EZIP: Easy Image Processing

Kevin Chiu   Tejas Nadkarni   Swati Kumar   Avanti Dharkar

{kgc2113, tgn2104, sak2144, ad2518}@columbia.edu

December 18, 2007

Contents

1 Introduction

It does not come as a surprise that we are surrounded by images. We use images for communication, transmitting information, creation of fiction etc. . In short we need image processing to understand, analyze and even change the world around us. Computer Vision techniques are widely applied in television, medicine, and even Hollywood movies.

We have developed efficient technologies to process digital images; however, Computer Vision remains a challenging domain. A deep understanding of the difficulties in the existing best practices has led us to the development of a suitable language for image processing. We call it EZIP, or Easy Image Processing.

1.1 Background

Image processing operations like smoothing and edge detection, and many more are very widely used in areas like Computer Vision. These operations are generally performed on images represented as matrices. We need a language using which can perform matrix operations like addition, subtraction, multiplication and convolution simply and easily.

The inspiration for our language stems from our own experiences with writing computer vision algorithms in C and C++. We noticed that while implementing simple image processing algorithms like blurring, edge detection, smoothing etc. most of our time was spent in writing the code rather than developing effective algorithms. In C and C++, the traditional implementation language for Computer Vision algorithms, the user is left to handle matrix operations using loops and conditional statements that make the code convoluted and difficult to read and understand.

The goal of our language is to provide the user with easy image manipulation techniques required for image processing. Generally image processing algorithms have an image and a kernel that works on that image. Our language is specifically designed to elegantly handle various kernel operations on the image and deliver an output image in a user-friendly manner. The user no longer has to worry about numerous unintuitive and error prone constructs, he only has to think about the operations he wants to perform on the image.

1.2 Related Works

1.2.1 Matlab

Matlab is a high-level language and interactive environment that enables the user to program complicated algorithms faster than with traditional programming languages such as C, C++, and Fortran. The image processing add on provides powerful libraries for image processing. It has native support for the class of Computer Vision algorithms that involve kernel convolutions and is also adeptly suited to performing global image operations such as histogram equalization and contrast enhancement.

However, Matlab is targeted at users working in the commercial or research fields who have plenty of funds to afford it. The cost of an individual license of Matlab exceeds the budget for many individual users. For an average individual looking to perform some basic image processing, Matlab is not a feasible option. Our language provides a suitable alternative for such users. At no cost, the user can perform basic image processing using few lines of code. The user need not even have much programming background since the syntax is intuitive.

1.2.2 C/C++

It is well established that C and C++ are extremely powerful and flexible languages for performing computationally intensive numerical operations. An experienced programmer can use C/C++ to implement complex image processing algorithms from first principles. However, implementation is not such a trivial matter for people with lesser programming skills. EZIP, although not as flexible as C/C++, is better suited to developers who need to quickly program Computer Vision algorithms without having to implement low level, elementary functions.

1.3 Goal

The goal of our language is to provide a simpler, more programmer-friendly way of doing image processing. We will make image processing accessible to the masses.

1.3.1 Ease-of-Use and Freedom

The most common use cases are de-noising, edge detection, and image enhancement. Most of these complex operations are accomplishable with only a few lines of code in our language. EZIP has a clear and concise way of defining matrix operations relevant to image processing. A user familiar with image processing can look at the code and easily understand the function of the program. Common data structures, such as kernels, are also be simple to create and straight forward to use.

Our language and implementation are free to use and modifiable by anyone under the MIT license. The combination of power and freedom will ensure that our language offers an attractive alternative to existing solutions. EZIP is also platform agnostic. It is a translated language targeted to Java, which runs on the vast majority of widely used platforms available today.

1.4 Main Language Features

In this subsection we describe some of main features in our language.

1.4.1 Image Operations

Elementary image operations are supported – convolution, addition, subtraction, division, and multiplication.

1.4.2 Pixel Access

The programmer can access individual pixels of an image with the following syntax.

```
image[x,y,"rgb"] = 123;
int a = image[x,y,"rgb"];
```

Where x and y are the x and y coordinates of the pixel you would like to read or assign to, and “rgb” is one of either “r”, “g”, or “b”, designating the color channel you would like to access.

1.4.3 Program Flow Control

The main statements of program flow control are implemented, including if, while, for, return and break.

1.4.4 User-defined Functions

Functions can be declared using the following syntax.

```
function name(type arg1, type arg2){

}
```

1.4.5 Internal Functions

Internal functions include print, view, load, save, sqrt, newImage, getWidth, getHeight, sin, and cos. Print prints ASCII representations of non-image objects to the command line. View shows a graphical representation of any image passed into it. Load and save handle loading images from, and saving images to, the disk. newImage creates a new image with a specified width and height. getWidth and getHeight examine an image and return the width or height respectively.

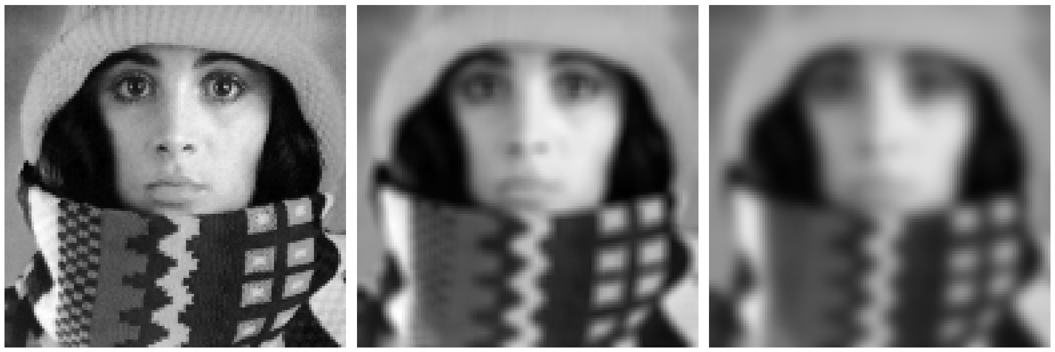


Figure 1: An example image blurred multiple times.

1.4.6 Code Sample

This sample code implements a blurring function similar to the one illustrated in Figure 1. The code gives an example of how we will comment code, make image declarations, make kernel declaration, and perform convolution. The syntax may change during implementation.

```
// blur image three times and then display

image i = load("myimage.jpg");
kernel k = {0,1,0:1,4,1:0,1,0};
image smoothed = i; int j = 0;
while(j < 3){
        smoothed = smoothed ** k; j = j+1;
}
view(smoothed);
```

2 Language Tutorial

In this section, you will learn how to use EZIP.

2.1 Getting Started

Write your program in any file. Suppose you write it in say example. Then to run your program go to the command prompt and type `java jar ezip.jar [your_file_name]` You can keep your file and the jar file in the same directory.

If your file is in a different directory, give the absolute path name from your file to the jar file. Once, your program runs, you can easily see if there are any errors on the command prompt and then go back and change the file. Save it and run it again, till you get no errors and the expected output is shown.

2.2 Variable Declaration and Assignment

Ezip has the following data types which can be used to declare a variable, namely int, double, boolean, kernel and image. You can use the following commands to declare variables:

- `int a;`
- `double b;`
- `boolean flag;`
- `kernel k;`
- `image i;`

Note that every declaration needs to be followed by a semicolon. We are using semicolon to terminate all lines of our program. You can also declare more than one variable separated by commas like `int a,b;`. Variables can be assigned some value after declaration. For e.g.

- `image i = load("xyz.bmp");` where xyz.bmp is the file name of the image on the local directory.
- `int a = 10;`
- `int a, b = 5;`
- `a = 7;`
- `kernel k = [10.0,2.0,4.0:3.0,5.0,7.0:6.0,8.0,0.0];` This is a 3X3 kernel where each row has been separated by commas. The kernel can have any number of rows and columns but each row should have the same number of elements. Also, each element of the kernel can be a double or an int or an expression that evaluates to a double or an int. Thus, kernel k = [5+9, 2-1,5+3]; is also acceptable. This is a 1X3 kernel having 1 row and 3 columns.
- `boolean b=true;` Boolean can take values of either true or false.
- `double d = 7.0`

Images can only take in values using the load function. We can also use the internal function `newImage(rows,columns)` to create a completely new image having the specified rows and columns. It works like `image newimg = newImage(30,30)` will create a new image having 30 rows and 30 columns with each pixel of the new image assigned the color black.

2.3 Arithmetic Operations

The binary arithmetic operations that can be performed are plus, minus, multiplication and division. The unary arithmetic operation is unary minus. For e.g.: if a and b are integers, then

```
int c = a+b; int c = a-b; int c = a/b; int c = a*b;
```

You can add a double to an integer and vice versa. When a double is added/subtracted/multiplied or divided to an integer, the result is a double. So, `double d = a+b` where `a` is an `int` and `b` is a `double`. You can perform arithmetic operations on images of the same size. For example,

```
image j = load("xyz.bmp");
image k = load("abc.bmp");
image result = i+j;
```

Please note that to add/subtract/multiply and divide integers or doubles to images, the image should be on the left side of the operator and the number on the right hand side. We can perform arithmetic operations only on images and numbers. The most important operator that image processing needs is convolution. We use ** to represent convolution. It can be used as follows.

```
image img1 = load("xyz.bmp");
kernel k = [1.0,1.0,1.0:1.0,1.0,1.0:1.0,1.0,1.0];
image averagedimg = img1 ** k;
```

The image to be convoluted needs to be on the left of the operator followed by the operator and the kernel. We dont perform any operations on kernels. To change the kernel values, you need to re-define the kernel or use expressions inside the kernel while assigning values.

2.4 Logical Operations

The logical operations are

!= > < >= <= < > ==

These can be used with numbers and booleans. They return a boolean after the operation. For example `int a = 4; boolean b = a<5;` These operations will be generally used in control flow statements.

2.5 Control Flow Statements

We have 6 types of control flow statements. “if statement”, “while loop” and “for loop”, “break”, “continue” and “return.” For example

```
if(a<5){ //do something} else { //do something }
```

We can also have an if without an else. The if statement takes a boolean value. The while loop can be used as follows.

```
int a = 0; while (a < 5) { //do something a= a+ 1;}
```

In this case the while loop iterates over a, 5 times and executes the body after each iteration. The condition part of the while loop ie.. a¡5 in this case, should also return a boolean. For loops can be used as follows.

```
for(i = 0; i< 5; i=i+1) {//do something}
```

In the for loop the three parts are separated by semicolons. You can also have just the condition and one or more parts empty. Thus, `for(;a<5;){//do something}` is a valid statement. The braces after the for loop are compulsory if any statements are part of the for loop, else if the for loop does not have a body, terminate it with a semicolon. The return statement can only be used inside functions. It is used to pass the control from the current function to previous calling function.

```
function xyz(int a)
{
        int b = a;
        print(b);
        if(b > 0){
                return;
        }
        print("This will not be printed");
}
        int a = 20;
        xyz(20);
        print("The function retuned");
```

The above will return back to the outer scope and print line “The function returned”. `b` is greater than zero, so the line “This will not be printed” will not be printed. The break statement will break out of a loop like a for loop or a while loop and execute the lines following the loop body. It can be used only inside loops.

```
for(i=1;i<5;i=i+1){
        print("This is printed only 3 times");
        if( i == 3){
                break;
        }
}
```

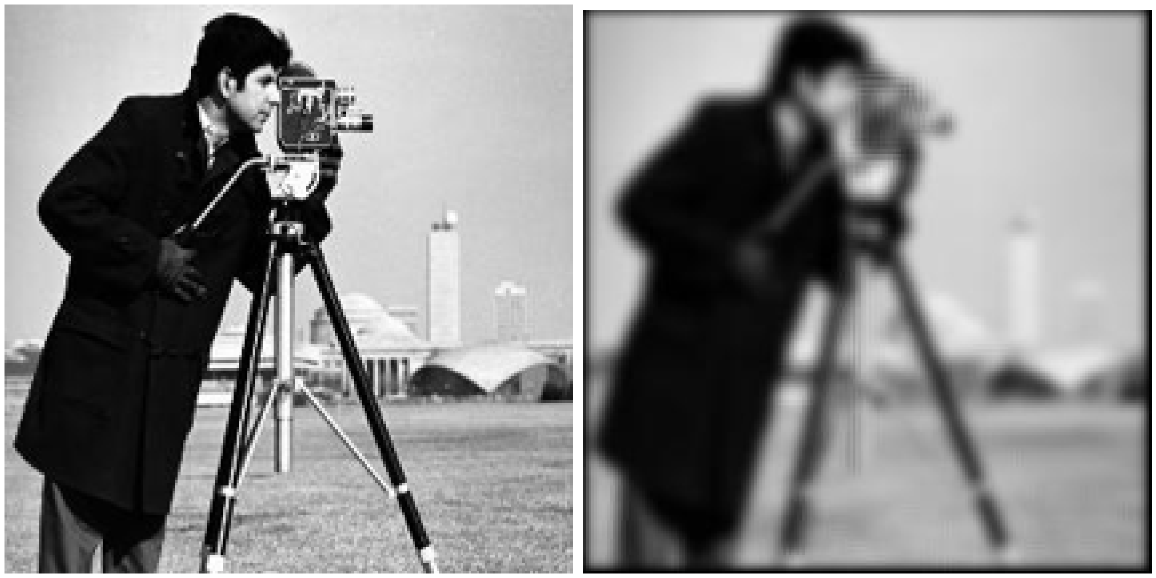
The statement “This is printed only 3 times” will be printed three times and then the control will break from the loop. The continue statement will jump to the condition check and execute the loop from the beginning.

```
while(i < 5){
        if(i == 3){
                continue;
        }else{
                print("Always prints it");
        }
```

We have some internal functions for working with images like load, view, print and save. These functions can be called directly and used as regular functions in your program. Thus, `load("abc.bmp")` loads an image into an image variable. `view(EzipImage i)` takes in an image and shows it on screen to the user. `print(EzipDataType d)` is used to print the value of any datatype except image and kernel.`save(EzipImage img, "xyz.bmp")` saves an image img to the file xyz.bmp. `newImage(width, height)` creates a new image with specified rows and columns, where width and height are integers. `getHeight(image img)` gives the number of rows in an image. `getWidth(image img)` gives the number of columns in an image. We also have some math functions like cos(double value), sin(double value) and sqrt(double value or integer value) for the ease of the user.

2.6 Some Simple Examples

2.6.1 Blurring An Image



```
function blur(image a,kernel k){
        int i;
        image result;
        for(i=0;i<5;i=i+1){
                result = a ** k;
        }
        view(result);
}
kernel k = [0.25,0,0.25:0,0,0:0.25,0,0.25];
image im = load("Hello World.bmp");
blur(im,k);
```

Here's the same process using recursion.

```
kernel blur = [0.25,0,0.25:0,0,0:0.25,0,0.25];
image im = load("cameraman.bmp");
function blur(image img,kernel k,int n){
        if(n>0)
                return blur(img,k,n-1)**k;
        else
                return img;
        }
image newimage = blur(im,blur,5);
view(newimage);
```

2.6.2 Median Filtering

```
function sort(image sortim){
int i,j,tmp;
for (i=0; i<8; i=i+1){
 for (j=0; j<8-i; j=j+1){
    if (sortim[j+1,0,"r"] < sortim[j,0,"r"]){
      tmp = sortim[j,0,"r"];
      sortim[j,0,"r"] = sortim[j+1,0,"r"];
      sortim[j+1,0,"r"] = tmp;
      }
 }
}
return sortim;
}

function median_filter(image img){
int i,j,r,c;

for(r = 1;r < getHeight(img)-1;r = r+1){
        for(c = 1;c < getWidth(img)-1;c = c+1){
                image sortimage = newImage(9,1);
                int countj = 0;
                for(i = -1;i <= 1;i = i+1){
                        for(j = -1;j <= 1;j = j+1){
                                sortimage[countj,0,"r"] = img[c+j,r+i,"r"];
                                countj = countj + 1;
                        }
                }
                image temp = sort(sortimage);
                for(i = -1;i <= 1;i = i+1){
                        for(j = -1;j <= 1;j = j+1){
                                img[c+j,r+i,"r"] = temp[4,0,"r"];
                        }
                }
        }
}
view(img);
}

image newim = load("cameraman.bmp");
view(newim);
median_filter(newim);
```

3 Language Manual

3.1 Lexical conventions

3.1.1 Comments

EZIP follows Cs's syntax for comments. Multiline comments start with `/*` and end with `*/`. Single line comments start with `//`.

3.1.2 Identifiers

An identifier is a sequence of alphanumeric characters and underscores. Identifiers must start with a letter. The identifiers are case sensitive. Identifiers have unlimited length.

3.1.3 Keywords

The following identifiers are used as keywords in EZIP. They may not be used for any other purpose.

```
for    image    function  return
if     kernel   for       AND
else   int      while     OR
true   double   break     NOT
false  boolean  continue
```

3.1.4 Constants

Ezip has `integer`, `double`, and String constants. `Integers` are a sequence of digits. `Doubles` are a sequence of digits followed by a period, followed by another sequence of digits. Strings are a sequence of characters surrounded by double quotes.

3.2 Types

- `int` values are integers from -2147483648 to 2147483647.
- `double` values are floating point numbers expressed in the format `(DIGIT)+.(DIGIT)+`. A `DIGIT` is a natural number between 0 and 9 inclusive.
- `boolean` values are either `true` or `false`.
- `image` is a proprietary type that is internally stored as a BufferedImage, but is exposed as a primitive data type. Individual pixels are accessible using bracket annotation, ID[x,y,"R"/"G"/"B"], to access to specific color pixels at the representative x and y coordinates of the image.
- `kernel` is a proprietary type that is internally stored as a matrix, but is exposed as a primitive data type.

3.3 Statements

3.3.1 Expression

A statement is an expression having the form

```
expression;
```

3.3.2 Block

A block is a type of statement that consists of several statements grouped together to form a block having the form

```
{ statement* }
```

3.3.3 Conditional

The format of the conditional statement is

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

In both cases, `statement` is executed when `expression` is `true`. The `else` is always associated with the last encountered `else`-less `if`.

3.3.4 While Loops

The format of the `while` statement is

```
while ( expression ) statement
```

The `statement` is executed repeated while the `expression` is `true`. The `expression` is checked before each execution of the `statement`.

3.3.5 For Loops

The format of the `for` statement is

```
for ( expression; expression; expression ) statement
```

The first `expression` is executed when the program enters the loop. The `statement` inside the loop continues to run as long as the second `expression` holds true. The value of the second `expression` before the `statement` is executed in each run of the loop. At the end of each loop, the third `expression` is evaluated.

3.3.6 Break

The format of the `break` statement is

```
break
```

The `break` statement can only be used inside loops, such as `for` or `while` statements. It terminates the innermost loop in which it is placed.

3.3.7 Continue

The `continue` statement can only be used inside loops. It skips the evaluation of any following statements

3.3.8 Return

The format of the `return` statement is

```
return expression;
return;
```

The `return` statement can only be used inside function definitions. It returns the value of the expression to the containing function's caller or exits the function if there is nothing to return.

3.3.9 Escape

The `escape` statement can only be used inside loops. Calling `escape` inside a loop breaks out of all enclosing loops.

3.3.10 Assignments

The format of an assignment is

```
ID = expression;
ID[x,y] = expression;
```

The expression on the right is evaluated, and the value resolved from that evaluation is bound to the ID on the left. In the case that the ID is an array, the array element associated with the provided index(es)

3.3.11 Function Calls

The format of an assignment is

```
ID( expression list )
```

The expressions inside the comma-separated `expression list` are passed to the function defined by the `ID` for execution. Functions are considered expressions. When a functions return type is `void` the value of the expression is `null`.

3.3.12 Declarations

The format of a declaration is

```
TYPE ID;
```

Declarations and instantiations can be combined. The format for the combined statement is

```
TYPE ID = expression;
```

where `TYPE` is `int`, `double`, `boolean`, `kernel`, or `image`. The `expression` on the right is instantiated and the result is bound to the `ID` on the left.

Declarations can also be chained in interesting ways. Instantiations can be chained and include an optional assignments.

```
TYPE (ID = expression,)* ID = expression;
```

3.4 Expressions

3.4.1 Primary Expressions

A primary expression can be an identifier (`ID`), constant (`image`, `kernel`, `integer`, `double`), boolean (`true`, `false`), an expression contained in parenthesis, or an element (`image[x,y]`). All expressions evaluate to a primary expression.

3.4.2 Unary Expressions

There is one unary operator, `-`, which negates the expression. The actual results of this operation are different for different types. It is not legal to negate a boolean value.

- `int` values are multiplied by -1.
- `double` values are multiplied by -1.

3.4.3 Arithmetic Expressions

The order of precedence from highest to lowest for operators on `integers` and `doubles` is

- `*` and `/`
- `+` and `-`

When `integers` and `doubles` are mixed in an expression, the expression is evaluated as a `double`.

The following list gives the order of precedence from highest to lowest for operators on `images` as well as explanations for their use.

- `**` takes an `image` on the left and a `kernel` on the right. Evaluating an expression containing a convolution involves executing a sum of products calculation over an image to produce a new image.
- `*` and `/` operators can be used in two ways with images. First, they can be used between a integer or double value and an image to multiply or divide the individual pixel entries in the image by the scalar value to produce a new image. Second, they can be used between two images to multiply or divide the corresponding pixel values in each image to produce a new image.
- `+` and `-` can be used in two ways with images. First, they can be used between a integer or double value and an image to add or subtract the individual pixel entries in the image to or from the scalar value to produce a new image. Second, they can be used between two images to add or subtract the corresponding pixel values in each image to produce a new image. In the case of subtraction, the image or scalar on the right is subtracted from the image or scalar on the left.

3.4.4 Logical Expressions

Logical expressions use the following operators to take pairs of boolean expressions and evaluate to boolean expressions, or in the case of `not`, a boolean to another boolean.

- `and` evaluates to `true` when both input expressions are `true` and `false` otherwise.
- `or` evaluates to `true` when either input expression is `true` and `false` otherwise.
- `not` evaluates to `true` when the input expression is `false` and evaluates to `true` otherwise.

3.4.5 Relational Expressions

The format of a relational expression is

```
EXPRESSION (relational operator) EXPRESSION
```

Both `EXPRESSION` statements must evaluate to the same type. The relational operators are as follows.

- `==` evaluates to `true` if both expressions resolve to equivalent values, and `false` otherwise. This operator can also be used to compare images. If both images have equal pixel values for every pixel defined in each image, then this evaluates to true, otherwise, it evaluates to `false`.
- != evaluates to `true` if both expressions resolve to unequal values, and `false` otherwise.
- `<` evaluates to `true` if the left expression resolves to a value less than the right expression, and `false` otherwise.
- `>` evaluates to `true` if the left expression resolves to a value greater than the right expression, and `false` otherwise.
- `<=` evaluates to `true` if the left expression resolves to a value less than or equal to the right expression, and `false` otherwise.
- `>=` evaluates to `true` if the left expression resolves to a value greater than or equal to the right expression, and `false` otherwise.

3.5 Function Definitions

The format of a function definition is

```
function (TYPE)? ID (((TYPE ID)(, TYPE ID)?)*) statement
```

A function begins with the keyword `function` followed by an optional `TYPE` specifier describing the return type of the function. The remainder of the function includes the `ID`, argument list, and function body.

3.6 Scope

3.6.1 Default Scope

If a variable is declared outside of a block it is assigned the default scope. The scope of this variable extends from where it is declared to the end of the file.

3.6.2 Local Scope

If a variable is declared inside of a block its scope is local to that block. The scope of this variable extends from where it is declared within the block to the end of the block.

3.6.3 Scope Conflicts

- A variable cannot be declared more than once in the current scope, not including surrounding or surrounded scopes.
- If a variable has been declared in a surrounding scope, that variable is accessible in the current scope.
- If a variable has been declared in a surrounding scope, redeclaring the variable in the current scope hides the variable that has been declared in the surrounding scope until execution leaves the bounds of the current scope.

4 Project Plan

4.1 Process

The process we used for planning, specification, development, and testing was we would schedule large blocks of time on the weekends to do an agile programming variant. When planning the meeting, we would come up with an agenda and expected artifacts. During the meeting, we would resolve any uncertainties presented in the agenda, and then implement the proposed solutions. At the end of the meeting, we would review our accomplishments, test the current implementation, and plan the agenda for the next meeting. This cycle of planning and execution would continue until the end of the project.

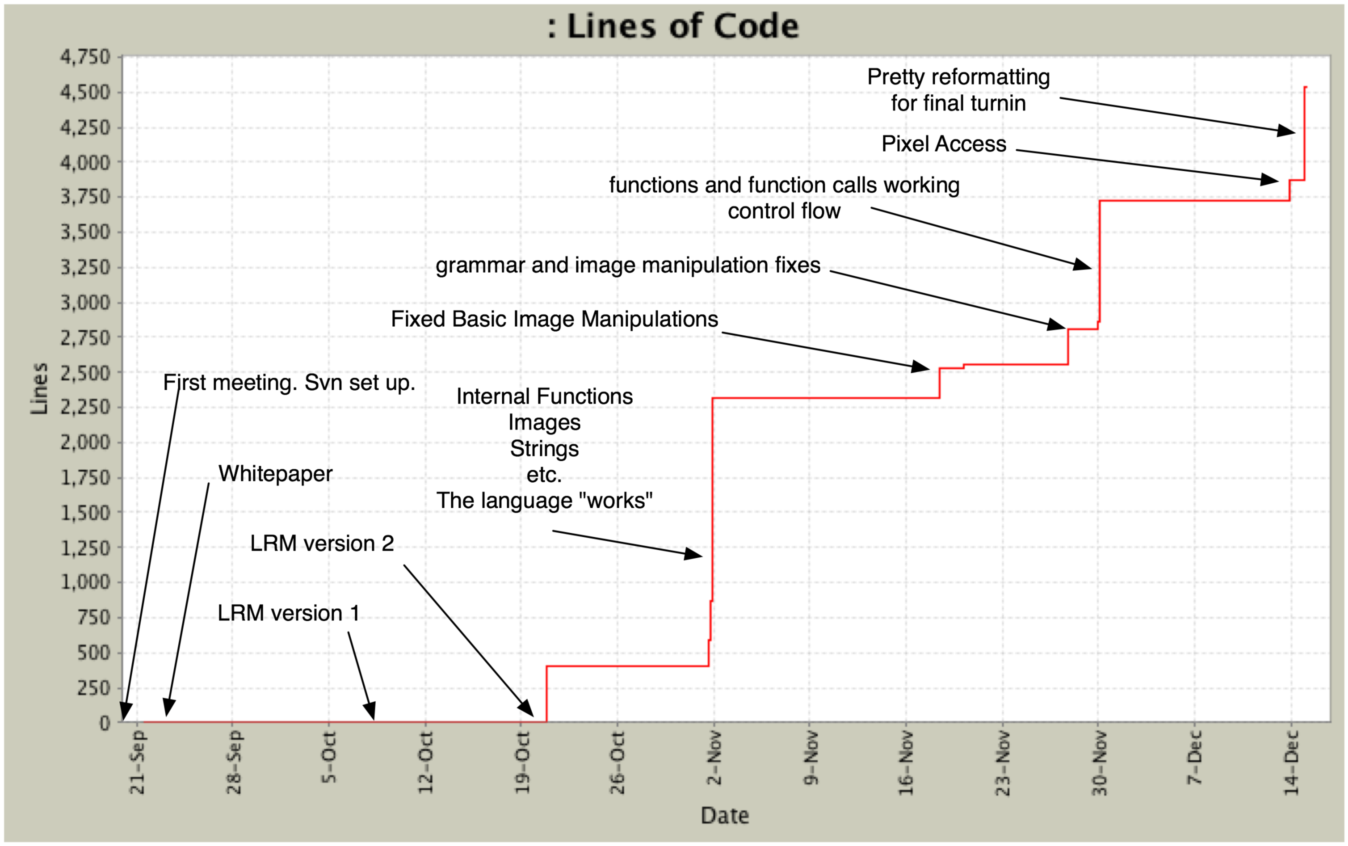
4.2 Programming Style

We used a conventional coding style commonly used in Java applications. Eclipse has a built-in automatic formatting setting that met our requirements, "Java conventions [built-in]." We also tried to keep Java source files at 80 columns width or less.

The following is a sample of java code using our coding conventions.

```java
/**
 * A sample source file for the code formatter preview
 */

package mypackage;

import java.util.LinkedList;

public class MyIntStack {
    private final LinkedList fStack;

    public MyIntStack() {
        fStack = new LinkedList();
        }

        public int pop() {
            return ((Integer) fStack.removeFirst()).intValue();
        }

        public void push(int elem) {
            fStack.addFirst(new Integer(elem));
        }

        public boolean isEmpty() {
            return fStack.isEmpty();
        }
    }
```

4.3 Timeline



4.4 Roles

- Tejas - front end lead

- Kevin - LaTeX, image algorithms, project support
- Swati - back end lead
- Avanti - testing lead

4.5 Tools and Languages

We used Eclipse with the ANTLR plugin. The languages we used to implement our translator were ANTLR, to generate the lexer, parser, and walker, and Java to implement the functions the walker calls.

4.6 Project Log

Full graphical log available at http://section7.org/ezip/log. Please beware the line-of-code counts cited in the development log. Many of them are wildly inaccurate.

```
------------------------------------------------------------------------
r126 | tejas | 2007-12-15 03:09:22 -0500 (Sat, 15 Dec 2007) | 1 line

fixed svn
------------------------------------------------------------------------
r125 | swati | 2007-12-15 02:50:40 -0500 (Sat, 15 Dec 2007) | 1 line

changed ezipimage and internal functions for error checking
------------------------------------------------------------------------
r124 | kevin | 2007-12-15 01:13:55 -0500 (Sat, 15 Dec 2007) | 1 line

final turnin
------------------------------------------------------------------------
r123 | kevin | 2007-12-15 01:03:39 -0500 (Sat, 15 Dec 2007) | 1 line

formatting for final report
------------------------------------------------------------------------
r122 | swati | 2007-12-15 00:05:31 -0500 (Sat, 15 Dec 2007) | 1 line

change for == and != to make ezipdouble compatible with ezipint
------------------------------------------------------------------------
r121 | tejas | 2007-12-14 23:56:15 -0500 (Fri, 14 Dec 2007) | 1 line

committing my latest code
------------------------------------------------------------------------
r120 | tejas | 2007-12-14 23:54:55 -0500 (Fri, 14 Dec 2007) | 1 line

restore EzipInterpreter.java
------------------------------------------------------------------------
r119 | kevin | 2007-12-14 23:38:24 -0500 (Fri, 14 Dec 2007) | 1 line
```

```
added safety to raster access
------------------------------------------------------------------------
r118 | kevin | 2007-12-14 23:32:04 -0500 (Fri, 14 Dec 2007) | 1 line

changed error messages for final report formatting
------------------------------------------------------------------------
r117 | kevin | 2007-12-14 23:30:00 -0500 (Fri, 14 Dec 2007) | 1 line

changed error messages for clarity
------------------------------------------------------------------------
r116 | swati | 2007-12-14 23:19:32 -0500 (Fri, 14 Dec 2007) | 1 line

change for == and != to make ezipdouble compatible with ezipint
------------------------------------------------------------------------
r115 | kevin | 2007-12-14 23:13:05 -0500 (Fri, 14 Dec 2007) | 1 line

formatting fixes
------------------------------------------------------------------------
r114 | swati | 2007-12-14 23:02:35 -0500 (Fri, 14 Dec 2007) | 1 line

Minor changes to ezipint and ezipdouble
------------------------------------------------------------------------
r113 | kevin | 2007-12-14 22:57:13 -0500 (Fri, 14 Dec 2007) | 1 line

fixed height/width issues
------------------------------------------------------------------------
r112 | tejas | 2007-12-14 22:23:47 -0500 (Fri, 14 Dec 2007) | 1 line

modified lrm3.tex
------------------------------------------------------------------------
r111 | tejas | 2007-12-14 19:55:59 -0500 (Fri, 14 Dec 2007) | 1 line

Added pixel access operator
------------------------------------------------------------------------
r110 | kevin | 2007-12-14 02:22:26 -0500 (Fri, 14 Dec 2007) | 1 line

sqrt overhaul, main change
------------------------------------------------------------------------
r109 | kevin | 2007-12-14 01:45:03 -0500 (Fri, 14 Dec 2007) | 1 line

new view function
------------------------------------------------------------------------
r108 | kevin | 2007-12-14 01:15:45 -0500 (Fri, 14 Dec 2007) | 1 line
```

```
removed old internal functions
------------------------------------------------------------------------
r107 | kevin | 2007-12-14 00:51:44 -0500 (Fri, 14 Dec 2007) | 1 line

sqrt now takes ints, get/set added to image class
------------------------------------------------------------------------
r106 | tejas | 2007-12-13 23:40:40 -0500 (Thu, 13 Dec 2007) | 1 line

changed something.
------------------------------------------------------------------------
r105 | kevin | 2007-12-13 23:34:42 -0500 (Thu, 13 Dec 2007) | 1 line

resoloved greyscale conflicts
------------------------------------------------------------------------
r104 | kevin | 2007-12-13 23:25:10 -0500 (Thu, 13 Dec 2007) | 1 line

additional internal functions + image datatype update - previous commit)
------------------------------------------------------------------------
r103 | kevin | 2007-12-13 23:19:55 -0500 (Thu, 13 Dec 2007) | 1 line

additional internal functions
------------------------------------------------------------------------
r102 | tejas | 2007-12-13 22:07:23 -0500 (Thu, 13 Dec 2007) | 1 line

moved test cases to test folder
------------------------------------------------------------------------
r101 | avanti | 2007-12-13 22:01:10 -0500 (Thu, 13 Dec 2007) | 1 line

Adding test case files
------------------------------------------------------------------------
r100 | tejas | 2007-12-13 21:26:49 -0500 (Thu, 13 Dec 2007) | 1 line

changed grammar
------------------------------------------------------------------------
r99 | kevin | 2007-12-13 21:25:08 -0500 (Thu, 13 Dec 2007) | 1 line

internal functions updated
------------------------------------------------------------------------
r98 | kevin | 2007-12-13 19:39:45 -0500 (Thu, 13 Dec 2007) | 1 line

 added save function + fixed print function
------------------------------------------------------------------------
r97 | swati | 2007-12-13 17:44:56 -0500 (Thu, 13 Dec 2007) | 1 line

Minor changes to datatypes file and added a check in internal functions
```

```
file for image print
------------------------------------------------------------------------
r96 | tejas | 2007-12-13 16:57:46 -0500 (Thu, 13 Dec 2007) | 1 line

Return now works.
------------------------------------------------------------------------
r95 | kevin | 2007-12-05 23:25:25 -0500 (Wed, 05 Dec 2007) | 1 line

headers for ezipimage and ezipinternalfunctions
------------------------------------------------------------------------
r94 | tejas | 2007-11-30 03:57:58 -0500 (Fri, 30 Nov 2007) | 1 line

Function Definitions and Function Calls now work.
------------------------------------------------------------------------
r93 | kevin | 2007-11-30 00:27:27 -0500 (Fri, 30 Nov 2007) | 1 line

final report checkin
------------------------------------------------------------------------
r92 | kevin | 2007-11-30 00:25:53 -0500 (Fri, 30 Nov 2007) | 1 line

fixing final report checkin
------------------------------------------------------------------------
r91 | tejas | 2007-11-29 23:47:44 -0500 (Thu, 29 Nov 2007) | 1 line

renamed EzipBool.java to EzipBoolean.java
------------------------------------------------------------------------
r90 | tejas | 2007-11-29 23:44:36 -0500 (Thu, 29 Nov 2007) | 1 line

Added break, continue functionality, scope handling.
------------------------------------------------------------------------
r89 | swati | 2007-11-29 20:03:05 -0500 (Thu, 29 Nov 2007) | 1 line

Changed double, int and boolean
------------------------------------------------------------------------
r88 | kevin | 2007-11-29 19:02:06 -0500 (Thu, 29 Nov 2007) | 1 line

image comparsion now does straight comparison between data buffers
------------------------------------------------------------------------
r87 | kevin | 2007-11-29 18:38:40 -0500 (Thu, 29 Nov 2007) | 1 line

image comparison no longer throws exception
------------------------------------------------------------------------
r86 | kevin | 2007-11-29 18:33:32 -0500 (Thu, 29 Nov 2007) | 1 line

grayscale works
```

```
------------------------------------------------------------------------
r85 | swati | 2007-11-29 05:02:25 -0500 (Thu, 29 Nov 2007) | 1 line

Added for statment
------------------------------------------------------------------------
r84 | swati | 2007-11-28 17:04:46 -0500 (Wed, 28 Nov 2007) | 1 line

Added while and if stmts
------------------------------------------------------------------------
r83 | swati | 2007-11-28 14:43:48 -0500 (Wed, 28 Nov 2007) | 1 line

Added while and if stmts
------------------------------------------------------------------------
r82 | tejas | 2007-11-27 23:56:04 -0500 (Tue, 27 Nov 2007) | 1 line

image stuff finally works MJUAHHAHAHAHAHAHHAHAHAHAHHAHA HAHAHAHAHAHHAHAH
A (evil laughter) - Kevin & Tejas
------------------------------------------------------------------------
r81 | kevin | 2007-11-27 22:53:03 -0500 (Tue, 27 Nov 2007) | 1 line

image algorithm tweaks - minor
------------------------------------------------------------------------
r80 | kevin | 2007-11-27 22:47:24 -0500 (Tue, 27 Nov 2007) | 1 line

checked in integer version of image class - instead of hex - TEJAS pffff
fft.
------------------------------------------------------------------------
r79 | tejas | 2007-11-27 20:05:43 -0500 (Tue, 27 Nov 2007) | 1 line

changed a few files
------------------------------------------------------------------------
r78 | kevin | 2007-11-27 19:58:44 -0500 (Tue, 27 Nov 2007) | 1 line

image functions updated
------------------------------------------------------------------------
r77 | kevin | 2007-11-27 19:56:31 -0500 (Tue, 27 Nov 2007) | 1 line

image functions updated
------------------------------------------------------------------------
r76 | kevin | 2007-11-27 19:50:33 -0500 (Tue, 27 Nov 2007) | 1 line

fixed stuff
------------------------------------------------------------------------
r75 | kevin | 2007-11-27 19:34:28 -0500 (Tue, 27 Nov 2007) | 1 line
```

```
resolved image
------------------------------------------------------------------------
r74 | tejas | 2007-11-27 19:23:28 -0500 (Tue, 27 Nov 2007) | 1 line

renamed EzipMatrix.java as EzipKernel.java
------------------------------------------------------------------------
r73 | tejas | 2007-11-27 19:22:22 -0500 (Tue, 27 Nov 2007) | 1 line

added to grammer
------------------------------------------------------------------------
r72 | kevin | 2007-11-27 19:04:37 -0500 (Tue, 27 Nov 2007) | 1 line

added print method, refactored internal functions
------------------------------------------------------------------------
r71 | kevin | 2007-11-27 19:01:16 -0500 (Tue, 27 Nov 2007) | 1 line

added print method
------------------------------------------------------------------------
r70 | kevin | 2007-11-27 18:34:53 -0500 (Tue, 27 Nov 2007) | 1 line

deleted old stuff
------------------------------------------------------------------------
r69 | kevin | 2007-11-27 18:31:59 -0500 (Tue, 27 Nov 2007) | 1 line

added nameless constructor and used where appropriate
------------------------------------------------------------------------
r68 | kevin | 2007-11-27 17:51:49 -0500 (Tue, 27 Nov 2007) | 1 line

removed old builtin methods, fixed current builtin methods, fixed images
.
------------------------------------------------------------------------
r67 | kevin | 2007-11-21 02:39:26 -0500 (Wed, 21 Nov 2007) | 1 line

image class updated - Can you check that I'm using the new constructor c
orrectly at the end of the long arithmetic functions?
------------------------------------------------------------------------
r66 | tejas | 2007-11-20 03:36:11 -0500 (Tue, 20 Nov 2007) | 1 line

Made changes to have Hello world working again.
------------------------------------------------------------------------
r65 | kevin | 2007-11-18 13:27:59 -0500 (Sun, 18 Nov 2007) | 1 line

fixed errors in image class
------------------------------------------------------------------------
r64 | kevin | 2007-11-18 12:59:37 -0500 (Sun, 18 Nov 2007) | 1 line
```

```
fixed errors in image class
------------------------------------------------------------------------
r63 | kevin | 2007-11-18 12:57:46 -0500 (Sun, 18 Nov 2007) | 1 line

fixed errors in image class
------------------------------------------------------------------------
r62 | kevin | 2007-11-18 12:57:09 -0500 (Sun, 18 Nov 2007) | 1 line

fixed errors in image class
------------------------------------------------------------------------
r61 | kevin | 2007-11-18 12:54:36 -0500 (Sun, 18 Nov 2007) | 1 line

fixed errors in image class
------------------------------------------------------------------------
r60 | kevin | 2007-11-18 12:50:28 -0500 (Sun, 18 Nov 2007) | 1 line

fixed errors in image class
------------------------------------------------------------------------
r59 | tejas | 2007-11-18 12:35:09 -0500 (Sun, 18 Nov 2007) | 1 line

Changed a few files.
------------------------------------------------------------------------
r58 | kevin | 2007-11-18 11:58:57 -0500 (Sun, 18 Nov 2007) | 1 line

added error checking to image class
------------------------------------------------------------------------
r57 | kevin | 2007-11-18 11:34:38 -0500 (Sun, 18 Nov 2007) | 1 line

filled in image class - added breaks
------------------------------------------------------------------------
r56 | kevin | 2007-11-18 11:21:31 -0500 (Sun, 18 Nov 2007) | 1 line

filled in image class
------------------------------------------------------------------------
r55 | kevin | 2007-11-18 10:20:42 -0500 (Sun, 18 Nov 2007) | 1 line

added stuff
------------------------------------------------------------------------
r54 | tejas | 2007-11-01 23:54:11 -0400 (Thu, 01 Nov 2007) | 1 line

Hello World now working properly
------------------------------------------------------------------------
r53 | tejas | 2007-11-01 23:52:02 -0400 (Thu, 01 Nov 2007) | 1 line
```

```
Moved remotely
------------------------------------------------------------------------
r52 | tejas | 2007-11-01 23:51:45 -0400 (Thu, 01 Nov 2007) | 1 line

Moved remotely
------------------------------------------------------------------------
r51 | tejas | 2007-11-01 23:51:36 -0400 (Thu, 01 Nov 2007) | 1 line

Moved remotely
------------------------------------------------------------------------
r50 | tejas | 2007-11-01 23:51:26 -0400 (Thu, 01 Nov 2007) | 1 line

Moved remotely
------------------------------------------------------------------------
r49 | tejas | 2007-11-01 23:51:17 -0400 (Thu, 01 Nov 2007) | 1 line

Moved remotely
------------------------------------------------------------------------
r48 | tejas | 2007-11-01 23:51:08 -0400 (Thu, 01 Nov 2007) | 1 line

Moved remotely
------------------------------------------------------------------------
r47 | tejas | 2007-11-01 23:50:03 -0400 (Thu, 01 Nov 2007) | 1 line

Docs folder created
------------------------------------------------------------------------
r46 | tejas | 2007-11-01 23:47:08 -0400 (Thu, 01 Nov 2007) | 1 line

Hello World is ready
------------------------------------------------------------------------
r45 | kevin | 2007-11-01 19:11:23 -0400 (Thu, 01 Nov 2007) | 1 line

added image and string classes
------------------------------------------------------------------------
r44 | kevin | 2007-11-01 19:10:52 -0400 (Thu, 01 Nov 2007) | 1 line

added image and string classes
------------------------------------------------------------------------
r43 | tejas | 2007-11-01 19:02:15 -0400 (Thu, 01 Nov 2007) | 1 line


------------------------------------------------------------------------
r42 | kevin | 2007-11-01 18:57:38 -0400 (Thu, 01 Nov 2007) | 1 line

updated built in functions
```

```
------------------------------------------------------------------------
r41 | kevin | 2007-11-01 18:43:30 -0400 (Thu, 01 Nov 2007) | 1 line

updated built in functions
------------------------------------------------------------------------
r40 | kevin | 2007-11-01 18:32:27 -0400 (Thu, 01 Nov 2007) | 1 line

updated built in functions
------------------------------------------------------------------------
r39 | kevin | 2007-11-01 18:29:22 -0400 (Thu, 01 Nov 2007) | 1 line

added built in methods
------------------------------------------------------------------------
r38 | tejas | 2007-11-01 17:50:30 -0400 (Thu, 01 Nov 2007) | 1 line

EzipDataType.java , EzipException.java , EzipSymbolTable.java added
------------------------------------------------------------------------
r37 | kevin | 2007-10-29 15:52:03 -0400 (Mon, 29 Oct 2007) | 1 line

updated final writeup
------------------------------------------------------------------------
r36 | kevin | 2007-10-29 15:46:09 -0400 (Mon, 29 Oct 2007) | 1 line

lrm2 updated for final writeup
------------------------------------------------------------------------
r35 | kevin | 2007-10-29 15:26:15 -0400 (Mon, 29 Oct 2007) | 1 line

lrm2 pdf
------------------------------------------------------------------------
r34 | tejas | 2007-10-20 22:43:04 -0400 (Sat, 20 Oct 2007) | 1 line

Added grammer.g file
------------------------------------------------------------------------
r33 | kevin | 2007-10-20 22:41:56 -0400 (Sat, 20 Oct 2007) | 1 line

lrm almost done
------------------------------------------------------------------------
r32 | kevin | 2007-10-20 18:02:43 -0400 (Sat, 20 Oct 2007) | 1 line

lrm2 underway
------------------------------------------------------------------------
r31 | kevin | 2007-10-20 13:26:19 -0400 (Sat, 20 Oct 2007) | 1 line

empty lrm2
------------------------------------------------------------------------
```

```
r30 | tejas | 2007-10-20 12:47:50 -0400 (Sat, 20 Oct 2007) | 1 line

modified the first few sections.
------------------------------------------------------------------------
r29 | kevin | 2007-10-20 12:35:24 -0400 (Sat, 20 Oct 2007) | 1 line

pdf
------------------------------------------------------------------------
r28 | kevin | 2007-10-20 12:34:51 -0400 (Sat, 20 Oct 2007) | 1 line

sdf
------------------------------------------------------------------------
r27 | kevin | 2007-10-14 13:43:10 -0400 (Sun, 14 Oct 2007) | 1 line

corrections
------------------------------------------------------------------------
r26 | swati | 2007-10-14 13:02:26 -0400 (Sun, 14 Oct 2007) | 1 line

Added conversion
------------------------------------------------------------------------
r25 | kevin | 2007-10-14 12:45:57 -0400 (Sun, 14 Oct 2007) | 1 line

middlings to nearing end of LRM
------------------------------------------------------------------------
r24 | kevin | 2007-10-14 12:36:12 -0400 (Sun, 14 Oct 2007) | 1 line

middlings to nearing end of LRM
------------------------------------------------------------------------
r23 | tejas | 2007-10-14 11:34:39 -0400 (Sun, 14 Oct 2007) | 3 lines

Added the Introduction.

-Tejas
------------------------------------------------------------------------
r22 | kevin | 2007-10-14 11:30:47 -0400 (Sun, 14 Oct 2007) | 1 line

middlings of LRM
------------------------------------------------------------------------
r21 | kevin | 2007-10-14 10:03:35 -0400 (Sun, 14 Oct 2007) | 1 line

beginnings of LRM
------------------------------------------------------------------------
r20 | kevin | 2007-10-13 14:44:36 -0400 (Sat, 13 Oct 2007) | 1 line

grammar
```

```
------------------------------------------------------------------------
r19 | kevin | 2007-09-24 17:55:06 -0400 (Mon, 24 Sep 2007) | 1 line

pdf done again
------------------------------------------------------------------------
r18 | kevin | 2007-09-24 17:34:30 -0400 (Mon, 24 Sep 2007) | 1 line

pdf done
------------------------------------------------------------------------
r17 | kevin | 2007-09-24 17:05:53 -0400 (Mon, 24 Sep 2007) | 1 line

text tweaked
------------------------------------------------------------------------
r16 | kevin | 2007-09-24 16:38:41 -0400 (Mon, 24 Sep 2007) | 1 line

text tweaked
------------------------------------------------------------------------
r15 | kevin | 2007-09-24 16:19:24 -0400 (Mon, 24 Sep 2007) | 1 line

pic centered
------------------------------------------------------------------------
r14 | swati | 2007-09-24 16:02:41 -0400 (Mon, 24 Sep 2007) | 1 line

Backgnd and inspiration added
------------------------------------------------------------------------
r13 | kevin | 2007-09-24 15:57:19 -0400 (Mon, 24 Sep 2007) | 1 line

woo
------------------------------------------------------------------------
r12 | kevin | 2007-09-24 12:08:37 -0400 (Mon, 24 Sep 2007) | 1 line

added some text to the whitepaper
------------------------------------------------------------------------
r11 | tejas | 2007-09-23 22:46:32 -0400 (Sun, 23 Sep 2007) | 1 line

Added the part on Matlab being too expensive
------------------------------------------------------------------------
r10 | kevin | 2007-09-21 16:36:56 -0400 (Fri, 21 Sep 2007) | 1 line

added whitepaper
------------------------------------------------------------------------
r9 | tejas | 2007-09-21 16:29:01 -0400 (Fri, 21 Sep 2007) | 1 line

I have resolved the conflict
------------------------------------------------------------------------
```

```
r8 | swati | 2007-09-21 16:26:36 -0400 (Fri, 21 Sep 2007) | 1 line

moving test3 to trunk
------------------------------------------------------------------------
r7 | swati | 2007-09-21 16:25:36 -0400 (Fri, 21 Sep 2007) | 1 line

Checking in test3
------------------------------------------------------------------------
r6 | swati | 2007-09-21 16:23:38 -0400 (Fri, 21 Sep 2007) | 1 line

ADDED FIELS
------------------------------------------------------------------------
r5 | tejas | 2007-09-21 16:14:35 -0400 (Fri, 21 Sep 2007) | 1 line

this is tejas
------------------------------------------------------------------------
r4 | kevin | 2007-09-21 16:12:44 -0400 (Fri, 21 Sep 2007) | 1 line

hi
------------------------------------------------------------------------
r3 | k | 2007-09-21 16:05:09 -0400 (Fri, 21 Sep 2007) | 1 line

added base folder structure
------------------------------------------------------------------------
r2 | t | 2007-09-21 16:03:04 -0400 (Fri, 21 Sep 2007) | 1 line

Test 2 ver 1
------------------------------------------------------------------------
r1 | k | 2007-09-21 15:26:07 -0400 (Fri, 21 Sep 2007) | 1 line

first checkin
------------------------------------------------------------------------
```

5 Architectural Design

The lexer, parser and tree walker are implemented using Antlr. They are contained in a single file called grammar.g. grammar.g contains three sections where EzipLexer, EzipParser and EzipWalker are implemented. The lexer takes in the source program and outputs a stream of tokens that are given to the parser.

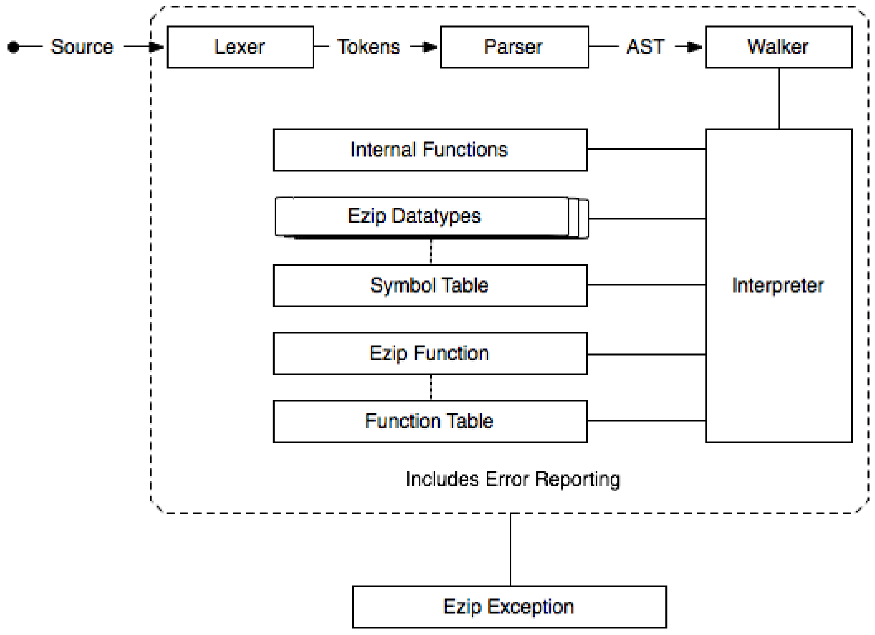
The parser parses these tokens and creates an Abstract syntax tree that is given to the walker.

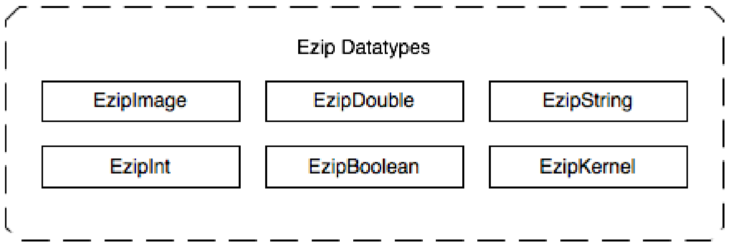
The source file of EzipWalker contains calls to functions defined in EzipInterpreter.java for implementing the walker.

The class EzipException is used for delivering all error messages. It is used by EzipFunction, EzipInterpreter, EzipDataType, EzipSymbolTable and EzipFunctionTable. It is also used by the parser and walker.

The back end consists of five major classes namely, the data type classes used in our language, internal functions, user-defined functions, symbol table and function table.

5.1 Block Diagram





5.2 Datatypes

These are derived from a single class called EzipDataType.java that defines the common functions and error handling for individual data type classes. The individual data type classes of EzipInt, EzipDouble, EzipBoolean, EzipString, EzipKernel and EzipImage override relevant functions in the EzipDataType.

The errors can be specific to a particular data type or it can be a common error for all data types. If the error is common to all data type classes then, it is defined in EzipDataType and creates an instance of EzipException otherwise the individual data type classes override the error function and create an instance of EzipException.

The data type classes of EzipInt, EzipDouble, EzipBoolean and EzipString are just wrapper classes of Java element types.

EzipKernal defines a kernel used to operate on an image. It internally used the Java Kernel class having height, width and an array of floating point values.

EzipImage defines an image. It internally uses the Java BufferedImage class.

5.3 Ezip Interpreter

EzipInterpreter contains all the helper functions needed by the walker to walk the tree and perform some action. It is used for creating new variables and functions, implementing control flows, function calls and managing symbol table and function table.

5.4 Internal Functions

Internal functions are written for the users convenience. These functions are the most commonly used functions like load, view, save and print. They are pre-defined in our language and have some specific syntax.

5.5 Symbol Table

The symbol table is used to keep track of name-value pairs for all the data types and variables defined in the user program. It is internally implemented as a list of hashtables where each hashtable stores the name and values of the variable within a particular scope. It has functions to store newly defined variables in the hashtable and update already defined ones. It also searches if the variable has been defined in the parents of the current scope, thus, implementing open scoping rules.

5.6 Ezip Function and Function Table

This class handles function execution, recursion, function return and places the functions in the function symbol table. We have global scope for all functions, so whenever a function gets created, it is added to the function table which is implemented just as a hashtable. The function table just stores the function name and the function itself.

5.7 Credits

5.7.1 Tejas

```
grammar.g (lexer, parser, walker)
Main.java
EzipDataType.java
EzipInterpreter.java
EzipSymbolTable.java
EzipKernel.java
EzipException.java
EzipFunctionTable.java
```

5.7.2 Swati

```
grammar.g (lexer, parser, walker)
EzipInt.java
EzipBoolean.java
EzipDouble.java
EzipString.java
EzipFunction.java
```

5.7.3 Kevin

```
EzipImage.java
EzipInternalFunctions.java
grammar.g (first version)
LaTeX, SVN (project support)
```

5.7.4 Avanti

```
Testing
grammar.g (lexer, parser)
```

All team members contributed to documentation.

6 Test Plan

6.1 Goals

No matter how well the language has been designed and coded, it will inevitably still contain defects. Testing is the process of executing it with the intent of finding faults (bugs). A successful test is one that finds errors, not one that doesnt find errors. Although testing can prove the presence of faults it cannot prove their absence but it can definitely increase confidence that our language works.

We tested our system under white box and black box testing. We tried to write efficient test cases which could test our language thoroughly and systematically so that we could be sure that the language was performing as desired.

6.2 Methods

The Ezip group wanted to run tests in parallel with each of three design phases. The tests are cumulative, so that each phase will integrate the tests in all phases previous to it with its tests.

6.2.1 Unit Testing of Individual Code Modules

When the essential components of the Ezip interpreter were being built, we used Unit Testing. Thus, when classes like EzipInt, EzipDouble, EzipBoolean were designed we used unit testing with white box so that we could test every method and every code branch inside every method.

6.2.2 Integration Testing with Correlation of Modules

At this point, all the major classes have been written and pieced together in their final arrangement. We wanted to ensure that the classes were collaborating as desired. Thus after all the classes were coded, we checked their interaction to see if its was working right. For ex: we checked if it was possible to add an int (from the class EzipInt) to an Image (from the class EzipImage). This is the phase where regression tests become important. As inconsistencies are discovered and corrected throughout the code, it was imperative that the flow of data was not affected.

Further, as our interpreter began to accept more complete grammars, we did not want to lose functionality on the initial grammars that were parsed correctly.

6.2.3 Black Box Testing

Finally after all the unit testing and integration testing is done, we perform black box testing at the system acceptance level to ensure that the entire system is working promptly. If was found necessary, the interpreter was modified as a result of these tests.

6.3 Tools

We used JUnit extensively for testing.

6.4 JUnit Testing

We tested Ezips lexer, Parser and Walker extensively for lexical, syntactic and semantic errors respectively.

6.4.1 Testing EzipLexer

For testing the EzipLexer we checked the various tokens. Ex: Checking for relational operators:

```
assertValidInput(">=",EzipLexer.GE);
assertValidInput("<=",EzipLexer.LE);
```

Checking for string.

```
assertValidInput("\"he said \"\"They liked you\"\"\"",EzipLexer.STRING,"he said \"They liked you\"");
```

Checking for comments. (simple and multiline)

```
assertValidInput("myVar=5; //my inline comment\n;",expectedTypes);
assertValidInput("num=5; /*my inline comment\ncarrieson*/;",expectedTypes);
```

The source code for testing the walker is included in EzipLexerTest.java.

6.4.2 Testing EzipParser

For checking the parser we had a set of valid and a set of invalid production rules. Examples of valid production rules:

```
assertProductionOk("program", "image im ;kernel k = [2,3,4:2,3,4:2,3,4];int i = 5; double d; boolean
assertProductionOk("func_def", "function f1(image im, int i) {}");
assertProductionOk("func_def", "function f1(double im, int i) {if(i ==0) {i=1; im =9.9;}}");
assertProductionOk("for_stmt", "for(i=1;i<5;i=5);");
assertProductionOk("break_stmt", "break;");
assertProductionOk("decl_stmt", "kernel k =[2.3,23,4.5,34:2.3,23,4.5,34:2.3,23,4.5,34];");
assertProductionOk("decl_stmt", "image im = 1;");
assertProductionOk("if_stmt", "if(i == 5){int j = 0; i = i+ 1;break;}else {int k = 0;}");
assertProductionOk("while_stmt", "while(i<5);");
assertProductionOk("while_stmt", "while(i<5) {i = i+1;}");
assertProductionOk("continue_stmt", "continue;");
```

Examples of invalid production rules:

```
assertProductionNotOk("program", "i");
assertProductionNotOk("program", "function f() {i = i+ 1}");
assertProductionNotOk("func_def", "function f1(double im) {if(i = 0) {i=1; im =9.9;}}");
assertProductionNotOk("for_stmt", "for(i=1;i<5;i=5)");
```

Along with this we had a set of test files which we checked, using a function, for correct parsing. The source code for testing the walker is included in EzipParserTest.java and BaseParserTest.java.

6.4.3 Testing the Walker

For Testing the walker we had various test files defined.

The test files that were checked for correctness in parsing were now checked for static semantic analysis.

We even had a different set of files which were syntactically right but semantically wrong. These included examples like uninitialized variables, convolution of an int with another int, incorrect design of loops etc. These parsed correctly but showed errors in semantics.

The source code for testing the walker is included in FunctionTest.java and BaseWalkerTest.java. All the test files are also attached.

6.5 Credits

- Testing Lead - Avanti
- Ad Hoc Testing - Tejas, Swati, Kevin

7 Lessons Learned

7.1 Individual Lessons

7.1.1 Kevin

I learned that projects proceed more smoothly if the team concentrates on the difficult, uncertain parts of the project first, so that lessons learned from challenges at the beginning of the project can be used to expedite the completion of the remainder of the project.

Additionally, I learned the basic structure of a language interpreter (parser, lexer, walker) as well as a useful tool for making one (ANTLR).

7.1.2 Tejas

Previous to this project, I had never been exposed to version control software such as CVS or SubVersion. The most sophisticated version control I used prior to this was creating a hierarchy of backup directories. Working with Subversion on this project has made me appreciate its features. Now I tend to use Subversion even with my individual projects.

I discovered how manipulation of objects by reference can lead to unexpected results. Many of the bugs occurring in our program were caused since two functions at different locations were acting on different references to same object.

7.1.3 Swati

One important lesson I learned was to create a parent class for all the common functions to be used by other classes. The EzipDatatype class contains many common functions that are implemented by all other data type classes. Hence having a main interface for repeated code helps in making the code more robust and bug free. Also, once the grammar had been decided it should not be changed because a small change in the grammar may result in other unseen problems. Lastly, defining the goal is not enough, we need to clearly state the path to be taken to reach the goal because, the same goal can be implemented in many different ways and which the shortest and easiest path may not be visible at that moment. So, getting help from the people who have already implemented it, makes the task much more easier.

7.1.4 Avanti

With the other deadlines which were much closer, I always procrastinated work related to this project. I realized that I should have managed my time better by being regular with this project.

I also realized that its difficult to coordinate a group of four people. Its important that everyone should always communicate via email so that there is no communication gap.

7.1.5 Advice for Future Teams

Although it is necessary to start early, make sure you have a strong design which makes sense. Syntax issues should be sorted out early. Changing the syntax at a later stage can be much more expensive.

It is very difficult to divide the entire project into 4 parts and to work on them separately. Instead, plan to meet every week and divide the week's tasks into the 4 members.

Concentrate most of your effort on the compiler and not implementing the backend. Use java libraries for the backend instead of writing your own functions. The whole point of the project is building a compiler and not how fancy or complicated the backend is.

Start early. Use version control.

A Code

Interpreter

```
 Lines   Words   Chars    File
    52     173    1518    EzipBoolean.java
   115     409    3050    EzipDataType.java
    81     256    2376    EzipDouble.java
     7      19     183    EzipException.java
   100     279    2913    EzipFunction.java
    26      74     637    EzipFunctionTable.java
   259    1039    8214    EzipImage.java
   123     347    3695    EzipInt.java
   203     762    6739    EzipInternalFunctions.java
   204     665    5982    EzipInterpreter.java
    53     109     947    EzipKernel.java
    22      58     495    EzipString.java
    71     215    1842    EzipSymbolTable.java
    68     235    2134    Main.java
   399    1308    9905    grammar.g
  1783    5948   50630    total
```

Testing

```
 Lines   Words   Chars     File
     8      24     124     testAddition.txt
     6      19      99     testAssign.txt
    23      54     264     testConvolution.txt
     8      25     117     testDivision.txt
    10      36     204     testImageAddition.txt
    14      48     245     testImageDivision.txt
    16      54     282     testImageMultiplication.txt
    16      48     250     testImageSubtraction.txt
     6      15      71     testIncorrect1.txt
     0       6      20     testIncorrect2.txt
     2       6      39     testIncorrect3.txt
     2      12      55     testIncorrect4.txt
     3      13      53     testIncorrect5.txt
     3      28     159     testIncorrect6.txt
     4      14      61     testIncorrect7.txt
     8      25     118     testMultiplication.txt
     8      25     119     testSubtraction.txt
    99     806    6364     test_plan.tex
   376     518    5152     BaseParserTest.java
    67      94     954     BaseWalkerTest.java
   250     352    4075     EzipLexerTest.java
   111     236    2813     EzipParserTest.java
```

```
  279     544    6219     FunctionTest.java
 1319    3002   27857     total
```

Demos

```
Lines   Words   Chars     File
   44     114     773     kernel.txt
   42      75     870     histogram.txt
   28      71     466     integer.txt
   26      56     509     threshold.txt
   23      41     374     invert.txt
  163     357    2992     total
```

A.1 Ezipboolean.java

Swati Kumar

```
public class EzipBoolean extends EzipDataType{
            private Boolean value = null;
            String type = "bool";

            EzipBoolean( boolean arg_value ) {
                super( null , "boolean" );
                value = new Boolean(arg_value);
            }

            EzipBoolean( String name ){
                super( name , "boolean" );
            }

            public String getType() {
                return this.type;
            }
            public Boolean getValue()
            {
                        if(this.value==null)
                                nullError();
                        return this.value;
            }
            public EzipDataType and( EzipDataType b ) {
                if ( b instanceof EzipBoolean )
                    return new EzipBoolean( this.getValue() && ((EzipBoolean)b).getValue
                return error( b, "AND" );
            }
            public EzipDataType or( EzipDataType b ) {
                if ( b instanceof EzipBoolean )
                    return new EzipBoolean( this.getValue() || ((EzipBoolean)b).getValue
                return error( b, "OR" );
            }

            public EzipDataType not() {
                return new EzipBoolean( !this.getValue() );
            }

            public EzipDataType equi( EzipDataType b ) {
                if ( b instanceof EzipBoolean )
                    return new EzipBoolean( ( this.getValue() == ((EzipBoolean)b).getVal
                return error( b, "==" );
            }

            public EzipDataType notequi( EzipDataType b ) {
                if ( b instanceof EzipBoolean )
                    return new EzipBoolean( ( this.getValue() !=((EzipBoolean)b).getValu
                return error( b, "!=" );
            }

}
```

A.2 EzipDataType.java

Tejas Nadkarni

```
public class EzipDataType {

        protected String name ;
        protected String type ;

        public EzipDataType() {
                name = null ;
                type = "undefined" ;
        }

        public EzipDataType(String arg_name) {
                name = arg_name ;
                type = "undefined" ;
        }

        public EzipDataType(String arg_name, String arg_type){
                name = arg_name ;
                type = arg_type ;
        }

        /*This is not required if the constructor is called*/
        public void setName(String arg_name){
                name = arg_name ;
        }
        /*This is not required if the constructor is called*/
        public void setType(String arg_type){
                type = arg_type;
        }

        public String getType() {
                return type;
        }

        public String getName() {
                return name;
        }

        public Object getValue(){
                return null;
        }

        public void nullError(){
                throw new EzipException("<"+getType()+">"+name+"␣has␣not␣been␣initialize
        }

        /*For unary operators or when the second operand for a binary operator is NULL*/
        public EzipDataType error( String operator ) {
        throw new EzipException( "illegal␣operation:␣" + operator
                                + "(␣<" + getType() + ">"
                                + ( name != null ? name : "<?>" )
                                + "␣)" );
    }

        public EzipDataType error(EzipDataType b, String operator){
                if(null == b)
                        return error(operator);
        throw new EzipException(
                "Illegal␣operation:␣"
                + "␣<" + getType() + ">␣"
                + ( getName() != null ? getName() : "" )
                + "␣" + operator + "␣"
                + "<" + b.getType() + "> "
                + ( b.getName() != null ? b.getName() : "" ));
        }

    public EzipDataType assign( EzipDataType b ) {
        return error( b, "=" );
    }
    public EzipDataType uminus() {
        return error( "-" );
    }
    public EzipDataType plus( EzipDataType b ) {
        return error( b, "+" );
    }
    public EzipDataType minus( EzipDataType b ) {
        return error( b, "-" );
    }
    public EzipDataType mult( EzipDataType b ) {
        return error( b, "*" );
    }
    public EzipDataType conv( EzipDataType b ) {
        return error(b, "**" );
    }
    public EzipDataType div( EzipDataType b ) {
        return error(b, "/");
    }
    public EzipDataType gt( EzipDataType b ) {
        return error(b, ">");
    }
    public EzipDataType ge( EzipDataType b ) {
        return error(b, ">=");
    }
    public EzipDataType lt( EzipDataType b ) {
        return error(b, "<");
    }
    public EzipDataType le( EzipDataType b ) {
        return error(b, "<=");
    }
    public EzipDataType equi( EzipDataType b ) {
        return error(b, "==");
    }
    public EzipDataType notequi( EzipDataType b ) {
        return error(b, "!=");
    }
    public EzipDataType and( EzipDataType b ) {
        return error(b, "AND");
    }
    public EzipDataType or( EzipDataType b ) {
        return error(b, "OR");
    }
    public EzipDataType not() {
        return error("NOT");
    }
}
```

A.3 EzipDouble.java

Swati Kumar

```
public class EzipDouble extends EzipDataType{
        private Double value = null;

        public EzipDouble( double arg_value ) {
        super( null , "double" );
        value = new Double(arg_value);
    }

        public EzipDouble( String name ){
        super( name , "double" );
    }
    public String getType() {
        return "double";
    }
    public static double doubleValue( EzipDataType b ) {
        if ( b instanceof EzipDouble )
            return ((EzipDouble)b).getValue();
        if ( b instanceof EzipInt )
            return (double) ((EzipInt)b).getValue();
        b.error( "cast to double" );
        return 0;
    }

    public Double getValue(){
                if(this.value==null)
                        nullError();
                return this.value;
    }

    public EzipDataType uminus() {
        return new EzipDouble( -this.getValue() );
    }
    public EzipDataType plus( EzipDataType b ) {
        return new EzipDouble( this.getValue() + doubleValue(b) );
    }

    public EzipDataType minus( EzipDataType b ) {
        return new EzipDouble( this.getValue() - doubleValue(b) );
    }

    public EzipDataType mult( EzipDataType b ) {
        return new EzipDouble( this.getValue() * doubleValue(b) );
    }

    public EzipDataType div( EzipDataType b ) {
        return new EzipDouble( this.getValue() / doubleValue(b));
    }
    public EzipDataType gt( EzipDataType b ) {
        return new EzipBoolean( this.getValue() > doubleValue(b) );
    }

    public EzipDataType ge( EzipDataType b ) {
        return new EzipBoolean( this.getValue() >= doubleValue(b) );
    }

    public EzipDataType lt( EzipDataType b ) {
        return new EzipBoolean( this.getValue() < doubleValue(b) );
    }

    public EzipDataType le( EzipDataType b ) {
        return new EzipBoolean( this.getValue() <= doubleValue(b) );
    }

    public EzipDataType equi( EzipDataType b ) {
        if(b instanceof EzipDouble)
                return new EzipBoolean( this.getValue().doubleValue() ==
(((EzipDouble)b).getValue()).doubleValue());
        error(b,"==");
        return null;
    }

    public EzipDataType notequi( EzipDataType b ) {
        if( b instanceof EzipDouble)
                return new EzipBoolean( this.getValue().doubleValue() != ((EzipDouble)b)
        error(b,"!=");
        return null;
    }
}
```

A.4 EzipException.java

Tejas Nadkarni

```
public class EzipException extends RuntimeException{
    EzipException( String msg ) {
        System.err.println( "Ezip Error : ");
        System.err.println( msg);
    }
}
```

A.5 EzipFunction.java

Swati Kumar

```
import antlr.collections.AST;
import java.io.PrintWriter;


public class EzipFunction {

        String name;
        EzipDataType[] arg ;
    AST body ;
    EzipSymbolTable parent ;
    boolean isInternal ;
    int id ;

    public String getName() {
                return name;
        }

        EzipFunction( String name, EzipDataType[] arg, AST body, EzipSymbolTable parent)
        this.name = name;
        this.arg = arg;
        this.body = body;
        this.parent = parent;
    }
    /*for internal functions*/
    EzipFunction( String name, int id ) {
        this.name = name;
        this.arg = null;
        this.id = id;
        parent = null;
        body = null;
    }

    public boolean isInternal(){
        return (this.body == null);
    }

    public final int getInternalId() {
        return id;
    }

    public String getFullName(){
        StringBuffer fullName = new StringBuffer("");
        fullName = fullName.append(this.name)
                                             .append("(");
        for( int i=0; i<arg.length ; i++ )
                if(i==0)
                        fullName = fullName.append("␣").append(arg[i].getType()).append(
                else
                        fullName = fullName.append(",␣").append(arg[i].getType()).append

        fullName = fullName.append("␣)");
        return fullName.toString();
    }

    public void execute(EzipDataType[] value, EzipWalker walker, EzipInterpreter ezip) t
        ezip.enterScope();
        EzipSymbolTable st = ezip.getSymbolTable();

        if(arg.length != value.length)
                throw new EzipException("Call␣to␣"+getFullName()+"␣requires␣"+arg.length
        for(int i=0 ; i<arg.length ; i++)
                if ( arg[i].getClass() == value[i].getClass() ){
                        value[i].setName(arg[i].getName());
                        st.addVariable(value[i]);
                }
                else
                        throw new EzipException("Trying␣to␣assign␣<"+value[i].getType()+
        walker.stmt(this.body);
        ezip.exitScope();
    }
    public void print( PrintWriter w ) {
        if ( body == null )
        {
            w.println( name + " = <internal-function> #" + id );
        }
        else
        {
            if ( name != null )
                w.print( name + " = " );
            w.print( "<function>(" );
            for ( int i=0; ; i++ )
            {
                w.print( arg[i] );
                if ( i >= arg.length - 1 )
                    break;
                w.print( "," );
            }
            w.println( ")" );
        }
    }
    public EzipSymbolTable getParentSymbolTable() {
        return this.parent;
    }

    public EzipFunction getCopy(){
        return new EzipFunction(this.name,this.arg,this.body,this.parent);
    }
}
```

A.6 EzipFunctionTable.java

Tejas Nadkarni

```
import java.util.Hashtable;

public class EzipFunctionTable extends Hashtable<String, EzipFunction>{

        // This variable needs to added since this class extends another class
        static final long serialVersionUID = 1;

        public EzipFunction getFunction(String name){
                EzipFunction func;

                func = get(name);
                if (func == null)
                        throw new EzipException("Function "+name+" has not been declared
                return func;
        }

        public void addFunction(EzipFunction func){
                String name = func.getName();

                if (name == null)
                        throw new EzipException("Function name not declared ");

                put (name, func);
        }
}
```

A.7 EzipImage.java

Kevin Chiu

```
/*
 * EzipImage.java : Image algorithms, operations, and data structures for EZIP.
 *
 * @author Kevin Chiu kgc2113@columbia.edu
 */


import java.awt.image.BufferedImage;
import java.awt.image.ConvolveOp;
import java.awt.image.DataBuffer;
import java.awt.image.Kernel;
import java.awt.image.Raster;
import java.awt.image.WritableRaster;

public class EzipImage extends EzipDataType {
        private BufferedImage value;

        public EzipImage(String name) {
                super(name, "image");
        }

        public EzipImage(BufferedImage arg_value) {
                super(null, "image");
                value = arg_value;
        }

        public EzipDataType equi(EzipDataType image) {
                if(image instanceof EzipImage){
```

```
                EzipImage ezi = (EzipImage)image;
                DataBuffer db1 = ezi.getValue().getRaster().getDataBuffer();
                DataBuffer db2 = getValue().getRaster().getDataBuffer();
                int size = db1.getSize();
                if(size != db2.getSize())
                        return new EzipBoolean(false);
                for(int i = 0; i < size; i++){
                        if(db1.getElem(i) != db2.getElem(i)){
                                return new EzipBoolean(false);
                        }
                }
                return new EzipBoolean(true);

        }
                return error(image, "==");
}

public EzipDataType assign(EzipDataType b) {
        if (b instanceof EzipImage) {

                BufferedImage source = (BufferedImage) b.getValue();
                WritableRaster w = source.getRaster();
                this.value = source.getSubimage(0, 0, w.getWidth(), w.getHeight(
                return this;
        } else {
                return error(b, "=");
        }
}

public int[] getRGB(int x, int y) {
        int[] rgb = new int[3];
        getValue().getRaster().getPixel(x, y, rgb);
        return rgb;
}

public void setRGB(int x, int y, int[] rgb) {
        getValue().getRaster().setPixel(x, y, rgb);
}

private enum Operation {
        ADD, SUBTRACT, MULTIPLY, DIVIDE
}

public EzipDataType plus(EzipDataType b) {
        if (b instanceof EzipImage) return imageOperation(Operation.ADD, (EzipIm
        if (b instanceof EzipInt) return integerOperation(Operation.ADD, (EzipIn
        return error(b, "+");
}

public EzipDataType minus(EzipDataType b) {
        if (b instanceof EzipImage) return imageOperation(Operation.SUBTRACT, (E
        if (b instanceof EzipInt) return integerOperation(Operation.SUBTRACT, (E
        return error(b, "-");
}

public EzipDataType mult(EzipDataType b) {
        if (b instanceof EzipImage) return imageOperation(Operation.MULTIPLY, (E
        if (b instanceof EzipInt) return integerOperation(Operation.MULTIPLY, (E
        return error(b, "*");
}

public EzipDataType div(EzipDataType b) {
        if (b instanceof EzipImage) return imageOperation(Operation.DIVIDE, (Ezi
        if (b instanceof EzipInt) return integerOperation(Operation.DIVIDE, (Ezi
        return error(b, "/");
}


private EzipDataType imageOperation(Operation operation, EzipImage i) {
        if (i.getValue().getWidth() != this.getValue().getWidth()
                        || i.getValue().getHeight() != this.getValue().getHeight
                        "image operations can only be used on pairs of images of

        int height = getValue().getHeight();
        int width = getValue().getWidth();
        int colors = getColors(i);
        int length = height * width * colors;
        int me[] = new int[length];
        int you[] = new int[length];
        getValue().getRaster().getPixels(0, 0, width, height, me);
        i.getValue().getRaster().getPixels(0, 0, width, height, you);

        int pixel = 0;
        for (int y = 0; y < height; y++)
                for (int x = 0; x < width; x++) {
                        for (int color = 0; color < colors; color++) {
                                pixel = y * width * colors + x * colors + color;
                                switch (operation) {
                                        case ADD:
                                                me[pixel] = (me[pixel] + you[pix
                                                break;
                                        case SUBTRACT:
                                                me[pixel] = (me[pixel] - you[pix
                                                break;
                                        case MULTIPLY:
                                                me[pixel] = (me[pixel] * you[pix
                                                break;
                                        case DIVIDE:
                                                if(you[pixel] > 0) me[pixel] /=
                                                else me[pixel] = 255;
                                                break;
                                }
                        }
                }

        BufferedImage answer = new BufferedImage(width, height, BufferedImage.TY
        answer.getRaster().setPixels(0, 0, width, height, me);
        return new EzipImage(answer);
}

private int getColors(EzipImage i) {
        switch(getValue().getType()){
                case BufferedImage.TYPE_INT_RGB:
                        return 3;
                case BufferedImage.TYPE_3BYTE_BGR:
                        return 3;
                //case BufferedImage.TYPE_BYTE_GRAY:
                //      return 1;
                //case BufferedImage.TYPE_USHORT_GRAY:
                //      return 1;
        }
throw new EzipException("unsupported image type");
}

private EzipDataType integerOperation(Operation operation, EzipInt argument) {
        int arg = argument.getValue();
        int width = getValue().getWidth();
        int height = getValue().getHeight();
        int colors = getColors(this);
        int length = width * height * colors;
        int me[] = new int[length];
        getValue().getRaster().getPixels(0, 0, width, height, me);
        int pixel = 0;
        for (int y = 0; y < height; y++)
                for (int x = 0; x < width; x++) {
```

```
                    for (int color = 0; color < colors; color++) {
                        pixel = y * width * colors + x * colors + color;
                        switch (operation) {
                            case ADD:
                                me[pixel] = (me[pixel] + arg) <=
                                break;
                            case SUBTRACT:
                                me[pixel] = (me[pixel] - arg) >=
                                break;
                            case MULTIPLY:
                                me[pixel] = (me[pixel] * arg) <=
                                break;
                            case DIVIDE:
                                if(arg > 0) me[pixel] /= arg;
                                else me[pixel] = 255;
                                break;
                        }
                    }
                }
            BufferedImage answer = new BufferedImage(width, height, BufferedImage.TY
            answer.getRaster().setPixels(0, 0, width, height, me);
            return new EzipImage(answer);
    }

    public BufferedImage getValue() {
            if(this.value==null)
                    nullError();
            return this.value;
    }

    public void setValue(BufferedImage bi) {
            value = bi;
    }

    public EzipDataType conv(EzipDataType b) {
            if (b instanceof EzipKernel) {
                    Kernel k = new Kernel(((EzipKernel) b).getWidth(), ((EzipKernel)
                    ConvolveOp cov = new ConvolveOp(k);
                    BufferedImage answer = new BufferedImage(getValue().getWidth(),
                    cov.filter(getValue(), answer);
                    return new EzipImage(answer);
            }
            return error(b, "**");

    }
```

```
public EzipDataType getPixel(EzipDataType x, EzipDataType y, String s) {
        int[] iArray = new int[3];
        Raster r = value.getRaster();
        int xp = (Integer) x.getValue();
        int yp = (Integer) y.getValue();

        int width = this.getValue().getWidth();
        int height = this.getValue().getHeight();
        if (xp >= width || xp < 0) throw new EzipException("The␣image␣is␣only␣"
                        + "␣pixels␣wide.␣You␣tried␣to␣access␣a␣pixel␣at␣horizont
        if (yp >= height || xp < 0) throw new EzipException("The␣image␣is␣only␣"
                        + "␣pixels␣tall.␣You␣tried␣to␣access␣a␣pixel␣at␣vertical
        r.getPixel(xp, yp, iArray);
        s = s.toLowerCase();
        if (s.equals("r")) {
                return new EzipInt(iArray[0]);
        } else if (s.equals("g")) {
                return new EzipInt(iArray[1]);
        } else if (s.equals("b")) {
                return new EzipInt(iArray[2]);
        }
        throw new EzipException("getPixel()␣takes␣image,␣x,␣y,␣String␣where␣Stri
}

public void setPixel(EzipDataType x, EzipDataType y, String s, EzipDataType inte
        int i = (Integer) intensity.getValue();
        int xp = (Integer) x.getValue();
        int yp = (Integer) y.getValue();
        WritableRaster r = this.getValue().getRaster();
        int width = this.getValue().getWidth();
        int height = this.getValue().getHeight();
        if (xp >= width || xp < 0) throw new EzipException("The␣image␣is␣only␣"
                        + "␣pixels␣wide.␣You␣tried␣to␣set␣a␣pixel␣at␣horizantal␣
        if (yp >= height || yp < 0) throw new EzipException("The␣image␣is␣only␣"
                        + "␣pixels␣tall.␣You␣tried␣to␣set␣a␣pixel␣at␣vertical␣in
        int[] array = new int[4];
        this.getValue().getRaster().getPixel(xp, yp, array);

        s = s.toLowerCase();
        if (s.equals("r")) {
                array[0] = i;
        } else if (s.equals("g")) {
                array[1] = i;
        } else if (s.equals("b")) {
```

```
                    array[2] = i;
                }
                r.setPixel(xp, yp, array);
            }

}
```

A.8 EzipInt.java

Swati Kumar

```
public class EzipInt extends EzipDataType{

        private Integer value = null;

    public EzipInt( int arg_value ) {
        super(null, "int");
        value = new Integer(arg_value);
    }

    public EzipInt( String name ){
        super( name , "int" );
    }

    public Integer getValue(){
                if(this.value==null)
                        nullError();
                return this.value;
    }

    public EzipDataType assign(EzipDataType b){
        if (b instanceof EzipInt)
                value = ((EzipInt)b).getValue();
        else
                error(b,"=");
        return this;
    }

    public EzipDataType uminus() {
        return new EzipInt( -this.getValue() );
    }

    public EzipDataType plus( EzipDataType b ) {
        if( b instanceof EzipInt)
                return new EzipInt(this.getValue() + ((EzipInt)b).getValue());
        if( b instanceof EzipDouble)
                return new EzipDouble(this.getValue() + ((EzipDouble)b).getValue());
        error(b,"+");
        return null;
    }

    public EzipDataType minus( EzipDataType b ) {
        if( b instanceof EzipInt)
                return new EzipInt(this.getValue() - ((EzipInt)b).getValue());
        if (b instanceof EzipDouble)
                return new EzipDouble(this.getValue() - ((EzipDouble)b).getValue());
        error(b,"-");
        return null;
    }

    public EzipDataType mult( EzipDataType b ) {
        if( b instanceof EzipInt)
                return new EzipInt(this.getValue() * ((EzipInt)b).getValue());
        if (b instanceof EzipDouble)
                return new EzipDouble(this.getValue() * ((EzipDouble)b).getValue());
        error(b,"*");
        return null;

    }

    public EzipDataType div( EzipDataType b ) {
        if( b instanceof EzipInt)
                return new EzipInt(this.getValue() / ((EzipInt)b).getValue());
        if( b instanceof EzipDouble)
                return new EzipDouble(this.getValue() / ((EzipDouble)b).getValue());
        error(b,"/");
        return null;

    }
    public EzipDataType gt( EzipDataType b ) {
        if( b instanceof EzipInt)
                return new EzipBoolean( this.getValue() > ((EzipInt)b).getValue());
        if( b instanceof EzipDouble)
                return new EzipBoolean(this.getValue() > ((EzipDouble)b).getValue());
        error(b,">");
        return null;
    }

    public EzipDataType ge( EzipDataType b ) {
        if( b instanceof EzipInt)
```

```
                return new EzipBoolean( this.getValue() >= ((EzipInt)b).getValue());
        if( b instanceof EzipDouble)
                return new EzipBoolean(this.getValue() >= ((EzipDouble)b).getValue());
        error(b,">=");
        return null;
    }
    public EzipDataType lt( EzipDataType b ) {
        if( b instanceof EzipInt)
                return new EzipBoolean( this.getValue() < ((EzipInt)b).getValue());
        if( b instanceof EzipDouble)
                return new EzipBoolean(this.getValue() < ((EzipDouble)b).getValue());
        error(b,"<");
        return null;

    }

    public EzipDataType le( EzipDataType b ) {
        if( b instanceof EzipInt)
                return new EzipBoolean( this.getValue() <= ((EzipInt)b).getValue());
        if( b instanceof EzipDouble)
                return new EzipBoolean(this.getValue() <= ((EzipDouble)b).getValue());
        error(b,"<=");
        return null;

    }

    public EzipDataType equi( EzipDataType b ) {
    if( b instanceof EzipInt)
                return new EzipBoolean( this.getValue().intValue() == (((EzipInt)b).getV
    error(b,"==");
        return null;

    }

    public EzipDataType notequi( EzipDataType b ) {
    if( b instanceof EzipInt)
                return new EzipBoolean( this.getValue().intValue() != ((EzipInt)b).getVa
        error(b,"!=");
        return null;

    }

}
```

A.9 EzipInternalFunctions.java

Kevin Chiu

```
/*
 * EzipInternalFunctions.java : Internal functions for EZIP.
 *
 * @author Kevin Chiu kgc2113@columbia.edu
 */
import java.awt.image.BufferedImage;
import java.awt.image.WritableRaster;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingUtilities;

public class EzipInternalFunctions {

        final static int f_load = 0;
        final static int f_view = 1;
        final static int f_print = 2;
        final static int f_save = 3;
        final static int f_getWidth = 6;
        final static int f_getHeight = 7;
        final static int f_newImage = 8;
        final static int f_sqrt = 9;
        final static int f_sin = 10;
        final static int f_cos = 11;

        public static void register(EzipFunctionTable ft) {
                ft.addFunction(new EzipFunction("load", f_load));
                ft.addFunction(new EzipFunction("view", f_view));
                ft.addFunction(new EzipFunction("print", f_print));
                ft.addFunction(new EzipFunction("save", f_save));

                ft.addFunction(new EzipFunction("getWidth", f_getWidth));
                ft.addFunction(new EzipFunction("getHeight", f_getHeight));

                ft.addFunction(new EzipFunction("newImage", f_newImage));
                ft.addFunction(new EzipFunction("sqrt", f_sqrt));
                ft.addFunction(new EzipFunction("sin", f_sin));
        ft.addFunction(new EzipFunction("cos", f_cos));
}

public static EzipDataType execute(int id, EzipDataType[] arg) {

        switch (id) {
                case f_load:
                        if (arg.length != 1) throw new EzipException("load(Strin
                        if (!(arg[0] instanceof EzipString)) throw new EzipExcep
                                        "load(<String>)␣does␣not␣take␣arguments␣
                        return load(arg[0]);

                case f_view:
                        if (arg.length != 1) throw new EzipException(
                                        "view(image)␣takes␣1␣parameter,␣the␣imag
                        if (!(arg[0] instanceof EzipImage)) throw new EzipExcept
                                        + arg[0].getType());
                        view(arg[0]);
                        return null;

                case f_print:
                        if (arg.length != 1) throw new EzipException(
                                        "print()␣takes␣1␣parameter,␣a␣non-image,
                        if ((arg[0] instanceof EzipImage) || arg[0] instanceof E
                                        "print()␣does␣not␣accept␣" + arg[0].getT
                        print(arg[0]);
                        return null;

                case f_save:
                        if (arg.length != 2) throw new EzipException("The␣syntax
                        if (arg[0] instanceof EzipImage && arg[1] instanceof Ezi
                                save(arg[0], arg[1]);
                        } else {
                                throw new EzipException("save(image,String)␣does
                                                + arg[1].getType());
                        }
                        return null;

                case f_getWidth:
                        if (arg.length != 1 || !(arg[0] instanceof EzipImage)) t
                                        "getWidth(image)␣takes␣an␣image␣and␣retu
                        else return getWidth(arg[0]);

                case f_getHeight:
                        if (arg.length != 1 || !(arg[0] instanceof EzipImage)) t
```

```
                                    "getHeight(image)␣takes␣an␣image␣and␣ret
                        else return getHeight(arg[0]);

                case f_newImage:
                        if (arg.length != 2)
                                throw new EzipException("newImage␣takes␣two␣argu
                        else if(!(arg[0] instanceof EzipInt && arg[1] instanceof
                                    "newImage(int␣height,int␣width)␣takes␣tw
                                                    + ",␣" + arg[1].getType(
                        else return newImage(arg[0], arg[1]);

                case f_sqrt:
                        if (arg.length != 1 || !(arg[0] instanceof EzipDouble ||
                                    "sqrt(double)␣takes␣a␣double␣as␣its␣sole
                        else return sqrt(arg[0]);

                case f_sin:
                        if (arg.length != 1 || !(arg[0] instanceof EzipDouble))
                                    "sin(double)␣takes␣a␣double␣as␣its␣sole␣
                        else return sin(arg[0]);

                case f_cos:
                        if (arg.length != 1 || !(arg[0] instanceof EzipDouble))
                                    "cos(double)␣takes␣a␣double␣as␣its␣sole␣
                        else return cos(arg[0]);

        }
        return null;
}

private static EzipDataType cos(EzipDataType a) {
        return new EzipDouble(Math.cos((Double) a.getValue()));
}

private static EzipDataType sin(EzipDataType a) {
        return new EzipDouble(Math.sin((Double) a.getValue()));
}

private static EzipDataType sqrt(EzipDataType a) {
        double val;
        if(a instanceof EzipInt){
                val = ((EzipInt)a).getValue() + 0.0;
        }
        else
                val = (Double) a.getValue();
```

```
		return new EzipDouble(Math.sqrt(val));
	}

	private static EzipImage load(EzipDataType file) {
		String fileString = (String) file.getValue();
		File f = new File(fileString);
		BufferedImage bi = null;
		try {
			bi = ImageIO.read(f);
		} catch (Exception e) {
			throw new EzipException("Error reading file " + "\"" + fileStrin
		}
		if (bi == null) throw new EzipException(fileString + " could not be read
		return new EzipImage(bi);
	}

	private static void view(EzipDataType image) {

		BufferedImage source = (BufferedImage) image.getValue();
		WritableRaster w = source.getRaster();
		BufferedImage destination = new BufferedImage(source.getWidth(), source.
		destination.setData(w);
		Icon icon = new ImageIcon(destination);

		JLabel label = new JLabel(icon);
		final JFrame f = new JFrame("View");
		f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		f.getContentPane().add(label);
		f.pack();
		SwingUtilities.invokeLater(new Runnable() {
			public void run() {
				f.setLocationRelativeTo(null);
				f.setVisible(true);
			}
		});
	}

	private static void print(EzipDataType e) {
		System.out.println(e.getValue());
	}

	private static void save(EzipDataType image, EzipDataType string) {
		BufferedImage bi = (BufferedImage) image.getValue();
		String filename = (String) string.getValue();
		File destination = new File(filename);
```

```
            try {
                    ImageIO.write(bi, "jpg", destination);
            } catch (IOException e) {
                    throw new EzipException("failed writing image " + filename);
            }
    }

    private static EzipInt getWidth(EzipDataType image) {
            BufferedImage bi = (BufferedImage) image.getValue();
            EzipInt i = new EzipInt(bi.getWidth());
            return i;
    }

    private static EzipInt getHeight(EzipDataType image) {
            BufferedImage bi = (BufferedImage) image.getValue();
            EzipInt i = new EzipInt(bi.getHeight());
            return i;
    }

    private static EzipImage newImage(EzipDataType width, EzipDataType height) {
            int nx = (Integer) width.getValue();
            int ny = (Integer) height.getValue();
            BufferedImage bi = new BufferedImage(nx, ny, BufferedImage.TYPE_INT_RGB)
            return new EzipImage(bi);
    }
}
```

A.10 EzipInterpreter.java

Tejas Nadkarni

```
import java.util.Vector;
import antlr.collections.AST;

public class EzipInterpreter {
        EzipSymbolTable symt ;
        EzipFunctionTable ft ;
        EzipDataType ezipReturn ;
        int loopCounter   ;
        int control ;
        int funcCounter ;

        static final int NORMAL = 0;
        static final int BREAK = 1;
        static final int CONTINUE = 2;
        static final int RETURN = 3;
```

```
public void startLoop() { loopCounter++; }
public void endLoop() { loopCounter--; }
public boolean isLoop() { return loopCounter > 0 ; }

public void startFunc() { funcCounter++; }
public void endFunc() { funcCounter--; }
public boolean isFunc() { return funcCounter > 0 ; }

public void setControl(int control) {
        this.control = control;
}

public void setReturn( EzipDataType a){
        this.ezipReturn = a;
}

public int getControl() {
        return this.control;
}
public boolean isControlNormal(){
        if(control == NORMAL)
                return true;
        else
                return false;
}

public void enterScope(){
        symt = new EzipSymbolTable(symt);
}

public void exitScope(){
        symt = symt.getParent();
}

public EzipSymbolTable getSymbolTable(){
        return this.symt;
}

EzipInterpreter(){
        symt = new EzipSymbolTable(null);
        ft = new EzipFunctionTable();
        EzipInternalFunctions.register(ft);
        control = NORMAL;
        loopCounter = 0;
```

```
            funcCounter = 0;
        }

        public EzipDataType funcInvoke(String name, EzipDataType[] args , EzipWalker wal
                EzipFunction func = ft.getFunction(name);
                if( func.isInternal()){
                        return EzipInternalFunctions.execute( func.getInternalId(), args
                }
                EzipSymbolTable old_symt = symt;
                symt = func.getParentSymbolTable();
                startFunc();
                func.execute(args,walker,this);
                endFunc();
                symt = old_symt;
                setControl(NORMAL);

                EzipDataType temp = ezipReturn;
                ezipReturn = null;
                return temp;
        }

//      public EzipDataType runInternal( int id , EzipDataType args ){
//              return EzipInternalFunctions.run( symt , id, args );
//      }

        public EzipDataType findID( String id ){
                return symt.getVariable(id);
        }

        public EzipDataType createVariable(String type,String name, boolean isArg){
                EzipDataType var = null ;

                if( type.equals("image") ) var = new EzipImage(name) ;
                if( type.equals("int") ) var = new EzipInt(name) ;
                if( type.equals("kernel") ) var = new EzipKernel(name) ;
                if( type.equals("boolean") ) var = new EzipBoolean(name) ;
                if( type.equals("double") ) var = new EzipDouble(name) ;

                if( null == var ) throw new EzipException("Unknown Datatype : "+type);
                if(!isArg) symt.addVariable(var) ;
                return var ;
        }

        public void assign(EzipDataType a, EzipDataType b){
                String aName, aType, bType ;
        aName = a.getName();
        aType = a.getType();
        bType = b.getType();

        EzipDataType var = symt.getVariable(aName) ;

        if(!aType.equals(bType)){
                throw new EzipException("Illegal operation trying to assign <"+b
        }

        var =  b;
        var.setName(aName);

        symt.setVariable(var);
}

public EzipDataType createNum(String value){
        if ( value.contains(".") || value.contains("e") )
                return new EzipDouble( Double.valueOf(value) );
        else
                return new EzipInt( Integer.valueOf(value) );
}

public EzipDataType createKernel( Vector<EzipKernel> v ){
        int height;
        int width;
        float[] f;
        Vector<Float> fl = new Vector<Float>();

        height = v.size();
        if ( height > 0 )
                width = v.elementAt(0).getValue().length;
        else
                throw new EzipException("No elements defined in Kernel");


        for(int i=0; i<height ; i++ ){
                f = v.elementAt(i).getValue();
                if( f.length != width )
                        throw new EzipException("Difference in size of row 1 and
                for( int j=0 ; j<width ; j++ )
                        fl.add(f[j]);
        }
        return new EzipKernel(width,height,fl.toArray(new Float[v.size()]));


    }

    public Float getFloat(EzipDataType a){

        if (a instanceof EzipInt)
            return ((EzipInt) a).getValue().floatValue();
        if (a instanceof EzipDouble)
            return ((EzipDouble) a).getValue().floatValue();

        throw new EzipException("Kernel can only contain float values");

    }

    public void createFunction(String name, Vector<EzipDataType> v, AST body){
        EzipFunction func = new EzipFunction(name, v.toArray(new EzipDataType[v.
        ft.addFunction(func);
    }

    public void assignPixel(String img,EzipDataType x,EzipDataType y,String rgb,Ezip
        EzipDataType image = symt.getVariable(img);

        if(!(image instanceof EzipImage))
            throw new EzipException("Pixel access operator only works on 'im

        if(!(x instanceof EzipInt) && !(y instanceof EzipInt))
            throw new EzipException("Image coords should be integers while u

        if(!(a instanceof EzipInt))
            throw new EzipException("Illegal Operation: Trying to assign "+a

        if(!rgb.equalsIgnoreCase("R") && !rgb.equalsIgnoreCase("G") && !rgb.equa
            throw new EzipException("Pixel type can only be 'R', 'G', or 'B'

        ((EzipImage)image).setPixel(x,y,rgb,a);
    }

    public EzipDataType getPixel(EzipDataType img, EzipDataType x, EzipDataType y,St

        if(!(img instanceof EzipImage))
            throw new EzipException("Pixel access operator only works on 'im

        if(!(x instanceof EzipInt) && !(y instanceof EzipInt))
```

```
                throw new EzipException("Image coords should be integers while u

            if(!rgb.equalsIgnoreCase("R") && !rgb.equalsIgnoreCase("G") && !rgb.equa
                throw new EzipException("Pixel type can only be 'R', 'G', or 'B'

            return ((EzipImage) img).getPixel(x, y, rgb);
        }

}
```

A.11 EzipKernel.java

Tejas Nadkarni

```
public class EzipKernel extends EzipDataType{

        private String type = "kernel";
        private int width;
        private int height;
        private float[] value;

        EzipKernel(Float[] value){
                super(null,"kernel");
                this.height = 1 ;
                this.width = value.length;
                this.value = getFloat(value);
        }

        EzipKernel(int width,int height, Float[] value)
        {
                super(null,"kernel");
                this.width = width;
                this.height = height;
                this.value = getFloat(value);
        }

        EzipKernel(String name){
                super(name,"kernel");
        }

        private float[] getFloat(Float[] f){
                float[] fl = new float[f.length];
                for (int i=0 ; i < f.length ; i++)
                        fl[i] = f[i];
                return fl;
        }

        public int getWidth()
        {
                return this.width;
        }
        public int getHeight()
        {
                return this.height;
        }
        public float[] getValue()
        {
                if(this.value==null)
                        nullError();
                return this.value;
        }
        public String getType()
        {
                return this.type;
        }
}
```

A.12 EzipString.java

Swati Kumar

```
public class EzipString extends EzipDataType{

        private String value;

        public EzipString(String arg_value) {
                super( null , "String" );
                this.value = arg_value ;
        }
        public String getValue() {
                if(this.value==null)
                        nullError();
                return this.value;
        }

    public EzipDataType plus( EzipDataType b ) {
        if ( b instanceof EzipString )
            return new EzipString( getValue() + ((EzipString)b).getValue() );
        return error( b, "+" );
    }
}
```

A.13 EzipSymbolTable.java

Tejas Nadkarni

```
import java.util.HashMap;


public class EzipSymbolTable extends HashMap<String, EzipDataType>{

        // This variable needs to added since this class extends another class
        static final long serialVersionUID = 1;

        EzipSymbolTable parent;

        public EzipSymbolTable(EzipSymbolTable arg_parent) {
                parent = arg_parent ;
        }

        public EzipSymbolTable getParent() {
                return parent;
        }

        public void setParent(EzipSymbolTable parent) {
                this.parent = parent;
        }

    public boolean containsVariable( String name ) {
        return containsKey( name );
    }

    public EzipDataType getVariable( String name ) {
        EzipSymbolTable st = this ;
        EzipDataType x = st.get( name ) ;

        while ( null == x && null != st.getParent() ){
                st = st.getParent();
                x = st.get( name );
        }

        if(null == x)
                throw new EzipException("Variable '"+name+"' has not been defined.") ;

        return x;
    }

    public void addVariable( EzipDataType data){
        String name = data.getName();

        if (name == null)
                throw new EzipException("Trying to store unnamed Variable in Symbol Tabl

        if ( containsVariable(name) )
                throw new EzipException("Redeclaration of variable '"+name+"'");

        put( name , data );
    }

    public void setVariable( EzipDataType data){
        String name = data.getName();

        if (name == null)
                throw new EzipException("Trying to update unnamed Variable in Symbol Tab

        EzipSymbolTable st = this ;
        do{
                if(st.containsVariable(name)) {
                        put( name, data ) ;
                        return;
                }
                st = st.getParent() ;
        }while(null != st);

        throw new EzipException("Variable '"+name+"' of type <"+data.getType()+"> has no
    }
}
```

A.14 Main.java

Tejas Nadkarni

```
/*
 * Simple front-end for an ANTLR lexer/parser that dumps the AST
 * textually and displays it graphically.  Good for debugging AST
 * construction rules.
 *
 * Behrooz Badii, Miguel Maldonado, and Stephen A. Edwards
 *
 * Modified by Tejas Nadkarni (tgn2104)
 */
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import antlr.CommonAST;
import antlr.RecognitionException;
import antlr.TokenStreamException;

public class Main {
  public static void main(String args[]) {
          String File;
          if (args.length < 1){
                  System.err.println("USAGE: ezip filename");
                  return;
          }
          else
                  File = args[0];

    try {
      FileInputStream input = new FileInputStream(File);
      EzipLexer lexer = new EzipLexer(input);
      EzipParser parser = new EzipParser(lexer);
      parser.program();
      CommonAST parseTree = (CommonAST)parser.getAST();
      EzipWalker walker = new EzipWalker();
      walker.stmt(parseTree) ;
    } catch(EzipException e) {
    } catch(TokenStreamException e) {
        System.err.println( "Ezip Token Error : ");
        System.err.println( e );
    } catch(RecognitionException e) {
        System.err.println( "Ezip Recognition Error : ");
        System.err.println( e );
    }  catch(FileNotFoundException e) {
        System.err.println( "Ezip File Error : ");
        System.err.println( e );
    }
    catch(Exception e) {
        System.err.println( "Ezip Error : ");
        System.err.println( e );
    }
  }
}
```

A.15 grammar.g

Tejas Nadkarni, Swati Kumar, Kevin Chiu, Avanti Dharkar

```
/*
 * grammar.g : the lexer and the parser of EZIP, in ANTLR grammar.
 *
 * @author Tejas Nadkarni - tgn2104@columbia.edu
 */

class EzipLexer extends Lexer;
options{
        testLiterals = false;
        charVocabulary = '\3'..'\177';
        k = 2;
}

{
        public void reportError(String err){
                throw new EzipException(err);
        }
        public void reportError( RecognitionException e ){
                throw new EzipException( e.toString() );
        }
}

protected ALPHA : ('a'..'z'|'A'..'Z'|'_') ;
protected DIGIT : ('0'..'9') ;

ID options { testLiterals = true ;}     : (ALPHA) ( ALPHA | DIGIT )* ;
NUM                : (DIGIT)+ ( '.' (DIGIT)+ )? ;

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

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

STRING
        : '"'!
                ( '"' '"'!
                | ~('"'|'\n'|'\r')
                )*
          '"'!
        ;

LPAREN  : '(';
RPAREN   :  ')';
MULT     :  '*';
CONV     :  "**";
PLUS     :  '+';
MINUS    :  '-';
DIV                 :  '/';
SEMI     :  ';';
LBRACE   :  '{';
RBRACE   :  '}';
LBLK     :  '[';
RBLK     :  ']';
ASGN     :  '=';
COMMA    :  ',';
GE                  :  ">=";
LE                  :  "<=";
GT                  :  '>';
LT                  :  '<';
EQUI     :  "==";
NEQUI    :  "!=";
COLON    :  ":";

MULTI_COMMENT    :    "/*" ( options {greedy=false;} : (NL) {} | ~( '\n' | '\r' ) )* "*/"
LINE_COMMENT     : "//" ( options {greedy=false;} : . )* '\n' {$setType(Token.SKIP); newl


class EzipParser extends Parser;

options{
    k = 2;
    buildAST = true;
}

tokens{
        DECLARATION;
        STATEMENT;
        FOR_CON;
        FUNC_CALL;
        ARG;
        MATRIX;
        ROW;
        PIXEL_ASSIGN;
        POSITIVE;
        NEGATIVE;
}
```

```
{
        public void reportError(String err){
                throw new EzipException(err);
        }
        public void reportError( RecognitionException e ){
                throw new EzipException( e.toString() );
        }
}

program
        : ( statement | func_def )* EOF!
            {#program = #([STATEMENT,"PROG"], program); }
        ;

func_def
        : "function"^ ID LPAREN! (arg (COMMA! arg)*)? RPAREN! func_body
        ;

arg
                : type ID
                        { #arg = #([ARG,"ARG"],arg); }
                ;

func_body
        : LBRACE! (statement)* RBRACE!
            {#func_body = #([STATEMENT,"FUNC_BODY"], func_body); }
        ;

statement
                : decl_stmt
                | if_stmt
                | for_stmt
                | while_stmt
                | break_stmt
                | continue_stmt
                | return_stmt
                | assign_stmt
                | pixel_assign_stmt
                | func_call_stmt
                | block
                        {#statement = #([STATEMENT,"STATEMENT"], statement); }
                ;

type
                : "image"
```

```
                    | "kernel"
                    | "int"
                    | "double"
                    | "boolean"
                    ;

decl_stmt
                    : type ( ID (ASGN expression)? ) ( COMMA! ( ID (ASGN expression)? ) )* S
                              {#decl_stmt = #([DECLARATION, "DECLARATION"], decl_stmt); }
                    ;

block     : LBRACE! (statement)* RBRACE! ;

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

for_stmt
                    : "for"^ LPAREN! (for_assgn)? SEMI! for_con SEMI! (for_assgn)? RPAREN! (


for_con
                    : (expression) {#for_con = #([FOR_CON,"FOR_CON"], for_con); }
                    ;

for_assgn
                    : ID ASGN^ expression
                    ;

while_stmt
                    : "while"^ LPAREN! expression RPAREN! ( statement | SEMI! ) ;

break_stmt
                    : "break" SEMI! ;

continue_stmt
                    : "continue" SEMI! ;

return_stmt
                    : "return"^ (expression)? SEMI! ;

assign_stmt
                    : ID ASGN^ expression SEMI! ;
```

```
pixel_assign_stmt
                : element ASGN! expression SEMI!
                        {#pixel_assign_stmt = #([PIXEL_ASSIGN,"PIXEL_ASSIGN"],pixel_assig
                ;

element : ID^ LBLK! expression COMMA! expression COMMA! STRING RBLK!
                ;

func_call_stmt
                : func_call SEMI!;

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

expr_list
                : ( ( expression | STRING ) (COMMA! (expression | STRING ))* )?
                        //{#expr_list = #([EXPR_LIST,"EXPR_LIST"], expr_list); }
                ;

expression
                : not_expr ( ( "AND"^ | "OR"^ ) not_expr )*
                ;

not_expr
                : ("NOT"^)? rltn_expr
                ;

rltn_expr
                : add_expr ( ( GE^ | LE^ | LT^ | GT^ | EQUI^ | NEQUI^ ) add_expr )?
                ;

add_expr
                : mult_expr ( ( PLUS^ | MINUS^ ) mult_expr )* ;

mult_expr
                : conv_expr ( ( MULT^ | DIV^ ) conv_expr )* ;

conv_expr
                : sign_atom ( (CONV^) sign_atom )*
                ;

sign_atom
                : PLUS! atom
```

```
                        {#sign_atom = #([POSITIVE,"POSITIVE"], sign_atom); }
                | MINUS! atom
                        {#sign_atom = #([NEGATIVE,"NEGATIVE"], sign_atom); }
                | atom
                ;

atom
                : ID
                | element
                | NUM
                | func_call
                | matrix
                | "true"
                | "false"
                | LPAREN! expression RPAREN!
                ;

matrix   : LBLK! row (COLON! row)* RBLK!
                        {#matrix = #([MATRIX,"MATRIX"], matrix); }
                ;

row             : expression (COMMA! expression )*
                        {#row = #([ROW,"ROW"], row); }
                ;




class EzipWalker extends TreeParser;
{
    EzipInterpreter ezip = new EzipInterpreter();
}

expr returns [ EzipDataType r ]
{
        EzipDataType a, b;
        r = null;
}
                : #(FUNC_CALL
                                func_name:ID ( b=expr                          { v.add(
                        )
                        {
                                r = ezip.funcInvoke(func_name.getText(), v.toArray(new E
                                if (r == null) throw new EzipException("Function␣'"+func
                        }
                | #(id:ID
                                (a=expr b=expr rgb:STRING                               { r = ez
                        )
                | #(str:STRING
                | #(num:NUM
                | #("true"
                | #("false"
                | #("AND" a=expr b=expr                                                 { r = a.
                | #("OR" a=expr b=expr                                                  { r = a.
                | #("NOT" a=expr
                | #(POSITIVE a=expr )                                                   { r = a;
                | #(NEGATIVE a=expr )                                                   { r = a.
                | #(PLUS a=expr b=expr)                                                 { r = a.
                | #(MINUS a=expr b=expr)                                                { r = a.
                | #(MULT a=expr b=expr)                                                 { r = a.
                | #(DIV a=expr b=expr)                                                  { r = a.
                | #(CONV a=expr b=expr)                                                 { r = a.
                | #(GE a=expr b=expr)                                                   { r = a.
                | #(LE a=expr b=expr)                                                   { r = a.
                | #(GT a=expr b=expr)                                                   { r = a.
                | #(LT a=expr b=expr)                                                   { r = a.
                | #(EQUI a=expr b=expr)                                                 { r = a.
                | #(NEQUI a=expr b=expr)                                                { r = a.
                | #(MATRIX
                                (a=expr
                                )*
                        )
                | #(ROW
                                (a=expr
                                )*
                        )
                | #(ARG type:. arg:ID                                                   { r = ez
                | #(FOR_CON r=expr )
                ;


stmt
{
        EzipDataType a, b, c;
}
                : #(STATEMENT
                        (s:.

                | #(DECLARATION type:.
                                (id:ID
                                        (ASGN b=expr
                                        )?
                                )*
                        )
                | #(FUNC_CALL
                                func_name:ID ( b=expr                              { v.add(
                        )
                | #(ASGN a=expr b=expr)                                            { a.assi
                | #(PIXEL_ASSIGN #(img:ID a=expr b=expr rgb:STRING) c=expr
                        {
                                ezip.assignPixel(img.getText(),a,b,rgb.getText(),c);
                        }
                )
                | #(nop:NOP
                | #("break"
                        {
                                if (ezip.isLoop())      ezip.setControl(ezip.BREAK);
                                else throw new EzipException("'break' can only be used i
                        })
                | #("continue"
                        {
                                if (ezip.isLoop())      ezip.setControl(ezip.CONTINUE);
                                else throw new EzipException("'continue' can only be use
                        })
                | #("return" ( a=expr                                              { ezip.s
                        {
                                if(ezip.isFunc()) ezip.setControl(ezip.RETURN);
                                else throw new EzipException("'return' can only be used
                        }
                | #("if" a=expr thenp:. (elsep:.)?
                  {
                        if ( !( a instanceof EzipBoolean ) )
                             throw new EzipException("if expression should be a bool");
                        if ( ((EzipBoolean)a).getValue())
                                stmt( #thenp );
                        else if ( null != elsep )
                                stmt( #elsep );
                  })
                |#("while" whilexpr:. (whilebody:.)?
                  {
                        ezip.startLoop();
                        if( !((expr(#whilexpr)) instanceof EzipBoolean) )
                                throw new EzipException("'while expression' should evalu
                while ( ((EzipBoolean)(expr(#whilexpr))).getValue() )
                {
```

```
                        if( whilebody!=null )stmt( #whilebody );
                        if(ezip.getControl() == ezip.BREAK){
                                ezip.setControl(ezip.NORMAL);
                                break;
                        }
                        if(ezip.getControl() == ezip.CONTINUE)
                                ezip.setControl(ezip.NORMAL);
                }
                ezip.endLoop();
        }
        )
      |#("for" (ex1:ASGN)? ex2:FOR_CON (ex3:ASGN)? (forbody:STATEMENT)?)
                {
                        if( ex1!=null ) stmt(ex1) ; a = expr(ex2);
                        ezip.startLoop();
                        if ( !( a instanceof EzipBoolean ) )
                                throw new EzipException("'for expression' should
                        while(((EzipBoolean)a).getValue())
                        {
                                if( forbody!=null ) stmt((#forbody));
                                if( ezip.getControl() == ezip.BREAK){
                                        ezip.setControl(ezip.NORMAL);
                                        break;
                                }
                                if(ezip.getControl() == ezip.CONTINUE)
                                        ezip.setControl(ezip.NORMAL);
                                if(ex3!=null)stmt(ex3);
                                a=expr(ex2);
                        }
                        ezip.endLoop();
                }
      |#("function" name:ID                                        { java.u
                (a=expr
                fbody:STATEMENT)
      ;
```

A.16 FunctionTest.java

Avanti Dharkar

```
import antlr.SemanticException;
import java.io.*;


/**
```

```
 * Tests for Static Semantic Analysis of functions

 * @author Avanti Dharkar (ad2518@columbia.edu)

 *

 */

public class FunctionTest extends BaseWalkerTest {

        public void testAddition() {

                try {
                        //works correctly
                    File DIR_WALKER_TEST_INPUT=new File("Test");
                        File SSAFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"testAddition.
                        EzipWalker walker = walkFile(SSAFUNC_TEST1);

                        // should not be any errors

                } catch (SemanticException e) {

                        fail(e.getMessage());

                }

        }


        public void testAssign() {

                try {
                          File DIR_WALKER_TEST_INPUT=new File("Test");
                                File SSAFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"testA
                                EzipWalker walker = walkFile(SSAFUNC_TEST1);
                        // should not be any errors
                } catch (SemanticException e) {
                        fail(e.getMessage());
                }

        }

        public void testConvolution() {
```

```
        try {
                    File DIR_WALKER_TEST_INPUT=new File("Test");
                          File SSAFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"testC
                          EzipWalker walker = walkFile(SSAFUNC_TEST1);
        } catch (SemanticException e) {

                fail(e.getMessage());
                System.err.println(e.getLine());

        }

}

public void testDivision() {
        try {

                File DIR_WALKER_TEST_INPUT=new File("Test");
                File SSAFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"testDivision.
                EzipWalker walker = walkFile(SSAFUNC_TEST1);

        } catch (SemanticException e) {

                fail(e.getMessage());

                System.err.println(e.getLine());
        }

}

public void testImageAddition() {
        try {
                File DIR_WALKER_TEST_INPUT=new File("Test");
                File SSAFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"testImageAddi
                EzipWalker walker = walkFile(SSAFUNC_TEST1);

        } catch (SemanticException e) {

                fail(e.getMessage());

                System.err.println(e.getLine());
        }

}

public void testImageDivision() {
        try {

                File DIR_WALKER_TEST_INPUT=new File("Test");
                File SSAFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"testImageDivi
                EzipWalker walker = walkFile(SSAFUNC_TEST1);

        } catch (SemanticException e) {

                fail(e.getMessage());

                System.err.println(e.getLine());
        }

}

public void testImageMultiplication() {

        try {

                File DIR_WALKER_TEST_INPUT=new File("Test");
                File SSAFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"testImageMul
                EzipWalker walker = walkFile(SSAFUNC_TEST1);

        } catch (SemanticException e) {

                fail(e.getMessage());

                System.err.println(e.getLine());
        }

}
public void testImageSubtraction() {

        try {

                File DIR_WALKER_TEST_INPUT=new File("Test");
                File SSAFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"testImageSubt
                EzipWalker walker = walkFile(SSAFUNC_TEST1);

        } catch (SemanticException e) {

                fail(e.getMessage());

                System.err.println(e.getLine());
```

```
            }

        }

        public void testMultiplication() {

                try {

                        File DIR_WALKER_TEST_INPUT=new File("Test");
                        File SSAFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"testMultiplic
                        EzipWalker walker = walkFile(SSAFUNC_TEST1);

                } catch (SemanticException e) {
                        fail(e.getMessage());
                        System.err.println(e.getLine());
                }
        }


        public void testNegVariable() {
                                try {

                        File DIR_WALKER_TEST_INPUT=new File("TestSemanticsIncorrect");
                        File SSANEGFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"test1.txt"
                        walkFile(SSANEGFUNC_TEST1);
                        fail("could walk function");
                        // variable is uninializaed.. so exception is thrown
                }
                                catch (EzipException e) {
                                        // expected

                                        assertTrue(e.getMessage().startsWith("Ezip Error
                                }
                                catch (Exception e) {
                        // expected

                        assertTrue(e.getMessage().startsWith("Ezip Error : "));
                }
        }

        // function is undefined
    public void testNegFunction() {
                try {

                        File DIR_WALKER_TEST_INPUT=new File("TestSemanticsIncorrect");
                        File SSANEGFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"test2.txt"
                        walkFile(SSANEGFUNC_TEST1);
                        fail("could walk function");
                // function is undefined
                        } catch (Exception e) {
                                // expected

                                assertTrue(e.getMessage().startsWith("Ezip Error : "));
                        }
        }
// adding int to string
    public void testNegIntToString() {
                try {

                        File DIR_WALKER_TEST_INPUT=new File("TestSemanticsIncorrect");
                        File SSANEGFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"test3.txt"
                        walkFile(SSANEGFUNC_TEST1);
                        fail("could walk function");
                // int to string
                        } catch (Exception e) {
                                // expected

                                assertTrue(e.getMessage().startsWith("Ezip Error : "));
                        }
        }
 // Convolving image with an int

    public void testNegImageToConv() {
                try {

                        File DIR_WALKER_TEST_INPUT=new File("TestSemanticsIncorrect");
                        File SSANEGFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"test4.txt"
                        walkFile(SSANEGFUNC_TEST1);
                        fail("could walk function");
                // Convolving image with an int
                        } catch (Exception e) {
                                // expected

                                assertTrue(e.getMessage().startsWith("Ezip Error : "));
                        }
        }
 // Convoling int with int
    public void testNegConvInt() {
                try {
                        File DIR_WALKER_TEST_INPUT=new File("TestSemanticsIncorrect");
                        File SSANEGFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"test5.txt"
                        walkFile(SSANEGFUNC_TEST1);
                        fail("could walk function");
                // Convoling int with int
                        } catch (Exception e) {
                                // expected

                                assertTrue(e.getMessage().startsWith("Ezip Error : "));
                        }
        }
// if condition with numeric condition inside
    public void testNegIfCond() {
                try {

                        File DIR_WALKER_TEST_INPUT=new File("TestSemanticsIncorrect");
                        File SSANEGFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"test6.txt"
                        walkFile(SSANEGFUNC_TEST1);
                        fail("could walk function");
                // Convoling int with int
                        } catch (Exception e) {
                                // expected

                                assertTrue(e.getMessage().startsWith("Ezip Error : "));
                        }
        }

  //Incorrect 'for' loop

    public void testNegForLoop() {
                try {

                        File DIR_WALKER_TEST_INPUT=new File("TestSemanticsIncorrect");
                        File SSANEGFUNC_TEST1=new File(DIR_WALKER_TEST_INPUT,"test7.txt"
                        walkFile(SSANEGFUNC_TEST1);
                        fail("could walk function");
                // Incorrect 'for' loop
                        } catch (Exception e) {
                                // expected

                                assertTrue(e.getMessage().startsWith("Ezip Error : "));
                        }
        }

}
```

A.17 BaseParserTest.java

Avanti Dharkar

```
import java.io.File;
import java.io.FileInputStream;

import java.io.FileNotFoundException;



import antlr.RecognitionException;

import antlr.SemanticException;

import antlr.TokenStreamException;

import antlr.collections.AST;



public abstract class BaseWalkerTest extends BaseParserTest {

        protected EzipWalker walkFile(File file) throws SemanticException {

                EzipParser parser = null;
                EzipWalker walker = new EzipWalker();
                try {

                        parser = makeParser(new FileInputStream(file));

                } catch (FileNotFoundException e) {

                        fail(e.getMessage());
                }

                try {

                        parser.program();

                } catch (RecognitionException e) {

                        fail(e.getMessage());

                } catch (TokenStreamException e) {
```

```
                fail(e.getMessage());
            }
            AST ast = parser.getAST();
            try {
                walker.stmt(ast);
            } catch (SemanticException e)   {
                throw e;
            } catch (RecognitionException e) {
                fail(e.getMessage()+" in line "+e.getLine());
            }
            return walker;
        }
}
```

A.18 EzipLexerTest.java

Avanti Dharkar

```
import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;
import antlr.ANTLRTokenTypes;
import antlr.Token;
import antlr.TokenStreamException;
import junit.framework.TestCase;



/**
 * Tests the lexer
 * @author Avanti Dharkar(ad2518)
 *
 */

public class EzipLexerTest extends TestCase {

        public void setUp() {
        }

        /**
         * Returns a lexer
     * @param input
     *   * @return

         */

        private EzipLexer getLexer(String input) {

        EzipLexer lexer = new EzipLexer(new StringReader(input));
        return lexer;
    }
        /**

         * Returns the first token produced from the given string

         * @param input string

         * @return a Token if matched

         * @throws TokenStreamException

         */

        private Token lexString(String input) throws TokenStreamException {
        EzipLexer lexer = new EzipLexer(new StringReader(input));
        Token token = lexer.nextToken();
        return token;
        }



        private void assertValidInput(String input,int type) {

                assertValidInput(input,type,null);
```

```
}

private void assertValidInput(String input,int type, String text) {
        Token token=null;
        try {
                token = lexString(input);
        } catch (TokenStreamException e) {
                fail(e.getMessage());
        }
        assertNotNull(token);
        if (text!=null) {
                assertEquals(text,token.getText());

        }
        assertEquals(type,token.getType());

}
private void assertValidInput(String input, int [] types) {
EzipLexer lexer = new EzipLexer(new StringReader(input));
        Token token=null;
        List tokens=new ArrayList();
        int tokenNum=1;
        boolean end=false;
```

```
do {
        try {
                token = lexer.nextToken();
                tokenNum++;
                assertNotNull(token);
        } catch (TokenStreamException e) {
                fail("token #"+tokenNum+" " + e.getMessage());
        }
        end=(token.getType()==ANTLRTokenTypes.EOF);
        if (!end) {
                tokens.add(token);
        }
} while (!end);
// are the two lists the same size
if (types.length!=tokens.size()) {
        assertEquals("did not get expected number of tokens",types.lengtl
}
for (int i=0;i<types.length;i++) {
        token=(Token)tokens.get(i);
        int type = types[i];
        assertEquals("failed on token with text "+token.getText(),type,t
}
}

private void assertInvalidInput(String input) {

        try {

                lexString(input);

                fail("accepted token "+input+", should not have");

        } catch (TokenStreamException e) {

                // expected

        }


}

public void testPunctuation() {

        assertValidInput("=",EzipLexer.ASGN);
        assertValidInput(":",EzipLexer.COLON);
        assertValidInput(",",EzipLexer.COMMA);
        assertValidInput(";",EzipLexer.SEMI);

}


public void testOperators() {

        assertValidInput(">=",EzipLexer.GE);
        assertValidInput("<=",EzipLexer.LE);
        assertValidInput(">",EzipLexer.GT);
        assertValidInput("<",EzipLexer.LT);
        assertValidInput("==",EzipLexer.EQUI);
        assertValidInput("!=",EzipLexer.NEQUI);
        assertValidInput("**",EzipLexer.CONV);
}

/**
```

```
 * Tests that the lexer ignores newline characters

 *

 */

public void testNewlines() {

        int [] expectedTypes={EzipLexer.ID,EzipLexer.SEMI,EzipLexer.ID,EzipLexer
        assertValidInput("myVar;\nmyOtherVar;",expectedTypes);
}



public void testIdentifier() {

        assertValidInput("iAmAVariable",EzipLexer.ID);

}

public void testSimpleComment() {

        int [] expectedTypes={EzipLexer.ID,EzipLexer.ASGN,EzipLexer.NUM,EzipLexe

        assertValidInput("myVar=5;␣//my␣inline␣comment\n;",expectedTypes);

}

public void testMultilineComment() {

        int [] expectedTypes={EzipLexer.ID,EzipLexer.ASGN,EzipLexer.NUM,EzipLexe

        assertValidInput("num=5;␣/*my␣inline␣comment\ncarrieson*/;",expectedType

}

public void testString() {

        assertValidInput("\"foo␣bar\"",EzipLexer.STRING,"foo␣bar");

        assertValidInput("\"he␣said␣\"\"They␣liked␣you\"\"\"",EzipLexer.STRING,"

}

public void testNumbers() {
```

```
                assertValidInput("1.3",EzipLexer.NUM);
                assertValidInput("1",EzipLexer.NUM);

        }

        }
```

A.19 EzipParserTest.java

Avanti Dharkar

```
import java.io.File;

import antlr.collections.AST;

/**

 * Tests for the Parser

 * @author Avanti Dharkar

 */

public class EzipParserTest extends BaseParserTest {

        /**

         * Simple valid test

         *

         */

        public void testValid() {

                assertProductionOk("program", "");

                assertProductionOk("program", "function␣f()␣{}");

                assertProductionOk("program", "image␣im;");
                assertProductionOk("program", "image␣im␣;kernel␣k;int␣i;␣double␣d;␣boole

                assertProductionOk("program", "image␣im␣;kernel␣k␣=␣[2,3,4:2,3,4:2,3,4];

                assertProductionOk("func_def", "function␣f1(image␣im,␣int␣i)␣{}");
```

```
            assertProductionOk("func_def", "function f1(double im, int i) {if(i ==0)

            assertProductionOk("for_stmt", "for(i=1;i<5;i=5);");
            assertProductionOk("break_stmt", "break;");
            assertProductionOk("decl_stmt", "kernel k =[2.3,23,4.5,34:2.3,23,4.5,34:
            assertProductionOk("decl_stmt", "image im = 1;");

            assertProductionOk("if_stmt", "if(i == 5){int j = 0;  i = i+ 1;break;} el
            assertProductionOk("while_stmt", "while(i<5);");
            assertProductionOk("while_stmt", "while(i<5) {i = i+1;}");
            assertProductionOk("continue_stmt", "continue;");


            assertProductionOk("matrix", "[2,2,2:3,3,3:4,4,4:5,5,5]");
            assertProductionOk("sign_atom", "true");

        assertProductionOk("assign_stmt", "i=5;");
            assertProductionOk("pixel_assign_stmt", "");
            assertProductionOk("sign_atom", "-true;");
            }
/**

        * Tests invalid input

        */

    public void testInvalid() {
    assertProductionNotOk("program", "i");
    assertProductionNotOk("program", "function f() {i = i+ 1}");
    assertProductionNotOk("while_stmt", "");
            assertProductionNotOk("program", "im;");
            assertProductionNotOk("program", "image ;kernel ;int; double; boolean ;"
            assertProductionNotOk("func_def", "function f1( im, int i) {}");
            assertProductionNotOk("func_def", "function f1(double im) {if(i = 0) {i=
            assertProductionNotOk("for_stmt", "for(i=1;i<5;i=5)");
        assertProductionNotOk("break_stmt", "break");
            assertProductionNotOk("decl_stmt", "im = 1;");
            assertProductionNotOk("while_stmt", "while(i<);");
            assertProductionNotOk("continue_stmt", "continue");
        assertProductionOk("assign_stmt", "i==5;");
            assertProductionNotOk("pixel_assign_stmt", "");
    }
    /**
```

```
     * Tests all the files in code/valid for parsability

     *

     */

    public void testAllValid() {  //check

      File DIR_PARSER_TEST_INPUT = new File("Test");
            assertTrue(DIR_PARSER_TEST_INPUT.isDirectory());

          for (String file:DIR_PARSER_TEST_INPUT.list()) {

                              File cd = new File(file);

                                   if (cd.isFile()) {
                                          assertParsedOk(cd);
                    }
     }
    }
}
```

A.20 BaseWalkerTest.java

Avanti Dharkar

```
import java.io.File;
import java.io.FileInputStream;

import java.io.FileNotFoundException;



import antlr.RecognitionException;

import antlr.SemanticException;

import antlr.TokenStreamException;

import antlr.collections.AST;



public abstract class BaseWalkerTest extends BaseParserTest {

        protected EzipWalker walkFile(File file) throws SemanticException {
```

```
        EzipParser parser = null;
        EzipWalker walker = new EzipWalker();
        try {

                parser = makeParser(new FileInputStream(file));

        } catch (FileNotFoundException e) {

                fail(e.getMessage());
        }

        try {

                parser.program();

        } catch (RecognitionException e) {

                fail(e.getMessage());

        } catch (TokenStreamException e) {

                fail(e.getMessage());

        }

        AST ast = parser.getAST();

        try {

                walker.stmt(ast);

        } catch (SemanticException e)   {

                throw e;

        } catch (RecognitionException e) {

                fail(e.getMessage()+" in line "+e.getLine());

        }

        return walker;

}
```

```
}
```

A.21 testAssign.txt

Avanti Dharkar

```
image im = load( "aLittleBlurred.bmp" );

image a = load( "Hello␣World.bmp" );

a = im;

view( a );
```

A.22 testAddition.txt

Avanti Dharkar

```
image im = load( "aLittleBlurred.bmp" );

image a = load( "butterfly450sq.jpeg" );

view ( im );

a = im + 150;

 view( a );
```

A.23 testImageMultiplication.txt

Avanti Dharkar

```

//Testing the multiplication operation for images. Making sure that both images are of t

image im = load( "butterfly450sq.jpeg" );

image a = load( "weirdo450x450.bmp" );

image b;
image c;

b = im * a;
c = a * im;

print ( "Image's␣are␣here" );
```

```
view( b );
view( c );
```

A.24 testSubtraction.txt

Avanti Dharkar

```
image im = load( "aLittleBlurred.bmp" );

image a = load( "Hello World.bmp" );

view ( im );

a = im - 150;

view( a );
```

A.25 testMultiplication.txt

Avanti Dharkar

```
image im = load( "aLittleBlurred.bmp" );

image a = load( "Hello World.bmp" );

view ( im );

a = im * 25;

view( a );
```

A.26 testImageSubtraction.txt

Avanti Dharkar

```


//Testing the subtraction operation for images. Making sure that both images are of the


image im = load( "butterfly450sq.jpeg" );

image a = load( "weirdo450x450.bmp" );

image b;
image c;
```

```
b = im - a;
c = a - im;

view( b );
view( c );
```

A.27 testImageAddition.txt

Avanti Dharkar

```
//Testing the addition operation for images. Making sure that both images are of the sam

image im = load( "butterfly450sq.jpeg" );

image a = load( "weirdo450x450.bmp" );


a = im + a;

view( a );
```

A.28 testImageDivision.txt

Avanti Dharkar

```
//Testing the division operation for images. Making sure that both images are of the sam

image im = load( "butterfly450sq.jpeg" );

image a = load( "weirdo450x450.bmp" );

image b;
image c;

b = im / a;
c = a / im;

view( b );
view( c );
```

A.29 testDivision.txt

Avanti Dharkar

```
image im = load( "aLittleBlurred.bmp" );

image a = load( "Hello␣World.bmp" );

view ( im );

a = im / 2;

view( a );
```

A.30 testConvolution.txt

Avanti Dharkar

```
image im = load( "aLittleBlurred.bmp" );

//image im = load( "Hello World.bmp" );


//int b = 200;
//int c = 100;

kernel k = [0,11,0:11,4,1:10,11,0];

//print(k);

im = im ** k ;

//int b = b + 10 ;
//a = a + 100 ;

view( im );



//view( a );
//print( b>=201 );
```

A.31 testIncorrect1.txt

Avanti Dharkar

Lexically and syntactically right but incorrect semantically.

```
//using unitialized variables

view ( im );
```

```
a = im + 150;

 view( a );
```

A.32 testIncorrect2.txt

Avanti Dharkar

Lexically and syntactically right but incorrect semantically.

```
int i = 5 + "hello";
```

A.33 testIncorrect3.txt

Avanti Dharkar

Lexically and syntactically right but incorrect semantically.

```

//undefined function
int i = add(2,3);
```

A.34 testIncorrect4.txt

Avanti Dharkar

Lexically and syntactically right but incorrect semantically.

```
//convolution of image with int
image im;
im = im ** 3;
```

A.35 testIncorrect5.txt

Avanti Dharkar

Lexically and syntactically right but incorrect semantically.

```
// convolving int with int

int i;
image im = i ** i;
```

A.36 testIncorrect6.txt

Avanti Dharkar

Lexically and syntactically right but incorrect semantically.

```
//Instead of the typical boolean expression in if we give a number
if(0){ print ("I send out an error";}
if(5+5) {print("Same here because I am an error too"}
```

A.37 testIncorrect7.txt

Avanti Dharkar

Lexically and syntactically right but incorrect semantically.

```
//incorrect thing in loop

for(i=i+5; i<3; )
          { i = i + 1;
          }
```

A.38 kernel.txt

The demos were a collaborative effort between all team members.

```
image i = load("cameraman.bmp");

view(i);

kernel emboss = [1  ,0, 0:
                                        0,        0,        0:
                                        0,        0,        -1];

kernel shake = [0.5  ,0, 0:
                                        0,        0,        0:
                                        0,        0,        0.5];

kernel edge = [ 0, 1, 0 : 1, -4, 1 : 0, 1, 0];

kernel blur = [ 0,        0.25,     0         :
                                        0.25,     0,        0.25:
                                        0,        0.25,     0         ];

kernel shake2 =    [       0,        0,        0.5       :
                                                  0,        0,        0         :
                                                  0.5,0,    0         ];


function applyKernel(image im,kernel k,int n){
          if(n>0)
                    return applyKernel(im,k,n-1)**k;
          else
                    return im;
}


image i1 = (i-(i- i**edge))*3;
image i2 = (applyKernel(i,shake,100)/2 + applyKernel(i,shake2,100)/2) ;
image i3 = applyKernel(i,blur,100);
image i4 = i**edge;
view(i1);
view(i2);
view(i3);
view((i-(i-i**emboss))*3+i);
view(i1+i2);
view(i3+i4);
view(i**emboss);
//view(i);
```

A.39 histogram.txt

The demos were a collaborative effort between all team members.

```
image img = load("cameraman.bmp");



function histogram(image img){
        int rows = getHeight(img);
        int cols = getWidth(img);

        int w=256,h=300;

        image hist = newImage(w+1,h+1);
        image cntr = newImage(w+1,1);
        int i,j;
        int pixel;

        for(i=0;i<cols;i=i+1){
                for(j=0;j<rows;j=j+1){
                        pixel = img[i,j,"R"];

                        if(h-(cntr[pixel,0,"G"] * 100 + cntr[pixel,0,"R"])/5 < h ){
                                hist[pixel,h-(cntr[pixel,0,"G"] * 100 + cntr[pixel,0,"R"
                                hist[pixel,h-(cntr[pixel,0,"G"] * 100 + cntr[pixel,0,"R"
                                hist[pixel,h-(cntr[pixel,0,"G"] * 100 + cntr[pixel,0,"R"
                        }

                        if (cntr[pixel,0,"R"]==99){
                                cntr[pixel,0,"R"]=0;
                                cntr[pixel,0,"G"]=cntr[pixel,0,"G"]+1;
                        }
                        else
                                cntr[pixel,0,"R"]=cntr[pixel,0,"R"]+1;

                }
        }
        return hist;
}


view(img);
view(histogram(img));
```

A.40 integer.txt

The demos were a collaborative effort between all team members.

```
/* Function to find the Factorial */
function fact ( int a )
{
        if( a > 0 )
                return fact(a-1)*a;
        else
                return 1;
}

/* Function to print the Fibonacci series */
function fibonacci(int a, int b, int c){
        if( (a + b) < c ){
                print(a+b);
                if( a < b )
                        fibonacci(a+b,b,c);
                else
                        fibonacci(a,a+b,c);
        }
}

// Fibonacci Series
print("Fibonacci␣Series␣:");
fibonacci(0,1,1000);

// Factorial
print("Factorial␣:␣");
print(fact(6));
```

A.41 threshold.txt

The demos were a collaborative effort between all team members.

```
function threshold(image img,int threshold){
        int rows = getHeight(img);
        int cols = getWidth(img);
        int i,j;

        for(i=0;i<rows;i=i+1){
                for(j=0;j<cols;j=j+1){
                        if((img[i,j,"R"]+img[i,j,"R"]+img[i,j,"R"])/3 < threshold){
                                img[i,j,"R"] = 0 ;
                                img[i,j,"G"] = 0 ;
                                img[i,j,"B"] = 0 ;
                        }else{
                                img[i,j,"R"] = 255 ;
                                img[i,j,"G"] = 255 ;
                                img[i,j,"B"] = 255 ;
                        }
                }
        }
        return img;
}

image cam = load("cameraman.bmp");

view(cam);
view(threshold(cam,100));
```

A.42 invert.txt

The demos were a collaborative effort between all team members.

```
function invert(image img){
        int rows = getHeight(img);
        int cols = getWidth(img);
        int i,j;

        for(i=0;i<rows;i=i+1){
                for(j=0;j<cols;j=j+1){
                        img[i,j,"R"] = 255 - img[i,j,"R"];
                        img[i,j,"G"] = 255 - img[i,j,"G"];
                        img[i,j,"B"] = 255 - img[i,j,"B"];
                }
        }
        return img;
}


image cam = load("cameraman.bmp");
view(cam);

view(invert(cam));
```

B References

- Antlr 2.0 Documentation `http://www.antlr2.org/doc/index.html`
- Antlr 2.0 Tutorial `http://www.javadude.com/articles/antlrtut/`
- MX Source Code `http://www1.cs.columbia.edu/~sedwards/classes/2003/w4115/mx030521.zip`
- C Reference Manual `http://cm.bell-labs.com/cm/cs/who/dmr/cman.pdf`
- ANSI C grammar for ANTLR v3 `http://www.antlr2.org/grammar/1153358328744/c.g`