Firefly:
An Educational Vector Graphics Language

Final Report

Authors:

Roy Aslan, Prerna Chikersal, Alexander Shnayder

{*ra2752, pc2667, ajs2119*}*@columbia.edu*

COMS W4115: Programming Languages and Translators
Summer 2014

Prof. Stephen A. Edwards

Contents

1 Introduction

Firefly (FF) is an imperative language that allows programmers to use their knowledge of vector algebra to create complex 2D shapes by drawing lines. Like the language LOGO[1, 2], Firefly can be used as an aid for teaching students how to program. Programmers can use control structures and special operators in Firefly to draw lines, which can be joined together to create complex shapes, including curves.

The backend of our compiler generates "FlatC", which we define as 3-address "assembly-like" code in C++. This "FlatC" can then be linked with the OpenGL and GLUT libraries and compiled with gcc or g++ to- produce an executable. The executable when run, displays a GLUT window with a shape as output of the FF program.

2 Motivation

Although, Firefly is inspired by the LOGO programming language, it is different from LOGO in many respects. LOGO[1, 2] is a multi-paradigm adaptation and dialect of Lisp, which is a functional programming language. Its statements are english-like commands, due to which it is only suitable for teaching very young students to program. Usually, at the high school or tertiary level, scripting languages like JavaScript and Python or imperative languages like C/C++ are used to teach students to program. These languages can however be too difficult and intimidating for beginners.

Keeping this in mind, our team came up with the idea of implementing a language called Firefly, which has complicated control structures and operators, instead of english-like commands, but is still domain-specific like LOGO. In Firefly, programmers write code to control a firefly and draw shapes using it. Apart from programming, our language can also be used to teach basic concepts in vector algebra like the meaning of vectors, direction vectors and rotation of vectors.

3 Background

This section describes the coordinate axis and how the position of the firefly is manipulated. It also gives an overview of basic vector algebra and how curves can be approximated by lines.

3.1 The Coordinate Axis

FF uses the Cartesian Coordinate System[3], as shown in figure 1.

3.2 The Firefly

The "Firefly" is defined as a point (x_0,y_0) in cartesian space, such that, when a line is drawn, it is drawn starting from (x_0,y_0) to some other point.

The Firefly's initial location is set to (0,0) and when a line is drawn from its current location to a new location, the new location becomes its current location.

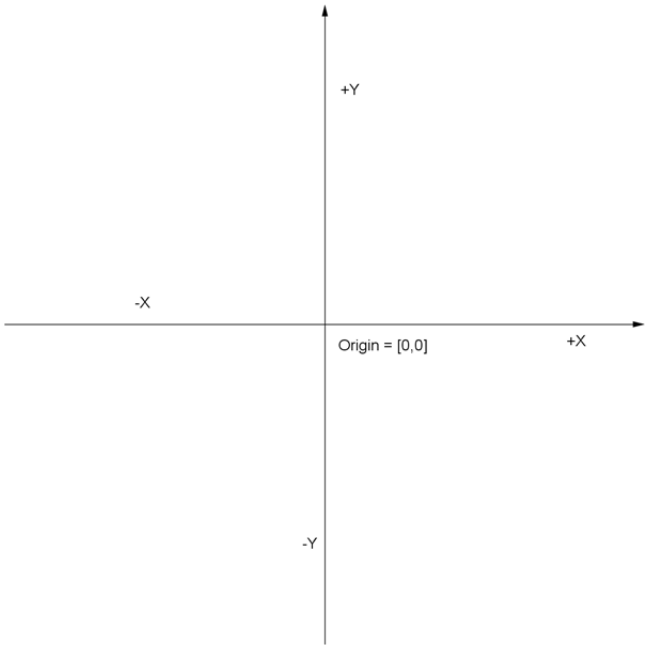

Figure 1: The Cartesian Coordinate System

3.3 Vector Algebra Primer

A **vector**[4] represents the length and direction of a line segment. Its length is denoted by $|V|$. A **unit vector** is a vector of length 1. The direction of a vector V is the unit vector U parallel to V: $U = V = |V|$. This unit vector U is also called the **direction vector** of V.

Consider [p, q] to be a vector in 2D space, such that p and q are the x and y components of the vector respectively. Rotating the vector [p,q] by an angle theta in radians, will give a **rotated vector [p', q']** as follows:

$$p' = p * cos(theta) - q * sin(theta)$$
$$q' = p * sin(theta) + q * cos(theta)$$

If theta is positive, the vector will be rotated in the anti-clockwise direction. Whereas, if theta is negative, the vector will be rotated in the clockwise direction.

4 Tutorial

This tutorial explains how to install the Firefly compiler, write a Firefly program, and then run it using the compiler.

4.1 Installation

First, download and untar the Firefly tarball. Then, within the toplevel directory, type the "make" command. Note: this requires that you have the "make" and "ocamlbuild" programs installed on your machine.

4.2 Compiling and running a Firefly program

Once the compiler has been built, you can compile a Firefly program (e.g. helloWorld.ff) by running the following command:

./firefly.byte < helloWorld.ff

If the compilation is successful, this will generate a file called "output.cpp". From there, you can compile and execute this C++ file using the GCC C++ compiler.

Alternatively, you can use the included helper script to compile a Firefly program, as well as automatically compile the resulting C++ file using the GCC C++ compiler. For example:

./runCompiler.sh helloWorld.ff:
generates an output C++ file called "helloWorld.cpp".

./runCompiler.sh -c helloWorld.ff:
generates the output C++ file and compiles it using g++ on Windows, and executes.

./runCompiler.sh -p helloWorld.ff:
same as using the "-c" option, but assumes g++ on Mac OS.

./runCompiler.sh -h:
shows usage options/help menu for the runCompiler script.

If at any point you wish to rebuild the compiler, you can simply execute "make clean" followed by "make" again.

4.3 Firefly Examples

A basic Firefly program can be used to draw simple geometric shapes. The example below shows how to draw a simple square with 4 individual line operations. Please note again, that at the beginning of each program, the firefly always starts at the origin.

tutorialEg.ff
0.5 on [1,0]
0.5 on [0,1]
0.5 on [-1,0]
0.5 on [0,-1]

The resulting output, after compiling and executing the C++ program, would look like figure 2

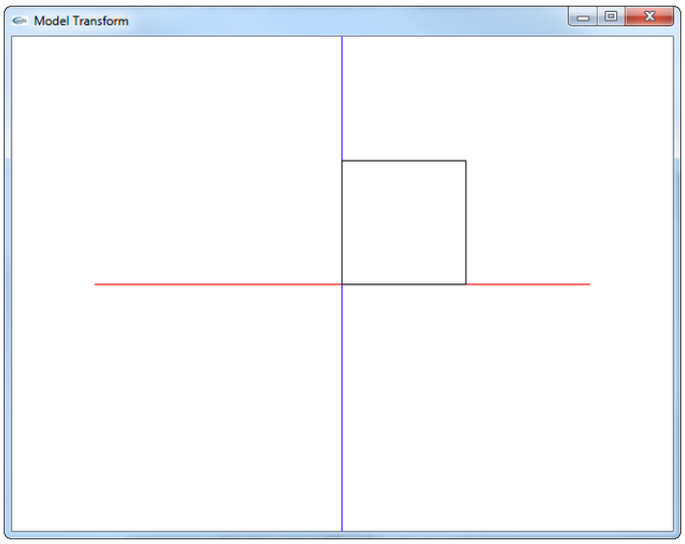

Figure 2: Firefly Tutorial Example

5 Language Reference Manual

This manual describes the Firefly (FF) language. Specifically, it explains the syntax and semantic conventions required to properly use the language. A FF program consists of a list of statements, such as, expressions, if-else statements, while loops, statement blocks, function calls and function definitions. The entry point of the program is the first statement in the file. All variables declared in the FF file are global, except those declared inside functions or passed to functions as arguments.

5.1 Lexical Conventions

There are five classes of tokens: identifiers, keywords, operators, constants and separators. Blanks, horizontal and vertical tabs and newlines, that is, all whitespace characters, as well as comments are ignored except when they separate tokens. If the input stream has been separated into tokens up to a given character, the next token is the longest string of characters that could constitute a token.

Firefly is a case sensitive language, that is, all tokens are case-sensitive.

5.1.1 Comments

Firefly supports block comments. Similar to C/C++ block comments, a comment block begins with /* and ends with */. The compiler will ignore any comment blocks embedded in the source code. Nested comments are not allowed. There is no unique syntax for single-line comments, but the comment block syntax can be used to surround a single line.

single_line_comment.ff

```
/*This is a single-line comment*/
```

multi_line_comment.ff

```
/*This is a
multi-line
comment*/
```

5.1.2 Whitespace

Whitespace consists of any sequence of space, tab, carriage return and newline characters. Whitespace can be used by the programmer for formatting programs and visually separating tokens. All whitespace is ignored by the Firefly compiler, except when it is used to separate tokens.

5.1.3 Identifiers

An identifier is a sequence of characters that consists of alphabets, numbers and underscore. An identifier cannot begin with a number or underscore. It is used to label/reference a variable or a function. An identifier must NOT be a keyword, as keywords are reserved words.

5.1.4 Keywords

Firefly supports the following keywords:

if, else, endif	Specifies conditional expression.
while	Designates a block of statements to be executed until a given predicate expression is qualified to be true
let	Designates a function definition.

5.1.5 Operators

Firefly has the following operators:

+ − * /	Arithmetic addition, subtraction, multiplication and division operators.
== != < <= > >=	Relational equal to, not equal to, less than, less than and equal to, greater than, greater than and equal to operators
=	Assignment operator.
&& || !	Boolean AND, OR and NOT operators.
on	Move firefly while drawing a line.
off	Move firefly without drawing a line.

5.1.6 Constants

$constant \rightarrow int|float$
$int \rightarrow [`+' `-']?[`0' - `9']+$

$float \rightarrow [`+' `-']?[`0' - `9'] + [`.'][`0' - `9']+$

Constants are expressions representing a fixed, literal value. They can either be of type int or float, and can have a negative value if prefixed with a "-". You can also prefix a constant with a "+", though this is optional.

5.1.7 Comma Separator

comma-separator→ ,

A comma-separator is used to separate the two elements of a vec2 expression. For example, a vec2 such as [3,4] uses the comma to separate the first element "3" from the second element "4".

5.1.8 Dollar($) Symbol

Refers to the $1 to $n locally scoped function variables.

5.2 Built-in Functions

The Firefly has the following built-in library functions:

sin(< *int* > | < *float* >)	Returns sine of an angle in degrees. Angle must be integer or float value or expression that evaluates to an integer or float value.
cos(< *int* > | < *float* >)	Returns cosine of an angle in degrees. Angle must be integer or float value or expression that evaluates to an integer or float value.
sqrt(< *int* > | < *float* >)	Returns square root of an integer or float value.

Please note that FF has no standard library functions, but the user is free to define and use his own functions.

5.3 Meaning of Identifiers

This section describes the purpose, scope and lifetime, and types of identifiers.

5.3.1 Purpose

Identifiers are used to denote either variables or functions. Functions have no return values, and their identifiers can be used for defining or calling a given function. Functions (along with the special $n local variable types) are described in depth under the statements section. Variables can be assigned values and can be referenced in FF expressions. A variable must be assigned a value (which can be any valid FF expression) before it is used in an expression. For example, the following program will be rejected by the FF compiler because myvar2 is used before it is assigned a value.

myvar1 = 0
myvar1 = myvar2 + 4
myvar2 = 1

The variable type is inferred from the first instance when it is assigned (it "inherits" the type of the expression it is assigned to). Subsequent assignments to this variable must adhere to this type. For example, the following program will be rejected since since myvar is being assigned a float value after it has been defined as a vec2.

```
myvar = [3.0, 4.0]
myvar = 2.0
```

5.3.2 Scope and Lifetime

All variables, regardless of whether they are assigned within or outside function definitions, are statically scoped global variables. This means that they can be accessed and assigned from anywhere in the program. To illustrate this, the following program will draw a line of length 0.5 by using a variable defined inside a function definition (note that the function must be called before using the variable, though the compiler does not enforce this requirement; see function definitions and calls in a subsequent section):

```
let myfun(1) =
{
    myvar1 = 5
    myvar2 = $1
}
myfun(3)
mynewvar = myvar1
```

In the above program, mynewvar will be assigned the value 5. As will be explained in the functions section, the special $1 variable signifies the value of the first (and in this case only) argument passed to the function. As we will discuss, $n variables are locally scoped, meaning that they are accessible within a function call (activation). $n variables only be assigned through a function call (by specifying their value in the function arguments).

5.3.3 Basic Types

FF supports two basic types:

- Integers
- Floating points

These types are signed, meaning that they support negative numbers.

5.3.4 Structured Types

vec2 is the only structured type. It is comprised of two floating point values (in brackets, separated by a comma). These values can be expressions containing variables as this program illustrates:

```
x = 1.0
y = 2.0
myvec = [x + 1.0, y * 2 - 4]
```

5.4 Expressions and Operators

This section describes the syntax and semantics of expressions in Firefly. Unless otherwise stated, all operators mentioned below are left-associative. They are listed in order of descending precedence.

5.4.1 Variable Expression

variable-expr → *variable*
variable → *identifier*

A variable x is an expression that evaluates to the value bound to x. A variable is represented by an identifier, as discussed in section 5.1.3.

5.4.2 Constant Expression

constant-expr → *constant*

A constant can either be an int or a float, and evaluates to the literal value of the constant, as discussed in section 5.1.6.

5.4.3 Vec2 Expression

vec2-expression → [*expression*, *expression*]

A vec2 is a two-element list and is itself an expression. Both expressions in a vec2 must have the same type, which can either be int or float.

5.4.4 Element Access Expression

element-access-expression → *vec2-expression.x* | *vec2-expression.y*

The elements of a vec2 can be accessed using an element access expression, which returns the evaluation of the specified element. The ".x" operator is used to return the first element of the specified vec2, and the ".y" operator is used to return the second element of the specified vec2. For example, "[1,2].x" evaluates to 1, and "[1,2].y" evaluates to 2.

5.4.5 Parenthesized Expression

parenthesized-expression → (*expression*)

An expression surrounded by parentheses simply evaluates to that expression. Parentheses are used for grouping in order to force execution of the inner expression, thus overriding default precedence and associativity rules. For example, "9 - (5 - 2)" forces "5 - 2" to be evaluated, preventing "9 - 5" from being evaluated as it normally would without parentheses.

5.4.6 Arithmetic Operator Expressions

arith-op-expression → *expression* + *expression* | *expression* − *expression*
| *expression* ∗ *expression* | *expression* / *expression*

An arithmetic operator expression is the result of applying an arithmetic operator on a pair of expressions. The pair of expressions must be of the same type, which can be int, float or vec2. Arithmetic operators are thus overloaded. The resulting expression will have the same type as the operand expressions. Integer division rounds to zero (i.e. any fractional remainder is truncated). Division by 0 is undefined.

Addition and subtraction have equal precedence. Multiplication and division also have equal precedence, but higher than that of addition/subtraction.

When applied to vectors, the arithmetic operators have the below behavior:

Arithmetic Operator	**Description**	**Example**
$+$	Matching element addition	$[1,2]+[3,5]=[1+3,2+5]=[4,7]$
$-$	Matching element subtraction	$[5,3]-[1,2]=[5-3,3-2]=[2,1]$
$*$	Matching element multiplication	$[1,2]*[3,4]=[1*3,2*4]=[3,8]$
$/$	Matching element division	$[4,9]/[2,3]=[4/2,9/3]=[2,3]$

5.4.7 Relational Operator Expressions

rel-op-expression → *expression* == *expression* | *expression* ! = *expression*
| *expression* < *expression* | *expression* <= *expression* | *expression* > *expression*
| *expression* >= *expression*

A relational operator expression is the result of applying a relational operator on a pair of expressions. The pair of expressions must be of the same type, which can be int or float. Relational operators are thus overloaded. The resulting expression will always have type bool. The below table explains the behavior of the relational operators. In terms of precedence, all the operators have the same precedence.

Relational Operator	**Description**	**Example**
==	Equal to	$1 == 1$ *returns true*, $1 == 0$ *returns false.*
! =	Not equal to	$1! = 2$ *returns true*, $1! = 1$ *returns false.*
<	Less than	$1 < 2$ *returns true.* $3 < 2$ *returns false.*
<=	Less than and equal to	$1 <= 1$ *returns true.* $2 <= 1$ *returns false.*
>	Greater than	$1 > 0$ *returns true.* $1 > 2$ *returns false.*
>=	Greater than and equal to	$1 >= 1$ *returns true*, $1 >= 2$ *returns false.*

It is recommended to avoid using == with float values, as the behavior could be unexpected.

5.4.8 Boolean Operator Expressions

bool-op-expression → (*expression* && *expression*) | (*expression* || *expression*) | (!*expression*)

A boolean operator expression is the result of applying a boolean operator on an expression or pair of expressions having type bool. The result of the boolean operation will always be type bool. The possible operators are listed in the below table in order of their precedence. The logical negation operator is right-associative. Note the presence of conditional evaluation.

Relational Operator	**Description**	**Example**
||	Logical Disjunction	$(b \mid\mid c) \rightarrow$ Evaluate b. If false, evaluate and return c. Else, return true.
&&	Logical Negation	$(b \;\&\&\; c) \rightarrow$ Evaluate b. If true, evaluate and return c. Else, return false.
!	Less than	$1 < 2$ $(!b) \rightarrow$ Evaluate b. If true, return false. Else, return true.

5.4.9 Firefly Expression

firefly-expression → *expression* on *expression* | *expression* off *expression*

A firefly expression moves the position of the firefly to a new location, and returns the vec2 representing the new location of the firefly. The left-hand expression of the firefly operator ("on" or "off") must be a float, and represents the distance by which the firefly will move. The right-hand expression must be a vec2, and represents the direction vector on which the firefly will move. Below is an explanation of the two firefly operators. Note that they have equal precedence and are both right-associative.

Firefly Operator	**Description**	**Example**
on	Move the firefly along the given direction vector (after normalizing it) by "x" units, where "x" represents the given distance. This will also draw a line that matches the firefly's path.	0.5 on $[1, 1]$
off	Move the firefly along the given direction vector (after normalizing it) by "x" units, where "x" represents the given distance. This will NOT draw a line that matched the firefly's path.	0.5 off $[1, 0]$

5.4.10 Assignment Expression

assign-expression → *variable* = *expression*

An assignment expression evaluates a given expression and binds the result to the given variable. The result of the assignment expression itself is the new value bound to the variable. For example, "x = 3" is an expression that will assign the value 3 to variable "x", and will also return the value 3.

If the given variable has not yet been assigned, then the assignment expression will set the variable?s type to be equal to the given expression?s type. If the given variable has already been assigned, then the type of the given expression must match the type of the variable.

Assignment is right associative.

5.5 Statements

A statement is a basic unit of execution in a Firefly program. Unless otherwise indicated, statements are executed in sequence.

5.5.1 Expression Statement

expression-statement $\rightarrow$ *assign-expression* | $firefly$-*expression*

The only valid expressions that can also exist as statements are assignment expressions and firefly expressions, since they have side effects.

5.5.2 Block Statement

block-statement $\rightarrow \{statement(statement)*\}$
A block statement is a statement that executes a group of one or more other statements enclosed in curly braces.

5.5.3 If-Else Statement

if-*else-statement* $\rightarrow$ if *expression block-statement*$_{true}$ *else block-statement*$_{false}$ $endif$

An if-else statement represents a control flow structure in which the given expression must be of type bool. The expression is evaluated, and if it's true, then only the first block-statement provided is executed. If the expression is false, then only the second block-statement provided is executed. Note that the "else" and "endif" keywords are required.

5.5.4 While Statement

while-statement $\rightarrow$ *while expression block-statement*

A while statement represents a control flow structure in which the given expression must be of type bool. The provided block-statement will be repeatedly executed until the given expression evaluates to false. The expression is evaluated before each execution of the block-statement.

5.5.5 Function Definition Statement

Functions are defined as follows:
let [function name] ([number of arguments]) = { [body of the function] }

The body of the function is comprised of FF statements. Enclosed in parenthesis is an integer that represents the number of arguments the function accepts. Function names follow the same syntactic rules as FF variables (must begin with letter, followed by sequence of letters and numbers). For example, the following defines a function named myfun that accepts 3 arguments:
let myfun(3) = { statement1 statement2 ...}

FF functions are void (i.e., they do not return a value). They can accept zero arguments. FF accepts floating point or integer arguments only.
Function arguments are referenced in the body of the function with a dollar sign ($) followed by an integer that represents the ordinal value of the argument. For example, the following function will add up its two arguments, and store the result in a variable x:
let myfun(2) = { x = $1 + $2 }

Any variable definition within the body of a function is equivalent to defining that variable outside of the function. That is, all variable definitions are statically accessible throughout the program. Arguments serve as a function?s sole local variables. Recursion is supported, and elaborated on in the next section.

Functions can be defined anywhere in the program, including within other functions. They can also be called from anywhere in the code that is below the function definition.

5.5.6 Function Call Statement

Functions are called as follows:
[function name] ([argument1, argument2, ...])

The list of arguments consists of (either an empty set or) valid FF expressions, separated by commas. For example, the following statement calls a function named myfun that accepts three arguments (x, y and z are previously defined variables):
*myfun(x + 3, y * 4.5, z)*

The above will cause the statements in the body of myfun to be executed. $1 in the body of myfun will be substituted with $x + 3$, with the other two arguments substituting $2 and $3 respectively.

A function must be defined in the code before it is called. For example, the following FF code would not be accepted by the compiler:
myfun()
myfun() = { ... }

Function calls must provide a number of arguments that is specified in the function definition. The below code will result in error due to a mismatch in argument count:
myfun(2) = { ... }
myfun(x, y, z)

Recursion is supported by FF, whereby a function can call it self with a new set of arguments. These new arguments can be expressions that contain existing arguments. Arguments are maintained through the call chain so that each set of arguments is localized. The following sample code and program output illustrates this:

```
h = 1
d = 0.1
i = 0
let f(2) =
{
        i = i + 1
```

```
        d on [$1, $2]
        if(i > 4){
                h = 1
        }
        else
        {
                f($1, $2 + 5)
        }
        endif
        d on [$1, $2]
}
f(5.0, 1.0)
```

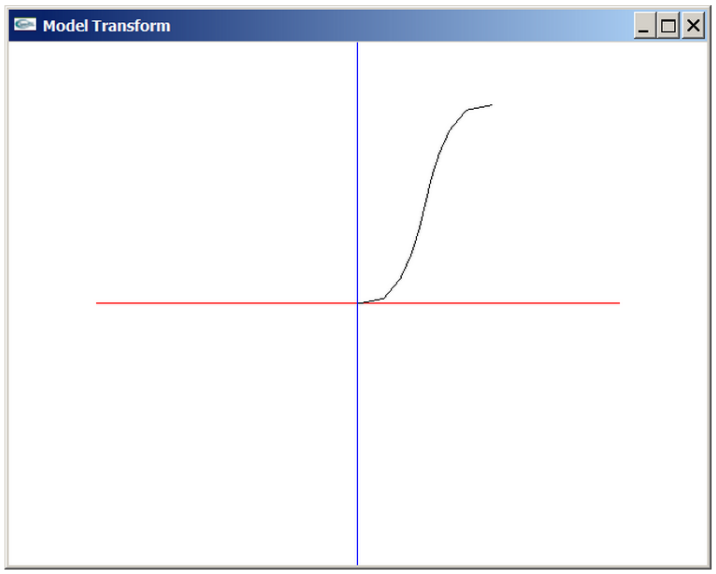

Figure 3: The image above shows how function arguments are advanced and subsequently rolled back during recursion (e.g., the first line, executed before the recursive call, and the last line, executed after recursion, have the same slope, showing the rolling back of $2).

6 Project Plan

This section gives an overview of our teamwork, project timeline, and software development environments used.

6.1 Teamwork and Collaboration

All members of our team contributed to all parts of the project equally. There were no well-defined roles and responsibilities, as we mostly sat together for hours to code.

6.1.1 Meetings

Initially, we met twice a week for 2-3 hours after class, on campus. However, once we started coding, we tried to meet almost everyday or every other day for 4-5 hours to sit together and code in one room. Towards the end, this time duration extended to 7-8 hours on Saturdays and Sundays.

6.1.2 Communication Between Meetings

Between meetings, we stayed in touch with each other via e-mail, whenever we worked on the code.

6.2 Timeline

This timeline includes class deadlines and what we finished working on.

Date	Progress
07-09-14	Proposal Submitted
07-22-14	Set up Git repository and installed OpenGL and GLUT
07-23-14	LRM Submitted
08-01-14	MakeFiles created and project setup complete
08-07-14	Initial working compiler with AST, Scanner, Parser, Compile files
08-08-14	Implemented "on" operator and drew first line in ff. Basic expressions complete.
08-11-14	Added in IF statement code
08-13-14	IF and WHILE fully working and Function definition implemented.
08-14-14	Function definition extended. Class Presentation.
08-15-14	Function call and recursion working. Created Activation Records for functions.
08-16-14	Function call with arguments implemented.
08-16-14	Final Report Submitted

6.3 Version Control

GitHub[5] was used for Version Control. Here is a link to our repository: https://github.com/prernaa/Firefly. The GitHub Log can be found in Appendix C.

6.4 Software Development Environment

OCaml compiler: v4.01
Graphics Interface: OpenGL v4.1 , GLUT v3.0
Testing environment: Shell Scripts

7 Compiler Architecture

This section explains the architecture of our compiler.

7.1 Overview

At its highest level, an FF program is a list of statements (and these include control structures and function definitions). In the first phase, the text of the source code is tokenized and parsed to build an Abstract Syntax Tree (AST). Then, each statement construct in the AST is broken down into lower-level syntactic elements (low-level syntax tree - LLST) that are closer to three address code (TAC) operations. These are passed off to semantic analysis to form a semantically check syntax tree (SAST). It is at this point that type checking is done and types for variables

are inferred. The next step is to translate these elements into TAC-like operations in C++. Once all statements are broken down and translated, the resultant intermediate code (comprised of C++ statements) is compiled into an executable using the G++ compiler.

7.2 Scanner (scanner.mll)

The scanner tokenizes the FF source code to be read by the parser.

7.3 Parser (parser.mly)

The parser pattern-matches tokens into syntax. Statements are the highest-level compositions of an FF program, as is reflected by the following code snipet:

```
program:
   /* nothing */                                              { [] }
 | program stmt                                           { ($2 :: $1) }

stmts:
    /* nothing */                                              { [] }
  | stmts stmt                                               { ($2 :: $1) }

stmt:
    expr_stmt                                                { Expr($1) }
  | LBRACE stmts RBRACE                                  { Block(List.rev $2) }
  | IF expr stmt ELSE stmt ENDIF                             { If($2, $3, $5) }
  | WHILE expr stmt                                          { While($2, $3) }
  | IDENTIFIER LPAREN actuals_opt RPAREN                 { Call ($1, $3) }
  | LET IDENTIFIER LPAREN INTEGER RPAREN ASSIGN stmt
                                                           { Fdef($2, $4, $7) }
```

control structure (IF, WHILE), function definitions and function calls are all considered statements.

7.4 Abstract and Lower-Level Syntax Tree (ast.ml and compiler.ml)

High-level, parsed FF statements are translated into smaller elements, that are closer to single operations reflect in three address code (TAC). Compiler.ml takes the high-level types constructed by ast.ml and breaks them down into these elements.

The following code snippet (compiler.ml) shows how function calls and function definitions are broken down into elements before they are fed into the semantic analyzer. Note the references to lower-level TAC elements such as labels (Lbl, Flbl) and goto statements (Goto). The end result is a lower-level syntax tree. Types for these elements are defined in semantics.ml; so the lower-level syntax tree has the same types and structure as the semantically check syntax tree (discussed in the next section). The basic structure consist of a three element tuple, containing the operation/construct, a string representation of the operation/construct, and a string representation of the data type (some elements will have an unknown data type at this point).

```
| Call(fn, args)        ->
        lbl_index := !lbl_index + 10;
        let argcount = List.length args in
        let li = !lbl_index in [SetReturn(li+1), "SET RET " ^ fn, "void"]
             @( List.concat
```

```
                    ( List.map
                            (
                                    fun e -> (gen_expr e) @
                                        [(SetLocal(fn), "SET LOC " ^ fn,
                                        "void")]
                            )       args
                    )
            )
        @ [(GotoFun(fn, argcount), "GOTO FUN " ^ fn, "void")]
        @ [(Lbl(li+1), "LBL " ^ string_of_int(li+1), "void")]
|   Fdef(fname, argcount, body)     ->
            funs.(!fun_index) <- (fname, argcount);
            fun_index := !fun_index + 1;
            lbl_index := !lbl_index + 10;
            let li = !lbl_index in
            [(Goto(li), "GOTO " ^ string_of_int(li), "void")]
            @ [(Flbl(fname), "FLBL " ^ fname, "void")]
            @ (gen_stmt body)
            @ [(GotoReturn, "GOTO RET ", "void")]
            @ [(Lbl(li), "LBL " ^ string_of_int(li), "void")]
            @ [(Endfdef, "ENDFDEF " ^ string_of_int(li), "bool")]
```

7.5 Semantic Analyser (semantics.ml

The elements of the lower-level syntax tree are manipulated/checked by the semantic analyzer in the following ways:

- **Type inference.** Previously unknown types are inferred. Types for highest precedence operations (i.e., leaves of the tree) are resolved and trickled up the chain in order to infer the datatype of the entire expression. The below image shows the diagnostic output of a program with a single statement $x = (2.3 + 5.6)$ on $[2.5 * 3.0, 9.0 - 7.0/2.0]$. Note how the unknown types in the SYNTACTIC STACK part (labeled TypeToInfer) are resolved in the SEMANTIC STACK part. The lowest level (highest precedence) operations are at the top of each stack. The bottom of the stack (or root of the statement subtree) represents the assigning of the return value of the "on" operator to the variable "x", which happens to be of type vec2cpp.

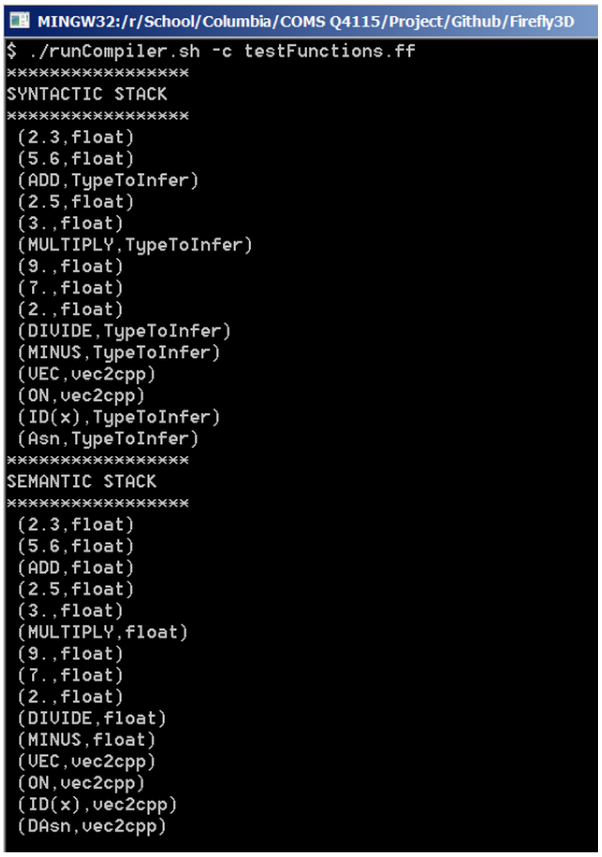

Figure 4:

- **Type checking.** Types of operands are evaluated to make sure they comply with the operator (e.g., assigning an expression of type A to a variable previously defined as a type B, making sure both operands of the polymorphic plus operation are the same, etc.).
- **Proper function use.** Function calls are checked to ensure they are calling a previously defined function and are using the correct number of arguments.

Additionally, The semantic analyzer further elaborates of the low-level syntax tree (some of the constructs of the element type defined in semantics.ml are not utilized by the LLST). The final result is a semantically check syntax tree (SAST) that can be translated to C++ TAC.

7.6 Translator (flatc.ml)

The elements of the SAST are translated into C++ code. At this point in the process, high level control flow constructs have already been broken down to near single operations.

The following illustrates high control flow is managed and maintained for function calls and how scoping is managed for function arguments (which are the called function?s local variables). Control flow for other constructs, such as while loops and if/else statements, can be found in the full code for flatc.ml in the appendix section. **Note that C++ functions are not actually utilized here (i.e., the generated code is flat and makes use of gotos and labels), and the word "function" in the following context denotes either an FF function or a conceptual simplification of the actual C++ implementation.** First, an activation

record is initiated via a C++ stack of pointers. A pointer to a label right beneath the function call is placed on the stack, and will be used as a return point:

```
|   SetReturn(i)        ->
        tvars.(!temp_counter) <- (id_ti, "actRecord");
        fprintf oc "\n\t%s" (ti ^ ".retFunc = &&_L" ^ string_of_int i ^";");
        fprintf oc "\n\t%s" ("fRecords.push(" ^ ti ^ ");");
        temp_counter := !temp_counter + 1;
```

Before the goto statement (to jump to the function definition), a frame (or base) pointer is placed on a stack. This will be used for accessing the functions local variables.

```
|   GotoFun(fn, argcount) ->
        fprintf oc "\n\t%s" ("lv_frameptr.push(lv_index);");
        fprintf oc "\n\t%s" ("goto _L" ^ fn ^ ";");
```

Another step before jumping to the function definition is to generate the argument expression and scope them correctly (as local variables of the destination function). This is done by storing the output of each argument expression in an array. Note the use of a floating point array, since FF only supports floating point arguments (whereby integer arguments are promoted).

```
|   SetLocal(fn) ->
        fprintf oc "\n\t%s" ("lclVars[lv_index] = (float)_t" ^
            string_of_int(!temp_counter - 1) ^ ";");
        fprintf oc "\n\t%s" ("lv_index = lv_index + 1;");
```

Once the control is handed off from the call to the code that implements the function definition argument values are read from the local variable stack. The argument number (i.e., ordinal position from left to right in the function call) sets the offset from frame pointer to retrieve the localized value (from the correct scope).

```
|   GetLocal(i) ->
        tvars.(!temp_counter) <- (id_ti, "float");
        fprintf oc "\n\t%s" (ti ^ " = lclVars[lv_frameptr.top() - " ^
            string_of_int i ^ "];");
        Stack.push ti stck;
        temp_counter := !temp_counter + 1;
```

Finally, once when the block of statements for the function definition are carried out in full, control is passed back to where the function called was made from. At this point the frame pointer and return record are popped from the stacks, terminating the local scope of the function call. The previous frame pointer will then be at the top of the stack; and, if the function was called by another function (as is in the case of recursion), the local scope will be rolled back to local argument values.

```
|   GotoReturn  ->
                tvars.(!temp_counter) <- (id_ti, "void *");
                fprintf oc "\n\t%s" ("lv_frameptr.pop();");
                fprintf oc "\n\t%s" ("_t"^string_of_int(!temp_counter)^" =
                    ((fRecords.top()).retFunc);");
                fprintf oc "\n\t%s" ("fRecords.pop();\n");
                fprintf oc "\n\t%s" ("goto
                    *_t"^string_of_int(!temp_counter)^";\n");
                temp_counter:=!temp_counter+1;
```

The above provides support for recursion (as illustrated in the function calls section of the LRM).

The code below is an example of the 3-address code generated by the compiler:

```
 _t17 = _ff;
_t18.retFunc = &&_L31;
fRecords.push(_t18);
_t19 = lclVars[lv_frameptr.top() - 1];
_t20 = lclVars[lv_frameptr.top() - 2];
_t21 = _t19 + _t20;
lclVars[lv_index] = (float)_t21;
lv_index = lv_index + 1;
_t22 = lclVars[lv_frameptr.top() - 2];
lclVars[lv_index] = (float)_t22;
lv_index = lv_index + 1;
lv_frameptr.push(lv_index);
goto _Lf;
_L31:
goto _L21;
_L20:
_t23 = 0;
a = _t23;
_L21:
lv_frameptr.pop();
_t24 = ((fRecords.top()).retFunc);
fRecords.pop();
```

7.7 Merging Everything Together (compiler.ml)

Compiler.ml parses the arrays of variables (both programer defined and temporary TAC variables) that were populated in the previous steps, and declarative C++ statements are generated (declaring all variables at the top of the CPP file avoids C++ scoping issues brought on by goto statements). It then merges the declarations and the translated C++ statements from flatc.ml, along with hard-coded C++ code (mostly pertaining to OpenGL and GLUT) into a finalized CPP file. This file is passed off to the g++ compiler.

7.8 g++ Compiler

We employed the g++ compiler to producing the executable file from the generated CPP file.

8 Testing

The Firefly compiler accepts a Firefly program (.ff source file) and generates a C++ program (.cpp output file). Our system test suite takes a list of simple, representative Firefly programs - each of which targets a unique feature of the language - and compiles those programs into their corresponding output C++ files. These output files are then compared to golden copies in order to ensure the output is as expected. Hence, we are system testing our compiler after integrating all its parts together.

One example test case is shown below. You will find other test cases in the “test” folder as well as in Appendix B.

8.1 Example Test Case

Below is an example for testing a simple “on” statement:

test-on.ff

1.0 on [1,1]

Below is a **snippet** from the "golden" "FlatC" file for the above. You'll find the complete code in Appendix B.

```
int myprogram(){
        float _t0 = 1.;
        int _t1 = 1;
        int _t2 = 1;
        vec2cpp _t3 = {_t1 , _t2};
        float _t4 = sqrt((_t3.x * _t3.x + _t3.y * _t3.y));
        _t3.x = _t3.x/_t4;
        _t3.y = _t3.y/_t4;
        vec2cpp _t5 = {_t0 * _t3.x + _ff.x , _t0 * _t3.y + _ff.y };
        glBegin(GL_LINES);
                glColor3f(0.0, 0.0, 0.0);
                glVertex2f(_ff.x, _ff.y);
                glVertex2f(_t5.x , _t5.y);
        glEnd();
        _ff.x = _t5.x;
        _ff.y = _t5.y;
        vec2cpp _t6 = _ff;
        return 0;
}
```

9 Sample Programs

This section shows a few sample programs written in FF, the output code generated for them by our compiler and the shapes the create.

9.1 Circle

INPUT:
circle.ff

```
/* DRAW CIRCLE */

v1 = [0.0,-1.0]
0.25 on [0.0,-1.0]

x = v1.x
y = v1.y

angle = 0.0

while (angle < 360.0)
{
        0.005 on [x,y]

        x2 = 0.005 * (cos angle)
        y2 = 0.005 * (sin angle)
```

```
        x = x2
        y = y2

        angle = angle + 1.0
}
```

OUTPUT snippet from circle.cpp

```
int myprogram(){
        vec2cpp _ff = {0,0};
        stack <actRecord> fRecords;
        float lclVars[1024];
        int lv_index = 0;
        stack <int> lv_frameptr;
        stack <int> lv_lastframeptr;

        vec2cpp v1;
        float x;
        float y;
        float angle;
        float x2;
        float y2;
        float _t0;
        float _t1;
        vec2cpp _t2;
        float _t3;
        float _t4;
        float _t5;
        vec2cpp _t6;
        float _t7;
        vec2cpp _t8;
        vec2cpp _t9;
        float _t10;
        float _t11;
        float _t12;
        float _t13;
        bool _t14;
        float _t15;
        vec2cpp _t16;
        float _t17;
        vec2cpp _t18;
        vec2cpp _t19;
        float _t20;
        float _t21;
        float _t22;
        float _t23;
        float _t24;
        float _t25;
        float _t26;
        float _t27;

        _t0 = 0.;
        _t1 = -1.;
        _t2.x = _t0;
        _t2.y = _t1;
        v1 = _t2;
        _t3 = 0.25;
        _t4 = 0.;
        _t5 = -1.;
        _t6.x = _t4;
```

```
_t6.y = _t5;
_t7 = sqrt((_t6.x * _t6.x + _t6.y * _t6.y));
_t6.x = _t6.x/_t7;
_t6.y = _t6.y/_t7;
_t8.x = _t3 * _t6.x + _ff.x;
_t8.y = _t3 * _t6.y + _ff.y;
glBegin(GL_LINES);
        glColor3f(0.0, 0.0, 0.0);
        glVertex2f(_ff.x, _ff.y);
        glVertex2f(_t8.x , _t8.y);
glEnd();
_ff.x = _t8.x;
_ff.y = _t8.y;

_t9 = _ff;
_t10 = v1.x;
x = _t10;
_t11 = v1.y;
y = _t11;
_t12 = 0.;
angle = _t12;
_L11:
_t13 = 360.;
_t14 = angle < _t13;
if (!_t14) { goto _L10; }
_t15 = 0.005;
_t16.x = x;
_t16.y = y;
_t17 = sqrt((_t16.x * _t16.x + _t16.y * _t16.y));
_t16.x = _t16.x/_t17;
_t16.y = _t16.y/_t17;
_t18.x = _t15 * _t16.x + _ff.x;
_t18.y = _t15 * _t16.y + _ff.y;
glBegin(GL_LINES);
        glColor3f(0.0, 0.0, 0.0);
        glVertex2f(_ff.x, _ff.y);
        glVertex2f(_t18.x , _t18.y);
glEnd();
_ff.x = _t18.x;
_ff.y = _t18.y;

_t19 = _ff;
_t20 = 0.005;
_t21 = cos(M_PI / 180 * angle);
_t22 = _t20 * _t21;
x2 = _t22;
_t23 = 0.005;
_t24 = sin(M_PI / 180 * angle);
_t25 = _t23 * _t24;
y2 = _t25;
x = x2;
y = y2;
_t26 = 1.;
_t27 = angle + _t26;
angle = _t27;
goto _L11;
_L10:


return 0;
```

```
}
```

GLUT WINDOW for circle.ff:

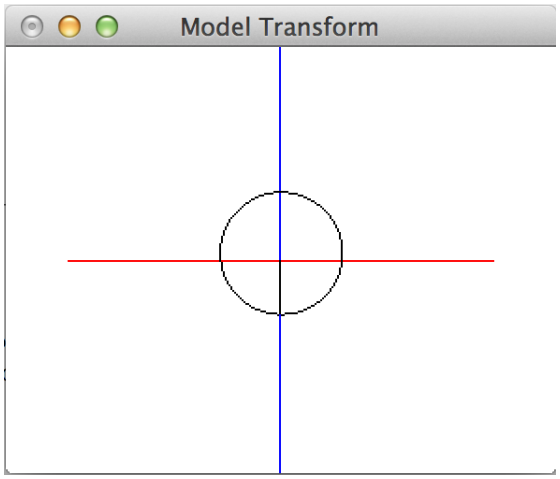

Figure 5: Glut Window Output of circle.ff

9.2 Square

INPUT:
square.ff

```
/* DRAW SQUARE */
0.2 off [1,1]
v1 = [1,0]
x = v1.x
y = v1.y

i = 4
while (i > 0)
{
        0.5 on [x,y]

        x2 = (cos 90) * x - (sin 90) * y
        y2 = (sin 90) * x + (cos 90) * y

        x = x2
        y = y2

        i = i - 1
}
```

OUTPUT snippet from
square.cpp

```
int myprogram(){
        vec2cpp _ff = {0,0};
        stack <actRecord> fRecords;
        float lclVars[1024];
        int lv_index = 0;
        stack <int> lv_frameptr;
        stack <int> lv_lastframeptr;
```

```
vec2cpp v1;
float x;
float y;
int i;
float x2;
float y2;
float _t0;
int _t1;
int _t2;
vec2cpp _t3;
float _t4;
vec2cpp _t5;
vec2cpp _t6;
int _t7;
int _t8;
vec2cpp _t9;
float _t10;
float _t11;
int _t12;
int _t13;
bool _t14;
float _t15;
vec2cpp _t16;
float _t17;
vec2cpp _t18;
vec2cpp _t19;
int _t20;
float _t21;
float _t22;
int _t23;
float _t24;
float _t25;
float _t26;
int _t27;
float _t28;
float _t29;
int _t30;
float _t31;
float _t32;
float _t33;
int _t34;
int _t35;

_t0 = 0.2;
_t1 = 1;
_t2 = 1;
_t3.x = _t1;
_t3.y = _t2;
_t4 = sqrt((_t3.x * _t3.x + _t3.y * _t3.y));
_t3.x = _t3.x/_t4;
_t3.y = _t3.y/_t4;
_t5.x = _t0 * _t3.x + _ff.x;
_t5.y = _t0 * _t3.y + _ff.y;
_ff.x = _t5.x;
_ff.y = _t5.y;

_t6 = _ff;
_t7 = 1;
_t8 = 0;
```

```
        _t9.x = _t7;
        _t9.y = _t8;
        v1 = _t9;
        _t10 = v1.x;
        x = _t10;
        _t11 = v1.y;
        y = _t11;
        _t12 = 4;
        i = _t12;
        _L11:
        _t13 = 0;
        _t14 = i > _t13;
        if (!_t14) { goto _L10; }
        _t15 = 0.5;
        _t16.x = x;
        _t16.y = y;
        _t17 = sqrt((_t16.x * _t16.x + _t16.y * _t16.y));
        _t16.x = _t16.x/_t17;
        _t16.y = _t16.y/_t17;
        _t18.x = _t15 * _t16.x + _ff.x;
        _t18.y = _t15 * _t16.y + _ff.y;
        glBegin(GL_LINES);
                glColor3f(0.0, 0.0, 0.0);
                glVertex2f(_ff.x, _ff.y);
                glVertex2f(_t18.x , _t18.y);
        glEnd();
        _ff.x = _t18.x;
        _ff.y = _t18.y;

        _t19 = _ff;
        _t20 = 90;
        _t21 = cos(M_PI / 180 * _t20);
        _t22 = _t21 * x;
        _t23 = 90;
        _t24 = sin(M_PI / 180 * _t23);
        _t25 = _t24 * y;
        _t26 = _t22 - _t25;
        x2 = _t26;
        _t27 = 90;
        _t28 = sin(M_PI / 180 * _t27);
        _t29 = _t28 * x;
        _t30 = 90;
        _t31 = cos(M_PI / 180 * _t30);
        _t32 = _t31 * y;
        _t33 = _t29 + _t32;
        y2 = _t33;
        x = x2;
        y = y2;
        _t34 = 1;
        _t35 = i - _t34;
        i = _t35;
        goto _L11;
        _L10:


        return 0;
}
```

GLUT WINDOW for square.ff:

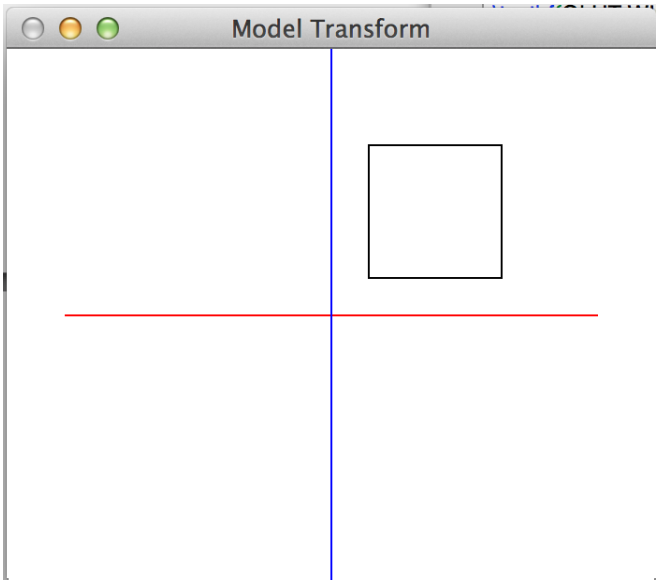

Figure 6: Glut Window Output of square.ff

9.3 Spiral

INPUT:
spiral.ff

```
/* DRAW SPIRAL */

v1 = [0.0,-1.0]

x = v1.x
y = v1.y

angle = 0.0

radius = 0.00001
finalRadius = 0.017
spiralIncrement = 0.000002

while (radius < finalRadius)
{
        radius on [x,y]

        x2 = radius * (cos angle)
        y2 = radius * (sin angle)

        x = x2
        y = y2

        angle = angle + 1.0
        radius = radius + spiralIncrement
}
```

OUTPUT snippet from
spiral.cpp

```
int myprogram(){
        vec2cpp _ff = {0,0};
```

```
stack <actRecord> fRecords;
float lclVars[1024];
int lv_index = 0;
stack <int> lv_frameptr;
stack <int> lv_lastframeptr;

vec2cpp v1;
float x;
float y;
float angle;
float radius;
float finalRadius;
float spiralIncrement;
float x2;
float y2;
float _t0;
float _t1;
vec2cpp _t2;
float _t3;
float _t4;
float _t5;
float _t6;
float _t7;
float _t8;
bool _t9;
vec2cpp _t10;
float _t11;
vec2cpp _t12;
vec2cpp _t13;
float _t14;
float _t15;
float _t16;
float _t17;
float _t18;
float _t19;
float _t20;

_t0 = 0.;
_t1 = -1.;
_t2.x = _t0;
_t2.y = _t1;
v1 = _t2;
_t3 = v1.x;
x = _t3;
_t4 = v1.y;
y = _t4;
_t5 = 0.;
angle = _t5;
_t6 = 1e-05;
radius = _t6;
_t7 = 0.017;
finalRadius = _t7;
_t8 = 2e-06;
spiralIncrement = _t8;
_L11:
_t9 = radius < finalRadius;
if (!_t9) { goto _L10; }
_t10.x = x;
_t10.y = y;
_t11 = sqrt((_t10.x * _t10.x + _t10.y * _t10.y));
```

```
        _t10.x = _t10.x/_t11;
        _t10.y = _t10.y/_t11;
        _t12.x = radius * _t10.x + _ff.x;
        _t12.y = radius * _t10.y + _ff.y;
        glBegin(GL_LINES);
                glColor3f(0.0, 0.0, 0.0);
                glVertex2f(_ff.x, _ff.y);
                glVertex2f(_t12.x , _t12.y);
        glEnd();
        _ff.x = _t12.x;
        _ff.y = _t12.y;

        _t13 = _ff;
        _t14 = cos(M_PI / 180 * angle);
        _t15 = radius * _t14;
        x2 = _t15;
        _t16 = sin(M_PI / 180 * angle);
        _t17 = radius * _t16;
        y2 = _t17;
        x = x2;
        y = y2;
        _t18 = 1.;
        _t19 = angle + _t18;
        angle = _t19;
        _t20 = radius + spiralIncrement;
        radius = _t20;
        goto _L11;
        _L10:


        return 0;
}
```

GLUT WINDOW for spiral.ff:

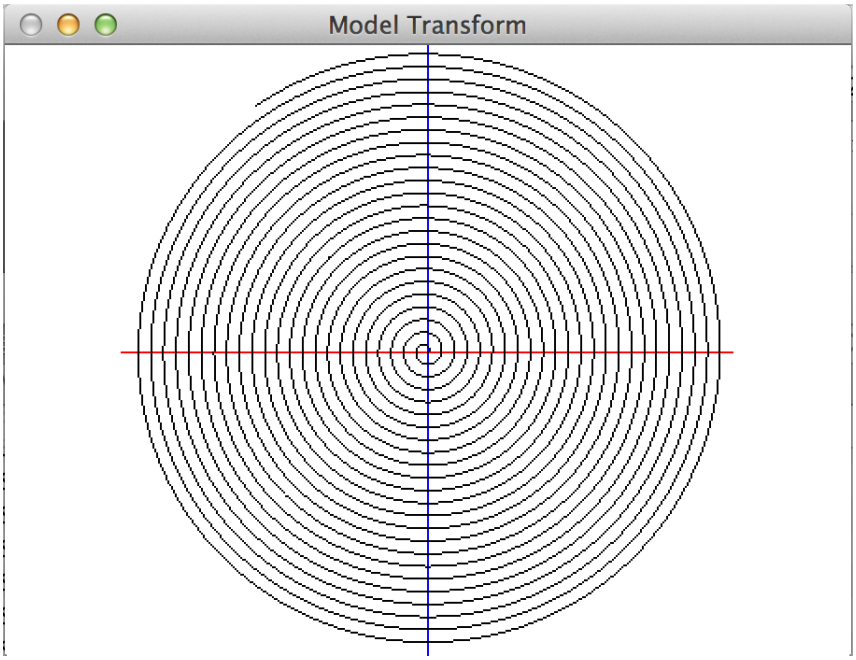

Figure 7: Glut Window Output of spiral.ff

9.4 Koch Snowflake

INPUT:
koch.ff

```
/*
        This is a sample program showing the Koch's snowflake, after its 3rd
            iteration

*/
0.2 off [1,1] /*repositioning firefly*/
0.1 off [0,1] /*repositioning firefly*/
dir=[1,0]
theta = 60
i = 12
while(i>0)
{
        0.1 on dir
        dir = [(cos 60) * dir.x - (sin 60) * dir.y, (sin 60) * dir.x + (cos
            60) * dir.y]
        0.1 on dir
        dir = [(cos -120) * dir.x - (sin -120) * dir.y, (sin -120) * dir.x +
            (cos -120) * dir.y]
        0.1 on dir
        dir = [(cos 60) * dir.x - (sin 60) * dir.y, (sin 60) * dir.x + (cos
            60) * dir.y]
        0.1 on dir
        if((i/2)*2==i)
                theta = -120
        else
                theta = 60
        endif
        dir = [(cos theta) * dir.x - (sin theta) * dir.y, (sin theta) * dir.x
            + (cos theta) * dir.y]
        i = i-1
}
```

OUTPUT snippet from
koch.cpp

```
int myprogram(){
        vec2cpp _ff = {0,0};
        stack <actRecord> fRecords;
        float lclVars[1024];
        int lv_index = 0;
        stack <int> lv_frameptr;
        stack <int> lv_lastframeptr;

        vec2cpp dir;
        int theta;
        int i;
        float _t0;
        int _t1;
        int _t2;
        vec2cpp _t3;
        float _t4;
        vec2cpp _t5;
        vec2cpp _t6;
        float _t7;
        int _t8;
        int _t9;
```

```
        vec2cpp _t10;
        float _t11;
        vec2cpp _t12;
        vec2cpp _t13;
        int _t14;
        int _t15;
        vec2cpp _t16;
        int _t17;
        int _t18;
        int _t19;
        bool _t20;
        float _t21;
        float _t22;
        vec2cpp _t23;
        vec2cpp _t24;
        int _t25;
        float _t26;
        float _t27;
        float _t28;
        int _t29;
        float _t30;
        float _t31;
        float _t32;
        float _t33;
        int _t34;
        float _t35;
        float _t36;
        float _t37;
        int _t38;
        float _t39;
        float _t40;
        float _t41;
        float _t42;
        vec2cpp _t43;
        float _t44;
        float _t45;
        vec2cpp _t46;
        vec2cpp _t47;
        int _t48;
        float _t49;
        float _t50;
        float _t51;
        int _t52;
        float _t53;
        float _t54;
        float _t55;
        float _t56;
        int _t57;
        float _t58;
        float _t59;
        float _t60;
        int _t61;
        float _t62;
        float _t63;
        float _t64;
        float _t65;
        vec2cpp _t66;
        float _t67;
        float _t68;
        vec2cpp _t69;
```

```
vec2cpp _t70;
int _t71;
float _t72;
float _t73;
float _t74;
int _t75;
float _t76;
float _t77;
float _t78;
float _t79;
int _t80;
float _t81;
float _t82;
float _t83;
int _t84;
float _t85;
float _t86;
float _t87;
float _t88;
vec2cpp _t89;
float _t90;
float _t91;
vec2cpp _t92;
vec2cpp _t93;
int _t94;
int _t95;
int _t96;
int _t97;
bool _t98;
int _t99;
int _t100;
float _t101;
float _t102;
float _t103;
float _t104;
float _t105;
float _t106;
float _t107;
float _t108;
float _t109;
float _t110;
float _t111;
float _t112;
float _t113;
float _t114;
vec2cpp _t115;
int _t116;
int _t117;

_t0 = 0.2;
_t1 = 1;
_t2 = 1;
_t3.x = _t1;
_t3.y = _t2;
_t4 = sqrt((_t3.x * _t3.x + _t3.y * _t3.y));
_t3.x = _t3.x/_t4;
_t3.y = _t3.y/_t4;
_t5.x = _t0 * _t3.x + _ff.x;
_t5.y = _t0 * _t3.y + _ff.y;
_ff.x = _t5.x;
```

```
_ff.y = _t5.y;

_t6 = _ff;
_t7 = 0.1;
_t8 = 0;
_t9 = 1;
_t10.x = _t8;
_t10.y = _t9;
_t11 = sqrt((_t10.x * _t10.x + _t10.y * _t10.y));
_t10.x = _t10.x/_t11;
_t10.y = _t10.y/_t11;
_t12.x = _t7 * _t10.x + _ff.x;
_t12.y = _t7 * _t10.y + _ff.y;
_ff.x = _t12.x;
_ff.y = _t12.y;

_t13 = _ff;
_t14 = 1;
_t15 = 0;
_t16.x = _t14;
_t16.y = _t15;
dir = _t16;
_t17 = 60;
theta = _t17;
_t18 = 12;
i = _t18;
_L11:
_t19 = 0;
_t20 = i > _t19;
if (!_t20) { goto _L10; }
_t21 = 0.1;
_t22 = sqrt((dir.x * dir.x + dir.y * dir.y));
dir.x = dir.x/_t22;
dir.y = dir.y/_t22;
_t23.x = _t21 * dir.x + _ff.x;
_t23.y = _t21 * dir.y + _ff.y;
glBegin(GL_LINES);
        glColor3f(0.0, 0.0, 0.0);
        glVertex2f(_ff.x, _ff.y);
        glVertex2f(_t23.x , _t23.y);
glEnd();
_ff.x = _t23.x;
_ff.y = _t23.y;

_t24 = _ff;
_t25 = 60;
_t26 = cos(M_PI / 180 * _t25);
_t27 = dir.x;
_t28 = _t26 * _t27;
_t29 = 60;
_t30 = sin(M_PI / 180 * _t29);
_t31 = dir.y;
_t32 = _t30 * _t31;
_t33 = _t28 - _t32;
_t34 = 60;
_t35 = sin(M_PI / 180 * _t34);
_t36 = dir.x;
_t37 = _t35 * _t36;
_t38 = 60;
_t39 = cos(M_PI / 180 * _t38);
```

```
_t40 = dir.y;
_t41 = _t39 * _t40;
_t42 = _t37 + _t41;
_t43.x = _t33;
_t43.y = _t42;
dir = _t43;
_t44 = 0.1;
_t45 = sqrt((dir.x * dir.x + dir.y * dir.y));
dir.x = dir.x/_t45;
dir.y = dir.y/_t45;
_t46.x = _t44 * dir.x + _ff.x;
_t46.y = _t44 * dir.y + _ff.y;
glBegin(GL_LINES);
        glColor3f(0.0, 0.0, 0.0);
        glVertex2f(_ff.x, _ff.y);
        glVertex2f(_t46.x , _t46.y);
glEnd();
_ff.x = _t46.x;
_ff.y = _t46.y;

_t47 = _ff;
_t48 = -120;
_t49 = cos(M_PI / 180 * _t48);
_t50 = dir.x;
_t51 = _t49 * _t50;
_t52 = -120;
_t53 = sin(M_PI / 180 * _t52);
_t54 = dir.y;
_t55 = _t53 * _t54;
_t56 = _t51 - _t55;
_t57 = -120;
_t58 = sin(M_PI / 180 * _t57);
_t59 = dir.x;
_t60 = _t58 * _t59;
_t61 = -120;
_t62 = cos(M_PI / 180 * _t61);
_t63 = dir.y;
_t64 = _t62 * _t63;
_t65 = _t60 + _t64;
_t66.x = _t56;
_t66.y = _t65;
dir = _t66;
_t67 = 0.1;
_t68 = sqrt((dir.x * dir.x + dir.y * dir.y));
dir.x = dir.x/_t68;
dir.y = dir.y/_t68;
_t69.x = _t67 * dir.x + _ff.x;
_t69.y = _t67 * dir.y + _ff.y;
glBegin(GL_LINES);
        glColor3f(0.0, 0.0, 0.0);
        glVertex2f(_ff.x, _ff.y);
        glVertex2f(_t69.x , _t69.y);
glEnd();
_ff.x = _t69.x;
_ff.y = _t69.y;

_t70 = _ff;
_t71 = 60;
_t72 = cos(M_PI / 180 * _t71);
_t73 = dir.x;
```

```
_t74 = _t72 * _t73;
_t75 = 60;
_t76 = sin(M_PI / 180 * _t75);
_t77 = dir.y;
_t78 = _t76 * _t77;
_t79 = _t74 - _t78;
_t80 = 60;
_t81 = sin(M_PI / 180 * _t80);
_t82 = dir.x;
_t83 = _t81 * _t82;
_t84 = 60;
_t85 = cos(M_PI / 180 * _t84);
_t86 = dir.y;
_t87 = _t85 * _t86;
_t88 = _t83 + _t87;
_t89.x = _t79;
_t89.y = _t88;
dir = _t89;
_t90 = 0.1;
_t91 = sqrt((dir.x * dir.x + dir.y * dir.y));
dir.x = dir.x/_t91;
dir.y = dir.y/_t91;
_t92.x = _t90 * dir.x + _ff.x;
_t92.y = _t90 * dir.y + _ff.y;
glBegin(GL_LINES);
        glColor3f(0.0, 0.0, 0.0);
        glVertex2f(_ff.x, _ff.y);
        glVertex2f(_t92.x , _t92.y);
glEnd();
_ff.x = _t92.x;
_ff.y = _t92.y;

_t93 = _ff;
_t94 = 2;
_t95 = i / _t94;
_t96 = 2;
_t97 = _t95 * _t96;
_t98 = _t97 == i;
if (_t98) { goto _L20; }
_t99 = 60;
theta = _t99;
goto _L21;
_L20:
_t100 = -120;
theta = _t100;
_L21:
_t101 = cos(M_PI / 180 * theta);
_t102 = dir.x;
_t103 = _t101 * _t102;
_t104 = sin(M_PI / 180 * theta);
_t105 = dir.y;
_t106 = _t104 * _t105;
_t107 = _t103 - _t106;
_t108 = sin(M_PI / 180 * theta);
_t109 = dir.x;
_t110 = _t108 * _t109;
_t111 = cos(M_PI / 180 * theta);
_t112 = dir.y;
_t113 = _t111 * _t112;
_t114 = _t110 + _t113;
```

```
        _t115.x = _t107;
        _t115.y = _t114;
        dir = _t115;
        _t116 = 1;
        _t117 = i - _t116;
        i = _t117;
        goto _L11;
        _L10:


        return 0;
}
```

GLUT WINDOW for koch.ff:

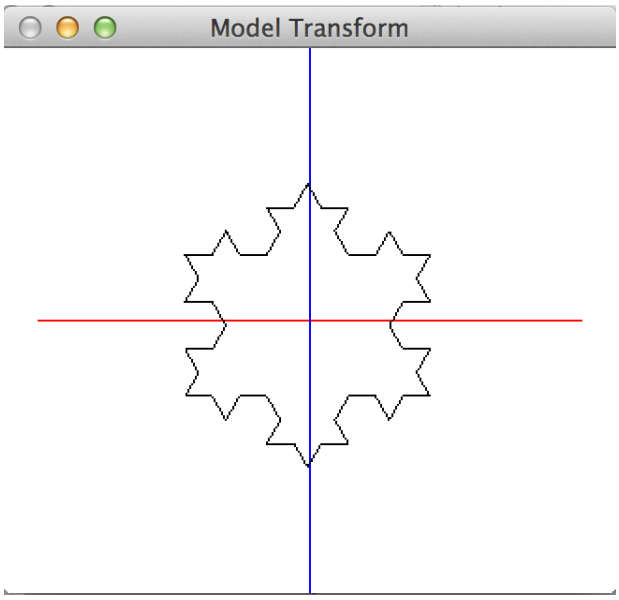

Figure 8: Glut Window Output of koch.ff

9.5 Fibonacci Stairs (Recursion example)

INPUT:
fibonacciStairs.ff

```
dir = [0.0,1.0]
dir2 = [1.0,0.0]
a = 1
let f(2) =
{
        if( $2 > 10.0){
                a = 0
        }
        else{
                $2 on dir
                0.1 on dir2
                f($2, $1+$2)
        }
        endif
}
```

```
f(0.0,0.05)
```

OUTPUT snippet from fibonacciStairs.cpp

```
int myprogram(){
        vec2cpp _ff = {0,0};
        stack <actRecord> fRecords;
        float lclVars[1024];
        int lv_index = 0;
        stack <int> lv_frameptr;
        stack <int> lv_lastframeptr;

        vec2cpp dir;
        vec2cpp dir2;
        int a;
        float _t0;
        float _t1;
        vec2cpp _t2;
        float _t3;
        float _t4;
        vec2cpp _t5;
        int _t6;
        float _t7;
        float _t8;
        bool _t9;
        float _t10;
        float _t11;
        vec2cpp _t12;
        vec2cpp _t13;
        float _t14;
        float _t15;
        vec2cpp _t16;
        vec2cpp _t17;
        actRecord _t18;
        float _t19;
        float _t20;
        float _t21;
        float _t22;
        int _t23;
        void * _t24;
        actRecord _t25;
        float _t26;
        float _t27;

        _t0 = 0.;
        _t1 = 1.;
        _t2.x = _t0;
        _t2.y = _t1;
        dir = _t2;
        _t3 = 1.;
        _t4 = 0.;
        _t5.x = _t3;
        _t5.y = _t4;
        dir2 = _t5;
        _t6 = 1;
        a = _t6;
        goto _L10;
        _Lf:
        _t7 = lclVars[lv_frameptr.top() - 2];
        _t8 = 10.;
```

```
_t9 = _t7 > _t8;
if (_t9) { goto _L20; }
_t10 = lclVars[lv_frameptr.top() - 2];
_t11 = sqrt((dir.x * dir.x + dir.y * dir.y));
dir.x = dir.x/_t11;
dir.y = dir.y/_t11;
_t12.x = _t10 * dir.x + _ff.x;
_t12.y = _t10 * dir.y + _ff.y;
glBegin(GL_LINES);
        glColor3f(0.0, 0.0, 0.0);
        glVertex2f(_ff.x, _ff.y);
        glVertex2f(_t12.x , _t12.y);
glEnd();
_ff.x = _t12.x;
_ff.y = _t12.y;

_t13 = _ff;
_t14 = 0.1;
_t15 = sqrt((dir2.x * dir2.x + dir2.y * dir2.y));
dir2.x = dir2.x/_t15;
dir2.y = dir2.y/_t15;
_t16.x = _t14 * dir2.x + _ff.x;
_t16.y = _t14 * dir2.y + _ff.y;
glBegin(GL_LINES);
        glColor3f(0.0, 0.0, 0.0);
        glVertex2f(_ff.x, _ff.y);
        glVertex2f(_t16.x , _t16.y);
glEnd();
_ff.x = _t16.x;
_ff.y = _t16.y;

_t17 = _ff;
_t18.retFunc = &&_L31;
fRecords.push(_t18);
_t19 = lclVars[lv_frameptr.top() - 1];
_t20 = lclVars[lv_frameptr.top() - 2];
_t21 = _t19 + _t20;
lclVars[lv_index] = (float)_t21;
lv_index = lv_index + 1;
_t22 = lclVars[lv_frameptr.top() - 2];
lclVars[lv_index] = (float)_t22;
lv_index = lv_index + 1;
lv_frameptr.push(lv_index);
goto _Lf;
_L31:
goto _L21;
_L20:
_t23 = 0;
a = _t23;
_L21:
lv_frameptr.pop();
_t24 = ((fRecords.top()).retFunc);
fRecords.pop();

goto *_t24;

_L10:
_t25.retFunc = &&_L41;
fRecords.push(_t25);
_t26 = 0.05;
```

```
        lclVars[lv_index] = (float)_t26;
        lv_index = lv_index + 1;
        _t27 = 0.;
        lclVars[lv_index] = (float)_t27;
        lv_index = lv_index + 1;
        lv_frameptr.push(lv_index);
        goto _Lf;
        _L41:


        return 0;
}
```

GLUT WINDOW for fibonacciStairs.ff:

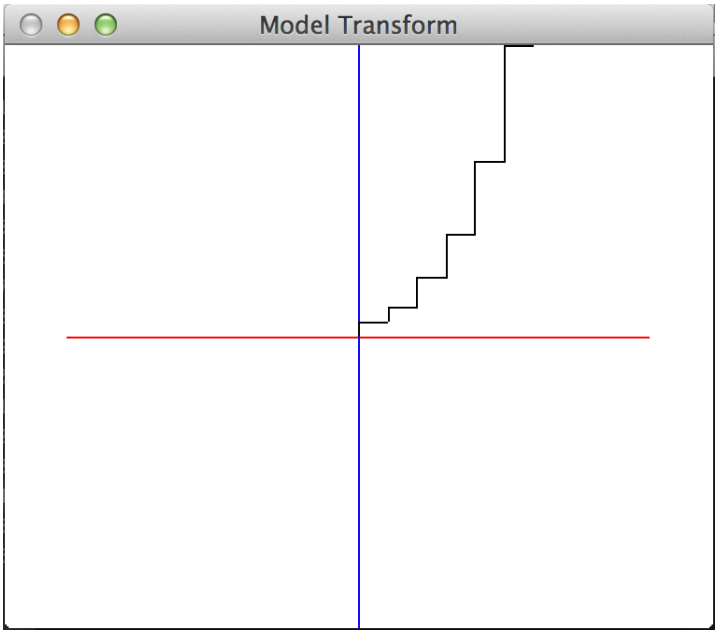

Figure 9: Glut Window Output of fibonacciStairs.ff

10 Lessons Learnt

10.1 Roy Aslan

1. Should have used homogenous C++ compiler from the beginning (we lost a lot of time due to this)
2. C++ scoping issues (due to not pre-declaring all variables at the top of the CPP program) should not have been band-aided. It caused delays and rework in the long run, and we ended up pre-declaring all variables at the end.
3. Should have utilized multiple passes. Most of our code generation occurs on the first pass. In hindsight, generating different elements across multiple passes would have probably made coding easier.
4. Should have clarified and documented priorities so that everyone would be on the same page of what is the next important thing to work on.

5. Should have read chapter 6 of the dragon book earlier. Semantic analysis and code generation took up a lot of time. Reading parts of chapter 6 after it was too late to change the architecture revealed things that we did suboptimally and things we did right for which the learning curve could have been shorter.

10.2 Prerna Chikersal

Initially, I was wary of using OCaml for this project, since I had never used a functional language before and it took multiple attempts to get the simplest of programs to compile in OCaml. However, towards the end of the project, I realised that OCaml is actually a very powerful tool for coding compilers from scratch. Had we used some other language like C++, we would have had to write much more code, making it nearly impossible for us to code a compiler from scratch in this short duration of time.

Along the course of the project, I also learnt that sitting in the same room with your teammates and coding really speeds up the process. In past group projects, my teammates and I only met for a few hours every week and finished different parts of the project on own accord. After this experience, I believe that working together is much more efficient. In fact, even working on different parts, while sitting together really helps.

I think our team's biggest shortcoming was time management, due to which we ended up implementing activation records for functions and recursion, which were the main features of our project, at the very end. I think we should have properly defined the scope of the project in the beginning and started coding a bit earlier.

I came to Columbia University from NTU, Singapore, for a summer exchange program. For me, this course is a final year "core" at NTU and I will be transferring the credits I earn back to NTU. Given the broad course content, the substantial project and the wonderful teammates I had, I believe this course has been a great learning experience and I am glad that I decided to take this course during my summer at Columbia.

10.3 Alex Shnayder

Besides the obvious "start early" lesson, there were several other lessons that I learned. First, I learned that having a specific design plan from the start would have helped avoid several problems down the road. Since our design plan was just high-level, we ended up tackling key problems (eg. passing information to and from the symbol table) only as they arrived, rather than with forethought, resulting in portions of our code that only worked for one situation, but not other situations.

Secondly, I learned that we should have put a greater focus on unit testing as our code base began to grow. Towards the end of our development lifecycle, in the interest of time, we conducted manual inspection of our intermediate representations when adding new features, rather than relying on unit tests to make sure these new features didn't break any pre-existing code. System testing helped with the final output, but not for the intermediate portions that sometimes behaved unexpectedly when making changes.

Thirdly, I learned the importance of becoming familiar with our subversioning system - Git - before actually using it. Using Git was a new experience for me, and my lack of familiarity with it led to occasional surprises that ultimately resulted in a general "paranoia" that I could

be overwriting something unexpectedly. This tangibly translated to slower progress, since there were times when I would hold off on making certain changes to a file that someone else was working on, out of fear that it would be too complicated to merge everything or restore any lost changes.

Finally, if I could go back in time and tell myself one piece of advice at the beginning of the course, it would be to read the Dragon book early on in the course. I only started reading the book halfway through the project, and frequently I would experience "A-ha!" moments after reading a section of the book that directly addressed a previous problem we had already faced. This could have helped us form a more concrete design plan from the get-go.

Appendix A

MakeFile, runCompiler.sh, ast.ml, scanner.mll, parser.mly, firefly.ml, compile.ml, semantics.ml, flatc.ml

MakeFile

```
OCAMLBUILD = ocamlbuild

all:
        $(OCAMLBUILD) -no-hygiene firefly.byte

clean:
        $(OCAMLBUILD) -clean
        rm -f firefly.byte
```

runCompiler.sh

```
#!/bin/sh

compileWin=0
compileMac=0
delete=0
help=0
silent=0
filename=""
dirname=""


Usage()
{
        echo "Usage: runCompiler.sh [OPTIONS] file.ff"
        echo ""
        echo "-c: Run C++ compiler on generated .cpp file, and then run the
            executable. (Windows only)"
        echo "-d: Delete all C++ files (*.cpp, *.obj, *.exe, etc.) related to
            source .ff file at the end of the script."
        echo "-h: Display help menu."
        echo "-p: Run C++ compiler on generated .cpp file, and then run the
            executable. (Mac OS X only)"
        echo "-s: Use with '-c' or '-p' option. This will suppress output from
            C++ compilation and execution."
        echo ""
        echo "This will run the Firefly compiler on file.ff, and generate an
            output cpp file."
}

DeleteCpp()
{
        rm -f $filename.cpp
        rm -f $filename.obj
        rm -f $filename.exe
        rm -f $filename.o
        rm -f $filename.out
        rm -f $filename
}

while getopts cdhps o
do
        case $o in
        c) # Run Windows C++ compiler and execute the executable file.
                compileWin=1
```

```
                ;;
        d) # Delete all C++ files related to source .ff file. Deletion happens
            at the end of the script.
                delete=1
                ;;
        h) # Show Usage information.
                help=1
                ;;
        p) # Run Mac OS X C++ compiler and execute the executable file.
                compileMac=1
                ;;
        s) # If the "-c" or "-p" option is used, suppress all output from C++
            compilation & execution.
                silent=1
                ;;
        esac
done

shift `expr $OPTIND - 1`

if [ $# -eq 1 ] # User supplied one argument, just as expected.
then

        filename=${1%.ff}
        dirname=${1%/*}

        if [ "$dirname" == "$1" ]
        then
                dirname="./"
        else
                :
        fi

        # Before compiling, delete any pre-existing C++ files (*.cpp, *.obj,
            etc.) related to source .ff file.
        DeleteCpp

        # Compile the source .ff file.
        if [ $silent = 1 ]
        then
                ./firefly.byte < $1 > nul
        else
                ./firefly.byte < $1
        fi
        mv output.cpp "$filename".cpp

        # If user chose "-c" option, run C++ compiler on Windows and execute
            the resulting .exe file.
        if [ $compileWin = 1 ]
        then
                if [ $silent = 1 ]
                then
                        g++ -o $filename -Wall $filename.cpp -mwindows
                            glut32.lib -lopengl32 -lglu32
                        ./$filename.exe
                        # start //B //WAIT cppCompile.bat "$filename" >nul
                            2>nul
                else
                        g++ -o $filename -Wall $filename.cpp -mwindows
                            glut32.lib -lopengl32 -lglu32
```

```
                        ./$filename.exe
                        # start //B //WAIT cppCompile.bat "$filename"
                fi
        else
                :
        fi

        if [ $compileMac = 1 ]
        then
                if [ $silent = 1 ] # Place code here to suppress C++
                    compilation & execution output. Useful for batch testing.
                then
                        g++ -w -o "$filename" "$filename".cpp -framework GLUT
                            -framework OpenGL
            ./"$filename"
                else
            g++ -w -o "$filename" "$filename".cpp -framework GLUT -framework
                OpenGL
            ./"$filename"
                fi
        else
                :
        fi

        # If user selected "-d" option, delete all C++ files (*.cpp, *.obj,
            etc.) related to source .ff file.
        if [ $delete = 1 ]
        then
                DeleteCpp
        else
                :
        fi

elif [ $# -eq 0 ] # User didn't supply any arguments. This is only allowed if
    they're just doing -h.
then
        if [ $help = 1 -a `expr $compileWin + $compileMac + $delete` = 0 ]
        then
                :
        else
                echo "Invalid number of arguments. There should be a single
                    Firefly source file supplied."
                echo ""
        fi

else # User supplied more than one argument. This is never allowed.
        echo "Invalid number of arguments. There should be a single Firefly
            source file supplied."
        echo ""
fi

# Show help menu.
if [ $help = 1 ]
then
        Usage
else
        :
fi
```

ast.ml

```
type op = Add | Minus | Multiply | Divide | LessThan | LessThanEq |
                        GreaterThan | GreaterThanEq | EqualsTo | NotEqualsTo |
                            On | Off |
                        Or | And

type constant =
        Integer of int
  | Float of float

type expr =
    Constant of constant
  | NegConstant of constant
  | Identifier of string (* can contain alphabets, numbers or underscores, but
     must begin with an alphabet*)
  | Vec2 of expr * expr
  | Binop of expr * op * expr
  | Assign of string * expr
  | Not of expr
  | Sqrt of expr
  | Sin of expr
  | Cos of expr
  | Getx of expr
  | Gety of expr
  | Local of int

type stmt =
        Expr of expr
  |     Block of stmt list
  | If of expr * stmt * stmt
  | While of expr * stmt
  | Call of string * (expr list)
  | Fdef of string * int * stmt

type stmts =
        stmt list

(*type fdef = *)


(*type fdefs =
        fdef list*)

type program =
        stmts
```

scanner.mll

```
{ open Parser }

let flt = ['0'-'9']+ ['.'] ['0'-'9']+

rule token = parse
  [' ' '\t' '\r' '\n']          { token lexbuf } (* Whitespace *)
| "/*"          {comment lexbuf}
| '+'           { PLUS }
| '-'           { MINUS }
| '*'           { TIMES }
| '/'           { DIVIDE }
| '('           { LPAREN }
| ')'           { RPAREN }
| '{'           { LBRACE }
```

```
| '}'           { RBRACE }
| '['           { OPENVEC }
| ','           { COMMA }
| ']'           { CLOSEVEC }
| '<'           { LESS }
| "<="          { LESSEQ }
| '>'           { GREATER }
| ">="          { GREATEREQ }
| "=="          { EQUALSTO }
| "!="          { NOTEQUALS }
| "&&"          { AND }
| "||"          { OR }
| '!'           { NOT }
| ['0'-'9']+ as lxm             { INTEGER(int_of_string lxm) }
| "on"                                          { ON }
| "off"                                         { OFF }
| "="                                           { ASSIGN }
| flt as lxm    { FLOAT(float_of_string lxm) }
| "if"          { IF }
| "else"        { ELSE }
| "while"       { WHILE }
| "let"         { LET }
| "endif"       { ENDIF }
| "sqrt"        { SQRT }
| "sin"         { SIN }
| "cos"         { COS }
| ".x"          { GETX }
| ".y"          { GETY }
| '$'           { LOCAL }
| eof { EOF }
| ['a'-'z' 'A'-'Z']+ ['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm {IDENTIFIER(lxm)}
| _ as char                                { raise (Failure("illegal character "
   ^ Char.escaped char)) }

and comment = parse
  "*/"          {token lexbuf}
| _                     {comment lexbuf}
```

parser.mly

```
%{ open Ast %}

%token <int> INTEGER
%token <float> FLOAT
%token LESS LESSEQ GREATER GREATEREQ EQUALSTO NOTEQUALS
%token AND OR NOT
%token LPAREN RPAREN LBRACE RBRACE
%token OPENVEC COMMA CLOSEVEC LET
%token SQRT SIN COS ASSIGN PLUS MINUS TIMES DIVIDE ON OFF IF ELSE WHILE ENDIF
%token EOF
%token <(float * float)> VEC2
%token GETX GETY
%token <string> IDENTIFIER
%token LOCAL

%left LPAREN RPAREN
%right SIN COS
%right SQRT
%right ASSIGN
%left OR
%left AND
```

```
%right NOT
%nonassoc LESS LESSEQ GREATER GREATEREQ EQUALSTO NOTEQUALS
%right ON OFF
%left OPENVEC COMMA CLOSEVEC
%left PLUS MINUS
%left TIMES DIVIDE
%nonassoc GETX GETY
%nonassoc UMINUS UPLUS

%start program
%type <Ast.program> program

%%

program:
   /* nothing */ { [] }
 | program stmt { ($2 :: $1) }

stmts:
         /* nothing */                  { [] }
        | stmts stmt                    { ($2 :: $1) }

stmt:
        expr_stmt                               { Expr($1) }
  |     LBRACE stmts RBRACE             { Block(List.rev $2) }
  | IF expr stmt ELSE stmt ENDIF        { If($2, $3, $5) }
  | WHILE expr stmt                             { While($2, $3) }
  | IDENTIFIER LPAREN actuals_opt RPAREN        { Call ($1, $3) }
  | LET IDENTIFIER LPAREN INTEGER RPAREN ASSIGN stmt    { Fdef($2, $4, $7) }

actuals_opt:
    /* nothing */                       { [] }
  | actuals_list                        { $1 }

actuals_list:
    expr                { [$1] }
  | actuals_list COMMA expr { $3 :: $1 }

vec2:
        OPENVEC expr COMMA expr CLOSEVEC        { Vec2($2,$4) }

expr_stmt:
        IDENTIFIER ASSIGN expr  { Assign ($1, $3)}
  | expr ASSIGN expr                    { raise (Failure ("Cannot have an
     expression on LHS of assignment!" ) ) }
  | expr ON expr                { Binop($1, On,   $3) }
  | expr OFF expr       { Binop($1, Off,   $3) }

expr:
        constant                                        { Constant($1) }
  | expr_stmt                                   { $1 }
  | vec2                                                { $1 }
  | IDENTIFIER                                  { Identifier($1) }
  | expr PLUS expr                              { Binop($1, Add,   $3) }
  | expr MINUS expr                             { Binop($1, Minus, $3) }
  | expr TIMES expr                             { Binop($1, Multiply, $3) }
  | expr DIVIDE expr                            { Binop($1, Divide, $3) }
  | expr LESS expr                              { Binop($1, LessThan, $3) }
  | expr LESSEQ expr                    { Binop($1, LessThanEq, $3) }
  | expr GREATER expr                   { Binop($1, GreaterThan, $3) }
```

```
  | expr GREATEREQ expr         { Binop($1, GreaterThanEq, $3) }
  | expr EQUALSTO expr                { Binop($1, EqualsTo, $3) }
  | expr NOTEQUALS expr         { Binop($1, NotEqualsTo, $3) }
  | expr OR expr                              { Binop($1, Or, $3) }
  | expr AND expr                             { Binop($1, And, $3) }
  | NOT expr                                  { Not($2) }
  | PLUS constant       %prec UPLUS           { Constant($2) }
  | MINUS constant      %prec UMINUS    { NegConstant($2) }
  | LPAREN expr RPAREN          { $2 }
  | SQRT expr                                 { Sqrt($2) }
  | SIN expr                                  { Sin($2) }
  | COS expr                                  { Cos($2) }
  | expr GETX                                 { Getx($1) }
  | expr GETY                                 { Gety($1) }
  | LOCAL INTEGER                             { Local($2) }

constant:
        INTEGER                               { Integer($1) }
  |     FLOAT                                 { Float($1)     }
```

compile.ml

```
open Ast
open Printf
open Semantics
open Flatc
open Stack

(*global vars management *)
let globals_index = ref (0)
let globals = Array.make 1024 ("", "")

(*functions management *)
let fun_index = ref (0)
let funs = Array.make 1024 ( "", 0 )

(* temp vars management *)
let tvar_index = ref (0)
let tempvars = Array.make 1024 ("", "")

(* Label managment *)
let lbl_index = ref (0)
let lblStack = Stack.create ()
let a = Stack.push !lbl_index lblStack

(* CPP output files *)
let file_head = "input_head.cpp"
let file = "output.cpp"
let oc_init = open_out file
let file_decs = "output_decs.cpp"
let oc_decs = open_out file_decs
let file_TAC = "output_TAC.cpp"
let oc_TAC = open_out file_TAC

let read_file file =
  let ic = open_in file in
  let lines = ref [] in
  try
    while true do
      let line = input_line ic in
      lines := line :: !lines
```

```
    done; assert false
  with End_of_file ->
    String.concat "\n" (List.rev !lines)

let closeCppFile = function
        _ ->
                        fprintf oc_TAC "%s\n\n" "\n\n
        return 0;
}";
                        close_out oc_TAC

let type_to_string = function
          Constant(x) ->
                (match x with
                          Integer(x) -> "int"
                        | Float(x) ->   "float")
        | NegConstant(x) ->
                (match x with
                          Integer(x) -> "int"
                        | Float(x) ->   "float")
        | Vec2(x,y) ->  "vec2cpp"
        | _     -> "InvalidType"

let rec gen_expr = function
        Constant(x) ->
                ( match x with
                        Integer(x) -> [(Int(x),string_of_int x,"int")]
                  | Float(x)   -> [(Flt(x),string_of_float x,"float")] )
  | NegConstant(x) ->
                ( match x with
                        Integer(x) -> [(Int(-x),string_of_int (-1 * x),"int")]
                  | Float(x)   -> [(Flt(-1.0 *. x),string_of_float (-1.0 *.
                      x),"float")] )
  | Vec2(x, y) -> gen_expr (x) @ gen_expr (y) @ [(Vec2_Op,"VEC","vec2cpp")]
  | Binop (e1, op, e2) -> let v1 = gen_expr e1 and v2 = gen_expr e2 in
                ( match op with
                        On -> v1 @ v2 @ [(On_Op,"ON","vec2cpp")]
                  | Off -> v1 @ v2 @ [(Off_Op,"OFF","vec2cpp")]
                  | Add -> v1 @ v2 @ [(Add_Op,"ADD","TypeToInfer")]
                  | Minus -> v1 @ v2 @ [(Minus_Op,"MINUS","TypeToInfer")]
                  | Multiply -> v1 @ v2 @
                      [(Multiply_Op,"MULTIPLY","TypeToInfer")]
                  | Divide -> v1 @ v2 @ [(Divide_Op,"DIVIDE","TypeToInfer")]
                  | LessThan -> v1 @ v2 @ [(LessThan_Op,"LESSTHAN","bool")]
                  | LessThanEq -> v1 @ v2 @
                      [(LessThanEq_Op,"LESSTHANEQ","bool")]
                  | GreaterThan -> v1 @ v2 @
                      [(GreaterThan_Op,"GREATERTHAN","bool")]
                  | GreaterThanEq -> v1 @ v2 @
                      [(GreaterThanEq_Op,"GREATERTHANEQ","bool")]
                  | EqualsTo -> v1 @ v2 @ [(EqualsTo_Op,"EQUALSTO","bool")]
                  | NotEqualsTo -> v1 @ v2 @
                      [(NotEqualsTo_Op,"NOTEQUALSTO","bool")]
                  | Or ->               lbl_index := !lbl_index + 10;
                                        Stack.push !lbl_index lblStack;
                                        let li = Stack.top lblStack in
                                        v1 @ [(Or_Op(li),"OR " ^
                                            string_of_int(li),"bool")] @ v2
                                        @ [(Lbl(li), "LBL " ^
                                            string_of_int(li), "void")]
```

```
                                    @ [(OrDone_Op,"ORDONE","bool")]
                  | And ->          lbl_index := !lbl_index + 10;
                                    Stack.push !lbl_index lblStack;
                                    let li = Stack.top lblStack in
                                    v1 @ [(And_Op(li),"AND " ^
                                        string_of_int(li),"bool")] @ v2
                                    @ [(Lbl(li), "LBL " ^
                                        string_of_int(li), "void")]
                                    @ [(AndDone_Op,"ANDDONE","bool")]
               )
  | Identifier(x) -> [(Id(x),"ID(" ^ x ^ ")","TypeToInfer")]
  | Assign(v,e) ->      gen_expr e @ gen_expr (Identifier(v)) @
    [(Asn_Op,"Asn","TypeToInfer")]
  | Not(e)      ->      gen_expr e @ [(Not_Op,"NOT","bool")]
  | Sqrt(e) -> gen_expr e @ [(Sqrt,"SQRT","TypeToInfer")]
  | Sin(e)      -> gen_expr e @ [(Sin_Op,"SIN","float")]
  | Cos(e)      -> gen_expr e @ [(Cos_Op,"COS","float")]
  | Getx(e)     -> gen_expr e @ [(Getx_Op,"GETX","float")]
  | Gety(e)     -> gen_expr e @ [(Gety_Op,"GETY","float")]
  | Local(i)-> [(GetLocal(i), "GET LOC " ^ string_of_int i, "float")]
  | _ -> []

let rec gen_stmt = function
        Expr e  ->      gen_expr e
  | If(e, ts, fs)       ->      lbl_index := !lbl_index + 10;
                                Stack.push !lbl_index lblStack;
                                let li = Stack.top lblStack in
                                gen_expr e
                                @ [(If_Op(li), "IF " ^ string_of_int(li),
                                    "bool")]
                                @ gen_stmt fs
                                @ [(Goto(li + 1), "GOTO " ^ string_of_int(li +
                                    1), "void")]
                                @ [(Lbl(li), "LBL " ^ string_of_int(li),
                                    "void")]
                                @ gen_stmt ts @ [(EndIf_Op, "ENDIF", "bool")]
                                @ [(Lbl(li + 1), "LBL " ^ string_of_int(li +
                                    1), "void")]
  | While(e, ts)        ->      lbl_index := !lbl_index + 10;
                                Stack.push !lbl_index lblStack;
                                let li = Stack.top lblStack in
                                [(Lbl(li + 1), "LBL " ^ string_of_int(li + 1),
                                    "void")]
                                @ gen_expr e
                                @ [(While_Op(li), "WHILE " ^
                                    string_of_int(li), "bool")]
                                @ gen_stmt ts
                                @ [(Goto(li + 1), "GOTO " ^ string_of_int(li +
                                    1), "void")]
                                @ [(Lbl(li), "LBL " ^ string_of_int(li),
                                    "void")]
                                @ [(EndWhile_Op, "ENDWHILE", "bool")]
  | Block(stmts)        ->      List.concat (List.map gen_stmt stmts)
  | Call(fn, args)      ->      lbl_index := !lbl_index + 10;
                                let argcount = List.length args in
                                let li = !lbl_index in
                                [SetReturn(li+1), "SET RET " ^ fn, "void"]
                                @       ( List.concat
                                          ( List.map
                                             (
```

```
                                                fun e -> (gen_expr e) @
                                                    [(SetLocal(fn), "SET LOC "
                                                    ^ fn, "void")]
                                            )  args
                                        )
                                    )
                                @ [(GotoFun(fn, argcount), "GOTO FUN " ^ fn,
                                    "void")]
                                @ [(Lbl(li+1), "LBL " ^ string_of_int(li+1),
                                    "void")]
  |     Fdef(fname, argcount, body)     ->
                                funs.(!fun_index) <- (fname, argcount);
                                fun_index := !fun_index + 1;
                                lbl_index := !lbl_index + 10;
                                let li = !lbl_index in
                                [(Goto(li), "GOTO " ^ string_of_int(li),
                                    "void")]
                                @ [(Flbl(fname), "FLBL " ^ fname, "void")]
                                @ (gen_stmt body)
                                @ [(GotoReturn, "GOTO RET ", "void")]
                                @ [(Lbl(li), "LBL " ^ string_of_int(li),
                                    "void")]
                                @ [(Endfdef, "ENDFDEF " ^ string_of_int(li),
                                    "bool")]

(* helper function to print array values as c++ variable declarations *)
let rec array_to_file g c i fl =
        (
                if i < c then (
                        let varname s = String.sub (s) (3) ((String.length s)
                            -4) in
                                fprintf fl "\n\t%s" ((snd g.(i)) ^ " " ^
                                    (varname (fst g.(i))) ^ ";");
                        array_to_file g c (i+1) fl
                )
                else ()
        )

(* for each statement, this function runs semantic analysis and generates flat
    C (TAC-like) code *)
let print_gen x = match x with
        _       ->      let syntaxtree = gen_stmt x (* build AST *)
                        in let sast = sa syntaxtree globals globals_index funs
                            fun_index (* build SAST via semantics.ml *)
                        in      generate_c sast tvar_index lbl_index oc_TAC
                            globals !globals_index tempvars; (* build flat C++
                            via flatc.ml *)
                        print_endline ""

let translate = function
        stmts   ->
                        fprintf oc_init "\n\t%s" (read_file file_head);
                        List.iter print_gen (List.rev stmts);
                        closeCppFile();
                        array_to_file globals !globals_index 0 oc_decs;
                        array_to_file tempvars (!temp_counter) 0 oc_decs;
                        close_out oc_decs;
                        fprintf oc_init "\n\t%s" ( (read_file file_decs) ^
                            "\n" ^ (read_file file_TAC) );
                        close_out oc_init
```

semantics.ml

```
open Stack

(* building blocks of SAST *)
type element =
                Asn_Op
        |       DAsn_Op
        |       Int of int
        |   Flt of float
        |       Id of string
        |       Vec2_Op
        |       Add_Op
        |       Minus_Op
        |       Multiply_Op
        |       Divide_Op
        |       On_Op
        |       Off_Op
        |       LessThan_Op
        |       LessThanEq_Op
        |       GreaterThan_Op
        |       GreaterThanEq_Op
        |       EqualsTo_Op
        |       NotEqualsTo_Op
        |       Or_Op of int
        |       OrDone_Op
        |       And_Op of int
        |       AndDone_Op
        |       Not_Op
        |       If_Op of int
        |       EndIf_Op
        |       Goto of int
        |       GotoFun of string * int
        |       GotoReturn
        |       Lbl of int
        |       Flbl of string
        |       While_Op of int
        |       EndWhile_Op
        |   Sqrt
        |       Sin_Op
        |       Cos_Op
        |       Getx_Op
        |       Gety_Op
        |   Endfdef
        |       SetReturn of int
        |       SetLocal of string
        |       GetLocal of int

let tempStack = Stack.create ()
let a = Stack.push (Int(1),"one","int") tempStack
let a = Stack.pop tempStack

let semStack = Stack.create ()
let a = Stack.push (Int(1),"one","int") semStack
let a = Stack.pop semStack

let frst (x,y,z) = x
let scnd (x,y,z) = y
let thrd (x,y,z) = z

(* This function takes a tuple; if the tuple refers to an undeclared var,
```

```
    throw an error.
Otherwise, return the tuple. *)
let checkUndeclaredVar v =
(
        match (frst v) with
                Id(_) when (thrd v) = "TypeToInfer" ->
                        raise ( Failure ("Variable " ^ (String.sub (scnd v)
                            (3) ((String.length (scnd v)) -4)) ^ " is
                            undeclared!") ) | _ -> ();
        v
)

(* perform semantic analysis on each syntax element *)
let evalTuple (x,y,z) g i f fi = (match x with
                Int(v) ->  Stack.push (x,y,z) (tempStack);Stack.push (x,y,z)
                    (semStack)
        |       Flt(v) -> Stack.push (x,y,z) (tempStack);Stack.push (x,y,z)
            (semStack)
        |       Add_Op  | Minus_Op | Multiply_Op | Divide_Op ->
                        let t1 = checkUndeclaredVar (Stack.pop tempStack)
                        and t2 = checkUndeclaredVar (Stack.pop tempStack) in
                        (
                                let v1 = (thrd t1) and v2 = (thrd t2) in
                                (
                                        if (v1 <> v2) && (v1 <> "pointer") &&
                                            (v2 <> "pointer") then
                                        (
                                                raise ( Failure ("Type
                                                    mismatch: " ^ (v1) ^ " and
                                                    " ^ (v2)) )
                                        )
                                        else
                                        (
                                                if (not (v1 = "vec2cpp" or v1
                                                    = "int" or v1 = "float" or
                                                    v1 = "pointer")) then
                                                (
                                                        raise ( Failure
                                                            ("Invalid type: " ^
                                                            (v1) ^ ":
                                                            Arithmetic operator
                                                            must take int,
                                                            float or vec2"))
                                                )
                                                else
                                                (
                                                        Stack.push (x,y,v1)
                                                            tempStack
                                                        ;Stack.push (x,y,v1)
                                                            semStack
                                                )
                                        )
                                )
                        )
        |       LessThan_Op     | LessThanEq_Op | GreaterThan_Op |
            GreaterThanEq_Op |
                 EqualsTo_Op | NotEqualsTo_Op   ->
                                let t1 = checkUndeclaredVar (Stack.pop
                                    tempStack)
                                and t2 = checkUndeclaredVar (Stack.pop
```

```
                                tempStack) in
                            (
                                    let v1 = (thrd t1) and v2 = (thrd t2)
                                        in
                                    (
                                            if (v1 <> v2) && (v1 <>
                                                "pointer") && (v2 <>
                                                "pointer") then
                                            (
                                                    raise ( Failure ("Type
                                                        mismatch: " ^ (v1)
                                                        ^ " and " ^ (v2)) )
                                            )
                                            else
                                            (
                                                    if (not (v1 = "int" or
                                                        v1 = "float")) then
                                                    (
                                                            raise (
                                                                Failure
                                                                ("Invalid
                                                                type: " ^
                                                                (v1) ^ ":
                                                                Relational
                                                                operator
                                                                must take
                                                                int or
                                                                float."))
                                                    )
                                                    else
                                                    (
                                                            Stack.push
                                                                (x,y,z)
                                                                tempStack
                                                            ;Stack.push
                                                                (x,y,z)
                                                                semStack
                                                    )
                                            )
                                    )
                            )
    |       OrDone_Op | AndDone_Op  ->
                            let t1 = checkUndeclaredVar (Stack.pop
                                tempStack) and tNone = Stack.pop tempStack
                            and t2 = checkUndeclaredVar (Stack.pop
                                tempStack)
                            in
                            (
                            let v1 = (thrd t1) and v2 = (thrd t2) in
                            (
                                    if (v2 <> "bool") then
                                    (
                                            if (x = OrDone_Op) then
                                                    raise ( Failure
                                                        ("Invalid type: " ^
                                                        v2 ^ "; left-hand
                                                        operand of || must
                                                        be bool"))
                                            else
                                                    raise ( Failure
```

```
                                                ("Invalid type: " ^
                                                v2 ^ "; left-hand
                                                operand of && must
                                                be bool"))
                                        );

                                        if (v1 <> "bool") then
                                        (
                                                if (x = OrDone_Op) then
                                                        raise ( Failure
                                                            ("Invalid type: " ^
                                                            v1 ^ "; right-hand
                                                            operand of || must
                                                            be bool"))
                                                else
                                                        raise ( Failure
                                                            ("Invalid type: " ^
                                                            v1 ^ "; right-hand
                                                            operand of && must
                                                            be bool"))
                                        );

                                        Stack.push (x,y,z) tempStack
                                        ;Stack.push (x,y,z) semStack
                                )
                        )
        |       Or_Op(i) | And_Op(i) -> Stack.push (x,y,z) tempStack;
            Stack.push (x,y,z) semStack
        |       Not_Op  ->
                        let t1 = checkUndeclaredVar (Stack.pop tempStack) in
                        (
                                let v1 = (thrd t1) in
                                (
                                        if (v1 <> "bool") then
                                        (
                                                raise ( Failure ("Invalid
                                                    type: " ^ v1 ^ "; NOT must
                                                    be applied to a bool.") )
                                        )
                                        else
                                        (
                                                Stack.push (x,y,z) tempStack
                                                ;Stack.push (x,y,z) semStack
                                        )
                                )
                        )
        |   Sqrt ->
                        let t1 = checkUndeclaredVar (Stack.pop tempStack) in
                        (
                                let v1 = (thrd t1) in
                                (
                                        if (v1 <> "int" && v1 <> "float") then
                                        (
                                                raise ( Failure ("Invalid
                                                    type: " ^ v1 ^ "; Sqrt can
                                                    only be applied to an int
                                                    or float.") )
                                        )
                                        else
                                        (
```

```
                                        Stack.push (x,y,v1) tempStack
                                        ;Stack.push (x,y,v1) semStack
                                )
                        )
                )
        |   Sin_Op ->
                        let t1 = checkUndeclaredVar (Stack.pop tempStack) in
                        (
                                let v1 = (thrd t1) in
                                (
                                        if (v1 <> "int" && v1 <> "float") then
                                        (
                                                raise ( Failure ("Invalid
                                                    type: " ^ v1 ^ "; Sqrt can
                                                    only be applied to an int
                                                    or float.") )
                                        )
                                                else
                                        (
                                                Stack.push (x,y,z) tempStack
                                                ;Stack.push (x,y,z) semStack
                                        )
                                )
                        )
        |   Cos_Op ->
                let t1 = checkUndeclaredVar (Stack.pop tempStack) in
                (
                        let v1 = (thrd t1) in
                        (
                                if (v1 <> "int" && v1 <> "float") then
                                (
                                        raise ( Failure ("Invalid type: " ^ v1
                                            ^ "; Sqrt can only be applied to an
                                            int or float.") )
                                )
                                else
                                (
                                        Stack.push (x,y,z) tempStack
                                        ;Stack.push (x,y,z) semStack
                                )
                        )
                )
        |   Getx_Op ->
                let t1 = checkUndeclaredVar (Stack.pop tempStack) in
                (
                        let v1 = (thrd t1) in
                        (
                                if (v1 <> "vec2cpp") then
                                (
                                        raise ( Failure ("Invalid type: " ^ v1
                                            ^ "; You can only get x-element of
                                            a vec2.") )
                                )
                                else
                                (
                                        Stack.push (x,y,z) tempStack
                                        ;Stack.push (x,y,z) semStack
                                )
                        )
                )
```

```
        |   Gety_Op ->
                let t1 = checkUndeclaredVar (Stack.pop tempStack) in
                (
                        let v1 = (thrd t1) in
                        (
                                if (v1 <> "vec2cpp") then
                                (
                                        raise ( Failure ("Invalid type: " ^ v1
                                            ^ "; You can only get y-element of
                                            a vec2.") )
                                )
                                else
                                (
                                        Stack.push (x,y,z) tempStack
                                        ;Stack.push (x,y,z) semStack
                                )
                        )
                )
        |       Id(v)    ->
                        if List.exists (fun s -> (fst s) = "ID(" ^ v ^ ")")
                            (Array.to_list g)
                        then
                        (
                                let f = List.find (fun s -> (fst s) = "ID(" ^
                                    v ^ ")") (Array.to_list g) in
                                Stack.push (x,y,(snd f)) tempStack
                                ;Stack.push (x,y,(snd f)) semStack (*
                                    temporary add!! *)
                                (* ^ We can add ID to semantic stack because
                                    we already know its type in this case. *)
                        )
                        else
                        (
                                Stack.push (x,y,z) tempStack
                                (* We do NOT add ID to semantic stack because
                                    we don't know its type in this case. *)
                        )
        |       Asn_Op  ->
                        let v = Stack.pop tempStack and e =
                            checkUndeclaredVar(Stack.pop tempStack) in
                        (
                                if ((thrd v) = "TypeToInfer") then (*
                                    Declaration of new var *)
                                (
                                        g.(!i) <- ((scnd v), (thrd e));
                                        i := !i + 1;
                                        Stack.push (DAsn_Op, "DAsn", (thrd e))
                                            tempStack
                                        ;Stack.push (frst v, scnd v, thrd e)
                                            semStack
                                        ;Stack.push (DAsn_Op, "DAsn", (thrd
                                            e)) semStack
                                )
                                else (* Assignment of existing var *)
                                (
                                        if ((thrd v) <> (thrd e)) && (thrd v
                                            <> "pointer") && (thrd e <>
                                            "pointer") then
                                        (
                                                raise ( Failure ("Type
```

```
                                        mismatch: " ^ (scnd v) ^ "
                                        - assigning " ^ (thrd e) ^
                                        " to " ^ (thrd v)))
                                )
                                else
                                (
                                        Stack.push (x,y,(thrd v))
                                            tempStack;
                                        Stack.push (x,y,(thrd v))
                                            semStack (* Temporary
                                            add!!! *)
                                )
                        )
                )
        |       Vec2_Op ->      let t1 = checkUndeclaredVar(Stack.pop
            tempStack)
                                and t2 = checkUndeclaredVar(Stack.pop
                                    tempStack) in
                                (
                                let v1 = (thrd t1) and v2 = (thrd t2)
                                    in
                                (
                                        if (v1 <> v2) && (v1 <>
                                            "pointer") && (v2 <>
                                            "pointer") then
                                        (
                                                raise ( Failure ("Type
                                                    mismatch: " ^ (v1)
                                                    ^ " and " ^ (v2)) )
                                        )
                                        else
                                        (
                                                Stack.push
                                                    (x,y,"vec2cpp")
                                                    tempStack
                                                ;Stack.push
                                                    (x,y,"vec2cpp")
                                                    semStack
                                        )
                                )
                        )
        |       On_Op   ->      let t1 = checkUndeclaredVar(Stack.pop
            tempStack)
                                and t2 = checkUndeclaredVar(Stack.pop
                                    tempStack) in
                                (
                                        let v1 = (thrd t1) and v2 =
                                            (thrd t2) in
                                        (
                                                if (v2 <> "float") &&
                                                    (v2 <> "pointer")
                                                    then
                                                (
                                                        raise (
                                                            Failure
                                                            ("Invalid
                                                            type: " ^
                                                            v2 ^ ";
                                                            left-hand
                                                            operand of
```

```
                                                        ON must be
                                                        float"))
                                                );

                                                if (v1 <> "vec2cpp")
                                                    then
                                                (
                                                        raise (
                                                            Failure
                                                            ("Invalid
                                                            type: " ^
                                                            v1 ^ ";
                                                            right-hand
                                                            operand of
                                                            ON must be
                                                            vec2"))
                                                );
                                                Stack.push (x,y,v1)
                                                    tempStack
                                                ;Stack.push (x,y,v1)
                                                    semStack
                                        )
                                    )
    |   Off_Op  ->  let t1 = checkUndeclaredVar(Stack.pop tempStack)
                                    and t2 = checkUndeclaredVar(Stack.pop
                                        tempStack) in
                                    (
                                            let v1 = (thrd t1) and v2 =
                                                (thrd t2) in
                                            (
                                                    if (v2 <> "float") &&
                                                        (v2 <> "pointer")
                                                        then
                                                    (
                                                            raise (
                                                                Failure
                                                                ("Invalid
                                                                type: " ^
                                                                v2 ^ ";
                                                                left-hand
                                                                operand of
                                                                OFF must be
                                                                float"))
                                                    );

                                                    if (v1 <> "vec2cpp")
                                                        then
                                                    (
                                                            raise (
                                                                Failure
                                                                ("Invalid
                                                                type: " ^
                                                                v1 ^ ";
                                                                right-hand
                                                                operand of
                                                                OFF must be
                                                                vec2"))
                                                    );
                                                    Stack.push (x,y,v1)
                                                        tempStack
```

```
                                                ;Stack.push (x,y,v1)
                                                    semStack
                                        )
                                    )
    |       If_Op(i)->      let t1 = checkUndeclaredVar(Stack.pop
        tempStack) in
                                        let v1 = (thrd t1) in
                                                if (v1 <> "bool") then
                                                (
                                                        raise (
                                                            Failure
                                                            ("Invalid
                                                            type: " ^
                                                            v1 ^ ";
                                                            conditional
                                                            expression
                                                            for IF must
                                                            be of type
                                                            bool."))
                                                )
                                                else
                                                (
                                                        Stack.push
                                                            (x,y,z)
                                                            semStack;
                                                )
    |       EndIf_Op        ->      Stack.push (x,y,z) semStack;
    |       While_Op(i)->   let t1 = checkUndeclaredVar(Stack.pop
        tempStack) in
                                        let v1 = (thrd t1) in
                                                if (v1 <> "bool") then
                                                (
                                                        raise (
                                                            Failure
                                                            ("Invalid
                                                            type: " ^
                                                            v1 ^ ";
                                                            conditional
                                                            expression
                                                            for WHILE
                                                            must be of
                                                            type
                                                            bool."))
                                                )
                                                else
                                                (
                                                        Stack.push
                                                            (x,y,z)
                                                            semStack;
                                                )
    |       EndWhile_Op     ->      Stack.push (x,y,z) semStack;
    |       Endfdef ->      Stack.push (x,y,z) semStack;
    |       Goto(i) ->      Stack.push (x, y, z) semStack;
    |       Goto(s) ->      Stack.push (x, y, z) semStack;
    |       GotoFun(fn, argcount) ->        (* Make sure function has been
        defined *)
                                        if List.exists (fun s -> (fst
                                            s) = fn) (Array.to_list f)
                                            then
                                        (
```

```
                                                    let m = List.find (fun
                                                        s -> (fst s) = fn)
                                                        (Array.to_list f) in
                                                    (* Make sure function
                                                        called with right
                                                        number of arguments
                                                        *)
                                                    if snd m <> argcount
                                                        then
                                                            raise (
                                                                Failure
                                                                ("Invalid
                                                                number of
                                                                arguments
                                                                for " ^ fn)
                                                                )
                                                    else Stack.push (x, y,
                                                        "void") semStack
                                            )
                                            else
                                            (
                                                    raise ( Failure
                                                        ("Calling undefined
                                                        function: " ^ fn))
                                            )
        |       GotoReturn      ->      Stack.push (x, y, z) semStack;
        |       Lbl(i)  ->      Stack.push (x, y, z) semStack;
        |       Flbl(s) ->      Stack.push (x, y, z) semStack;
        |       SetReturn(i)    ->      Stack.push (x, y, z) semStack;
        |       SetLocal(fn)    ->      Stack.push (x, y, z) semStack;
        |       GetLocal(i)             ->      Stack.push (x, y, z) tempStack;
                                                    Stack.push (x, y, z)
                                                        semStack;
        |       _ -> ()
        )

let sa lst g i f fi =   Stack.clear tempStack;
                                            Stack.clear semStack;

                                            (* Printing Syntactic Stack *)
                                            print_endline
                                                "*****************";
                                            print_endline "SYNTACTIC
                                                STACK";
                                            print_endline
                                                "*****************";
                                            List.iter (fun (fs, sn, thr) ->
                                                    print_endline (" (" ^
                                                        sn ^ "," ^ thr ^
                                                        ")")) lst;
                                            List.iter (fun (x) ->
                                                evalTuple x g i f fi) lst;

                                            (* Convert SAST stack to SAST
                                                list *)
                                            let rec buildSemList (l) =
                                                    if Stack.is_empty
                                                        semStack then
                                                            l
                                                    else
```

```
                                                    (
                                                            buildSemList
                                                                ((Stack.pop
                                                                semStack)
                                                                ::  l)
                                                    )
                                            in
                                            buildSemList []
```

flatc.ml

```
open Semantics
open Stack
open Printf

(*let file = "output.cpp"
let oc = open_out file*)

let tvarTbl = Hashtbl.create 100
let a = Hashtbl.add tvarTbl 1 "int"
let a = Hashtbl.clear tvarTbl

let getTempType index =
(
        if Hashtbl.mem tvarTbl index then
                Hashtbl.find tvarTbl index
        else
                ""
)

let stck = Stack.create ()
let a = Stack.push ("hey") stck
let a = Stack.pop stck

(*we should delete this if not needed --RA *)
let lstck = Stack.create ()
let a = Stack.push (-1) lstck
(*let a = Stack.pop lstck*)

let temp_counter = ref(0)
(*we should delete this if not needed --RA *)
let lbl_counter = ref(0)

(*let rec globals_to_file g c i oc =
        if i < c then (
                fprintf oc "\n\t%s" ((snd g.(i))^" "^(fst g.(i))^";\n");
                globals_to_file g c (i+1) oc
        )
        else ()*)

(*let firefly = ref (0.0,0.0) (* The compiler keeps a track of the firefly's
    position *) *)

(* We assume that a variable called _firefly is declared the initcpp code *)

let c_statement (x, y, z) ti li oc tvars = let tempType =
    getTempType(!temp_counter) in
        let typePrefix = match tempType with "" -> z | _ -> "" in
        let ti = "_t" ^ string_of_int(!temp_counter) in
        let ti_plus1 = "_t" ^ string_of_int(!temp_counter + 1) in
        let ti_plus2 = "_t" ^ string_of_int(!temp_counter + 2) in
```

```
let id_ti = "ID(" ^ ti ^ ")" in
let id_ti_plus1 = "ID(" ^ ti_plus1 ^ ")" in
let id_ti_plus2 = "ID(" ^ ti_plus2 ^ ")" in
match x with
        Int(v)  ->  tvars.(!temp_counter) <- (id_ti, typePrefix);
                                fprintf oc "\n\t%s" (ti ^ " = " ^
                                    string_of_int(v) ^ ";");
                                Stack.push
                                    ("_t"^string_of_int(!temp_counter))
                                    stck;
                                temp_counter := !temp_counter + 1;
                                ()
|       Flt(v)  ->      tvars.(!temp_counter) <- (id_ti, typePrefix);
                                fprintf oc "\n\t%s" (ti ^ " = " ^
                                    string_of_float(v) ^ ";");
                                Stack.push
                                    ("_t"^string_of_int(!temp_counter))
                                    stck;
                                temp_counter:=!temp_counter+1;
                                ()
|       Add_Op  ->      tvars.(!temp_counter) <- (id_ti, typePrefix);
                                let op1 = Stack.pop stck and op2 =
                                    Stack.pop stck in
                                fprintf oc "\n\t%s" (ti ^ " = " ^ op2
                                    ^ " + " ^ op1 ^ ";");
                                Stack.push
                                    ("_t"^string_of_int(!temp_counter))
                                    stck;
                                temp_counter:=!temp_counter+1;
                                ()
|       Minus_Op -> tvars.(!temp_counter) <- (id_ti, typePrefix);
                                let op1 = Stack.pop stck and op2 =
                                    Stack.pop stck in
                                fprintf oc "\n\t%s" (ti ^ " = " ^ op2
                                    ^ " - " ^ op1 ^ ";");
                                Stack.push
                                    ("_t"^string_of_int(!temp_counter))
                                    stck;
                                temp_counter:=!temp_counter+1;
                                ()
|       Multiply_Op -> tvars.(!temp_counter) <- (id_ti, typePrefix);
                                let op1 = Stack.pop stck and op2 =
                                    Stack.pop stck in
                                fprintf oc "\n\t%s" ("_t"^
                                    string_of_int(!temp_counter)^ " =
                                    "^op2^" * "^op1^";");
                                Stack.push
                                    ("_t"^string_of_int(!temp_counter))
                                    stck;
                                temp_counter:=!temp_counter+1;
                                ()
|       Divide_Op -> tvars.(!temp_counter) <- (id_ti, typePrefix);
                                let op1 = Stack.pop stck and op2 =
                                    Stack.pop stck in
                                fprintf oc "\n\t%s" ("_t"^
                                    string_of_int(!temp_counter)^ " =
                                    "^op2^" / "^op1^";");
                                Stack.push
                                    ("_t"^string_of_int(!temp_counter))
                                    stck;
```

```
                                temp_counter:=!temp_counter+1;
                                ()
    |       LessThan_Op      ->
                    tvars.(!temp_counter) <- (id_ti, typePrefix);
                    let op1 = Stack.pop stck and op2 = Stack.pop stck in
                    fprintf oc "\n\t%s" ("_t"^
                        string_of_int(!temp_counter)^ " = "^op2^" <
                        "^op1^";");
                    Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                    temp_counter:=!temp_counter+1;
                    ()
    |       LessThanEq_Op   ->
                    tvars.(!temp_counter) <- (id_ti, typePrefix);
                    let op1 = Stack.pop stck and op2 = Stack.pop stck in
                    fprintf oc "\n\t%s" ("_t"^
                        string_of_int(!temp_counter)^ " = "^op2^" <=
                        "^op1^";");
                    Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                    temp_counter:=!temp_counter+1;
                    ()
    |   GreaterThan_Op ->
                    tvars.(!temp_counter) <- (id_ti, typePrefix);
                    let op1 = Stack.pop stck and op2 = Stack.pop stck in
                    fprintf oc "\n\t%s" ("_t"^
                        string_of_int(!temp_counter)^ " = "^op2^" >
                        "^op1^";");
                    Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                    temp_counter:=!temp_counter+1;
                      ()
    |       GreaterThanEq_Op ->
                    tvars.(!temp_counter) <- (id_ti, typePrefix);
                    let op1 = Stack.pop stck and op2 = Stack.pop stck in
                    fprintf oc "\n\t%s" ("_t"^
                        string_of_int(!temp_counter)^ " = "^op2^" >=
                        "^op1^";");
                    Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                    temp_counter:=!temp_counter+1;
                    ()
    |       EqualsTo_Op     ->
                    tvars.(!temp_counter) <- (id_ti, typePrefix);
                    let op1 = Stack.pop stck and op2 = Stack.pop stck in
                    fprintf oc "\n\t%s" ("_t"^
                        string_of_int(!temp_counter)^ " = "^op2^" ==
                        "^op1^";");
                    Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                    temp_counter:=!temp_counter+1;
                    ()
    |       NotEqualsTo_Op  ->
                    tvars.(!temp_counter) <- (id_ti, typePrefix);
                    let op1 = Stack.pop stck and op2 = Stack.pop stck in
                    fprintf oc "\n\t%s" ("_t"^
                        string_of_int(!temp_counter)^ " = "^op2^" !=
                        "^op1^";");
                    Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                    temp_counter:=!temp_counter+1;
                    ()
    |       Not_Op  ->
                    tvars.(!temp_counter) <- (id_ti, typePrefix);
                    let op1 = Stack.pop stck in
                    fprintf oc "\n\t%s" ("_t"^
```

```
                    string_of_int(!temp_counter)^ " = !"^op1^";");
                Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                temp_counter:=!temp_counter+1;
                ()
    |   Sqrt    ->
                tvars.(!temp_counter) <- (id_ti, typePrefix);
                let op1 = Stack.pop stck in
                fprintf oc "\n\t%s" ("_t"^
                    string_of_int(!temp_counter)^ " = sqrt("^op1^");");
                Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                temp_counter:=!temp_counter+1;
                ()
    |       Sin_Op  ->
                tvars.(!temp_counter) <- (id_ti, typePrefix);
                let op1 = Stack.pop stck in
                fprintf oc "\n\t%s" ("_t"^
                    string_of_int(!temp_counter)^ " = sin(M_PI / 180 *
                    "^op1^");");
                Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                temp_counter:=!temp_counter+1;
                ()
    |       Cos_Op  ->
                tvars.(!temp_counter) <- (id_ti, typePrefix);
                let op1 = Stack.pop stck in
                fprintf oc "\n\t%s" ("_t"^
                    string_of_int(!temp_counter)^ " = cos(M_PI / 180 *
                    "^op1^");");
                Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                temp_counter:=!temp_counter+1;
                ()
    |       Getx_Op ->
                tvars.(!temp_counter) <- (id_ti, typePrefix);
                let op1 = Stack.pop stck in
                fprintf oc "\n\t%s" ("_t"^
                    string_of_int(!temp_counter)^ " = "^op1^".x;");
                Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                temp_counter:=!temp_counter+1;
                ()
    |       Gety_Op ->
                tvars.(!temp_counter) <- (id_ti, typePrefix);
                let op1 = Stack.pop stck in
                fprintf oc "\n\t%s" ("_t"^
                    string_of_int(!temp_counter)^ " = "^op1^".y;");
                Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                temp_counter:=!temp_counter+1;
                ()
    |       Vec2_Op ->
                tvars.(!temp_counter) <- (id_ti, typePrefix);
                let op1 = Stack.pop stck and op2 = Stack.pop stck in
                fprintf oc "\n\t%s" ("_t"^
                    string_of_int(!temp_counter)^ ".x = "^op2^";");
                fprintf oc "\n\t%s" ("_t"^
                    string_of_int(!temp_counter)^ ".y = "^op1^";");
                Stack.push ("_t"^string_of_int(!temp_counter)) stck;
                temp_counter:=!temp_counter+1;
                ()
    |       On_Op   ->
                tvars.(!temp_counter) <- (id_ti, "float");
                tvars.(!temp_counter + 1) <- (id_ti_plus1, "vec2cpp");
                tvars.(!temp_counter + 2) <- (id_ti_plus2, "vec2cpp");
```

```
                        let dir = Stack.pop stck and dist = Stack.pop stck in
                        fprintf oc "\n\t%s"
                            ("_t"^string_of_int(!temp_counter)^" =
                            sqrt(("^dir^".x * "^dir^".x + "^dir^".y *
                            "^dir^".y));
"^dir^".x = "^dir^".x/_t"^string_of_int(!temp_counter)^";
"^dir^".y = "^dir^".y/_t"^string_of_int(!temp_counter)^";
_t"^string_of_int(!temp_counter+1)^".x = "^dist^" * "^dir^".x + _ff.x;
_t"^string_of_int(!temp_counter+1)^".y = "^dist^" * "^dir^".y + _ff.y;
glBegin(GL_LINES);
\tglColor3f(0.0, 0.0, 0.0);
\tglVertex2f(_ff.x, _ff.y);
\tglVertex2f(_t"^string_of_int(!temp_counter+1)^".x ,
    _t"^string_of_int(!temp_counter+1)^".y);
glEnd();
_ff.x = _t"^string_of_int(!temp_counter+1)^".x;
_ff.y = _t"^string_of_int(!temp_counter+1)^".y;
");
                                        temp_counter:=!temp_counter+1;
                                        temp_counter:=!temp_counter+1;
                                        fprintf oc "\n\t%s" ("_t"^
                                            string_of_int(!temp_counter)^ " =
                                            _ff;");
                                        Stack.push
                                            ("_t"^string_of_int(!temp_counter))
                                            stck;
                                        temp_counter:=!temp_counter+1;
                                        ()
        |   Off_Op      ->      tvars.(!temp_counter) <- (id_ti, "float");
                                        tvars.(!temp_counter + 1) <-
                                            (id_ti_plus1, "vec2cpp");
                                        tvars.(!temp_counter + 2) <-
                                            (id_ti_plus2, "vec2cpp");
                                        let dir = Stack.pop stck and dist =
                                            Stack.pop stck in
                                        fprintf oc "\n\t%s"
                                            ("_t"^string_of_int(!temp_counter)^"
                                            = sqrt(("^dir^".x * "^dir^".x +
                                            "^dir^".y * "^dir^".y));
"^dir^".x = "^dir^".x/_t"^string_of_int(!temp_counter)^";
"^dir^".y = "^dir^".y/_t"^string_of_int(!temp_counter)^";
_t"^string_of_int(!temp_counter+1)^".x = "^dist^" * "^dir^".x + _ff.x;
_t"^string_of_int(!temp_counter+1)^".y = "^dist^" * "^dir^".y + _ff.y;
_ff.x = _t"^string_of_int(!temp_counter+1)^".x;
_ff.y = _t"^string_of_int(!temp_counter+1)^".y;
");
                                        temp_counter:=!temp_counter+1;
                                        temp_counter:=!temp_counter+1;
                                        fprintf oc "\n\t%s" ("_t"^
                                            string_of_int(!temp_counter)^ " =
                                            _ff;");
                                        Stack.push
                                            ("_t"^string_of_int(!temp_counter))
                                            stck;
                                        temp_counter:=!temp_counter+1;
                                        ()
        |   DAsn_Op ->
                                        let op1 = Stack.pop stck and op2 =
                                            Stack.pop stck in
                                        fprintf oc "\n\t%s" (op1 ^ " = " ^ op2
```

```
                                        ^ ";");
                                    (*fprintf oc "\n\t%s" (op1 ^ " =
                                        "^op2^";");*)
                                    Stack.push (op1) stck;
                                    ()
    |       Asn_Op   ->  let op1 = Stack.pop stck and op2 = Stack.pop stck
        in
                                    fprintf oc "\n\t%s" (op1 ^ " =
                                        "^op2^";");
                                    Stack.push (op1) stck;
                                    ()
    |       Id(v)    ->      let lenSubs = ((String.length y) -4) in
                                    let varname = String.sub (y) (3)
                                        lenSubs in
                                    Stack.push (varname) stck;
                                    ()
    |       If_Op(i)->      let e = Stack.pop stck in
                                        fprintf oc "\n\t%s" ("if (" ^
                                            e ^ ") { goto _L" ^
                                            string_of_int(i) ^ "; }");
    (*|     EndIf_Op->      fprintf oc "\n\t%s" ("}}") *)
    |       Or_Op(i) ->     let e = Stack.pop stck in
                                        fprintf oc "\n\t%s" ("if (" ^
                                            e ^ ") { goto _L" ^
                                            string_of_int(i) ^ "; }");
|   And_Op(i) ->
            tvars.(!temp_counter) <- (id_ti, typePrefix);
            let e = Stack.pop stck in
            fprintf oc "\n\t%s" ("bool
                _t"^string_of_int(!temp_counter+2)^" = false;");
            Hashtbl.replace tvarTbl (!temp_counter+2) "bool";
            fprintf oc "\n\t%s" ("if (!" ^ e ^ ") { goto _L" ^
                string_of_int(i) ^ "; }");
|   OrDone_Op ->
            tvars.(!temp_counter) <- (id_ti, typePrefix);
            fprintf oc "\n\t%s" ("_t"^ string_of_int(!temp_counter)^ " =
                _t"^string_of_int(!temp_counter-4)^" ||
                "^"_t"^string_of_int(!temp_counter-1)^";");
            Stack.push ("_t"^string_of_int(!temp_counter)) stck;
            temp_counter:=!temp_counter+1;
|   AndDone_Op ->
            tvars.(!temp_counter) <- (id_ti, typePrefix);
            fprintf oc "\n\t%s" ("_t"^ string_of_int(!temp_counter)^ " =
                _t"^string_of_int(!temp_counter-4)^" &&
                "^"_t"^string_of_int(!temp_counter-1)^";");
            Stack.push ("_t"^string_of_int(!temp_counter)) stck;
            temp_counter:=!temp_counter+1;
|   While_Op(i)     ->
            let e = Stack.pop stck in
            fprintf oc "\n\t%s" ("if (!" ^ e ^ ") { goto _L" ^
                string_of_int(i) ^ "; }");
|   Goto(i)         ->
            fprintf oc "\n\t%s" ("goto _L" ^ string_of_int(i)^";");
|   GotoFun(fn, argcount)   ->
            fprintf oc "\n\t%s" ("lv_frameptr.push(lv_index);");
            fprintf oc "\n\t%s" ("goto _L" ^ fn ^ ";");
|   GotoReturn      ->
            tvars.(!temp_counter) <- (id_ti, "void *");
            fprintf oc "\n\t%s" ("lv_frameptr.pop();");
            fprintf oc "\n\t%s" ("_t"^string_of_int(!temp_counter)^" =
```

```
                    ((fRecords.top()).retFunc);");
                fprintf oc "\n\t%s" ("fRecords.pop();\n");
                fprintf oc "\n\t%s" ("goto
                    *_t"^string_of_int(!temp_counter)^";\n");
                temp_counter:=!temp_counter+1;
    |   Lbl(i)  ->      if(i mod 2 = 0) then
                            fprintf oc "\n\t%s" ("_L"^string_of_int(i) ^
                                ":");
                        if(i mod 2 = 1) then
                            fprintf oc "\n\t%s" ("_L"^string_of_int(i) ^
                                ":");
    |   Flbl(s) ->
                fprintf oc "\n\t%s" ("_L" ^ s ^ ":");
    |   SetReturn(i)    ->
                tvars.(!temp_counter) <- (id_ti, "actRecord");
                fprintf oc "\n\t%s" (ti ^ ".retFunc = &&_L" ^ string_of_int i
                    ^";");
                fprintf oc "\n\t%s" ("fRecords.push(" ^ ti ^ ");");
                temp_counter := !temp_counter + 1;
    |   SetLocal(fn)    ->
                fprintf oc "\n\t%s" ("lclVars[lv_index] = (float)_t" ^
                    string_of_int(!temp_counter - 1) ^ ";");
                fprintf oc "\n\t%s" ("lv_index = lv_index + 1;");
    |   GetLocal(i)             ->
                tvars.(!temp_counter) <- (id_ti, "float");
                fprintf oc "\n\t%s" (ti ^ " = lclVars[lv_frameptr.top() - " ^
                    string_of_int i ^ "];");
                Stack.push ti stck;
                temp_counter := !temp_counter + 1;
        |       _               ->      ()

let generate_c lst ti li oc globals globals_count tvars =
        Stack.clear stck;
        (*globals_to_file globals globals_count 0 oc;*)
        print_endline "*****************";
        print_endline "SEMANTIC STACK";
        print_endline "*****************";
        List.iter (fun (fs, sn, thr) ->
                                        print_endline (" (" ^ sn ^ "," ^ thr ^
                                            ")")) lst;
        List.iter (fun (x) -> c_statement x ti li oc tvars) lst;

        ()
```

Appendix B

Test Suites (including testAll.sh)

testAll.sh

```
#!/bin/sh

keep=0
delete=0
numFail=0
numPass=0
failure=0
globallog=testAll.log
TESTDIR="tests"
EXPDIR="tests/exp"

Usage()
{
        echo "-d: Delete intermediate files (.cpp, .diff) on failure."
        echo "-h: Display this help message."
        echo "-k: Keep intermediate files (.cpp, .diff) on success."
}

Check()
{
        failure=0 # reset failed test status
        basename=`basename $1`
        basename="${basename%.*}"
        basedir=`dirname $1`

        ./runCompiler.sh -s $1

        Compare $basedir/$basename.cpp $EXPDIR/$basename.out
            $basedir/$basename.diff

        if [ $failure -eq 1 ]
        then
                numFail=`expr $numFail + 1`
                echo "FAILED $basename.cpp differs from $basename.out" 1>&2
                if [ $delete -eq 1 ]
                then
                        rm -f $basedir/$basename.cpp $basedir/$basename.diff
                fi
        else
                numPass=`expr $numPass + 1`
                if [ $keep -eq 0 ]
                then
                        rm -f $basedir/$basename.cpp $basedir/$basename.diff
                fi
        fi
}

Compare()
{
        # Compare input file to golden copy, and output any differences to a
            .diff file.
        diff -b "$1" "$2" > "$3" 2>&1 || {
    failure=1
    }
}
```

```
Cleanup()
{
        # Cleans up any pre-existing .cpp and .diff files in the test
            directory.
        `find $TESTDIR -name "*.cpp" | xargs rm -f`
        `find $TESTDIR -name "*.diff" | xargs rm -f`
}

Cleanup

while getopts kdh c
do
    case $c in
    k) # Keep intermediate files
        keep=1
        ;;
        d) # delete intermediate files on fail
        delete=1
        ;;
    h) # Help
        Usage
        ;;
        esac
done

shift `expr $OPTIND - 1`

if [ $# -ge 1 ]
then
    files=$@
else
        files=`find $TESTDIR -name "*.ff"`
fi

for file in $files
do
        Check $file
done

echo "Passed: $numPass"
echo "Failed: $numFail"
```

There are 17 test FF programs:

(1) test-arith1.ff

```
x = 1 + 2
y = 3.1 - 0.0
z = -1.0 * 2.5
p = 5.0 / 2.0
```

(2) test-arith2.ff

```
x = 9 - 5 - 2 + 3 / 3 + 1 * 0
```

(3) test-block.ff

```
0.5 on [1,1]

{
        0.5 on [0,-1]
        0.5 on [-1,0]
```

```
}

0.5 on [-1,-1]
```

(4) test-ifelse.ff

```
i = 5.0
if i > 1.0
{
        i = 1.0
}
else
{
        i = 0.0
}
endif

i on [1,1]
```

(5) test-ifelse2.ff

```
i = 5.0
if i > 1.0
{
        if (i > 2.0)
        {
                i = 1.0
        }
        else
        {
                i = 0.0
        }
        endif
}
else
{
        i = 0.0
}
endif

i on [1,1]
```

(6) test-ifelse3.ff

```
i = 5.0
if i < 1.0
{
        i = 0.0
}
else
{
        if (i > 2.0)
        {
                i = 1.0
        }
        else
        {
                i = 0.0
        }
        endif
}
endif
```

```
i on [1,1]
```

(7) test-mathlib.ff

```
v = (sqrt 9.0)/2.0 - 1.0
y = sin 90
x = cos 180
v on [x,y]
```

(8) test-off.ff

```
0.5 off [1,1]
0.5 on [1,1]
```

(9) test-on.ff

```
1.0 on [1,1]
```

(10) test-paren.ff

```
x = 9.0-(5.0-4.0)
x on [1,1]
```

(11) test-relop.ff

```
i = 0.0
if (3 > 2) {i = 1.0} else { i = 0.0 } endif i on [0,1]
if (3 < 2) {i = 0.0} else { i = 1.0 } endif i on [1,0]
if (3 == 3) {i = 1.0} else { i = 0.0 } endif i on [0,-1]
if (0 != 2) {i = 1.0} else { i = 0.0 } endif i on [-1,0]
```

(12) test-square.ff

```
v1 = [1,0]
x = v1.x
y = v1.y

i = 4
while (i > 0)
{
        0.5 on [x,y]

        x2 = (cos 90) * x - (sin 90) * y
        y2 = (sin 90) * x + (cos 90) * y

        x = x2
        y = y2

        i = i - 1
}
```

(13) test-varassign.ff

```
x = 0
y = x
```

(14) test-vardecl.ff

```
x = 0
```

(15) test-vec1.ff

```
x = [0,1]
y = [3.5,2.1]
```

(16) test-vec2.ff

```
v1 = [0.5,0.5]
x = v1.x
y = v1.y
x on v1
y on v1
```

(17) test-while.ff

```
i = 4
while i > 1
{
        0.25 on [1,1]
        i = i - 1
}
```

Appendix C

Sample Programs Here are some FF sample programs:

(1) circle.ff

```
/* DRAW CIRCLE */

v1 = [0.0,-1.0]
0.25 on [0.0,-1.0]

x = v1.x
y = v1.y

angle = 0.0

while (angle < 360.0)
{
        0.005 on [x,y]

        x2 = 0.005 * (cos angle)
        y2 = 0.005 * (sin angle)

        x = x2
        y = y2

        angle = angle + 1.0
}
```

(2) square.ff

```
/* DRAW SQUARE */

0.2 off [1,1]
v1 = [1,0]
x = v1.x
y = v1.y

i = 4
while (i > 0)
{
        0.5 on [x,y]

        x2 = (cos 90) * x - (sin 90) * y
        y2 = (sin 90) * x + (cos 90) * y

        x = x2
        y = y2

        i = i - 1
}
```

(3) spiral.ff

```
/* DRAW SPIRAL */

v1 = [0.0,-1.0]

x = v1.x
y = v1.y
```

```
angle = 0.0

radius = 0.00001
finalRadius = 0.017
spiralIncrement = 0.000002

while (radius < finalRadius)
{
        radius on [x,y]

        x2 = radius * (cos angle)
        y2 = radius * (sin angle)

        x = x2
        y = y2

        angle = angle + 1.0
        radius = radius + spiralIncrement
}
```

(4) koch.ff

```
/*
        This is a sample program showing the Koch's snowflake, after its 3rd
            iteration

*/
0.2 off [1,1] /*repositioning firefly*/
0.1 off [0,1] /*repositioning firefly*/
dir=[1,0]
theta = 60
i = 12
while(i>0)
{
        0.1 on dir
        dir = [(cos 60) * dir.x - (sin 60) * dir.y, (sin 60) * dir.x + (cos
            60) * dir.y]
        0.1 on dir
        dir = [(cos -120) * dir.x - (sin -120) * dir.y, (sin -120) * dir.x +
            (cos -120) * dir.y]
        0.1 on dir
        dir = [(cos 60) * dir.x - (sin 60) * dir.y, (sin 60) * dir.x + (cos
            60) * dir.y]
        0.1 on dir
        if((i/2)*2==i)
                theta = -120
        else
                theta = 60
        endif
        dir = [(cos theta) * dir.x - (sin theta) * dir.y, (sin theta) * dir.x
            + (cos theta) * dir.y]
        i = i-1
}
```

(5) fibonacciStairs.ff

```
dir = [0.0,1.0]
dir2 = [1.0,0.0]
a = 1
let f(2) =
```

```
{
        if( $2 > 10.0){
                a = 0
        }
        else{
                $2 on dir
                0.1 on dir2
                f($2, $1+$2)
        }
        endif
}
f(0.0,0.05)
```

Appendix D

GitHub Log

```
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c2464174147155545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion
:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c2464174147155545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
```

```
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c2464174147155545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 20:36:22 2014 -0400
:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c246417414715545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 20:36:22 2014 -0400

    Updates to helper scripts & Makefile

:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
```

```
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c2464174147155545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 20:36:22 2014 -0400

    Updates to helper scripts & Makefile

commit e6386840261ddd12c11ef13003d17b4de49f28e7
Author: Alex Shnayder <shnaydera@gmail.com>
:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c2464174147155545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 20:36:22 2014 -0400

    Updates to helper scripts & Makefile

commit e6386840261ddd12c11ef13003d17b4de49f28e7
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:14:41 2014 -0400

    removing cppCompile.sh. Not necessary anymore
```

```
:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c246417414715545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 20:36:22 2014 -0400

    Updates to helper scripts & Makefile

commit e6386840261ddd12c11ef13003d17b4de49f28e7
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:14:41 2014 -0400

    removing cppCompile.sh. Not necessary anymore

commit b9dd1a4d621e7e62872648aa6220567e4f93f838
Author: Alex Shnayder <shnaydera@gmail.com>
:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c246417414715545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
```

```
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 20:36:22 2014 -0400

    Updates to helper scripts & Makefile

commit e6386840261ddd12c11ef13003d17b4de49f28e7
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:14:41 2014 -0400

    removing cppCompile.sh. Not necessary anymore

commit b9dd1a4d621e7e62872648aa6220567e4f93f838
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:18:06 2014 -0400

:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c246417414715545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 20:36:22 2014 -0400

    Updates to helper scripts & Makefile

commit e6386840261ddd12c11ef13003d17b4de49f28e7
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:14:41 2014 -0400

    removing cppCompile.sh. Not necessary anymore

commit b9dd1a4d621e7e62872648aa6220567e4f93f838
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:18:06 2014 -0400
```

```
    Slight modification to Makefile
:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c2464174147155545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 20:36:22 2014 -0400

    Updates to helper scripts & Makefile

commit e6386840261ddd12c11ef13003d17b4de49f28e7
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:14:41 2014 -0400

    removing cppCompile.sh. Not necessary anymore

commit b9dd1a4d621e7e62872648aa6220567e4f93f838
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:18:06 2014 -0400

    Slight modification to Makefile

:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
```

```
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c246417414715545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 20:36:22 2014 -0400

    Updates to helper scripts & Makefile

commit e6386840261ddd12c11ef13003d17b4de49f28e7
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:14:41 2014 -0400

    removing cppCompile.sh. Not necessary anymore

commit b9dd1a4d621e7e62872648aa6220567e4f93f838
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:18:06 2014 -0400

    Slight modification to Makefile

commit a9cd95f9897d145d34e1447ab8529e225cf359e6
:
commit deb21f50c39e6263a03efb5a95bcf44b59d14bd5
Author: Prerna Chikersal <prerna1@e.ntu.edu.sg>
Date:   Mon Jul 21 17:53:24 2014 -0400

    Initial commit

commit e4c042a928ee45bd3c45fffd35f0120ad136f850
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Jul 30 21:14:19 2014 -0400

    hello world by alex

commit 7fbe99761e33ee24e14ea68c5dc07ced3ce14c52
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 22:14:21 2014 -0400

    Changed runCompiler to have ./

commit c246417414715545f496e77dd8d2f7a1803916b0
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Jul 30 23:18:38 2014 -0400

    Glut insertion

commit 0f6e2df2a9497a112ffb18605e4633a96980d64a
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 20:36:22 2014 -0400

    Updates to helper scripts & Makefile
```

```
commit e6386840261ddd12c11ef13003d17b4de49f28e7
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:14:41 2014 -0400

    removing cppCompile.sh. Not necessary anymore

commit b9dd1a4d621e7e62872648aa6220567e4f93f838
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Jul 31 21:18:06 2014 -0400

    Slight modification to Makefile

commit a9cd95f9897d145d34e1447ab8529e225cf359e6
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Jul 31 21:43:45 2014 -0400

    Added literal and plus to parser/scanner/ast

    Glut stuf is now working and compiling (not sure what fixed it, but I no
    longer get the CR/LF error). Please see if you can get it to work on
    yours. Need to add a file that is equivilant to bytecode.ml of microC.
    Maybe cflat.ml or something. Assuming we don't need an interperter (for
    diagnostics), we need to restructure compile.ml to read from flatc.ml.

commit 3d36bd20325943f591ee9edd86695ef73d920a62
Merge: a9cd95f b9dd1a4
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Jul 31 21:51:53 2014 -0400

    Merge remote-tracking branch 'origin/master'

    Conflicts:
        helloWorld.cpp
        helloWorld.exe

commit 48032068c712445794c3317e9bbeae33f92a54c0
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Fri Aug 1 01:06:26 2014 -0400

    Alex: repushing updates from today 8/1/2014

commit cc6af1bb5aba91153b319f835e60d0d7292f0e32
Merge: 4803206 3d36bd2
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Fri Aug 1 01:14:03 2014 -0400

    Trying to resync again.

commit af6dea6815c4977483f8e093d77745a7ed33c38f
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Fri Aug 1 01:16:25 2014 -0400

    Removing unnecessary files.

commit edb0eabf71e55135996144fa4a0a9646a9541f18
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Fri Aug 1 18:47:42 2014 -0400

    Alex: Updated compiler so that it compiles properly for now. Updated helper
```

```
commit 24a90d5e07d08cdf0c5bd1da90f1be3599154d24
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Mon Aug 4 15:36:04 2014 -0400

    Created basic batch test script.

commit 5aef6cd79d1d3e3082eefe5a7dd5334e17d2562b
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Wed Aug 6 19:54:13 2014 -0400

    Added in code for compiling and execution c++ file on a mac. Updated the
        REA

commit ba10bdc12c7d25f4e6bb14cb56c158a0c54c8c92
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Wed Aug 6 19:55:23 2014 -0400

    updating readme

commit 9940f2daef2f69c36317936f98ff5c99e7ed79a7
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 7 17:44:18 2014 -0400

    Output literal and vec2

commit 078f02c7383784871bfee156279aa20ac0f0e9ff
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 7 18:34:51 2014 -0400

    Small update to compile.ml to possibly prevent blank output file.

commit f32acdfcead5629e047f51542671c51be416685c
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 7 19:04:34 2014 -0400

    Float added, literal renamed

commit eddb9052816cd0e8c9d7ae1116d4de32fff75a88
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 7 20:56:01 2014 -0400

    Implemented multiple tokens via expr list

commit b11691d587e9e8d7e84f0cb86807397eaf3c4ec0
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 7 21:38:45 2014 -0400

    ON somewhat working and vec2 is flt*flt

commit 96d46deab7ace28368d099c3fbf35e14f7570d12
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 04:10:59 2014 -0400

    cpp generated now compiled

commit c1d1f4d8d5e3165bb311c9735adb7c99c5894eb8
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 04:16:44 2014 -0400

    removing _build to tidy things up
```

```
commit a8079564d51ee097c2b9611a2e33b005c8bb4a75
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 04:44:18 2014 -0400

    added in code to parse identifier

commit 4701931ace454242186f5024fda7bcecbe667a64
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 04:56:44 2014 -0400

    corrected small mistake in parsing identifiers. Added in code for parsing
        as

commit 88f7f905bfbcec0fe3191bc0f6f1011b0c88c6e5
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 06:24:58 2014 -0400

    variable declaration ROUGHLY DONE. Still need to refine, store declarations

commit 09aa324c3e47e71625491742e3f1a498de0b52b5
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 07:04:33 2014 -0400

    touching up last commit

commit e0cd40503c59654160a3df1e8052caaeee63deff
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 07:57:26 2014 -0400

    changes to Opengl code

commit f79edbcc8e890a6138ffaa544631a78fa6d94ddf
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 09:05:56 2014 -0400

    changed some opengl stuff. Worked out assignment for vec2s. Still not
        worked

commit 30a04d2eb2d5c8446063c2ecf4c5a79a0990e902
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 09:07:38 2014 -0400

    removed the old eval function. refer to commit history to find it again

commit 9faad75f0ee98af42380fbfaabeafd72f04e3f85
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 12:08:29 2014 -0400

    On works! We can now draw a line on GLUT Window

commit e826de72a4a3d4efab82b057917deb66597393fb
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 12:18:44 2014 -0400

    corrected issues in scanning vec2. Vec2 still doesn't take in negatives

commit 5769422fee8a345699172c588c2d65bfe42c56cf
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Fri Aug 8 15:37:05 2014 -0400
```

```
    Negative vector values working. May need to change to make more robust.

commit f2f2c8a6195246b4904f4e6ec1df1db023dd3cfc
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 16:37:01 2014 -0400

    all works for me

commit a8cbb67d80bc0d56a64f21fd85cf15285f4a9b06
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 16:41:24 2014 -0400

    made + sign in front of value also legal

commit bf1cbd65043032d7a12f640d251ef0cb2651320a
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Fri Aug 8 16:52:19 2014 -0400

    off operator working

commit 2b82007166e8f24985afb4614abb2694974569b2
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Fri Aug 8 17:29:11 2014 -0400

    Added unary plus and minus. Added constant/negative constant type.

commit 02a1bd33ecffb83a3308d92888fc7e7b87c63d12
Merge: 2b82007 bf1cbd6
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Fri Aug 8 17:44:58 2014 -0400

    Reconciled updates to make + and - work.

commit e35c4942ec94c23f2f4dda1378f806433521f8da
Author: royaslan <roy.aslan@outlook.com>
Date:   Fri Aug 8 21:33:13 2014 -0400

    stmnt now added, stacking added

commit 44ef616855ea787a38feb8ea5ab50184cad7ed10
Author: royaslan <roy.aslan@outlook.com>
Date:   Fri Aug 8 23:18:38 2014 -0400

    Assignment globals added

commit e3c8d67899c27fe7bd39c2a07410c6040121e710
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Sat Aug 9 15:09:07 2014 -0400

    Added type checking

commit c09c2482f0d43283757f429ad332ef4d490ab3e5
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Sat Aug 9 16:10:49 2014 -0400

    ID(x) semantics stack completed but need to do Assignment

commit cfff49f6b1e5404b2b50303ff82e1f88ccb0399e
Author: royaslan <roy.aslan@outlook.com>
```

```
Date:   Sun Aug 10 01:47:47 2014 -0400

    Assignment is partially added to semantic analyzer

commit 89c45208f435f31fef2628184c0fdd6a3023b1af
Author: royaslan <roy.aslan@outlook.com>
Date:   Sun Aug 10 03:35:16 2014 -0400

    Flatc is partially working (only for literals)

commit 7dac021f7b98ad32f09332811803c0b2fe55cea3
Author: royaslan <roy.aslan@outlook.com>
Date:   Sun Aug 10 04:10:57 2014 -0400

    Flatc working for some operators (add, vec2)

commit 4b8bf7aa7c761afcbd0b7689061bf627583cc1cf
Author: royaslan <roy.aslan@outlook.com>
Date:   Sun Aug 10 04:55:44 2014 -0400

    Further extended flat c

commit 15778fc55d06767724287de22efd854ca4a8b8db
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Sun Aug 10 11:12:55 2014 -0400

    Vec2 can have expressions; added minus and rel operators; added parentheses

commit 920c23786951adb625737a84e598d94b38840ec4
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Sun Aug 10 16:54:06 2014 -0400

    Adding to the semantic stack/type checking phase.

commit 4babfff53ce8028bd9af727c37591b76c3dc391b
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Sun Aug 10 17:45:17 2014 -0400

    Updated semantic analysis for proper static checking.

commit e141ac91f0b7190ec046e1d9f772c49328a49ccd
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Sun Aug 10 20:29:24 2014 -0400

    Added flatc code gen for Int, Flt, Add_Op, Minus_Op, DAsn_Op, Asn_Op, Id.

commit dbac0ec5115f38054ccdac3f512f874d29b2e610
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Sun Aug 10 20:36:29 2014 -0400

    added in flatc code gen for vec2cpp

commit 8f8bc858f18efe9568005ff9400af98e91d5aef7
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Sun Aug 10 21:22:20 2014 -0400

    ON code gen incomplete

commit b44abb3e56abed33ae4bd5bcb505246ee4a752f0
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
```

```
Date:   Sun Aug 10 21:26:31 2014 -0400

    All temp variables now begin with underscore (_)

commit fa4ea9f67ee6baa93302931c67f802cbceff0600
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Sun Aug 10 21:45:47 2014 -0400

    on code gen almost done, but not tested yet

commit ae2dfa1b3d40fb70bfd2da50bd2a0516c520622d
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Sun Aug 10 21:49:13 2014 -0400

    /*.....*/ now indicates a comment

commit 26e45a7a4cd055f8b3b272769115db5be06b01f8
Author: royaslan <roy.aslan@outlook.com>
Date:   Sun Aug 10 22:11:16 2014 -0400

    Updated syntax files for IF statement

commit be94c9f3c94577142a57bdb07c39a6ed57b4d961
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Sun Aug 10 22:50:57 2014 -0400

    still some issues in printing cpp

commit b4dc7921bf74dd62fc5d5a599094735a45569614
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Sun Aug 10 23:26:17 2014 -0400

    c++ code now printing to file properly

commit 1b3214717862a212321be3b2f3ded0d7f7773948
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Mon Aug 11 00:37:30 2014 -0400

    flat c for greater than done

commit 5254d029a829057c62ba2d8f6ef39e531a6d6afe
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Mon Aug 11 00:40:29 2014 -0400

    greater than and less than added

commit a1eca1c27a7ca19f90174cc821c8c93844234df1
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Mon Aug 11 00:42:40 2014 -0400

    added in <, >, <=, >= and ==. != is still left.

commit 7c93ddfb3b5367ae465d1f71db9e48ee5b973cb2
Author: royaslan <roy.aslan@outlook.com>
Date:   Mon Aug 11 01:41:20 2014 -0400

    IF partially working in compile and semantics

commit 82cd4256ea91362e44e21bde710fe6cce69b0b42
Merge: 7c93ddf a1eca1c
```

```
Author: royaslan <roy.aslan@outlook.com>
Date:   Mon Aug 11 01:41:39 2014 -0400

    Merge branch 'master' of https://github.com/prernaa/Firefly3D

commit 1eaa26d5f82213c82e225ac186dee91cc42fb02b
Author: royaslan <roy.aslan@outlook.com>
Date:   Mon Aug 11 02:12:22 2014 -0400

    Additional tweaking of IF handling

commit 2518aa64460b86857e29a32a9ffb47007121f85d
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Mon Aug 11 14:43:53 2014 -0400

    Fixed semantic stack algorithm to allow for parenthesis. Also added check
       fo

commit f07edad6c6806a5bcd3cd2b132611b0b5fb7e9e4
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Mon Aug 11 15:06:36 2014 -0400

    replaced with working semantics.ml file

commit c9c1f93e056d110cd772696f0175a34e30fc011b
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Mon Aug 11 16:33:49 2014 -0400

    Updated semantic stack and flatc to work together.

commit 376179db7c054be9534debeafc5097ce413532ed
Author: royaslan <roy.aslan@outlook.com>
Date:   Mon Aug 11 22:40:30 2014 -0400

    Starting to build labeling mechanism

commit 91460ca5f3dcc4c47f680ac6afdbc13fd7125d06
Author: royaslan <roy.aslan@outlook.com>
Date:   Mon Aug 11 22:43:28 2014 -0400

    Push flatc.ml

commit a6906808c8f68e9a770475376a431b56c0423c40
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Mon Aug 11 22:49:02 2014 -0400

    labels working for single if-else

commit b9ccb9ff1aca3ce1fd6a3d33b21319e9b5481183
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Tue Aug 12 01:24:27 2014 -0400

    trying to solve scope issues with gotos

commit e98664aea18e22fefdda10e03ba31c7eeed7bfc7
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Tue Aug 12 01:39:40 2014 -0400

    Made assign right assoc
```

```
commit 37f5e28b9bf4d5954c66638919ece03acdf7ee98
Author: royaslan <roy.aslan@outlook.com>
Date:   Tue Aug 12 13:07:57 2014 -0400

    Nested IF is now working and compiling on windows

commit 6b9920a72472905ec68d0bec1e3a63307658df49
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Tue Aug 12 13:15:58 2014 -0400

    Removed != from semantics; added blocks to front-end; updated runCompiler
        to

commit a5ac1950a160195d92a0b3754979230beeb444fc
Merge: 6b9920a 37f5e28
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Tue Aug 12 13:16:38 2014 -0400

    Merge remote-tracking branch 'origin'

commit fd1a59733ee9059e1777222f992924f528dfe88f
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Tue Aug 12 14:21:41 2014 -0400

    Added block statement to syntax tree generation.

commit 4209d8e4ecf1fd6deb1565b6c231e50d2657df35
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Tue Aug 12 15:29:03 2014 -0400

    Fixed vector to return _ff. Added while loop.

commit 7550865709921c87c34336ba8f4999496da14427
Author: royaslan <roy.aslan@outlook.com>
Date:   Tue Aug 12 16:25:13 2014 -0400

    WHILE and IF are fully function and nesting correctly

commit cfa8e09a644192b03bc3c943040e6a29ac6f3702
Author: royaslan <roy.aslan@outlook.com>
Date:   Tue Aug 12 22:57:46 2014 -0400

    Function defintion is parsed/scanned - nothing meaningful yet

commit 01feaba4651c37b6b24b104f32b216a62b19e70d
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Tue Aug 12 23:10:37 2014 -0400

    Alex: A program is now a series of statements. Boolean operators working.
        Mu

commit f1079b285c49fe47bcc24a643cc070b6fd72d1d6
Merge: 01feaba cfa8e09
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Wed Aug 13 01:22:26 2014 -0400

    Added check for undeclared vars. Added curly braces to if/while output in
        c+

commit 64b958bb164d1d853689e0e928f751972ee880f6
```

```
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Aug 13 21:05:08 2014 -0400

    ENDIF implemented

commit a292f07f92e12117fff6ff0fc7ed4af935cddf5c
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Aug 13 21:34:02 2014 -0400

    IF and WHILE are terminating correctly

commit 8f48c7babc922b59e1774cfc9672a3754750133c
Author: royaslan <roy.aslan@outlook.com>
Date:   Wed Aug 13 21:37:46 2014 -0400

    IF and WHILE are terminating....

commit 4e9f85c8ecece56a7496b422ada6c219254e85ff
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Wed Aug 13 22:23:21 2014 -0400

    compilable code for if, nested if and while

commit 81563dca0994335f1df3fa4aa776e5e52ac81fe5
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Wed Aug 13 22:26:01 2014 -0400

    minor change curly braces

commit b50bef682ba58a99ef3d2421433991842bedbcb5
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Wed Aug 13 22:36:42 2014 -0400

    test goto *ptr cpp

commit 2b62d6f6287a625365a8acf1d5b45756df47acfe
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Wed Aug 13 22:44:27 2014 -0400

    curve ff file

commit 0863c2b281018d35028cd8ae1269949a555fa412
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 02:25:56 2014 -0400

    Functions can now be defined anywhere in the program

commit 78f3bdb8ab14ae320a861012477c5f3f321bbae5
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Thu Aug 14 02:32:06 2014 -0400

    vec2=vec2 assignment, vec2+vec2, vec2-vec2, bool vec2==vec2 and bool
        vec2!=v

commit f86cd6e01a3ecfe2bfb15d29a987eadac9c488b6
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Thu Aug 14 02:51:48 2014 -0400

    made some really minor changes, but pushing to avoid conflicts later
```

```
commit 22887d0a02d521c9ec3ae63bbbd59ac345c3b884
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 03:26:39 2014 -0400

    I'm pushing glut32.lib since it's need for g++ on windows

commit 970fdcee99cda626c4de8dc6e85689f1eecc5a16
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Thu Aug 14 04:08:02 2014 -0400

    sqrt implemented

commit 5b09553cf7c861be0015297dab5b79b6278d3dc5
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Thu Aug 14 04:27:13 2014 -0400

    OFF operator problem fixed

commit 7bd3b79890a51344595e3ef288b3a6eadcfbc5ee
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 14 10:32:25 2014 -0400

    Adding windows.h preprocessor code.

commit 05e8221c2f2df2549875c2ccf4768f09362fb925
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 14 10:44:07 2014 -0400

    runCompiler -c option now uses g++

commit b497b56bc60d0b3417fc01595d4dd23e8f9e3dd6
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 14 11:11:20 2014 -0400

    Added sin/cos of degrees

commit f68b71fbf702435383c16f8ca2a1a00e4602dd7f
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 14 11:52:38 2014 -0400

    Added x and y vector access.

commit 4fcaed541ecb54188ac4600afc9aa8d39e954084
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 14 13:27:06 2014 -0400

    updating regression tests

commit 1676ce1187cee07ad91a98b5c40127f052506746
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 14:15:57 2014 -0400

    Function definition extended

commit dd0059812543c521379b79d0456b14dad1f86995
Merge: 1676ce1 4fcaed5
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 14:16:15 2014 -0400

    Merge branch 'master' of https://github.com/prernaa/Firefly3D
```

```
commit 57f7d0238a088ccda260b030bd10f2de0c9b2a7d
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Thu Aug 14 14:37:57 2014 -0400

    fixed curly braces for fdefs

commit c791f13ec16bb3097333fae949bdab1203bd1350
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 14 14:48:02 2014 -0400

    Drawing circle sample

commit 823283cb5f725395090ed9b85d1f9de711982347
Merge: c791f13 57f7d02
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 14 14:48:37 2014 -0400

    Merge branch 'master' of github.com:prernaa/Firefly3D

commit cf9a901d8479ba0ce024edd5f869d172615651fd
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 14 14:50:55 2014 -0400

    Updating helloworld.ff

commit 56dd168b0e77588275125ec69e6ed264521b36ad
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 15:26:17 2014 -0400

    Function calls taking shape

commit e48d5cc30819baa4cacf996ba838d7a13387ab8d
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Thu Aug 14 15:42:19 2014 -0400

    solved func call curly braces issue AND PUSHED SAMPLE KOCH PROGRAM

commit faffc98ff53fd16424137586 77d62689568ad680
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 16:19:37 2014 -0400

    Pointer for return path implement but buggy

commit de1d656adda5533907129de953f2dced02f25981
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Thu Aug 14 19:09:22 2014 -0400

    Prettyprinting stacks. added more test source files

commit 00403ee30624c7762dbd8b54fd978d7ad6c44b61
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 20:44:21 2014 -0400

    Prepping recursion

commit 6094376defc6252b2a09ed2ff4a519e2ab2353ca
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Thu Aug 14 20:53:09 2014 -0400
```

```
    changed location of global var f. Added in struct and stack for function
        act

commit 7604efa12c31468c598605c77d989e233a974712
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 21:08:40 2014 -0400

    Switching to stack for return path

commit 56939d840035aa0a072183e1c43dbb8be26817c1
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Thu Aug 14 21:16:40 2014 -0400

    activation record stack push and pop done

commit b66e9d438286281c5c1b52fcf7a20c9eb21e2a56
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Thu Aug 14 22:00:11 2014 -0400

    recursion works

commit 178a0ba6f05a75d4a7662ad4ea03479b17507e4e
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 22:29:48 2014 -0400

    pushing testfuncitons

commit 92f976e1ded857e24d4690b8471067f1b59a1554
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 22:50:33 2014 -0400

    Single are added to syntax only

commit 503c2c8c3ec5aec6fd2663c1703925211db459d3
Author: royaslan <roy.aslan@outlook.com>
Date:   Thu Aug 14 22:57:01 2014 -0400

    Args are working in fun calls only syntacticly for now

commit 9cb715cc5168286935ffc30961e3d7dcdf97a965
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Fri Aug 15 10:58:36 2014 -0400

    House cleaning on semantic analyzer and overloaded vec2 ops.

commit 9cf8436463bdaffb400989160717d2636c36fb68
Author: Alex Shnayder <shnaydera@gmail.com>
Date:   Fri Aug 15 16:01:04 2014 -0400

    Rename firefly3D to firefly.

commit 53c1a4a1f11a4ed4c9cbfd4e83072746fe36e7a1
Author: royaslan <roy.aslan@outlook.com>
Date:   Fri Aug 15 20:33:54 2014 -0400

    Declaration of globals and _ti at the top is working

commit 242667362e903087e2fd2e8883abccef421cd88e
Author: royaslan <roy.aslan@outlook.com>
Date:   Fri Aug 15 20:51:57 2014 -0400
```

```
    Removed some of the (now) unneeded braces from IFs

commit 9e0690d344789671b7060553dbec3b582545061b
Author: royaslan <roy.aslan@outlook.com>
Date:   Fri Aug 15 21:14:30 2014 -0400

    Initializing cpp is now in an input file (input_head.ccp)

commit 5448eec1e5e34cee9038d2d232e93f25e86a2020
Author: royaslan <roy.aslan@outlook.com>
Date:   Fri Aug 15 21:58:20 2014 -0400

    Cleaned up code a bit

commit be38d652bb066f2682965abd6954443b53fd77a9
Author: royaslan <roy.aslan@outlook.com>
Date:   Fri Aug 15 22:44:32 2014 -0400

    function calls now check if function has been defined

commit cb676672cd0de4db5e3f02ea869673fd1d1e3ee6
Author: royaslan <roy.aslan@outlook.com>
Date:   Fri Aug 15 23:06:22 2014 -0400

    Another round of testing and fixing bugs

commit 75febe02637827793175a8628bba21cdd9e0cb9f
Author: royaslan <roy.aslan@outlook.com>
Date:   Sat Aug 16 00:12:35 2014 -0400

    Function calls can set variables (but this does not use activation records;

commit c889456e933da9c4ba9ca983ad64d7924e760259
Author: royaslan <roy.aslan@outlook.com>
Date:   Sat Aug 16 19:58:45 2014 -0400

    Function args (i.e., local vars) are 90% done; just need to offset stack by

commit 3a6e9067c04f6ffafcda0ad7151f902c0beafc17
Author: royaslan <roy.aslan@outlook.com>
Date:   Sat Aug 16 21:13:31 2014 -0400

    Args with recursion is finally working!

commit 37c4fed22063472677a6fcf0e6e0498f8a4c3069
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Sun Aug 17 06:54:39 2014 +0530

    fixed struct assign prob

commit c9b92158257929bad7a9379ac108b35257a0bdd7
Author: royaslan <roy.aslan@outlook.com>
Date:   Sat Aug 16 22:58:52 2014 -0400

    Frame pointer stack is behaving correctly

commit 76f51f7e7b5a5185052fde215194f8348c56d2e3
Author: #PRERNA CHIKERSAL# <PRERNA1@e.ntu.edu.sg>
Date:   Sun Aug 17 12:31:37 2014 +0530
```

```
    updated test suites

commit 71509ced5be2cde61094cd8be1e2c43499c4e7d8
Author: royaslan <roy.aslan@outlook.com>
Date:   Sun Aug 17 04:07:26 2014 -0400

    fixed bug with "pointer"

commit 1140e8e7922471b173336b5ffb0e162e39787157
Merge: 71509ce 76f51f7
Author: royaslan <roy.aslan@outlook.com>
Date:   Sun Aug 17 04:07:32 2014 -0400

    Merge branch 'master' of https://github.com/prernaa/Firefly3D
```

References

[1] "The logo programming language." http://el.media.mit.edu/logo-foundation/logo/programming.html.

[2] "Logo – wikipedia." http://en.wikipedia.org/wiki/Logo_(programming_language).

[3] "The cartesian coordinate system." http://www.mathsisfun.com/data/cartesian-coordinates.html.

[4] "Vector algebra – basics." http://www.math.utah.edu/online/2210/notes/ch13.pdf.

[5] "Github." https://github.com/.