }
    }
}

import java.io.*;
import antlr.*;
//import antlr.CommonAST;
import antlr.collections.AST;
import antlr.debug.misc.ASTFrame;
//import antlr.TokenStreamException;
//import antlr.Token;

public class TestParser
{
    public static void main(String[] args)
    {
        try{
            DataInputStream input = new DataInputStream(System.in);

            // Create the lexer and parser and feed them the input

            ELMOLexer lexer = new ELMOLexer(input);
        /*
            //this code manually runs the lexer for debugging purposes
            System.out.println("running lexer");
            while(true)
            {
                try
                {
                    if(lexer.nextToken().getType()==Token.EOF_TYPE)
                        break;
                }
                catch(TokenStreamException tse)
                {
                    System.err.println("token error: "+tse);
                    //keep reading (or attempt to)
                    lexer.consume();
                }
            }

            System.out.println("done lexing");
        */

            ELMOParser parser = new ELMOParser(lexer);

            System.out.println("parsing file");
            parser.program(); // "file" is the main rule in the parser
            System.out.println("done parsing");
            if((parser.myFlags & parser.FLAG_ERROR)!=0)
            {
                System.out.println("The file did not parse correctly.");
                System.out.println("Please fix it and try again.");
            }
            else
            {
                // Get the AST from the parser
                CommonAST parseTree = (CommonAST)parser.getAST();

                if(parseTree!=null)
                {
                    // Print the AST in a human-readable format
                    System.out.println(parseTree.toStringList());

                    // Open a window in which the AST is displayed graphically
                    ASTFrame frame = new ASTFrame("AST from the ELMO parser", parseTree);
                    frame.setVisible(true);
                    /*
                    System.out.println("time to walk it: ");

                    ELMOWalker w = new ELMOWalker();
                    w.program(parseTree);

                    System.out.println("done walking");
                    */
                }
                else
                    System.out.println("null parse tree!");
            }
        }
        catch(RecognitionException e)
        {
            System.out.print("RecognitionException: ");
            System.out.println(e.toString());
        }
        catch(TokenStreamException e)
        {
            System.out.print("TokenStreamException: ");
            System.out.println(e.toString());
        }
    }
}

import java.io.*;
import antlr.CommonAST;
import antlr.collections.AST;
import antlr.TokenStreamException;
import antlr.Token;
import antlr.RecognitionException;
import antlr.SemanticException;

import antlr.debug.misc.ASTFrame;

class Elmoc
{
    public static void main(String[] args)
    {
        int i;

        //parse input
        if(args.length!=2)
        {
            System.err.println("usage:");
            System.err.println("    java -jar elmoc <elmo script filename> <output .obj file>");
            return;
        }
        File infile = new File(args[0]);
        if(!infile.exists())
        {
            System.err.println("could not find the file '"+infile+"'");
            return;
        }
        File outfile = new File(args[1]);
        if(outfile.exists())
        {
            System.out.print("file '"+outfile+"' exists; overwrite (y/n)? ");
            try{
                i = System.in.read();
                if(i!='y' && i!= 'Y' && i!=-1)
                {
                    System.out.println("aborting");
                    return;
                }
            }
            catch(IOException e)
            {
                System.out.println("error reading y/n response");
                return;
            }
            System.out.println("");
        }

System.out.println("input file: "+infile);
System.out.println("output file: "+outfile);

        //ready to run program
        ELMOLexer lexer=null;
        ELMOParser parser=null;
        ELMOWalker walker=null;

//ASTFrame frame=null;

        //first, parse the program, and catch those errors
        try{
            // Create the lexer and parser and feed them the input
            lexer = new ELMOLexer(new BufferedInputStream(new FileInputStream(infile)));
            parser = new ELMOParser(lexer);

//System.out.println("parsing file");
            parser.program();           // "program" is the main rule in the parser
//System.out.println("done parsing");
        }
        catch(SemanticException e)
        {
            System.out.print("Semantic Error: at ");
            System.out.println(e.toString());
        }
        catch(RecognitionException e)
        {
            System.out.print("Syntax Error: at ");
            System.out.println(e.toString());
        }
        catch(TokenStreamException e)
        {
            System.out.print("Lexer Error: at ");
            System.out.println(e.toString());
        }
        catch(Exception e)
        {
            System.out.println("General Error: ");
            System.out.println(e.toString());
            e.printStackTrace();
        }

        //now, assuming there weren't any egregeous errors,
        //walk the AST tree that has been produced (actually ELMOAST)
        if((parser.myFlags & parser.FLAG_ERROR)!=0)
        {
            System.out.println("The file did not parse correctly.");
            System.out.println("Please fix it and try again.");
        }
        else
        {
            try
            {
                // Get the AST from the parser
                CommonAST parseTree = (CommonAST)parser.getAST();
                if(parseTree!=null)
                {
// print the AST lisp-style
//System.out.println(parseTree.toStringList());
// show the AST in a window
//frame = new ASTFrame("AST from the ELMO parser", parseTree);
//frame.setVisible(true);

//System.out.println("time to walk it: ");

                    walker = new ELMOWalker();
                    walker.program(parseTree);
                    ELMOObjectWriter.flush(outfile.getCanonicalPath());

//System.out.println("done walking");
                }
                else
                    System.out.println("General Parser Error");
            }
            catch(SemanticException e)
            {
                System.out.print("Semantic Error: line ");
                System.out.println(e.getLine()+":"+e.getMessage());
            }
            catch(RecognitionException e)
            {
                System.out.print("Syntax Error: line ");
                System.out.println(e.getLine()+":"+e.getMessage());
            }
            catch(Exception e)
            {
                System.out.println("General Runtime Error: ");
                System.out.println(e.getMessage());
            }
        }
    }
}
```

Appendix D

CVS Logs

```
RCS file: /cvs_files/repository/PLT/elmo/README.txt,v
Working file: README.txt
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/11/11 06:05:11;  author: jw;  state: Exp;  lines: +5 -14
about to work on parser
----------------------------
revision 1.1
date: 2004/11/05 02:27:09;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Added documentation
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/build.xml,v
Working file: build.xml
head: 1.14
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.14
    arelease: 1.1.1.2
    avendor: 1.1.1
    start: 1.1.1.1
    cu: 1.1.1
keyword substitution: kv
total revisions: 16;    selected revisions: 14
description:
----------------------------
revision 1.14
date: 2004/12/21 04:27:02;  author: jw;  state: Exp;  lines: +2 -2
working on rand functioning...
----------------------------
revision 1.13
date: 2004/12/21 01:52:52;  author: jw;  state: Exp;  lines: +28 -25
added elmoc stuff
----------------------------
revision 1.12
date: 2004/12/04 16:45:33;  author: jw;  state: Exp;  lines: +1 -1
added "copy" method to ELMODataType and its subclasses
----------------------------
revision 1.11
date: 2004/12/04 07:49:00;  author: cvs_sl2285;  state: Exp;  lines: +5 -4
Function invocation works.
Pass-by-reference works.
RandomNumberRange works.
ELMOVector can be initialized from integers.
ELMOFloat is now a double underneath.
```

```
"ant test" now refers to a -Dinput property for the input file.
----------------------------
revision 1.10
date: 2004/12/03 17:14:09;  author: cvs_sl2285;  state: Exp;  lines: +5 -7
Added copy java files in src/java to buildsrc
----------------------------
revision 1.9
date: 2004/12/02 22:39:22;  author: jw;  state: Exp;  lines: +33 -8
altered build.xml to support paraphrases
added a little error-checking
----------------------------
revision 1.8
date: 2004/11/20 06:04:06;  author: cvs_sl2285;  state: Exp;  lines: +3 -0
Added some symbol table stuff
----------------------------
revision 1.7
date: 2004/11/11 06:05:12;  author: jw;  state: Exp;  lines: +1 -1
about to work on parser
----------------------------
revision 1.6
date: 2004/11/07 00:48:11;  author: jw;  state: Exp;  lines: +54 -4
updated the build.xml file with some more targets for finer granularity
everything should be backwards-compatible
i just made it so I could do "ant testparser" without worrying about whether the walker is in a compile-able state
----------------------------
revision 1.5
date: 2004/11/05 22:37:05;  author: cvs_sl2285;  state: Exp;  lines: +1 -0
Added "test" target in the build file
Enable tree walking in Tester
----------------------------
revision 1.4
date: 2004/11/05 21:54:12;  author: cvs_sl2285;  state: Exp;  lines: +7 -0
Yay, the walker walks!
----------------------------
revision 1.3
date: 2004/11/05 06:58:43;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
Work begins on ELMO_Walker
----------------------------
revision 1.2
date: 2004/11/05 02:22:45;  author: cvs_sl2285;  state: Exp;  lines: +3 -1
VL -> ELMO
----------------------------
revision 1.1
date: 2004/11/05 01:48:42;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Initial revision
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/docs/NodeTypes.txt,v
Working file: docs/NodeTypes.txt
head: 1.1
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.1
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 2; selected revisions: 1
description:
----------------------------
revision 1.1
date: 2004/11/05 02:27:09;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Added documentation
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/antlr.jar,v
Working file: src/antlr.jar
head: 1.1
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.1
    final-submission: 1.1
keyword substitution: b
total revisions: 1; selected revisions: 1
description:
----------------------------
```

```
revision 1.1
date: 2004/12/21 01:52:52;  author: jw;  state: Exp;
added elmoc stuff
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/antlr/.ELMOLexer.g.swp,v
Working file: src/antlr/.ELMOLexer.g.swp
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    start: 1.1.1.1
    cu: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/11/05 02:33:08;  author: cvs_sl2285;  state: dead;  lines: +0 -0
removed vim dropping
----------------------------
revision 1.1
date: 2004/11/05 01:48:43;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Initial revision
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/antlr/ELMOLexer.g,v
Working file: src/antlr/ELMOLexer.g
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    start: 1.1.1.1
    cu: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/11/05 02:21:04;  author: cvs_sl2285;  state: dead;  lines: +0 -0
Added the "final" Antlr grammars for ELMO
----------------------------
revision 1.1
date: 2004/11/05 01:48:43;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Initial revision
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/antlr/ELMO_Lexer.g,v
Working file: src/antlr/ELMO_Lexer.g
head: 1.11
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.11
    final-submission: 1.10
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 12;    selected revisions: 11
description:
----------------------------
revision 1.11
date: 2004/12/21 04:51:57;  author: jw;  state: Exp;  lines: +0 -59
ELMO release!
----------------------------
revision 1.10
date: 2004/12/19 20:20:11;  author: cvs_jw;  state: Exp;  lines: +4 -3
asdf
----------------------------
revision 1.9
date: 2004/12/02 22:39:22;  author: jw;  state: Exp;  lines: +19 -6
altered build.xml to support paraphrases
added a little error-checking
----------------------------
revision 1.8
date: 2004/11/20 23:47:57;  author: jw;  state: Exp;  lines: +17 -8
```

```
added elmo-specific commands (rotate, scale, etc) to parser
also added a 'print' command, for debugging porpoises
added some tests of the above to script.txt
----------------------------
revision 1.7
date: 2004/11/20 06:04:06;  author: cvs_sl2285;  state: Exp;  lines: +2 -1
Added some symbol table stuff
----------------------------
revision 1.6
date: 2004/11/15 05:29:07;  author: jw;  state: Exp;  lines: +1 -0
no message
----------------------------
revision 1.5
date: 2004/11/12 19:35:25;  author: jw;  state: Exp;  lines: +3 -0
declarations are parsed
includes func. decls
----------------------------
revision 1.4
date: 2004/11/11 09:04:10;  author: jw;  state: Exp;  lines: +1 -0
parser produces If and IfElse nodes now
----------------------------
revision 1.3
date: 2004/11/05 21:54:12;  author: cvs_sl2285;  state: Exp;  lines: +1 -0
Yay, the walker walks!
----------------------------
revision 1.2
date: 2004/11/05 02:22:46;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
VL -> ELMO
----------------------------
revision 1.1
date: 2004/11/05 02:21:04;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Added the "final" Antlr grammars for ELMO
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/antlr/ELMO_Parser.g,v
Working file: src/antlr/ELMO_Parser.g
head: 1.49
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.49
    final-submission: 1.49
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 51;    selected revisions: 49
description:
----------------------------
revision 1.49
date: 2004/12/21 01:52:52;  author: jw;  state: Exp;  lines: +1 -1
added elmoc stuff
----------------------------
revision 1.48
date: 2004/12/20 21:37:14;  author: cvs_jw;  state: Exp;  lines: +20 -35
error reporting update
----------------------------
revision 1.47
date: 2004/12/20 18:02:24;  author: jw;  state: Exp;  lines: +86 -108
parser has error line/col stuff
----------------------------
revision 1.46
date: 2004/12/20 00:46:58;  author: jw;  state: Exp;  lines: +1 -1
scaling by a constant works (worked before, but change got lost...)
----------------------------
revision 1.45
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +10 -4
new error line number stuff
----------------------------
revision 1.44
date: 2004/12/19 21:42:34;  author: cvs_jw;  state: Exp;  lines: +12 -11
we are done!   NOT
----------------------------
revision 1.43
date: 2004/12/19 20:20:11;  author: cvs_jw;  state: Exp;  lines: +20 -4
asdf
----------------------------
revision 1.42
date: 2004/12/18 21:59:28;  author: jw;  state: Exp;  lines: +25 -4
scale "around" is optional now (0,0,0 default)
```

```
if/else don't demand curly braces anymore
----------------------------
revision 1.41
date: 2004/12/18 17:22:33;  author: cvs_sl2285;  state: Exp;  lines: +4 -10
Added prefix operators
----------------------------
revision 1.40
date: 2004/12/17 20:27:43;  author: cvs_jw;  state: Exp;  lines: +1 -1
rockstars
----------------------------
revision 1.39
date: 2004/12/17 01:08:07;  author: jw;  state: Exp;  lines: +1 -1
got some actual object manipulation workin!
see objtest.elmo

scale causes an error :(
----------------------------
revision 1.38
date: 2004/12/16 17:11:18;  author: cvs_sl2285;  state: Exp;  lines: +17 -4
Last part of For loop no longer matches a semi-colon.
----------------------------
revision 1.37
date: 2004/12/16 07:45:06;  author: cvs_sl2285;  state: Exp;  lines: +5 -2
Added return w/o value.
Added Postfix Increment/Decrement.
----------------------------
revision 1.36
date: 2004/12/11 07:37:19;  author: cvs_sl2285;  state: Exp;  lines: +10 -20
Moved DotLittle Operators into separate AST node
----------------------------
revision 1.35
date: 2004/12/11 06:44:50;  author: cvs_sl2285;  state: Exp;  lines: +9 -6
Merged Assignment Operators into one AST node
----------------------------
revision 1.34
date: 2004/12/09 19:27:11;  author: cvs_sl2285;  state: Exp;  lines: +5 -7
Object's recognized by the parser/walker now.  Should work once
    ELMOObject.setCopy() and .copy() are implemented properly
----------------------------
revision 1.33
date: 2004/12/07 06:51:34;  author: cvs_sl2285;  state: Exp;  lines: +4 -6
Removed a bit of code from the walker and parser
----------------------------
revision 1.32
date: 2004/12/07 03:23:32;  author: cvs_sl2285;  state: Exp;  lines: +4 -6
No more nested int's and float's -- parser will create int/float nodes implicitly
----------------------------
revision 1.31
date: 2004/12/07 02:42:24;  author: cvs_sl2285;  state: Exp;  lines: +13 -19
Merged similiar operators together
----------------------------
revision 1.30
date: 2004/12/07 02:30:04;  author: jw;  state: Exp;  lines: +3 -1
cmpdstmts can have declarations in them now
----------------------------
revision 1.29
date: 2004/12/05 20:53:34;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
added group node walking
----------------------------
revision 1.28
date: 2004/12/05 17:51:22;  author: jw;  state: Exp;  lines: +2 -0
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
revision 1.27
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;  lines: +9 -10
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
----------------------------
revision 1.26
date: 2004/12/05 05:24:05;  author: cvs_sl2285;  state: Exp;  lines: +2 -2
Fixed ForIter
Added for loops
Fixed (pre)increment(post)
Fixed assign by reference
----------------------------
revision 1.25
date: 2004/12/04 07:49:00;  author: cvs_sl2285;  state: Exp;  lines: +8 -3
Function invocation works.
Pass-by-reference works.
```

```
RandomNumberRange works.
ELMOVector can be initialized from integers.
ELMOFloat is now a double underneath.
"ant test" now refers to a -Dinput property for the input file.
----------------------------
revision 1.24
date: 2004/12/04 04:10:49;  author: cvs_sl2285;  state: Exp;  lines: +22 -7
Incremental fixes towards getting functions/scopes to work
----------------------------
revision 1.23
date: 2004/12/03 19:54:01;  author: cvs_sl2285;  state: Exp;  lines: +1 -0
Added function invokation
----------------------------
revision 1.22
date: 2004/12/03 16:24:03;  author: jw;  state: Exp;  lines: +18 -5
no change?
----------------------------
revision 1.21
date: 2004/12/02 22:39:22;  author: jw;  state: Exp;  lines: +74 -6
altered build.xml to support paraphrases
added a little error-checking
----------------------------
revision 1.20
date: 2004/11/21 19:46:14;  author: cvs_jw;  state: Exp;  lines: +1 -0
++thingies work...we guess
----------------------------
revision 1.19
date: 2004/11/21 19:18:51;  author: cvs_sl2285;  state: Exp;  lines: +7 -7
added print statement!
----------------------------
revision 1.18
date: 2004/11/21 18:38:19;  author: cvs_jw;  state: Exp;  lines: +1 -1
foo
----------------------------
revision 1.17
date: 2004/11/21 04:27:16;  author: cvs_sl2285;  state: Exp;  lines: +6 -2
Added a class to encapsulate a function.
    - todo: add the actual function invokation

Added rudimentary scope checking and symbol table building

Collected function inputs under one list
----------------------------
revision 1.16
date: 2004/11/20 23:47:58;  author: jw;  state: Exp;  lines: +41 -8
added elmo-specific commands (rotate, scale, etc) to parser
also added a 'print' command, for debugging porpoises
added some tests of the above to script.txt
----------------------------
revision 1.15
date: 2004/11/20 18:25:21;  author: jw;  state: Exp;  lines: +1 -0
no message
----------------------------
revision 1.14
date: 2004/11/20 17:54:35;  author: jw;  state: Exp;  lines: +1 -1
no message
----------------------------
revision 1.13
date: 2004/11/20 06:04:06;  author: cvs_sl2285;  state: Exp;  lines: +5 -1
Added some symbol table stuff
----------------------------
revision 1.12
date: 2004/11/15 05:28:59;  author: jw;  state: Exp;  lines: +4 -4
no message
----------------------------
revision 1.11
date: 2004/11/13 22:39:03;  author: jw;  state: Exp;  lines: +15 -7
fundamental ambiguity with our <a,b,c> vector notation and the greater-than operator
ex: a = <1,2, 3 > - b;
versus
a = <1,2, (3>-b) >;
----------------------------
revision 1.10
date: 2004/11/12 19:35:25;  author: jw;  state: Exp;  lines: +51 -4
declarations are parsed
includes func. decls
----------------------------
revision 1.9
date: 2004/11/11 09:06:12;  author: jw;  state: Exp;  lines: +2 -2
parser produces If and IfElse nodes now
----------------------------
```

```
revision 1.8
date: 2004/11/11 09:04:10;  author: jw;  state: Exp;  lines: +80 -24
parser produces If and IfElse nodes now
----------------------------
revision 1.7
date: 2004/11/11 06:05:12;  author: jw;  state: Exp;  lines: +1 -3
about to work on parser
----------------------------
revision 1.6
date: 2004/11/06 00:10:07;  author: cvs_sl2285;  state: Exp;  lines: +2 -1
Added more nodes for Tree Walker to walk.
----------------------------
revision 1.5
date: 2004/11/05 21:54:12;  author: cvs_sl2285;  state: Exp;  lines: +31 -24
Yay, the walker walks!
----------------------------
revision 1.4
date: 2004/11/05 16:53:32;  author: cvs_sl2285;  state: Exp;  lines: +20 -1
Added statements
----------------------------
revision 1.3
date: 2004/11/05 02:24:16;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
forgot an instance of VL
----------------------------
revision 1.2
date: 2004/11/05 02:22:46;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
VL -> ELMO
----------------------------
revision 1.1
date: 2004/11/05 02:21:04;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Added the "final" Antlr grammars for ELMO
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/antlr/ELMO_Walker.g,v
Working file: src/antlr/ELMO_Walker.g
head: 1.60
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.59
    final-submission: 1.59
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 62;    selected revisions: 60
description:
----------------------------
revision 1.60
date: 2004/12/21 18:20:42;  author: cvs_jw;  state: Exp;  lines: +25 -6
conversion from deg to rad
----------------------------
revision 1.59
date: 2004/12/21 04:45:42;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
I swear the prefix operator stuff works now!
----------------------------
revision 1.58
date: 2004/12/21 04:38:48;  author: cvs_sl2285;  state: Exp;  lines: +5 -3
Re-added prefix operator stuff
----------------------------
revision 1.57
date: 2004/12/20 21:37:15;  author: cvs_jw;  state: Exp;  lines: +42 -58
error reporting update
----------------------------
revision 1.56
date: 2004/12/20 00:46:59;  author: jw;  state: Exp;  lines: +18 -0
scaling by a constant works (worked before, but change got lost...)
----------------------------
revision 1.55
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +33 -32
new error line number stuff
----------------------------
revision 1.54
date: 2004/12/19 21:42:34;  author: cvs_jw;  state: Exp;  lines: +42 -30
we are done!   NOT
----------------------------
revision 1.53
date: 2004/12/19 20:23:49;  author: cvs_sl2285;  state: Exp;  lines: +12 -12
Readded postfix "fix"
----------------------------
```

```
revision 1.52
date: 2004/12/19 20:20:21;  author: cvs_jw;  state: Exp;  lines: +32 -37
asdf
----------------------------
revision 1.51
date: 2004/12/19 20:20:11;  author: cvs_jw;  state: Exp;  lines: +4 -1
asdf
----------------------------
revision 1.50
date: 2004/12/18 21:59:29;  author: jw;  state: Exp;  lines: +20 -3
scale "around" is optional now (0,0,0 default)
if/else don't demand curly braces anymore
----------------------------
revision 1.49
date: 2004/12/18 19:38:35;  author: cvs_sl2285;  state: Exp;  lines: +14 -26
Almost fixed postfix operator
----------------------------
revision 1.48
date: 2004/12/18 17:22:33;  author: cvs_sl2285;  state: Exp;  lines: +8 -3
Added prefix operators
----------------------------
revision 1.47
date: 2004/12/17 22:40:19;  author: cvs_jw;  state: Exp;  lines: +1 -0
Outputs objects correctly!
----------------------------
revision 1.46
date: 2004/12/17 21:04:29;  author: cvs_sl2285;  state: Exp;  lines: +4 -2
Fixed recursion (hopefully)
----------------------------
revision 1.45
date: 2004/12/17 20:27:43;  author: cvs_jw;  state: Exp;  lines: +13 -3
rockstars
----------------------------
revision 1.44
date: 2004/12/17 16:58:39;  author: jw;  state: Exp;  lines: +10 -3
scaling works
----------------------------
revision 1.43
date: 2004/12/17 01:08:07;  author: jw;  state: Exp;  lines: +30 -5
got some actual object manipulation workin!
see objtest.elmo

scale causes an error :(
----------------------------
revision 1.42
date: 2004/12/16 17:11:19;  author: cvs_sl2285;  state: Exp;  lines: +7 -2
Last part of For loop no longer matches a semi-colon.
----------------------------
revision 1.41
date: 2004/12/16 07:45:06;  author: cvs_sl2285;  state: Exp;  lines: +29 -14
Added return w/o value.
Added Postfix Increment/Decrement.
----------------------------
revision 1.40
date: 2004/12/11 07:37:19;  author: cvs_sl2285;  state: Exp;  lines: +4 -31
Moved DotLittle Operators into separate AST node
----------------------------
revision 1.39
date: 2004/12/11 06:44:50;  author: cvs_sl2285;  state: Exp;  lines: +2 -71
Merged Assignment Operators into one AST node
----------------------------
revision 1.38
date: 2004/12/09 19:27:11;  author: cvs_sl2285;  state: Exp;  lines: +6 -8
Object's recognized by the parser/walker now.  Should work once
    ELMOObject.setCopy() and .copy() are implemented properly
----------------------------
revision 1.37
date: 2004/12/09 18:32:55;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
Fixed "break reference"
    If two vars, a and b, are linked by reference, then assigning b to an
        anonymous value will break its reference to a.
----------------------------
revision 1.36
date: 2004/12/09 02:44:11;  author: jw;  state: Exp;  lines: +3 -0
no message
----------------------------
revision 1.35
date: 2004/12/07 06:51:34;  author: cvs_sl2285;  state: Exp;  lines: +6 -49
Removed a bit of code from the walker and parser
----------------------------
revision 1.34
date: 2004/12/07 03:23:32;  author: cvs_sl2285;  state: Exp;  lines: +9 -4
No more nested int's and float's -- parser will create int/float nodes implicitly
----------------------------
revision 1.33
date: 2004/12/07 03:05:08;  author: jw;  state: Exp;  lines: +3 -3
some ) } thingies
----------------------------
revision 1.32
date: 2004/12/07 02:42:24;  author: cvs_sl2285;  state: Exp;  lines: +24 -128
Merged similiar operators together
----------------------------
revision 1.31
date: 2004/12/07 02:30:05;  author: jw;  state: Exp;  lines: +15 -8
cmpdstmts can have declarations in them now
----------------------------
revision 1.30
date: 2004/12/07 00:56:37;  author: jw;  state: Exp;  lines: +56 -65
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.29
date: 2004/12/06 06:23:56;  author: jw;  state: Exp;  lines: +41 -28
added preinc
need to think about how to do postinc...
----------------------------
revision 1.28
date: 2004/12/06 05:43:46;  author: jw;  state: Exp;  lines: +52 -52
ELMOSymbol renamed to ELMOIndirect (since it's more generic than just for symbol table stuff)
----------------------------
revision 1.27
date: 2004/12/06 05:35:22;  author: jw;  state: Exp;  lines: +26 -13
references work and functions do to (need more thorough testing)
----------------------------
revision 1.26
date: 2004/12/06 05:10:20;  author: jw;  state: Exp;  lines: +171 -191
func calls with refs seem to work
got rid of ELMODataHolder (now it's just ELMOSymbol, which is soon to be renamed)
----------------------------
revision 1.25
date: 2004/12/06 02:30:32;  author: jw;  state: Exp;  lines: +1 -1
print fix
----------------------------
revision 1.24
date: 2004/12/05 22:36:13;  author: jw;  state: Exp;  lines: +11 -11
it compiles
----------------------------
revision 1.23
date: 2004/12/05 22:32:39;  author: jw;  state: Exp;  lines: +11 -11
9 errors
----------------------------
revision 1.22
date: 2004/12/05 22:30:00;  author: jw;  state: Exp;  lines: +11 -11
19 errors
----------------------------
revision 1.21
date: 2004/12/05 22:27:55;  author: jw;  state: Exp;  lines: +1 -1
down to 30 errors
----------------------------
revision 1.20
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +154 -149
mucho errors
----------------------------
revision 1.19
date: 2004/12/05 20:53:34;  author: cvs_sl2285;  state: Exp;  lines: +136 -14
added group node walking
----------------------------
revision 1.18
date: 2004/12/05 18:28:16;  author: jw;  state: Exp;  lines: +43 -22
walker compiles - still needs buncha work
----------------------------
revision 1.17
date: 2004/12/05 18:15:05;  author: jw;  state: Exp;  lines: +41 -24
more to do in the walker with ELMODataHolder...
ELMO base types (Integer/Float) need work, too
----------------------------
revision 1.16
date: 2004/12/05 17:51:22;  author: jw;  state: Exp;  lines: +72 -57
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
```

```
revision 1.15
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;  lines: +87 -123
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
----------------------------
revision 1.14
date: 2004/12/05 05:24:05;  author: cvs_sl2285;  state: Exp;  lines: +62 -15
Fixed ForIter
Added for loops
Fixed (pre)increment(post)
Fixed assign by reference
----------------------------
revision 1.13
date: 2004/12/04 23:10:47;  author: cvs_sl2285;  state: Exp;  lines: +17 -10
If(Else) works
----------------------------
revision 1.12
date: 2004/12/04 17:15:24;  author: cvs_sl2285;  state: Exp;  lines: +58 -0
Added If(Else) statements and >, >=
----------------------------
revision 1.11
date: 2004/12/04 07:49:00;  author: cvs_sl2285;  state: Exp;  lines: +191 -74
Function invocation works.
Pass-by-reference works.
RandomNumberRange works.
ELMOVector can be initialized from integers.
ELMOFloat is now a double underneath.
"ant test" now refers to a -Dinput property for the input file.
----------------------------
revision 1.10
date: 2004/12/04 04:10:49;  author: cvs_sl2285;  state: Exp;  lines: +130 -61
Incremental fixes towards getting functions/scopes to work
----------------------------
revision 1.9
date: 2004/12/03 19:54:01;  author: cvs_sl2285;  state: Exp;  lines: +35 -35
Added function invokation
----------------------------
revision 1.8
date: 2004/11/21 19:18:51;  author: cvs_sl2285;  state: Exp;  lines: +10 -0
added print statement!
----------------------------
revision 1.7
date: 2004/11/21 18:53:23;  author: cvs_jw;  state: Exp;  lines: +3 -0
wakawaka
----------------------------
revision 1.6
date: 2004/11/21 04:27:16;  author: cvs_sl2285;  state: Exp;  lines: +92 -21
Added a class to encapsulate a function.
    - todo: add the actual function invokation

Added rudimentary scope checking and symbol table building

Collected function inputs under one list
----------------------------
revision 1.5
date: 2004/11/20 06:04:06;  author: cvs_sl2285;  state: Exp;  lines: +176 -143
Added some symbol table stuff
----------------------------
revision 1.4
date: 2004/11/06 00:10:07;  author: cvs_sl2285;  state: Exp;  lines: +122 -9
Added more nodes for Tree Walker to walk.
----------------------------
revision 1.3
date: 2004/11/05 21:54:12;  author: cvs_sl2285;  state: Exp;  lines: +56 -10
Yay, the walker walks!
----------------------------
revision 1.2
date: 2004/11/05 06:58:43;  author: cvs_sl2285;  state: Exp;  lines: +76 -0
Work begins on ELMO_Walker
----------------------------
revision 1.1
date: 2004/11/05 02:21:04;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Added the "final" Antlr grammars for ELMO
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/BadELMOTypeException.java,v
Working file: src/java/BadELMOTypeException.java
head: 1.6
branch:
locks: strict
```

```
access list:
symbolic names:
    hand_inable: 1.6
    final-submission: 1.6
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 7; selected revisions: 6
description:
----------------------------
revision 1.6
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +2 -2
new error line number stuff
----------------------------
revision 1.5
date: 2004/12/07 00:56:37;  author: jw;  state: Exp;  lines: +1 -1
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.4
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +3 -1
mucho errors
----------------------------
revision 1.3
date: 2004/12/05 18:28:16;  author: jw;  state: Exp;  lines: +1 -1
walker compiles - still needs buncha work
----------------------------
revision 1.2
date: 2004/12/05 18:15:06;  author: jw;  state: Exp;  lines: +4 -0
more to do in the walker with ELMODataHolder...
ELMO base types (Integer/Float) need work, too
----------------------------
revision 1.1
date: 2004/12/05 17:51:22;  author: jw;  state: Exp;
branches:  1.1.1;
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOAST.java,v
Working file: src/java/ELMOAST.java
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
    final-submission: 1.2
keyword substitution: kv
total revisions: 2; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/12/21 01:52:53;  author: jw;  state: Exp;  lines: +5 -2
added elmoc stuff
----------------------------
revision 1.1
date: 2004/12/20 00:25:56;  author: cvs_jw;  state: Exp;
added ELMOAST.java
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOArgument.java,v
Working file: src/java/ELMOArgument.java
head: 1.6
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.6
    final-submission: 1.6
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 7; selected revisions: 6
description:
----------------------------
revision 1.6
date: 2004/12/07 02:30:05;  author: jw;  state: Exp;  lines: +5 -7
cmpdstmts can have declarations in them now
----------------------------
```

```
revision 1.5
date: 2004/12/06 05:43:46;  author: jw;  state: Exp;  lines: +3 -3
ELMOSymbol renamed to ELMOIndirect (since it's more generic than just for symbol table stuff)
----------------------------
revision 1.4
date: 2004/12/06 05:10:20;  author: jw;  state: Exp;  lines: +11 -8
func calls with refs seem to work
got rid of ELMODataHolder (now it's just ELMOSymbol, which is soon to be renamed)
----------------------------
revision 1.3
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +2 -2
*** empty log message ***
----------------------------
revision 1.2
date: 2004/12/03 19:54:01;  author: cvs_sl2285;  state: Exp;  lines: +11 -0
Added function invokation
----------------------------
revision 1.1
date: 2004/12/03 17:13:13;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOAssignmentOperatorAST.java,v
Working file: src/java/ELMOAssignmentOperatorAST.java
head: 1.3
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.3
    final-submission: 1.3
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 4; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2004/12/20 22:03:17;  author: cvs_jw;  state: Exp;  lines: +2 -3
intermediate error stuff
----------------------------
revision 1.2
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +7 -6
new error line number stuff
----------------------------
revision 1.1
date: 2004/12/11 00:06:13;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Collapsed all assignment operators into one
    Assignment operators AST node.
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOBinaryOperatorAST.java,v
Working file: src/java/ELMOBinaryOperatorAST.java
head: 1.5
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.5
    final-submission: 1.5
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 6; selected revisions: 5
description:
----------------------------
revision 1.5
date: 2004/12/20 22:03:17;  author: cvs_jw;  state: Exp;  lines: +1 -1
intermediate error stuff
----------------------------
revision 1.4
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +1 -1
new error line number stuff
----------------------------
revision 1.3
date: 2004/12/07 02:42:25;  author: cvs_sl2285;  state: Exp;  lines: +2 -2
Merged similiar operators together
----------------------------
revision 1.2
```

```
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +3 -3
mucho errors
----------------------------
revision 1.1
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMODataHolder.java,v
Working file: src/java/ELMODataHolder.java
head: 1.2
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 2; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/12/06 05:10:20;  author: jw;  state: dead;  lines: +0 -0
func calls with refs seem to work
got rid of ELMODataHolder (now it's just ELMOSymbol, which is soon to be renamed)
----------------------------
revision 1.1
date: 2004/12/05 17:51:22;  author: jw;  state: Exp;
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMODataType.java,v
Working file: src/java/ELMODataType.java
head: 1.16
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.16
    final-submission: 1.16
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 18;    selected revisions: 16
description:
----------------------------
revision 1.16
date: 2004/12/20 22:03:17;  author: cvs_jw;  state: Exp;  lines: +17 -17
intermediate error stuff
----------------------------
revision 1.15
date: 2004/12/18 17:22:33;  author: cvs_sl2285;  state: Exp;  lines: +4 -0
Added prefix operators
----------------------------
revision 1.14
date: 2004/12/17 22:40:19;  author: cvs_jw;  state: Exp;  lines: +23 -0
Outputs objects correctly!
----------------------------
revision 1.13
date: 2004/12/17 01:08:07;  author: jw;  state: Exp;  lines: +7 -3
got some actual object manipulation workin!
see objtest.elmo

scale causes an error :(
----------------------------
revision 1.12
date: 2004/12/09 19:14:18;  author: jw;  state: Exp;  lines: +6 -6
some more name stuff
----------------------------
revision 1.11
date: 2004/12/07 02:42:25;  author: cvs_sl2285;  state: Exp;  lines: +56 -11
Merged similiar operators together
----------------------------
revision 1.10
date: 2004/12/07 02:30:05;  author: jw;  state: Exp;  lines: +6 -6
cmpdstmts can have declarations in them now
----------------------------
revision 1.9
date: 2004/12/07 00:56:37;  author: jw;  state: Exp;  lines: +25 -1
```

```
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.8
date: 2004/12/06 05:10:20;  author: jw;  state: Exp;  lines: +4 -0
func calls with refs seem to work
got rid of ELMODataHolder (now it's just ELMOSymbol, which is soon to be renamed)
----------------------------
revision 1.7
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +5 -5
mucho errors
----------------------------
revision 1.6
date: 2004/12/05 17:51:22;  author: jw;  state: Exp;  lines: +19 -0
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
revision 1.5
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;  lines: +23 -0
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
----------------------------
revision 1.4
date: 2004/12/04 16:45:34;  author: jw;  state: Exp;  lines: +3 -0
added "copy" method to ELMODataType and its subclasses
----------------------------
revision 1.3
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +43 -0
*** empty log message ***
----------------------------
revision 1.2
date: 2004/12/03 19:54:01;  author: cvs_sl2285;  state: Exp;  lines: +2 -0
Added function invokation
----------------------------
revision 1.1
date: 2004/12/03 17:13:13;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMODotBigAST.java,v
Working file: src/java/ELMODotBigAST.java
head: 1.3
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.3
    final-submission: 1.3
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 4; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2004/12/20 22:03:17;  author: cvs_jw;  state: Exp;  lines: +2 -2
intermediate error stuff
----------------------------
revision 1.2
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +1 -1
new error line number stuff
----------------------------
revision 1.1
date: 2004/12/07 06:51:17;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Moved a lot of code from the walker to DotBigAST class.
RandomFloat extends Float and takes one or two ELMODataType's to
    initialize a random number; basically more walker cleaning
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMODotLittleAST.java,v
Working file: src/java/ELMODotLittleAST.java
head: 1.3
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.3
    final-submission: 1.3
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 4; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2004/12/20 22:03:17;  author: cvs_jw;  state: Exp;  lines: +2 -2
intermediate error stuff
----------------------------
revision 1.2
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +1 -1
new error line number stuff
----------------------------
revision 1.1
date: 2004/12/11 07:37:19;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Moved DotLittle Operators into separate AST node
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOFace.java,v
Working file: src/java/ELMOFace.java
head: 1.6
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.6
    final-submission: 1.6
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 7; selected revisions: 6
description:
----------------------------
revision 1.6
date: 2004/12/17 22:40:19;  author: cvs_jw;  state: Exp;  lines: +17 -1
Outputs objects correctly!
----------------------------
revision 1.5
date: 2004/12/17 21:02:05;  author: cvs_edp2002;  state: Exp;  lines: +8 -2
beatch
----------------------------
revision 1.4
date: 2004/12/17 18:17:55;  author: cvs_jw;  state: Exp;  lines: +1 -1
normals
----------------------------
revision 1.3
date: 2004/12/07 00:56:37;  author: jw;  state: Exp;  lines: +1 -0
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.2
date: 2004/12/06 22:13:44;  author: jw;  state: Exp;  lines: +1 -0
added some .elmo files to /tests/
----------------------------
revision 1.1
date: 2004/12/06 21:53:23;  author: cvs_edp2002;  state: Exp;
branches:  1.1.1;
TODOs Done
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOFloat.java,v
Working file: src/java/ELMOFloat.java
head: 1.17
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.17
    final-submission: 1.17
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 18;    selected revisions: 17
description:
----------------------------
revision 1.17
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +6 -6
new error line number stuff
----------------------------
```

```
revision 1.16
date: 2004/12/18 17:22:33;  author: cvs_sl2285;  state: Exp;  lines: +4 -0
Added prefix operators
----------------------------
revision 1.15
date: 2004/12/07 06:51:17;  author: cvs_sl2285;  state: Exp;  lines: +1 -2
Moved a lot of code from the walker to DotBigAST class.
RandomFloat extends Float and takes one or two ELMODataType's to
    initialize a random number; basically more walker cleaning
----------------------------
revision 1.14
date: 2004/12/07 03:25:23;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
ELMOFloat.toString()  was printing int instead of float
----------------------------
revision 1.13
date: 2004/12/07 03:23:32;  author: cvs_sl2285;  state: Exp;  lines: +15 -1
No more nested int's and float's -- parser will create int/float nodes implicitly
----------------------------
revision 1.12
date: 2004/12/07 02:42:25;  author: cvs_sl2285;  state: Exp;  lines: +78 -0
Merged similiar operators together
----------------------------
revision 1.11
date: 2004/12/07 00:56:37;  author: jw;  state: Exp;  lines: +10 -5
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.10
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +10 -13
mucho errors
----------------------------
revision 1.9
date: 2004/12/05 18:28:17;  author: jw;  state: Exp;  lines: +1 -1
walker compiles - still needs buncha work
----------------------------
revision 1.8
date: 2004/12/05 18:15:06;  author: jw;  state: Exp;  lines: +1 -1
more to do in the walker with ELMODataHolder...
ELMO base types (Integer/Float) need work, too
----------------------------
revision 1.7
date: 2004/12/05 17:51:23;  author: jw;  state: Exp;  lines: +30 -0
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
revision 1.6
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;  lines: +68 -30
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
----------------------------
revision 1.5
date: 2004/12/04 16:45:34;  author: jw;  state: Exp;  lines: +4 -0
added "copy" method to ELMODataType and its subclasses
----------------------------
revision 1.4
date: 2004/12/04 07:49:01;  author: cvs_sl2285;  state: Exp;  lines: +5 -5
Function invocation works.
Pass-by-reference works.
RandomNumberRange works.
ELMOVector can be initialized from integers.
ELMOFloat is now a double underneath.
"ant test" now refers to a -Dinput property for the input file.
----------------------------
revision 1.3
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
*** empty log message ***
----------------------------
revision 1.2
date: 2004/12/03 19:54:01;  author: cvs_sl2285;  state: Exp;  lines: +11 -1
Added function invokation
----------------------------
revision 1.1
date: 2004/12/03 17:13:13;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOFunction.java,v
Working file: src/java/ELMOFunction.java
head: 1.14
```

```
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.14
    final-submission: 1.14
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 15;    selected revisions: 14
description:
----------------------------
revision 1.14
date: 2004/12/20 22:03:17;  author: cvs_jw;  state: Exp;  lines: +3 -3
intermediate error stuff
----------------------------
revision 1.13
date: 2004/12/17 21:16:03;  author: cvs_sl2285;  state: Exp;  lines: +4 -4
they better work now
----------------------------
revision 1.12
date: 2004/12/17 21:04:29;  author: cvs_sl2285;  state: Exp;  lines: +8 -4
Fixed recursion (hopefully)
----------------------------
revision 1.11
date: 2004/12/17 20:27:43;  author: cvs_jw;  state: Exp;  lines: +5 -0
rockstars
----------------------------
revision 1.10
date: 2004/12/07 02:42:25;  author: cvs_sl2285;  state: Exp;  lines: +2 -0
Merged similiar operators together
----------------------------
revision 1.9
date: 2004/12/07 02:30:05;  author: jw;  state: Exp;  lines: +3 -5
cmpdstmts can have declarations in them now
----------------------------
revision 1.8
date: 2004/12/06 05:43:46;  author: jw;  state: Exp;  lines: +2 -2
ELMOSymbol renamed to ELMOIndirect (since it's more generic than just for symbol table stuff)
----------------------------
revision 1.7
date: 2004/12/06 05:10:20;  author: jw;  state: Exp;  lines: +9 -7
func calls with refs seem to work
got rid of ELMODataHolder (now it's just ELMOSymbol, which is soon to be renamed)
----------------------------
revision 1.6
date: 2004/12/05 22:37:25;  author: cvs_sl2285;  state: Exp;  lines: +2 -2
Fixed lots of bugs
----------------------------
revision 1.5
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +3 -3
mucho errors
----------------------------
revision 1.4
date: 2004/12/04 07:49:01;  author: cvs_sl2285;  state: Exp;  lines: +2 -1
Function invocation works.
Pass-by-reference works.
RandomNumberRange works.
ELMOVector can be initialized from integers.
ELMOFloat is now a double underneath.
"ant test" now refers to a -Dinput property for the input file.
----------------------------
revision 1.3
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +2 -2
*** empty log message ***
----------------------------
revision 1.2
date: 2004/12/03 19:54:01;  author: cvs_sl2285;  state: Exp;  lines: +58 -9
Added function invokation
----------------------------
revision 1.1
date: 2004/12/03 17:13:13;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOGroup.java,v
Working file: src/java/ELMOGroup.java
head: 1.30
branch:
locks: strict
```

```
access list:
symbolic names:
    hand_inable: 1.30
    final-submission: 1.30
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 32;    selected revisions: 30
description:
----------------------------
revision 1.30
date: 2004/12/20 22:03:17;  author: cvs_jw;  state: Exp;  lines: +8 -61
intermediate error stuff
----------------------------
revision 1.29
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +1 -1
new error line number stuff
----------------------------
revision 1.28
date: 2004/12/19 20:20:11;  author: cvs_jw;  state: Exp;  lines: +8 -11
asdf
----------------------------
revision 1.27
date: 2004/12/18 09:33:44;  author: cvs_edp2002;  state: Exp;  lines: +7 -0
no message
----------------------------
revision 1.26
date: 2004/12/18 08:53:19;  author: cvs_edp2002;  state: Exp;  lines: +2 -1
no message
----------------------------
revision 1.25
date: 2004/12/18 07:58:28;  author: cvs_edp2002;  state: Exp;  lines: +13 -10
no message
----------------------------
revision 1.24
date: 2004/12/16 03:57:40;  author: jw;  state: Exp;  lines: +3 -3
scaling
----------------------------
revision 1.23
date: 2004/12/16 03:46:46;  author: cvs_edp2002;  state: Exp;  lines: +3 -3
no message
----------------------------
revision 1.22
date: 2004/12/16 02:57:04;  author: jw;  state: Exp;  lines: +1 -1
scaling
----------------------------
revision 1.21
date: 2004/12/16 00:23:09;  author: jw;  state: Exp;  lines: +7 -0
testing import/transform/export
----------------------------
revision 1.20
date: 2004/12/15 22:29:15;  author: jw;  state: Exp;  lines: +5 -4
some changes to make groups/stamping/rotating work (via erik AIM)
----------------------------
revision 1.19
date: 2004/12/15 21:59:38;  author: cvs_edp2002;  state: Exp;  lines: +6 -3
no message
----------------------------
revision 1.18
date: 2004/12/09 19:27:11;  author: cvs_sl2285;  state: Exp;  lines: +5 -0
Object's recognized by the parser/walker now.  Should work once
    ELMOObject.setCopy() and .copy() are implemented properly
----------------------------
revision 1.17
date: 2004/12/09 02:44:11;  author: jw;  state: Exp;  lines: +2 -2
no message
----------------------------
revision 1.16
date: 2004/12/07 10:44:13;  author: cvs_jmc2108;  state: Exp;  lines: +30 -16
added ancestry check to prevent cycles when attaching groups
----------------------------
revision 1.15
date: 2004/12/07 02:42:25;  author: cvs_sl2285;  state: Exp;  lines: +4 -6
Merged similiar operators together
----------------------------
revision 1.14
date: 2004/12/07 00:56:37;  author: jw;  state: Exp;  lines: +3 -3
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.13
```

```
date: 2004/12/06 20:39:06;  author: cvs_edp2002;  state: Exp;  lines: +3 -3
Some of my TODO's are done
----------------------------
revision 1.12
date: 2004/12/06 20:38:08;  author: cvs_edp2002;  state: Exp;  lines: +34 -37
Some of my TODO's are done
----------------------------
revision 1.11
date: 2004/12/05 22:37:25;  author: cvs_sl2285;  state: Exp;  lines: +19 -24
Fixed lots of bugs
----------------------------
revision 1.10
date: 2004/12/05 22:27:55;  author: jw;  state: Exp;  lines: +34 -9
down to 30 errors
----------------------------
revision 1.9
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +15 -13
mucho errors
----------------------------
revision 1.8
date: 2004/12/05 20:52:02;  author: cvs_sl2285;  state: Exp;  lines: +130 -6
Added group operator stuff
----------------------------
revision 1.7
date: 2004/12/05 18:15:06;  author: jw;  state: Exp;  lines: +1 -1
more to do in the walker with ELMODataHolder...
ELMO base types (Integer/Float) need work, too
----------------------------
revision 1.6
date: 2004/12/05 17:51:23;  author: jw;  state: Exp;  lines: +13 -1
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
revision 1.5
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;  lines: +43 -3
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
----------------------------
revision 1.4
date: 2004/12/04 16:45:34;  author: jw;  state: Exp;  lines: +10 -0
added "copy" method to ELMODataType and its subclasses
----------------------------
revision 1.3
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +2 -3
*** empty log message ***
----------------------------
revision 1.2
date: 2004/12/03 19:54:01;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
Added function invokation
----------------------------
revision 1.1
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOIndirect.java,v
Working file: src/java/ELMOIndirect.java
head: 1.1
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.1
    final-submission: 1.1
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 2; selected revisions: 1
description:
----------------------------
revision 1.1
date: 2004/12/07 00:56:37;  author: jw;  state: Exp;
branches:  1.1.1;
buncha fixes
dot xyz stuff going smoothly now
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOInteger.java,v
Working file: src/java/ELMOInteger.java
```

```
head: 1.17
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.17
    final-submission: 1.17
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 18;    selected revisions: 17
description:
----------------------------
revision 1.17
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +1 -22
new error line number stuff
----------------------------
revision 1.16
date: 2004/12/18 17:22:33;  author: cvs_sl2285;  state: Exp;  lines: +4 -0
Added prefix operators
----------------------------
revision 1.15
date: 2004/12/07 03:23:32;  author: cvs_sl2285;  state: Exp;  lines: +27 -7
No more nested int's and float's -- parser will create int/float nodes implicitly
----------------------------
revision 1.14
date: 2004/12/07 02:42:25;  author: cvs_sl2285;  state: Exp;  lines: +110 -5
Merged similiar operators together
----------------------------
revision 1.13
date: 2004/12/07 02:30:05;  author: jw;  state: Exp;  lines: +9 -5
cmpdstmts can have declarations in them now
----------------------------
revision 1.12
date: 2004/12/07 00:56:37;  author: jw;  state: Exp;  lines: +2 -16
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.11
date: 2004/12/05 22:37:25;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
Fixed lots of bugs
----------------------------
revision 1.10
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +10 -10
mucho errors
----------------------------
revision 1.9
date: 2004/12/05 18:15:06;  author: jw;  state: Exp;  lines: +24 -3
more to do in the walker with ELMODataHolder...
ELMO base types (Integer/Float) need work, too
----------------------------
revision 1.8
date: 2004/12/05 17:51:23;  author: jw;  state: Exp;  lines: +19 -1
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
revision 1.7
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;  lines: +81 -33
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
----------------------------
revision 1.6
date: 2004/12/05 05:25:27;  author: cvs_sl2285;  state: Exp;  lines: +4 -0
Added putSymbol() which recursively puts a key->symbol up the chain of
    symbol tables
Added increment() to ELMOInteger
----------------------------
revision 1.5
date: 2004/12/04 16:45:34;  author: jw;  state: Exp;  lines: +4 -0
added "copy" method to ELMODataType and its subclasses
----------------------------
revision 1.4
date: 2004/12/04 07:49:01;  author: cvs_sl2285;  state: Exp;  lines: +5 -0
Function invocation works.
Pass-by-reference works.
RandomNumberRange works.
ELMOVector can be initialized from integers.
ELMOFloat is now a double underneath.
"ant test" now refers to a -Dinput property for the input file.
----------------------------
```

```
revision 1.3
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +0 -1
*** empty log message ***
----------------------------
revision 1.2
date: 2004/12/03 19:54:01;  author: cvs_sl2285;  state: Exp;  lines: +10 -1
Added function invokation
----------------------------
revision 1.1
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOLogicalOperatorAST.java,v
Working file: src/java/ELMOLogicalOperatorAST.java
head: 1.3
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.3
    final-submission: 1.3
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 4; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2004/12/20 22:07:30;  author: cvs_jw;  state: Exp;  lines: +2 -2
error stuff done
----------------------------
revision 1.2
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +1 -1
new error line number stuff
----------------------------
revision 1.1
date: 2004/12/07 02:42:25;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Merged similiar operators together
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOMatrix.java,v
Working file: src/java/ELMOMatrix.java
head: 1.16
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.16
    final-submission: 1.16
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 18;    selected revisions: 16
description:
----------------------------
revision 1.16
date: 2004/12/20 22:07:30;  author: cvs_jw;  state: Exp;  lines: +1 -1
error stuff done
----------------------------
revision 1.15
date: 2004/12/16 04:01:54;  author: jw;  state: Exp;  lines: +1 -1
groupies
----------------------------
revision 1.14
date: 2004/12/16 03:54:35;  author: cvs_edp2002;  state: Exp;  lines: +1 -2
no message
----------------------------
revision 1.13
date: 2004/12/16 00:23:09;  author: jw;  state: Exp;  lines: +2 -2
testing import/transform/export
----------------------------
revision 1.12
date: 2004/12/15 23:01:42;  author: jw;  state: Exp;  lines: +1 -1
testing import/transform/export
----------------------------
revision 1.11
date: 2004/12/15 22:27:33;  author: cvs_edp2002;  state: Exp;  lines: +3 -1
no message
```

```
----------------------------
revision 1.10
date: 2004/12/15 06:25:30;  author: cvs_edp2002;  state: Exp;  lines: +22 -9
no message
----------------------------
revision 1.9
date: 2004/12/07 00:56:37;  author: jw;  state: Exp;  lines: +13 -13
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.8
date: 2004/12/06 20:39:06;  author: cvs_edp2002;  state: Exp;  lines: +3 -3
Some of my TODO's are done
----------------------------
revision 1.7
date: 2004/12/06 20:38:09;  author: cvs_edp2002;  state: Exp;  lines: +15 -13
Some of my TODO's are done
----------------------------
revision 1.6
date: 2004/12/05 22:37:25;  author: cvs_sl2285;  state: Exp;  lines: +14 -10
Fixed lots of bugs
----------------------------
revision 1.5
date: 2004/12/05 22:27:56;  author: jw;  state: Exp;  lines: +7 -4
down to 30 errors
----------------------------
revision 1.4
date: 2004/12/05 20:52:02;  author: cvs_sl2285;  state: Exp;  lines: +14 -2
Added group operator stuff
----------------------------
revision 1.3
date: 2004/12/05 17:51:23;  author: jw;  state: Exp;  lines: +12 -12
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
revision 1.2
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +11 -12
*** empty log message ***
----------------------------
revision 1.1
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOObject.java,v
Working file: src/java/ELMOObject.java
head: 1.24
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.24
    final-submission: 1.24
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 26;    selected revisions: 24
description:
----------------------------
revision 1.24
date: 2004/12/20 22:03:18;  author: cvs_jw;  state: Exp;  lines: +0 -1
intermediate error stuff
----------------------------
revision 1.23
date: 2004/12/20 21:37:15;  author: cvs_jw;  state: Exp;  lines: +6 -4
error reporting update
----------------------------
revision 1.22
date: 2004/12/19 23:08:23;  author: cvs_jw;  state: Exp;  lines: +1 -1
new error line number stuff
----------------------------
revision 1.21
date: 2004/12/18 09:33:38;  author: cvs_edp2002;  state: Exp;  lines: +1 -0
no message
----------------------------
revision 1.20
date: 2004/12/18 07:58:22;  author: cvs_edp2002;  state: Exp;  lines: +2 -1
no message
----------------------------
```

```
revision 1.19
date: 2004/12/17 22:40:19;  author: cvs_jw;  state: Exp;  lines: +22 -5
Outputs objects correctly!
----------------------------
revision 1.18
date: 2004/12/17 20:27:43;  author: cvs_jw;  state: Exp;  lines: +8 -7
rockstars
----------------------------
revision 1.17
date: 2004/12/17 01:08:08;  author: jw;  state: Exp;  lines: +22 -8
got some actual object manipulation workin!
see objtest.elmo

scale causes an error :(
----------------------------
revision 1.16
date: 2004/12/15 22:29:15;  author: jw;  state: Exp;  lines: +0 -6
some changes to make groups/stamping/rotating work (via erik AIM)
----------------------------
revision 1.15
date: 2004/12/15 06:33:53;  author: cvs_edp2002;  state: Exp;  lines: +5 -4
no message
----------------------------
revision 1.14
date: 2004/12/09 19:27:11;  author: cvs_sl2285;  state: Exp;  lines: +27 -0
Object's recognized by the parser/walker now.  Should work once
    ELMOObject.setCopy() and .copy() are implemented properly
----------------------------
revision 1.13
date: 2004/12/07 00:56:37;  author: jw;  state: Exp;  lines: +3 -3
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.12
date: 2004/12/06 21:52:57;  author: cvs_edp2002;  state: Exp;  lines: +3 -2
TODOs Done
----------------------------
revision 1.11
date: 2004/12/06 05:35:23;  author: jw;  state: Exp;  lines: +2 -0
references work and functions do to (need more thorough testing)
----------------------------
revision 1.10
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +8 -1
mucho errors
----------------------------
revision 1.9
date: 2004/12/05 20:52:02;  author: cvs_sl2285;  state: Exp;  lines: +0 -16
Added group operator stuff
----------------------------
revision 1.8
date: 2004/12/05 18:15:06;  author: jw;  state: Exp;  lines: +1 -1
more to do in the walker with ELMODataHolder...
ELMO base types (Integer/Float) need work, too
----------------------------
revision 1.7
date: 2004/12/05 17:51:23;  author: jw;  state: Exp;  lines: +12 -0
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
revision 1.6
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;  lines: +3 -2
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
----------------------------
revision 1.5
date: 2004/12/04 16:45:34;  author: jw;  state: Exp;  lines: +9 -0
added "copy" method to ELMODataType and its subclasses
----------------------------
revision 1.4
date: 2004/12/04 07:49:01;  author: cvs_sl2285;  state: Exp;  lines: +2 -3
Function invocation works.
Pass-by-reference works.
RandomNumberRange works.
ELMOVector can be initialized from integers.
ELMOFloat is now a double underneath.
"ant test" now refers to a -Dinput property for the input file.
----------------------------
revision 1.3
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
*** empty log message ***
```

```
----------------------------
revision 1.2
date: 2004/12/03 19:54:01;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
Added function invokation
----------------------------
revision 1.1
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOObjectWriter.java,v
Working file: src/java/ELMOObjectWriter.java
head: 1.12
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.12
    final-submission: 1.12
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 14;    selected revisions: 12
description:
----------------------------
revision 1.12
date: 2004/12/17 22:40:19;  author: cvs_jw;  state: Exp;  lines: +6 -7
Outputs objects correctly!
----------------------------
revision 1.11
date: 2004/12/16 01:38:22;  author: jw;  state: Exp;  lines: +31 -93
used ArrayList s in the ObjectWriter instead of re-allocating arrays all the time
----------------------------
revision 1.10
date: 2004/12/16 00:23:09;  author: jw;  state: Exp;  lines: +2 -1
testing import/transform/export
----------------------------
revision 1.9
date: 2004/12/16 00:04:23;  author: cvs_edp2002;  state: Exp;  lines: +8 -8
no message
----------------------------
revision 1.8
date: 2004/12/15 23:59:16;  author: cvs_edp2002;  state: Exp;  lines: +1 -1
no message
----------------------------
revision 1.7
date: 2004/12/15 23:56:26;  author: cvs_edp2002;  state: Exp;  lines: +17 -7
no message
----------------------------
revision 1.6
date: 2004/12/15 23:47:46;  author: cvs_edp2002;  state: Exp;  lines: +12 -9
no message
----------------------------
revision 1.5
date: 2004/12/15 21:12:53;  author: jw;  state: Exp;  lines: +9 -6
compiles now...
----------------------------
revision 1.4
date: 2004/12/15 06:33:53;  author: cvs_edp2002;  state: Exp;  lines: +26 -8
no message
----------------------------
revision 1.3
date: 2004/12/07 00:56:38;  author: jw;  state: Exp;  lines: +4 -4
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.2
date: 2004/12/06 21:52:57;  author: cvs_edp2002;  state: Exp;  lines: +75 -0
TODOs Done
----------------------------
revision 1.1
date: 2004/12/05 20:52:02;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Added group operator stuff
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOPostfixOperatorAST.java,v
Working file: src/java/ELMOPostfixOperatorAST.java
head: 1.4
branch:
```

```
locks: strict
access list:
symbolic names:
    hand_inable: 1.4
    final-submission: 1.4
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 5; selected revisions: 4
description:
----------------------------
revision 1.4
date: 2004/12/20 22:07:30;  author: cvs_jw;  state: Exp;  lines: +2 -2
error stuff done
----------------------------
revision 1.3
date: 2004/12/19 23:08:24;  author: cvs_jw;  state: Exp;  lines: +1 -1
new error line number stuff
----------------------------
revision 1.2
date: 2004/12/18 19:38:35;  author: cvs_sl2285;  state: Exp;  lines: +16 -0
Almost fixed postfix operator
----------------------------
revision 1.1
date: 2004/12/16 07:45:07;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Added return w/o value.
Added Postfix Increment/Decrement.
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOPrefixOperatorAST.java,v
Working file: src/java/ELMOPrefixOperatorAST.java
head: 1.3
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.3
    final-submission: 1.3
keyword substitution: kv
total revisions: 3; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2004/12/20 22:07:30;  author: cvs_jw;  state: Exp;  lines: +1 -1
error stuff done
----------------------------
revision 1.2
date: 2004/12/19 23:08:24;  author: cvs_jw;  state: Exp;  lines: +1 -1
new error line number stuff
----------------------------
revision 1.1
date: 2004/12/18 17:22:33;  author: cvs_sl2285;  state: Exp;
Added prefix operators
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMORandomFloat.java,v
Working file: src/java/ELMORandomFloat.java
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
    final-submission: 1.2
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/12/20 22:07:30;  author: cvs_jw;  state: Exp;  lines: +2 -2
error stuff done
----------------------------
revision 1.1
date: 2004/12/07 06:51:17;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Moved a lot of code from the walker to DotBigAST class.
RandomFloat extends Float and takes one or two ELMODataType's to
    initialize a random number; basically more walker cleaning
```

```
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMORelationalOperatorAST.java,v
Working file: src/java/ELMORelationalOperatorAST.java
head: 1.3
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.3
    final-submission: 1.3
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 4; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2004/12/20 22:07:30;  author: cvs_jw;  state: Exp;  lines: +1 -1
error stuff done
----------------------------
revision 1.2
date: 2004/12/19 23:08:24;  author: cvs_jw;  state: Exp;  lines: +1 -1
new error line number stuff
----------------------------
revision 1.1
date: 2004/12/07 02:42:25;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Merged similiar operators together
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOSimple.java,v
Working file: src/java/ELMOSimple.java
head: 1.7
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.7
    final-submission: 1.7
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 8; selected revisions: 7
description:
----------------------------
revision 1.7
date: 2004/12/19 23:08:24;  author: cvs_jw;  state: Exp;  lines: +6 -1
new error line number stuff
----------------------------
revision 1.6
date: 2004/12/17 21:14:48;  author: cvs_jw;  state: Exp;  lines: +22 -0
simple thing
----------------------------
revision 1.5
date: 2004/12/17 20:28:45;  author: cvs_jw;  state: Exp;  lines: +0 -22
rockstars
----------------------------
revision 1.4
date: 2004/12/07 00:56:38;  author: jw;  state: Exp;  lines: +1 -0
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.3
date: 2004/12/06 05:35:23;  author: jw;  state: Exp;  lines: +12 -0
references work and functions do to (need more thorough testing)
----------------------------
revision 1.2
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;  lines: +9 -1
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
----------------------------
revision 1.1
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOSymbol.java,v
Working file: src/java/ELMOSymbol.java
head: 1.3
```

```
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 3; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2004/12/06 05:43:46;  author: jw;  state: dead;  lines: +0 -0
ELMOSymbol renamed to ELMOIndirect (since it's more generic than just for symbol table stuff)
----------------------------
revision 1.2
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +1 -1
mucho errors
----------------------------
revision 1.1
date: 2004/12/05 17:51:23;  author: jw;  state: Exp;
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOSymbolTable.java,v
Working file: src/java/ELMOSymbolTable.java
head: 1.8
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.8
    final-submission: 1.8
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 9; selected revisions: 8
description:
----------------------------
revision 1.8
date: 2004/12/17 20:27:43;  author: cvs_jw;  state: Exp;  lines: +31 -0
rockstars
----------------------------
revision 1.7
date: 2004/12/06 05:43:46;  author: jw;  state: Exp;  lines: +3 -3
ELMOSymbol renamed to ELMOIndirect (since it's more generic than just for symbol table stuff)
----------------------------
revision 1.6
date: 2004/12/05 22:37:25;  author: cvs_sl2285;  state: Exp;  lines: +23 -24
Fixed lots of bugs
----------------------------
revision 1.5
date: 2004/12/05 17:51:23;  author: jw;  state: Exp;  lines: +6 -3
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
revision 1.4
date: 2004/12/05 05:25:27;  author: cvs_sl2285;  state: Exp;  lines: +57 -3
Added putSymbol() which recursively puts a key->symbol up the chain of
    symbol tables
Added increment() to ELMOInteger
----------------------------
revision 1.3
date: 2004/12/04 07:49:01;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
Function invocation works.
Pass-by-reference works.
RandomNumberRange works.
ELMOVector can be initialized from integers.
ELMOFloat is now a double underneath.
"ant test" now refers to a -Dinput property for the input file.
----------------------------
revision 1.2
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
*** empty log message ***
----------------------------
revision 1.1
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================
RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOVector.java,v
Working file: src/java/ELMOVector.java
head: 1.21
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.21
    final-submission: 1.21
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 23;    selected revisions: 21
description:
----------------------------
revision 1.21
date: 2004/12/19 23:08:24;  author: cvs_jw;  state: Exp;  lines: +5 -5
new error line number stuff
----------------------------
revision 1.20
date: 2004/12/18 17:22:33;  author: cvs_sl2285;  state: Exp;  lines: +4 -0
Added prefix operators
----------------------------
revision 1.19
date: 2004/12/15 23:29:13;  author: cvs_edp2002;  state: Exp;  lines: +3 -3
no message
----------------------------
revision 1.18
date: 2004/12/15 22:27:33;  author: cvs_edp2002;  state: Exp;  lines: +6 -4
no message
----------------------------
revision 1.17
date: 2004/12/15 06:25:30;  author: cvs_edp2002;  state: Exp;  lines: +4 -11
no message
----------------------------
revision 1.16
date: 2004/12/09 19:14:18;  author: jw;  state: Exp;  lines: +24 -19
some more name stuff
----------------------------
revision 1.15
date: 2004/12/07 02:42:25;  author: cvs_sl2285;  state: Exp;  lines: +18 -0
Merged similiar operators together
----------------------------
revision 1.14
date: 2004/12/07 00:56:38;  author: jw;  state: Exp;  lines: +133 -86
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.13
date: 2004/12/06 20:39:06;  author: cvs_edp2002;  state: Exp;  lines: +3 -3
Some of my TODO's are done
----------------------------
revision 1.12
date: 2004/12/06 20:38:09;  author: cvs_edp2002;  state: Exp;  lines: +6 -2
Some of my TODO's are done
----------------------------
revision 1.11
date: 2004/12/05 22:27:56;  author: jw;  state: Exp;  lines: +4 -4
down to 30 errors
----------------------------
revision 1.10
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +12 -12
mucho errors
----------------------------
revision 1.9
date: 2004/12/05 20:52:02;  author: cvs_sl2285;  state: Exp;  lines: +2 -2
Added group operator stuff
----------------------------
revision 1.8
date: 2004/12/05 18:15:06;  author: jw;  state: Exp;  lines: +1 -1
more to do in the walker with ELMODataHolder...
ELMO base types (Integer/Float) need work, too
----------------------------
revision 1.7
date: 2004/12/05 17:51:23;  author: jw;  state: Exp;  lines: +82 -28
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
revision 1.6
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;  lines: +52 -20
```

```
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
----------------------------
revision 1.5
date: 2004/12/04 16:45:34;  author: jw;  state: Exp;  lines: +4 -0
added "copy" method to ELMODataType and its subclasses
----------------------------
revision 1.4
date: 2004/12/04 07:49:01;  author: cvs_sl2285;  state: Exp;  lines: +35 -0
Function invocation works.
Pass-by-reference works.
RandomNumberRange works.
ELMOVector can be initialized from integers.
ELMOFloat is now a double underneath.
"ant test" now refers to a -Dinput property for the input file.
----------------------------
revision 1.3
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
*** empty log message ***
----------------------------
revision 1.2
date: 2004/12/03 19:54:01;  author: cvs_sl2285;  state: Exp;  lines: +10 -1
Added function invokation
----------------------------
revision 1.1
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/ELMOVoid.java,v
Working file: src/java/ELMOVoid.java
head: 1.7
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.7
    final-submission: 1.7
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 8; selected revisions: 7
description:
----------------------------
revision 1.7
date: 2004/12/19 23:08:24;  author: cvs_jw;  state: Exp;  lines: +1 -1
new error line number stuff
----------------------------
revision 1.6
date: 2004/12/07 02:42:25;  author: cvs_sl2285;  state: Exp;  lines: +0 -25
Merged similiar operators together
----------------------------
revision 1.5
date: 2004/12/05 21:19:56;  author: cvs_jw;  state: Exp;  lines: +17 -11
mucho errors
----------------------------
revision 1.4
date: 2004/12/05 16:50:03;  author: cvs_sl2285;  state: Exp;  lines: +28 -1
Organized all the Binary Operators into one AST node
All basic ELMO types now implement a number of standard methods (plus,minus,star,slash,carat)
----------------------------
revision 1.3
date: 2004/12/04 16:45:34;  author: jw;  state: Exp;  lines: +4 -0
added "copy" method to ELMODataType and its subclasses
----------------------------
revision 1.2
date: 2004/12/04 04:10:27;  author: cvs_sl2285;  state: Exp;  lines: +2 -0
*** empty log message ***
----------------------------
revision 1.1
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/Elmoc.java,v
Working file: src/java/Elmoc.java
head: 1.1
branch:
locks: strict
```

```
access list:
symbolic names:
    hand_inable: 1.1
    final-submission: 1.1
keyword substitution: kv
total revisions: 1; selected revisions: 1
description:
----------------------------
revision 1.1
date: 2004/12/21 01:52:53;  author: jw;  state: Exp;
added elmoc stuff
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/OBJFile.java,v
Working file: src/java/OBJFile.java
head: 1.17
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.17
    final-submission: 1.17
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 19;    selected revisions: 17
description:
----------------------------
revision 1.17
date: 2004/12/20 21:37:15;  author: cvs_jw;  state: Exp;  lines: +4 -0
error reporting update
----------------------------
revision 1.16
date: 2004/12/17 22:40:19;  author: cvs_jw;  state: Exp;  lines: +4 -1
Outputs objects correctly!
----------------------------
revision 1.15
date: 2004/12/17 21:31:57;  author: cvs_jw;  state: Exp;  lines: +2 -2
tiny
----------------------------
revision 1.14
date: 2004/12/17 21:29:54;  author: cvs_edp2002;  state: Exp;  lines: +2 -2
beatch
----------------------------
revision 1.13
date: 2004/12/17 21:02:05;  author: cvs_edp2002;  state: Exp;  lines: +4 -2
beatch
----------------------------
revision 1.12
date: 2004/12/17 20:59:38;  author: cvs_jw;  state: Exp;  lines: +0 -2
fix it Stephen
----------------------------
revision 1.11
date: 2004/12/17 20:27:43;  author: cvs_jw;  state: Exp;  lines: +4 -1
rockstars
----------------------------
revision 1.10
date: 2004/12/17 01:08:08;  author: jw;  state: Exp;  lines: +25 -15
got some actual object manipulation workin!
see objtest.elmo

scale causes an error :(
----------------------------
revision 1.9
date: 2004/12/16 01:38:22;  author: jw;  state: Exp;  lines: +1 -0
used ArrayList s in the ObjectWriter instead of re-allocating arrays all the time
----------------------------
revision 1.8
date: 2004/12/16 00:23:09;  author: jw;  state: Exp;  lines: +0 -1
testing import/transform/export
----------------------------
revision 1.7
date: 2004/12/15 23:01:42;  author: jw;  state: Exp;  lines: +6 -6
testing import/transform/export
----------------------------
revision 1.6
date: 2004/12/15 21:12:53;  author: jw;  state: Exp;  lines: +1 -1
compiles now...
----------------------------
revision 1.5
date: 2004/12/15 06:33:53;  author: cvs_edp2002;  state: Exp;  lines: +84 -27
```

```
no message
----------------------------
revision 1.4
date: 2004/12/07 10:26:16;  author: cvs_jmc2108;  state: Exp;  lines: +9 -9
no message
----------------------------
revision 1.3
date: 2004/12/07 00:56:38;  author: jw;  state: Exp;  lines: +4 -4
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.2
date: 2004/12/06 21:52:57;  author: cvs_edp2002;  state: Exp;  lines: +65 -30
TODOs Done
----------------------------
revision 1.1
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/StringFile.java,v
Working file: src/java/StringFile.java
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
    final-submission: 1.2
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/12/06 21:52:58;  author: cvs_edp2002;  state: Exp;  lines: +6 -0
TODOs Done
----------------------------
revision 1.1
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Removed java classes from packages and moved to higher package namespace
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/TestParser.java,v
Working file: src/java/TestParser.java
head: 1.1
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.1
    final-submission: 1.1
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 2; selected revisions: 1
description:
----------------------------
revision 1.1
date: 2004/12/04 16:45:34;  author: jw;  state: Exp;
branches:  1.1.1;
added "copy" method to ELMODataType and its subclasses
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/src/java/Tester.java,v
Working file: src/java/Tester.java
head: 1.10
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.10
    final-submission: 1.10
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 12;    selected revisions: 10
description:
```

```
----------------------------
revision 1.10
date: 2004/12/20 21:37:15;  author: cvs_jw;  state: Exp;  lines: +37 -16
error reporting update
----------------------------
revision 1.9
date: 2004/12/18 17:22:33;  author: cvs_sl2285;  state: Exp;  lines: +1 -1
Added prefix operators
----------------------------
revision 1.8
date: 2004/12/17 20:27:43;  author: cvs_jw;  state: Exp;  lines: +8 -4
rockstars
----------------------------
revision 1.7
date: 2004/12/17 01:08:08;  author: jw;  state: Exp;  lines: +1 -0
got some actual object manipulation workin!
see objtest.elmo

scale causes an error :(
----------------------------
revision 1.6
date: 2004/12/07 00:56:38;  author: jw;  state: Exp;  lines: +14 -1
buncha fixes
dot xyz stuff going smoothly now
----------------------------
revision 1.5
date: 2004/12/05 17:51:23;  author: jw;  state: Exp;  lines: +33 -15
added the ELMODataHolder layer of indirection
right in the middle of things with the Walker
(search for "STOPPED IN THE MIDDLE")
----------------------------
revision 1.4
date: 2004/12/03 17:13:14;  author: cvs_sl2285;  state: Exp;  lines: +1 -0
Removed java classes from packages and moved to higher package namespace
----------------------------
revision 1.3
date: 2004/12/02 22:39:22;  author: jw;  state: Exp;  lines: +11 -1
altered build.xml to support paraphrases
added a little error-checking
----------------------------
revision 1.2
date: 2004/11/05 22:37:05;  author: cvs_sl2285;  state: Exp;  lines: +5 -7
Added "test" target in the build file
Enable tree walking in Tester
----------------------------
revision 1.1
date: 2004/11/05 22:30:45;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Moved Tester to java src
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/Tester.java,v
Working file: tests/Tester.java
head: 1.2
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 2; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/11/05 22:30:45;  author: cvs_sl2285;  state: dead;  lines: +0 -0
Moved Tester to java src
----------------------------
revision 1.1
date: 2004/11/05 02:49:46;  author: jw;  state: Exp;
added tests dir
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/curl.elmo,v
Working file: tests/curl.elmo
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
keyword substitution: kv
total revisions: 2; selected revisions: 2
```

```
description:
----------------------------
revision 1.2
date: 2004/12/18 12:06:15;  author: cvs_edp2002;  state: Exp;  lines: +3 -3
no message
----------------------------
revision 1.1
date: 2004/12/18 11:23:57;  author: cvs_edp2002;  state: Exp;
no message
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/cylinder.obj,v
Working file: tests/cylinder.obj
head: 1.1
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.1
keyword substitution: kv
total revisions: 1; selected revisions: 1
description:
----------------------------
revision 1.1
date: 2004/12/17 17:35:24;  author: cvs_edp2002;  state: Exp;
no message
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/cylinder_max.obj,v
Working file: tests/cylinder_max.obj
head: 1.1
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.1
keyword substitution: kv
total revisions: 1; selected revisions: 1
description:
----------------------------
revision 1.1
date: 2004/12/18 19:48:35;  author: jw;  state: Exp;
tree.elmo makes a tree now!
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/dottest.elmo,v
Working file: tests/dottest.elmo
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/12/07 02:30:05;  author: jw;  state: Exp;  lines: +9 -1
cmpdstmts can have declarations in them now
----------------------------
revision 1.1
date: 2004/12/07 00:56:38;  author: jw;  state: Exp;
branches:  1.1.1;
buncha fixes
dot xyz stuff going smoothly now
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/foriter.elmo,v
Working file: tests/foriter.elmo
head: 1.1
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.1
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
```

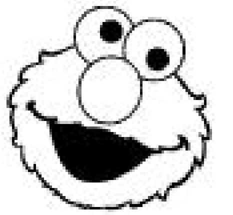
```
total revisions: 2; selected revisions: 1
description:
----------------------------
revision 1.1
date: 2004/12/16 17:11:19;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Last part of For loop no longer matches a semi-colon.
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/funcs.elmo,v
Working file: tests/funcs.elmo
head: 1.3
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.3
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 4; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2004/12/17 20:27:44;  author: cvs_jw;  state: Exp;  lines: +11 -1
rockstars
----------------------------
revision 1.2
date: 2004/12/07 02:30:05;  author: jw;  state: Exp;  lines: +40 -1
cmpdstmts can have declarations in them now
----------------------------
revision 1.1
date: 2004/12/07 00:56:38;  author: jw;  state: Exp;
branches:  1.1.1;
buncha fixes
dot xyz stuff going smoothly now
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/objtest.elmo,v
Working file: tests/objtest.elmo
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/12/17 16:58:39;  author: jw;  state: Exp;  lines: +4 -5
scaling works
----------------------------
revision 1.1
date: 2004/12/17 01:08:08;  author: jw;  state: Exp;
branches:  1.1.1;
got some actual object manipulation workin!
see objtest.elmo

scale causes an error :(
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/postinc.elmo,v
Working file: tests/postinc.elmo
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/12/18 19:38:35;  author: cvs_sl2285;  state: Exp;  lines: +6 -0
```

```
Almost fixed postfix operator
----------------------------
revision 1.1
date: 2004/12/16 07:45:25;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
Added test case for post increment
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/recurse.elmo,v
Working file: tests/recurse.elmo
head: 1.3
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.3
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 4; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2004/12/17 21:31:57;  author: cvs_jw;  state: Exp;  lines: +11 -21
tiny
----------------------------
revision 1.2
date: 2004/12/17 20:59:38;  author: cvs_jw;  state: Exp;  lines: +21 -10
fix it Stephen
----------------------------
revision 1.1
date: 2004/12/09 02:30:52;  author: cvs_sl2285;  state: Exp;
branches:  1.1.1;
added an example which breaks elmo
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/script.txt,v
Working file: tests/script.txt
head: 1.16
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.16
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 18;    selected revisions: 16
description:
----------------------------
revision 1.16
date: 2004/12/09 02:30:52;  author: cvs_sl2285;  state: Exp;  lines: +23 -0
added an example which breaks elmo
----------------------------
revision 1.15
date: 2004/12/06 05:45:10;  author: jw;  state: Exp;  lines: +9 -5
updated script.txt
----------------------------
revision 1.14
date: 2004/12/06 05:10:21;  author: jw;  state: Exp;  lines: +5 -2
func calls with refs seem to work
got rid of ELMODataHolder (now it's just ELMOSymbol, which is soon to be renamed)
----------------------------
revision 1.13
date: 2004/12/04 16:45:34;  author: jw;  state: Exp;  lines: +2 -0
added "copy" method to ELMODataType and its subclasses
----------------------------
revision 1.12
date: 2004/12/04 07:49:01;  author: cvs_sl2285;  state: Exp;  lines: +10 -93
Function invocation works.
Pass-by-reference works.
RandomNumberRange works.
ELMOVector can be initialized from integers.
ELMOFloat is now a double underneath.
"ant test" now refers to a -Dinput property for the input file.
----------------------------
revision 1.11
date: 2004/12/03 16:24:04;  author: jw;  state: Exp;  lines: +9 -2
no change?
----------------------------
revision 1.10
```

```
date: 2004/12/02 22:39:23;  author: jw;  state: Exp;  lines: +7 -0
altered build.xml to support paraphrases
added a little error-checking
----------------------------
revision 1.9
date: 2004/11/20 23:47:58;  author: jw;  state: Exp;  lines: +13 -1
added elmo-specific commands (rotate, scale, etc) to parser
also added a 'print' command, for debugging porpoises
added some tests of the above to script.txt
----------------------------
revision 1.8
date: 2004/11/20 18:25:22;  author: jw;  state: Exp;  lines: +3 -2
no message
----------------------------
revision 1.7
date: 2004/11/20 06:04:07;  author: cvs_sl2285;  state: Exp;  lines: +6 -13
Added some symbol table stuff
----------------------------
revision 1.6
date: 2004/11/13 22:39:03;  author: jw;  state: Exp;  lines: +5 -1
fundamental ambiguity with our <a,b,c> vector notation and the greater-than operator
ex: a = <1,2, 3 > - b;
versus
a = <1,2, (3>-b) >;
----------------------------
revision 1.5
date: 2004/11/12 19:35:25;  author: jw;  state: Exp;  lines: +9 -5
declarations are parsed
includes func. decls
----------------------------
revision 1.4
date: 2004/11/11 09:04:10;  author: jw;  state: Exp;  lines: +30 -1
parser produces If and IfElse nodes now
----------------------------
revision 1.3
date: 2004/11/11 06:05:12;  author: jw;  state: Exp;  lines: +5 -1
about to work on parser
----------------------------
revision 1.2
date: 2004/11/05 21:54:12;  author: cvs_sl2285;  state: Exp;  lines: +10 -3
Yay, the walker walks!
----------------------------
revision 1.1
date: 2004/11/05 02:49:46;  author: jw;  state: Exp;
branches:  1.1.1;
added tests dir
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/sphere.obj,v
Working file: tests/sphere.obj
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
keyword substitution: kv
total revisions: 2; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/12/18 12:06:15;  author: cvs_edp2002;  state: Exp;  lines: +265 -115
no message
----------------------------
revision 1.1
date: 2004/12/18 11:23:57;  author: cvs_edp2002;  state: Exp;
no message
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/teapot.obj,v
Working file: tests/teapot.obj
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 2
```

```
description:
----------------------------
revision 1.2
date: 2004/12/16 00:23:09;  author: jw;  state: Exp;  lines: +532 -532
testing import/transform/export
----------------------------
revision 1.1
date: 2004/12/15 16:48:13;  author: jw;  state: Exp;
branches:  1.1.1;
added some test OBJ files
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/testrand.elmo,v
Working file: tests/testrand.elmo
head: 1.1
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 1; selected revisions: 1
description:
----------------------------
revision 1.1
date: 2004/12/21 05:12:43;  author: jw;  state: Exp;
testrand.elmo
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/torus.obj,v
Working file: tests/torus.obj
head: 1.1
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.1
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 2; selected revisions: 1
description:
----------------------------
revision 1.1
date: 2004/12/15 16:48:14;  author: jw;  state: Exp;
branches:  1.1.1;
added some test OBJ files
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/torus_simple.obj,v
Working file: tests/torus_simple.obj
head: 1.2
branch:
locks: strict
access list:
symbolic names:
    hand_inable: 1.2
    arelease: 1.1.1.1
    avendor: 1.1.1
keyword substitution: kv
total revisions: 3; selected revisions: 2
description:
----------------------------
revision 1.2
date: 2004/12/17 01:08:08;  author: jw;  state: Exp;  lines: +11 -11
got some actual object manipulation workin!
see objtest.elmo

scale causes an error :(
----------------------------
revision 1.1
date: 2004/12/15 16:48:14;  author: jw;  state: Exp;
branches:  1.1.1;
added some test OBJ files
=============================================================================

RCS file: /cvs_files/repository/PLT/elmo/tests/tree.elmo,v
Working file: tests/tree.elmo
head: 1.15
branch:
locks: strict
access list:
```

```
symbolic names:
    hand_inable: 1.15
keyword substitution: kv
total revisions: 15;    selected revisions: 15
description:
----------------------------
revision 1.15
date: 2004/12/21 04:27:03;  author: jw;  state: Exp;  lines: +4 -4
working on rand functioning...
----------------------------
revision 1.14
date: 2004/12/21 01:52:54;  author: jw;  state: Exp;  lines: +2 -1
added elmoc stuff
----------------------------
revision 1.13
date: 2004/12/20 04:36:47;  author: jw;  state: Exp;  lines: +3 -9
tree.elmo
----------------------------
revision 1.12
date: 2004/12/20 00:46:59;  author: jw;  state: Exp;  lines: +9 -1
scaling by a constant works (worked before, but change got lost...)
----------------------------
revision 1.11
date: 2004/12/19 20:20:11;  author: cvs_jw;  state: Exp;  lines: +2 -58
asdf
----------------------------
revision 1.10
date: 2004/12/18 21:59:29;  author: jw;  state: Exp;  lines: +2 -4
scale "around" is optional now (0,0,0 default)
if/else don't demand curly braces anymore
----------------------------
revision 1.9
date: 2004/12/18 19:48:35;  author: jw;  state: Exp;  lines: +58 -8
tree.elmo makes a tree now!
----------------------------
revision 1.8
date: 2004/12/18 11:23:57;  author: cvs_edp2002;  state: Exp;  lines: +15 -17
no message
----------------------------
revision 1.7
date: 2004/12/18 08:37:09;  author: cvs_edp2002;  state: Exp;  lines: +2 -2
no message
----------------------------
revision 1.6
date: 2004/12/18 06:16:26;  author: jw;  state: Exp;  lines: +17 -5
copy assign no copy transform....bad?
----------------------------
revision 1.5
date: 2004/12/17 22:40:43;  author: cvs_jw;  state: Exp;  lines: +2 -2
Updated example
----------------------------
revision 1.4
date: 2004/12/17 21:31:57;  author: cvs_jw;  state: Exp;  lines: +1 -1
tiny
----------------------------
revision 1.3
date: 2004/12/17 20:59:38;  author: cvs_jw;  state: Exp;  lines: +5 -5
fix it Stephen
----------------------------
revision 1.2
date: 2004/12/17 20:27:44;  author: cvs_jw;  state: Exp;  lines: +16 -12
rockstars
----------------------------
revision 1.1
date: 2004/12/17 17:35:24;  author: cvs_edp2002;  state: Exp;
no message
=============================================================================
```



We swear that this is really the end.